Universidade Federal de Pernambuco

Departamento de Eletrônica e Sistemas

Curso de Engenharia Eletrônica

Robô garçom baseado em plataforma de robô seguidor de linha com visão computacional

Trabalho de Conclusão de Curso de Graduação

por

Bruno Campello Tôrres de Azevedo Teles

Orientador: João Marcelo Teixeira

Recife, Janeiro de 2023

Bruno Campello Tôrres de Azevedo Teles

Robô garçom baseado em plataforma de robô seguidor de linha com visão computacional

Monografia apresentada ao Curso de Engenharia Eletrônica, como requisito parcial para a obtenção do Título de Bacharel em Engenharia Eletrônica, Centro de Tecnologia e Geociências da Universidade Federal de Pernambuco.

Orientador: João Marcelo Teixeira

Banca Examinadora

Documento assinado digitalmente
gov.br
JOAO MARCELO XAVIER NATARIO TEIXEIRA
Data: 23/01/2023 19:41:07-0300
Verifique em https://verificador.iti.br

Prof. João Marcelo Teixeira

Documento assinado digitalmente
gov.br
REUBEN PALMER REZENDE DE SOUSA
Data: 23/01/2023 20:10:42-0300
Verifique em https://verificador.iti.br

Prof. Reuben Palmer Rezende de Sousa

Recife

2023

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Teles, Bruno Campello Tôrres de Azevedo.
Robô garçom baseado em plataforma de robô seguidor de linha com visão computacional / Bruno Campello Tôrres de Azevedo Teles. - Recife, 2023.
44 p. : il., tab.

Orientador(a): Joao Marcelo Xavier Natario Teixeira
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Engenharia Eletrônica - Bacharelado, 2023.

1. Robótica. 2. Seguidor de Linha. 3. Visão Computacional. 4. Busca em Grafos. I. Teixeira, Joao Marcelo Xavier Natario. (Orientação). II. Título.

620 CDD (22.ed.)

Agradecimentos

Aos meus pais, irmão e a toda minha família que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida.

Aos amigos que fiz durante o curso, particularmente Isaac, Romário, Pedro Henrique, Túlio, Vinícius, Arthur e Victor. Sem vocês, não seria possível ver a reta final deste desafio.

A todos os professores do Departamento de Eletrônica e Sistemas, mas principalmente ao professor João Marcelo, que se colocou à disposição de forma irrestrita para que eu pudesse realizar este projeto, mesmo nas situações mais adversas.

O sucesso é ir de fracasso em fracasso sem perder entusiasmo
para a sua produção ou a sua construção.

Winston Churchill

RESUMO

Seguindo tendências mundias de automação, o transporte de cargas também vem se modernizando, criando soluções autônomas capazes de levar diferentes tipos de volumes para pontos de interesse. O setor de restaurantes vem aumentando seu interesse na utilização de tal tecnologia aplicada à área, os *Robôs Garçons*. Capazes de levar pedidos para as mesas de forma rápida e eficiente, a demanda destes robôs cresceu bastante por causa também da pandemia de Covid-19, que aumentou o interesse por qualquer tecnologia que diminuísse o contato humano.

Neste trabalho é apresentada uma solução de robô garçom utilizando uma plataforma de robô seguidor de linha, capaz de encontrar o menor caminho possível entre dois pontos utilizando teoria dos grafos. As informações dos caminhos existentes são obtidas através do uso de visão computacional e a transmissão de dados entre computador e microcontrolador é realizada por meio de *Bluetooth*.

A solução desenvolvida atingiu o objetivo esperado, tanto por parte da navegação (robô pode se mover entre nós bastando apenas indicar qual é o nó de destino), quanto para a parte de visão computacional (foi possível obter um grafo a partir de uma imagem representando todos os nós de forma correta).

Palavras-chave: Robô Garçom, Pololu 3pi, Robô Seguidor de Linha, Visão Computacional.

ABSTRACT

Following global automation trends, cargo transport has also been modernizing, creating autonomous solutions capable of taking different types of volumes to points of interest. The restaurant sector has been increasing its interest in the use of such technology applied to the area, the *Robot Waiters*. Capable of taking orders to tables quickly and efficiently, the demand for these robots has grown a lot because of the Covid-19 pandemic, which has increased interest in any technology that reduces human contact.

In this work, a robot waiter solution is presented using a line follower robot platform, capable of finding the shortest possible path between two points using graph theory. Information on existing paths is obtained through the use of computer vision and data transmission between computer and microcontroller is performed via *Bluetooth*.

The developed solution achieved the expected objective, both in terms of navigation (the robot can move between nodes simply by indicating which is the destination node) and for the computer vision part (it was possible to obtain a graph from an image representing all nodes correctly).

Key-words: Robot Waiter, Pololu 3pi, Line Follower Robot, Computer Vision.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AGV	Veículo autoguiado
UFPE	Universidade Federal de Pernambuco
LDR	Sensor de luminosidade
RGB	Vermelho, Verde e Azul, sistema de cores
RFID	Identificação por radiofrequência
IDE	Ambiente de desenvolvimento integrado
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
CC	Corrente continua
RPI	Raspberry Pi
ROI	Região de Interesse
ISP	In-system programming
Ni-MH	Níquel-Hidreto Metálico
GHz	Giga Hertz
V	Tensão
kg	Quilograma
mm	Milímetro
m	Metro
VCC	Alimentação circuito integrado
GND	Terra
E/S	Entrada/Saída

SUMÁRIO

1 INTRODUÇÃO

Com a crescente aplicação de tecnologias a atividades cotidianas, tornou-se comum automatizar o maior número de tarefas, das mais simples às mais complexas. A automação tem potencial de ser aplicada em qualquer ambiente, dentre eles, em restaurantes.

Já existem automações bastante conhecidas e adotadas em restaurantes, como os sistemas de pedidos (que unificam todos os pedidos dos clientes, de forma rápida e organizada, e os repassam para a cozinha), softwares de gestão de estoque, de gestão financeira, etc [4]. Outra aplicação conhecida é a esteira utilizada em restaurantes de sushi, que existe desde 1958 [5].

A aplicação que se tornou o destaque e o foco deste trabalho é o *Robô Garçom*, utilizado em restaurantes, e que parte do princípio de um robô seguidor de linha industrial, o *AGV* (*Automated Guided Vehicle*, ou Veículo autoguiado), com aplicação de transporte de cargas de diferentes setores [6], como: ambientes automotivos, indústria de alimentos e bebidas, unidades de saúde, indústria de empacotamento e distribuição em armazéns.

Os primeiros robôs garçons foram utilizados já em 1983 no restaurante *Two Panda Deli* em Pasadena, California, EUA [7]. Apelidados de *Tanbo R-1* e *Tanbo R-2*, os robôs serviam comida aos clientes enquanto contavam piadas e tocavam músicas. Porém, por ser uma implementação inicial de tal tecnologia, que até hoje se mostra difícil de viabilizar, *Tanbo R-1* e *Tanbo R-2* não executavam suas tarefas de forma consistente, derrubando a comida ou andando em círculos quando rádios policiais estavam próximos, ou arrastando palavras, como se estivesse bêbado, quando suas baterias estavam acabando. Além disso, cada um custava *US$ 20.000,00* e pesava aproximadamente 36 kg.

É natural, portanto, que esta aplicação não tenha sido rapidamente difundida e amplamente utilizada nos anos seguintes. Garçons humanos continuavam sendo uma opção mais barata, prática e consistente.

No entanto, em anos recentes, com o crescimento exponencial de tecnologias - possibilitando uma implementação mais completa, barata, leve e robusta - e com a repentina falta de funcionários e necessidade de redução de contato humano devido à pandemia de Covid-19 [8] [9] [10], restaurantes ao redor do globo começaram a empregar robôs para servirem seus clientes. Segundo Grace Dickinson [11], outros fatores decisivos também são: robôs não se cansam, podendo exercer sua função com a mesma eficiência do pri-

meiro ao último momento do dia, enquanto uma pessoa se cansará após algumas horas de trabalho; salários de garçons estão sempre aumentando ao longo do tempo, enquanto o custo de uma solução robótica é cada vez mais acessível; assim como em 1983, os robôs em si servem como atrações para os restaurantes, atraindo clientes curiosos ao negócio.

Este projeto surgiu como projeto final de duas disciplinas de graduação do curso de Engenharia Eletrônica: Microcomputadores e Processamento de Imagem. O intuito original era de criar um robô que se locomovesse por um espaço delimitado branco com caminhos delineados em preto pré-definidos e usaria visão computacional para observar todas as mesas (destinos dos caminhos) para se encaminhar para uma mesa que tenha um sinal que signifique que está chamando o robô, como ilustrado na Figura 1. O sinal utilizado foi um pedaço de papel verde (e em seu verso, vermelho) ao lado da mesa, que por meio do processamento da imagem era reconhecido e então enviado ao robô que a mesa estava lhe chamando. A comunicação entre o robô e o software de processamento foi viabilizada utilizando *Bluetooth*, transmitindo a informação da mesa para qual o robô deveria ir, e após isso voltar para um ponto inicial, considerado como a cozinha. Parte do vídeo da apresentação desta versão inicial do trabalho pode ser acessada por este link [12].

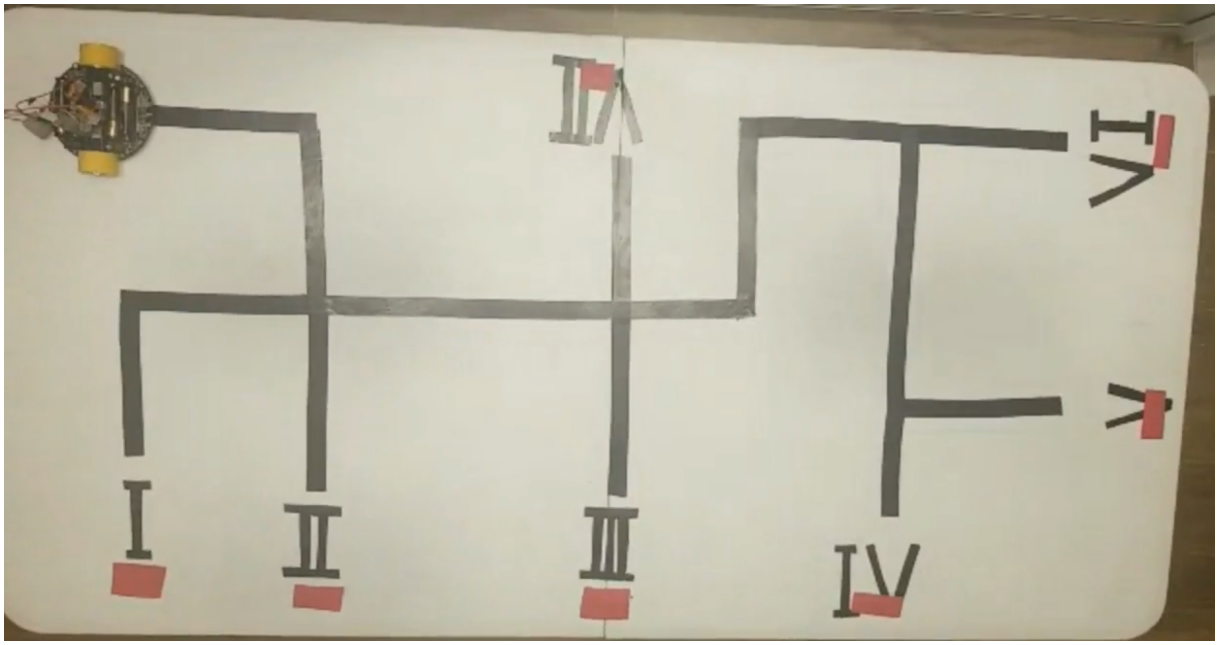

Figura 1: Imagem da apresentação do projeto na durante aula da disciplina de Microcomputadores.

Após a implementação inicial com sucesso, foram modificadas algumas premissas do projeto: não mais teria um observador constante (câmera posicionada no teto) para saber qual mesa estava chamando o robô, isto seria feito por um sinal enviado pelo usuário;

a informação a ser armazenada no *hardware* passou a ser obtida a partir de uma foto de todos os caminhos, sendo feito processamento de imagem para reconhecer todas as mesas, seus caminhos, e as curvas que o robô deve realizar para se mover um ponto a outro, criando então um grafo que permite a navegação ponto a ponto (mesa a mesa). Esta modificação tinha como objetivo facilitar a configuração do "mapa" do robô em um novo ambiente com mesas dispostas em posições distintas. Com estas modificações, a comunicação entre o robô e o software passou a transmitir mais informações, como o grafo com as informações das posições da mesa e onde o robô se encontrava em seu momento inicial.

1.1 Objetivos

1.1.1 Objetivo Geral

Criar robô que conseguisse se locomover de um ponto a qualquer outro ponto de uma maquete branca com caminhos definidos em preto, reconhecidos e transmitidos de forma automática para o mesmo através de um software executando em um computador.

1.1.2 Objetivos Específicos

Para atingir o objetivo geral, foram definidos os seguintes objetivos específicos:

- Estudar sobre projetos de robôs seguidores de linha.
- Estudar sobre possibilidades para criação de um grafo e busca mais eficiente de caminhos nele.
- Estudar sobre processamento de imagem, com foco em reconhecimento dos caminhos na foto.
- Especificar a forma como os caminhos serão armazenados e pesquisados.
- Modificar a versão já existente de navegação do robô garçom para navegação por grafo.
- Desenvolver software para encontrar caminhos usando visão computacional.
- Desenvolver software para transmitir dados para o *hardware*.

- Conduzir testes para validação funcional do protótipo.

1.2 Organização do texto

Este trabalho está dividido em seis capítulos, os quais abordam a contextualização e motivação à realização desse trabalho, assim como os principais trabalhos relacionados, o desenvolvimento do trabalho em si, assim como os testes realizados, principais dificuldades encontradas e as conclusões obtidas do mesmo. A separação utilizada para os capítulos é explicada abaixo, assim como os tópicos que cada um contém:

- Capítulo 1: Introdução, engloba a contextualização do trabalho e seus objetivos.
- Capítulo 2: Trabalhos Relacionados, lista alguns trabalhos encontrados que se relacionam diretamente com a presente proposta.
- Capítulo 3: Solução proposta, traz a definição do projeto desenvolvido nesse trabalho e detalha as tecnologias nele utilizadas, juntamente com o detalhamento do desenvolvimento realizado.
- Capítulo 4: Testes, detalha a execução dos testes de validação da proposta.
- Capítulo 5: Dificuldades Encontradas, lista as dificuldades encontradas nesse trabalho e como elas foram superadas.
- Capítulo 6: Conclusão e Trabalhos Futuros, apresenta as conclusões acerca do trabalho desenvolvido e lista possibilidades futuras para continuidade do mesmo.

2 TRABALHOS RELACIONADOS

Com o interesse crescente nos últimos tempos, e devido à complexidade e à possibilidade de aprendizado, a implementação de robôs seguidores de linha e robôs garçons virou um projeto interessante para alunos de áreas de atuação que envolvem conhecimentos de *hardware* e *software*, principalmente microcontroladores e visão computacional. Em seguida, serão listados alguns projetos com uma premissa de criar um robô garçom, com diferentes abordagens.

2.1 Robô seguidor de linha alimentado por fonte de energia alternativa

Este Trabalho de Conclusão de Curso foi desenvolvido em 2019 por Machado e Moura [1] com o intuito de unir uma aplicação de robô seguidor de linha à alimentação por fonte de energia alternativa a ser utilizado para entrega e distribuição de materiais dentro da Unidade Pedra Branca da Universidade do Sul de Santa Catarina.

Ao longo deste projeto foram desenvolvidas duas versões. A primeira versão foi desenvolvida utilizando os componentes listados a seguir, e é ilustrada na Figura 2 e Figura 3:

- Um chassi comercial
- Rodas
- Arduino
- *Driver* de acionamento dos motores
- Sensores de luminosidade (*LDR*)
- Motores CC alimentados por pilhas AA

Esta primeira versão foi testada percorrendo percursos pré-definidos, compostos por retas e trechos sinuosos (onde apresentou dificuldades inicialmente), em bancadas de laboratórios com condições de luminosidade controladas.

Realizados os ajustes necessários, foram feitos novos testes no piso do corredor da Universidade, onde foi constatada a limitação de sensores de luz para a detecção de caminho devido a variações de condições de iluminação, e então tomada a decisão de

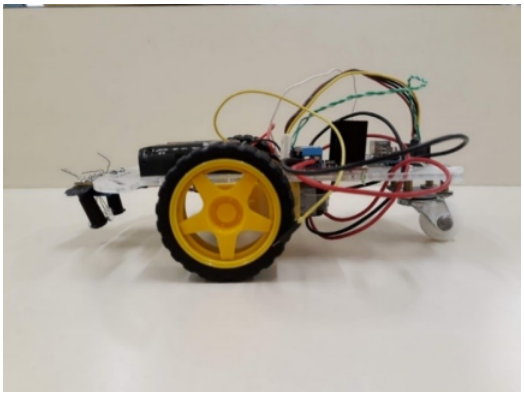

Figura 2: Vista lateral da primeira versão do protótipo de seguidor de linha de Machado e Moura [1].

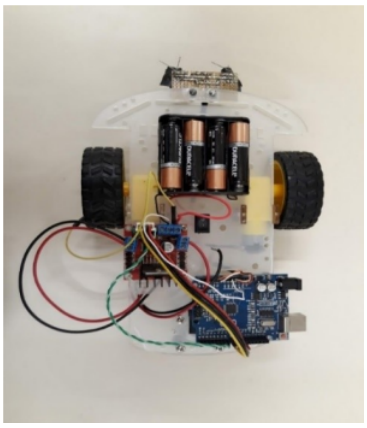

Figura 3: Vista superior da primeira versão do protótipo de seguidor de linha de Machado e Moura [1].

utilizar um módulo com sensores infravermelhos, conforme ilustrado na Figura 4. Com a adição dos novos sensores, o comportamento se tornou estável, comportando-se melhor sob diferentes condições de iluminação. Adicionalmente, foram instalados sensores de colisão frontal.

Na segunda versão também foi adicionado um sensor de cor RGB para identificar sua posição. Devido ao aumento da quantidade de sensores, foi observada a necessidade de mudar o microcontrolador utilizado, de um Arduino para uma ESP32, que possuía todas as portas necessárias. Nesta etapa também foi adicionado um sistema de localização via internet, graças à característica da placa ESP32, que permite tal conexão.

Outra grande mudança da segunda versão foi o chassi utilizado, modificado para um chassi de MDF maior e mais robusto, para poder exercer sua função de transporte de materiais com maior segurança. Devido ao aumento de tamanho e peso da chassi, os motores foram trocados por maiores com mais torque, já que os primeiros motores

Figura 4: Módulo Seguidor de Linha 5 Canais com Fim de Curso.

utilizados eram adequados para o primeiro chassi, de plástico. As mudanças podem ser visualizadas na Figura 5.

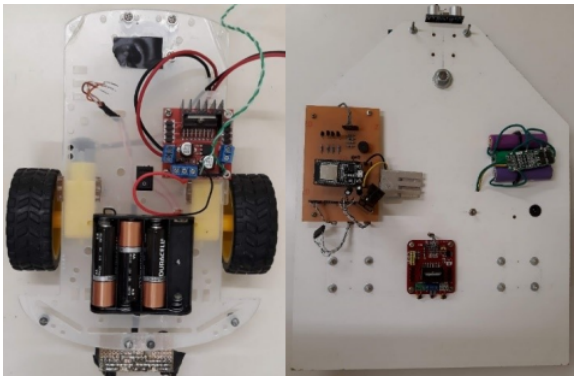

Figura 5: Comparação de vistas superiores das duas versões do robô

Para respeitar o conceito inicial de ser alimentado por fonte de energia alternativa, as baterias desta versão foram trocadas por pilhas de íon-lítio recarregáveis de 4,2 V que foram reaproveitadas de notebooks usados. A mudança também foi necessária devido ao aumento de tensão fornecida aos novos motores e à maior capacidade de armazenamento de energia das novas baterias.

Com isto, foi desenvolvido por completo o robô, capaz de se locomover seguindo as linhas definidas e sendo controlado via Wi-Fi, por onde recebe os comandos da localização que deve ir. Também foi possível atualizar o *firmware* do projeto a partir da conexão Wi-Fi.

2.2 Robçom, o Robô Garçom

Este robô [13] utiliza a mesma plataforma de robô seguidor de linha com uma finalidade um pouco mais específica: transportar bebidas de acordo com pedidos feitos

por clientes. Além disso, as bebidas transportadas por Robçom são preparadas de forma autônoma, com auxílio de bombas hidráulicas.

Existem duas partes diferentes que compõem o projeto: a estação base, onde fica um *dispenser* de copos e são preparadas as bebidas e o robô seguidor de linha, com um chassi omnidirecional de 3 rodas com suporte para carregar copos.

Para a construção do seguidor de linha foram utilizados os seguintes componentes:

- Chassi omnidirecional
- Servo motores
- Sensor Infravermelho 5 vias BfD-1000
- Sensor RFID
- Ponte H
- Rodas
- Arduino Mega 2560 R3
- *Power Banks*

A alimentação do robô se deu por via de 3 *Power Banks* para energizar o Arduino e os 3 motores. A versão final do Robçom pode ser visualizada na Figura 6.

O algoritmo inicial de movimentação do Robçom utilizou curvas omnidirecionais sobre seu eixo central. No entanto, a performance com este algoritmo apresentou baixa precisão, gerando a necessidade de trocar o algoritmo para um baseado em controle proporcional, que apresentou um comportamento melhor.

Figura 6: Versão final do protótipo - Robçom.

A segunda parte do projeto consiste na Estação Base, que recebe pedidos de um site desenvolvido utilizando HTML(*Hypertext Markup Language*) e CSS(*Cascading Style Sheets*) (conforme ilustrado na Figura 7) e os prepara. A comunicação é feita através de Wi-Fi, oferecendo seis combinações possíveis, à escolha do usuário.

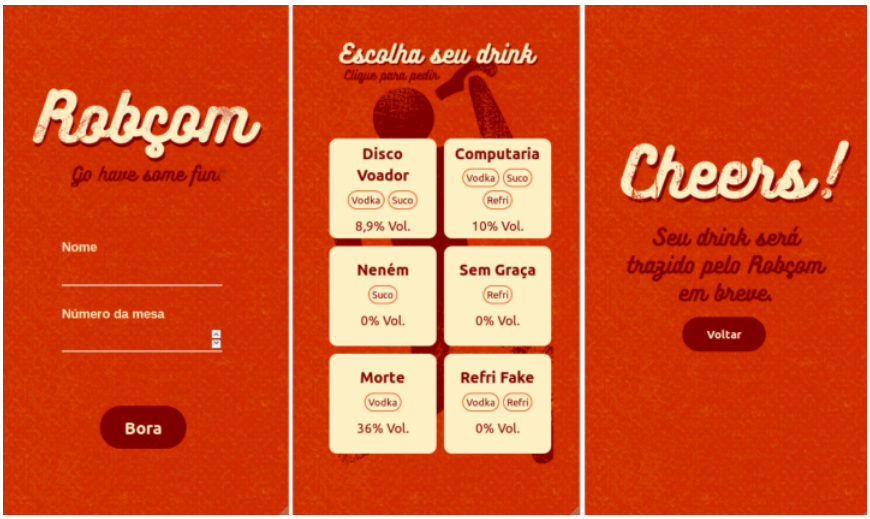

Figura 7: Interface com o cliente na forma de páginas web - Robçom.

A comunicação entre a máquina de *drinks* e o robô foi realizada através de *Bluetooth*. Foram transmitidas informações como: posicionamento do Robçom e a mesa destino do pedido.

Para melhor visualização do resultado do projeto, foi criado um vídeo, demonstrando o funcionamento do Robçom (link).

2.3 Alfred: O Robô Garçom

Semelhante a Robçom (Seção 2.2), este trabalho foi desenvolvido para a disciplina Oficina de Integração 3 da Universidade Tecnológica Federal do Paraná, porém este é um antecessor ao mencionado, realizado em 2016. O intuito de Alfred é o mesmo, de criar um robô garçom conectado a aplicativos *Android*. Porém, um fator interessante deste projeto é que a detecção de linhas foi realizada por visão computacional utilizando imagens obtidas por uma câmera ligada ou robô, diferente dos outros exemplos, que usaram sensores. A versão final de Alfred é ilustrada na Figura 8.

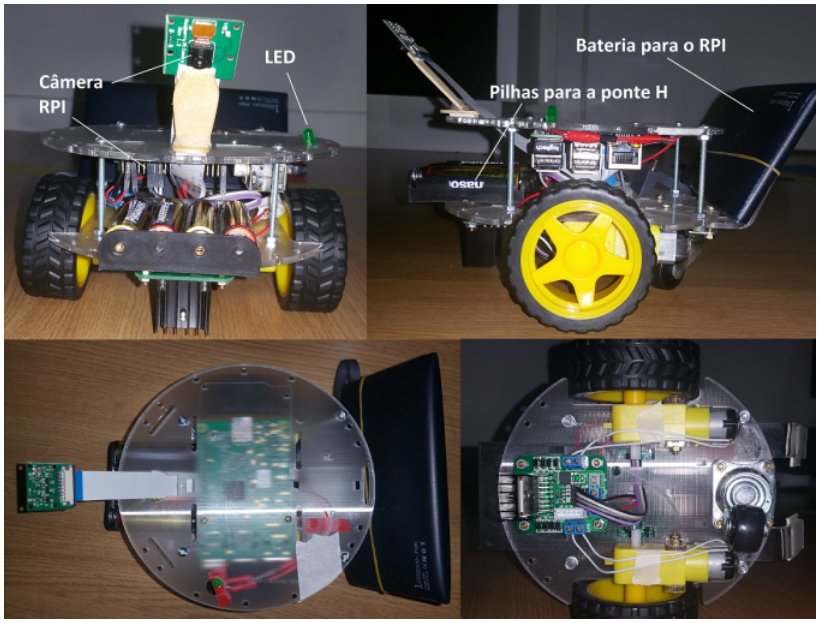

Figura 8: Versão final do protótipo Alfred.

Os componentes utilizados para construir Alfred foram:

- Raspberry Pi 3
- Módulo de câmera RPI
- Chassi comercial em acrílico
- Motores CC
- Rodas
- Ponte H L298N

A Estação Base de Alfred consiste de dois aplicativos Android: o aplicativo Cliente e o aplicativo Gerente. Como o nome indica, o aplicativo Cliente (ilustrado na Figura 9) é a interface onde um cliente é cadastrado, informando o nome a mesa em que se encontra. Nele é possível fazer um pedido, consultar seu *status* e fechar a conta. Já no aplicativo Gerente (ilustrado na Figura 10), é possível adicionar ou remover itens do cardápio, acompanhar os pedidos dos clientes listados por ordem de chegada, e a opção de enviar pedido, utilizada quando o pedido é colocado em cima do robô garçom.

Figura 9: Interface do aplicativo Cliente Alfred.

Figura 10: Interface do aplicativo Gerente Alfred.

Como mencionado, diferente de outros seguidores de linha que utilizam sensores para navegação (Seção 3.1.1), Alfred utiliza visão computacional, mais especificamente, a biblioteca *OpenCV* [14], amplamente utilizada em projetos da área. Cada imagem obtida pela câmera é transformada, analisada e processada para retornar um ponto branco para

cada ponto dentro de um limite determinado, indicando então o caminho a ser seguido em branco e o restante em preto, como ilustrado na Figura 11.

Saindo de um ponto inicial, definido como Cozinha, o robô tem duas opções de caminhos a seguir, cada um com uma cor distinta e pré-definida. Portanto, quando um destino é escolhido, a imagem binária resultante somente mostrará uma linha. Tendo a imagem com a linha claramente definida, são definidas três regiões de interesse (*ROI*, *Region of interest*), como exibido pela Figura 11. O sentido no qual o robô irá rotacionar será o que apresentar mais *pixels* brancos.

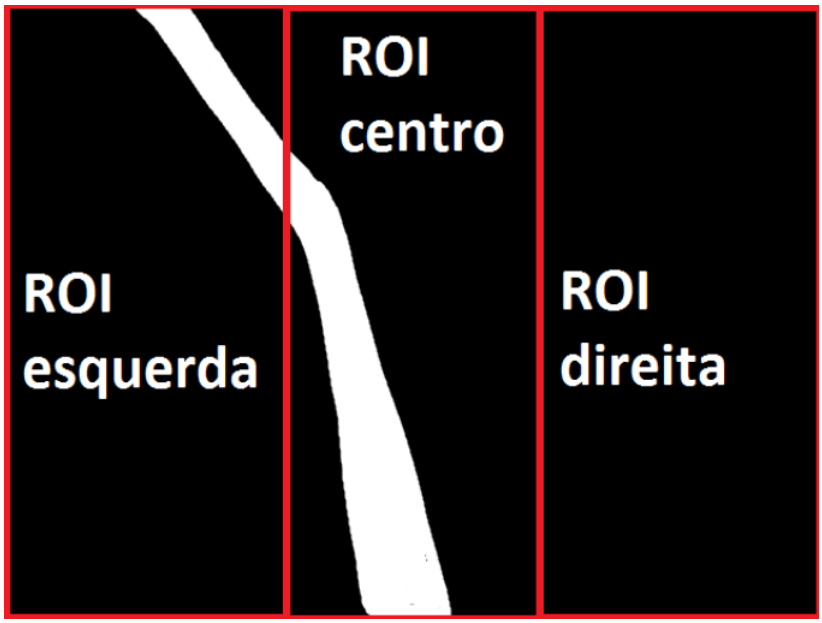

Figura 11: Representação de como as *ROI* funcionam (Alfred).

3 SOLUÇÃO PROPOSTA

A solução proposta neste trabalho consiste na criação de um protótipo de um sistema de robô garçom capaz de obter um grafo representando os caminhos para as mesas através de visão computacional e transferir os dados relevantes para o robô, que deve encontrar o caminho mais curto da mesa em que se encontra para a mesa de destino. Todo processamento feito por parte de software utilizou a linguagem de programação *Python* [15] e todo o código utilizado pode ser encontrado no repositório do *Github* criado para o trabalho [16]. O diagrama da solução proposta pode ser visualizado na Figura 12.

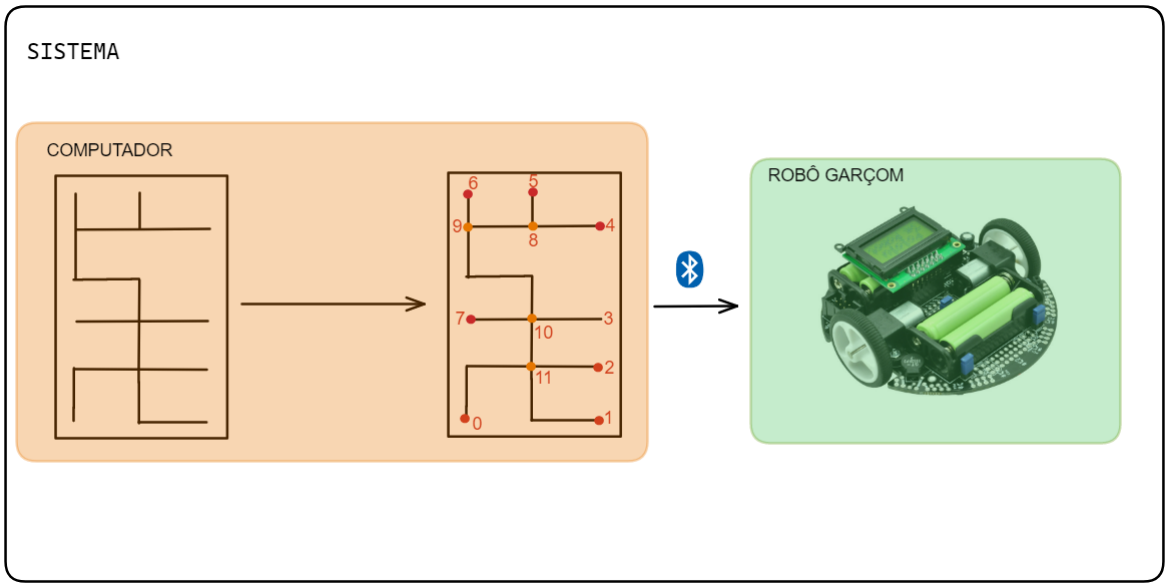

Figura 12: Diagrama do sistema proposto. O computador processa o mapa utilizando visão computacional (esquerda) e envia as informações através de bluetooth para o robô seguidor de linha (direita).

3.1 O robô garçom

Nesta seção serão abordadas as especificações dos componentes de hardware utilizados para desenvolver o projeto proposto nesse trabalho.

Para o desenvolvimento deste projeto, também foi utilizada uma plataforma de robô seguidor de linha, já definida e utilizada para a primeira implementação, o robô Pololu 3pi [2], abordado a seguir.

3.1.1 Pololu 3pi

O Pololu 3pi (ilustrado no diagrama da Figura 15) é uma solução comercial completa para aplicações envolvendo robôs seguidores de linhas. Alimentado por 4 pilhas AAA, os principais componentes com os quais este robô já vem equipado estão listados a seguir, ilustrados também na Figura 13 e na Figura 14.

- Dois motores CC com escovas
- Rodas
- *Buzzer* GT-0950RP3
- Display LCD 8x2 removível
- Microcontrolador Atmel AVR MEGA328
- Cinco sensores de Reflectância

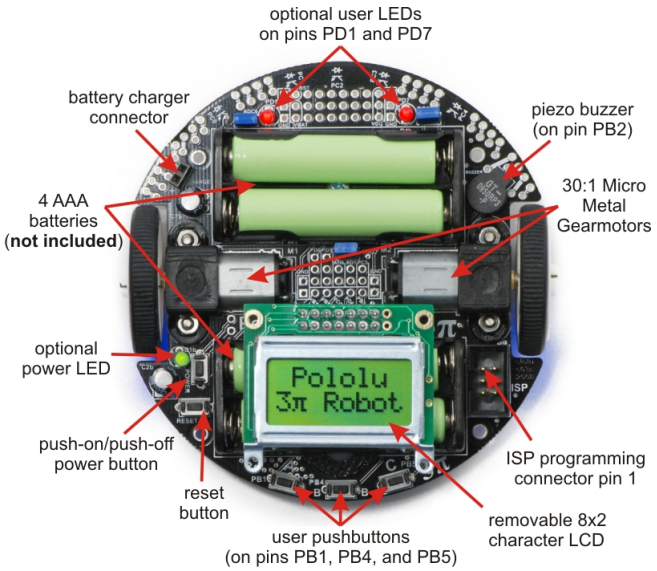

Figura 13: Visão de cima de características gerais do robô Pololu 3pi [2].

Um componente merece destaque: o Microcontrolador. Graças à escolha de utilizá-lo, é possível desenvolver todo o *firmware* (código utilizado no robô) usando a IDE do Arduino (Arduino Sketch), tornando o processo muito mais prático. Ademais, a própria empresa (Pololu) criou módulos de operação e exemplos de códigos das várias funções do robô, facilitando a curva de aprendizado e reduzindo o tempo necessário para se habituar com o mesmo.

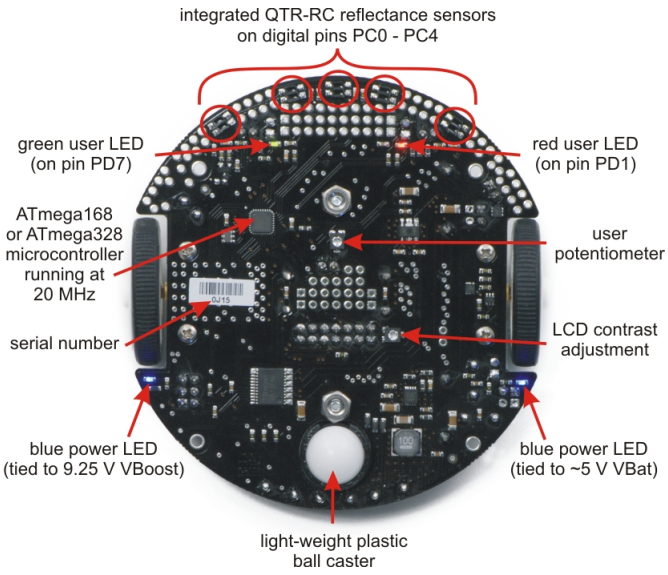

Figura 14: Visão de baixo de características gerais do robô Pololu 3pi [2].

Outro ponto de atenção são os sensores de reflectância, pontos focais para a funcionalidade como seguidor de linha. O posicionamento dos sensores (como é possível observar na Figura 14) permite que interseções entre segmentos sejam encontrados de forma mais fácil.

O robô não possui uma forma de se comunicar com outros equipamentos que não seja pelo seu conector ISP (Figura 13), portanto, para alcançar o objetivo da solução proposta, foi necessário acoplar um módulo *Bluetooth* para estabelecer a conexão entre o Pololu 3pi e o *software*.

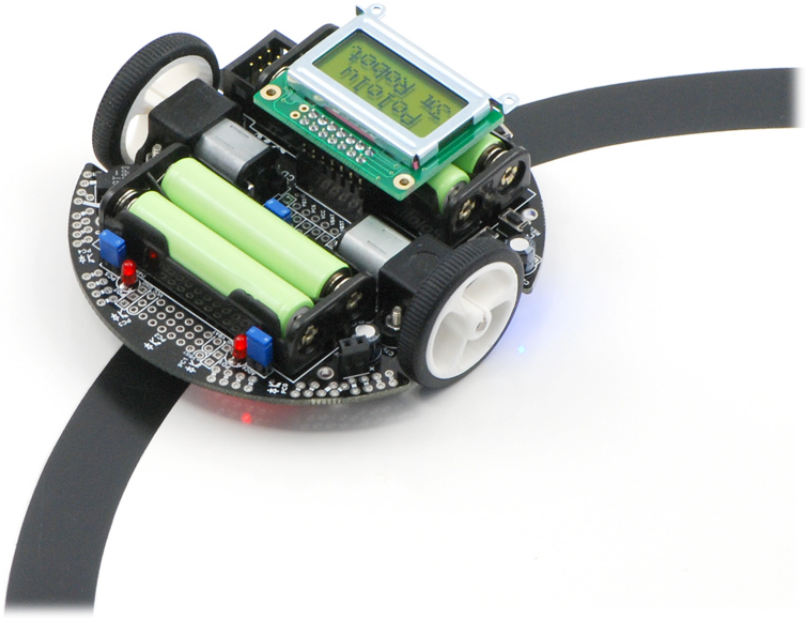

Figura 15: Robô Pololu 3pi em uma linha preta de 3/4" [3].

3.1.2 Módulo *Bluetooth* HC06

Bluetooth é uma tecnologia sem fio de curto alcance usada em diversas aplicações, inclusive por apresentar diversas vantagens quando comparada a outras tecnologias sem fio, como baixo consumo de energia, fácil obtenção, módulos pequenos (facilitando utilizar em qualquer aplicação), e baixo custo [17].

O módulo utilizado neste projeto foi o HC06, devido ao seu tamanho, peso, custo, e capacidade de atender a demanda necessária para transmissão de dados. Algumas características [18] deste módulo são:

- Compatibilidade com Arduino, Raspberry PI, ARM, AVR, PIC, etc.
- Operação na frequência 2,4 GHz
- Antena embutida
- Dimensões: 38 x 15,7 x 3,5 mm
- Pesa 3 g
- Alcance do sinal de até 10 m
- Tensão de alimentação de 5 V
- Tensão do sinal de transmissão de 3,3 V

O módulo foi conectado ao robô garçom, usando pinos VCC e GND, o pino de transmissão (TXD) do módulo foi conectado ao pino PD0 [3] [19] do Pololu 3pi, enquanto o pino de recepção do módulo (RXD) foi conectado ao pino PD1 do robô após passar por um divisor de tensão, devido ao fato que PD0 e PD1 são pinos digitais de Entrada/Saída (E/S ou *I/O*) e operam com sinais de até 5 V, enquanto o módulo só opera até 3,3 V em seus pinos de transmissão, como mencionado anteriormente.

Figura 16: Módulo Bluetooth HC06.

3.1.3 Pilhas de Ni-MH

As pilhas são um componente igualmente essencial do sistema pelo fato de serem a fonte de energia do robô, permitindo o acionamento dos motores, microcontrolador e módulo *Bluetooth*. Elas funcionam a partir de uma reação química que move elétrons do seu polo negativo (Ânodo) para o polo positivo (Cátodo) através de um circuito externo conectado a elas.

Ao longo do desenvolvimento do trabalho, foi constatada a necessidade de pilhas de Ni-MH devido ao uso intenso do robô, seguindo orientação da documentação da própria fabricante do robô [3], principalmente por serem recarregáveis. Portanto, as pilhas alcalinas antes utilizadas foram substituídas pelas pilhas recarregáveis, para evitar um consumo elevado de pilhas não recarregáveis.

3.1.4 Estimativa de Orçamento

Tendo todos os componentes definidos, é possível estimar o custo da implementação. Vale comentar que este é apenas um caminho tomado, tendo múltiplas formas de implementar uma solução que alcance os resultados desejados.

O seguidor de linha utilizado no projeto foi descontinuado, portanto, foi listado em seu lugar uma versão equivalente da mesma marca e com mesmas características.

Tabela 1: Preços estimados para componentes descritos na solução

Item	Preço (US$)	Preço (R$)
Pololu 3pi+	159.95 [20]	835,88
Módulo HC06	-	42,66 [18]
Pilhas de Ni-MH	-	25,29 [21]
Total		903,83

3.2 Obtenção de grafo através de visão computacional

A obtenção do grafo é realizada utilizando a biblioteca *OpenCV* [14], já citada anteriormente. Esta biblioteca é portada para diversas linguagens de programação, contendo várias técnicas de processamento de imagem, como a linguagem de programação escolhida para o projeto foi *Python*, foi utilizada a biblioteca desta linguagem.

O processo pode ser ser separado em duas partes:

- Encontrar as linhas que compõem os caminhos a partir da imagem de entrada
- Processar as linhas obtidas e obter os pontos dos grafos

3.2.1 Detecção de linhas

A detecção de linhas parte de uma imagem da superfície com os caminhos (como na Figura 17). É então aplicado um limiar binário na imagem em escala de cinza, fazendo com que a imagem só possa ter dois valores possíveis, dependendo do valor original e do valor do limiar definido com o intuito de remover informações desnecessárias da imagem e simplificar o processo. Na Figura 18 já fica muito mais fácil de identificar os caminhos, porém ainda existem elementos indesejados.

Da imagem binária, é utilizada a função findContours [22], que retornará o contorno de todos os elementos contínuos (conectados) em branco na Figura 18.

No entanto, o formato retornado para contornos não é tão interessante para a análise a ser realizada, criando retas duplicadas paralelas e retornando um formato de dado que dificultaria a manipulação. Por isso, é retornada uma imagem com somente o maior contorno, que será o contorno dos caminhos. Este contorno é expandido, de forma que as linhas paralelas se tornem uma só linha, mais grossa. Com isso, a imagem dos contornos terá uma função como uma máscara a ser utilizada mais adiante.

Também é aplicada a Transformada de Hough Probabilítica [23] na imagem binária,

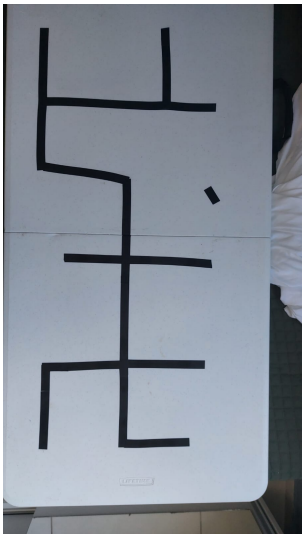

Figura 17: Foto de entrada com os caminhos representados pela fita preta, antes de qualquer tratamento.

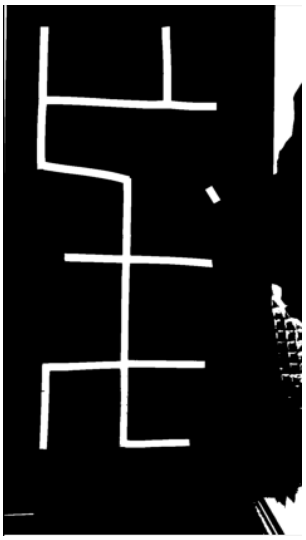

Figura 18: Imagem binária após limiar aplicado.

que encontra segmentos de reta, representados como (x_i, y_i, x_f, y_f), as coordenadas x e y do ponto inicial e as coordenadas x e y do ponto final do segmento, respectivamente.

Para poder construir o grafo, o formato retornado de uma lista de segmentos é mais interessante, porém, como é possível observar na Figura 21, existem muitas linhas em locais errados. Por isso é aplicado um filtro lógico *AND*, que terá o papel de unir os resultados obtidos pelos Contornos e pela Transformada.

Finalmente, a Transformada de Hough Probabilística é utilizada novamente na imagem, porém agora resultando em menos que 100 linhas e todas representando algum trecho dos caminhos.

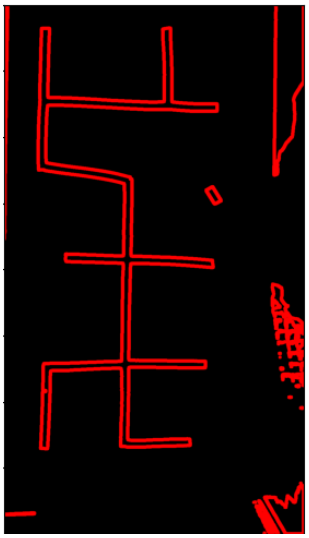

Figura 19: Imagem com os contornos encontrados em vermelho.

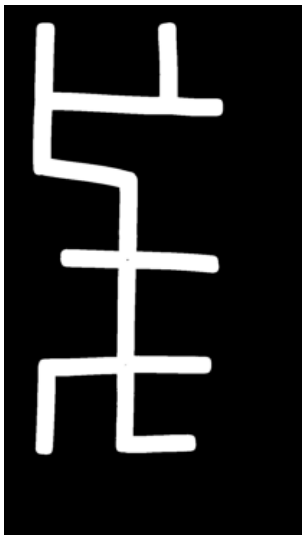

Figura 20: Imagem com o maior contorno em branco (para facilitar operação que terá em seguida).

3.2.2 Processamento das linhas obtidas

Após todo o processamento, ainda existem muito mais linhas do que o desejado. Por isso, é utilizado um algoritmo que identifica linhas semelhantes e as combina [24], baseado no ângulo e distância entre elas. Após a aplicação desta função, são obtidas uma linha para cada segmento do caminho, por exemplo, nove linhas para a imagem da Figura 17.

O próximo passo é encontrar as interseções das retas obtidas, utilizando o algoritmo da Figura 23, que além de retornar os pontos de interseção, também converte todas as retas em objetos *Line*, da biblioteca *Sympy* [25], que será útil para próximos passos da

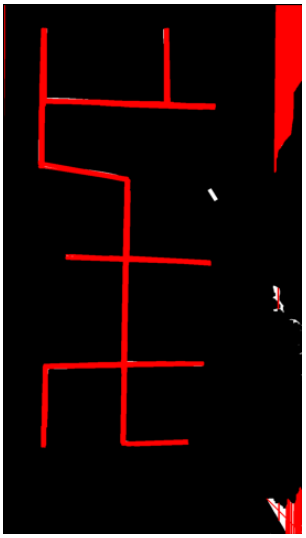

Figura 21: Resultado da Transformada de Hough Probabilítica, com mais de 200 linhas encontradas.

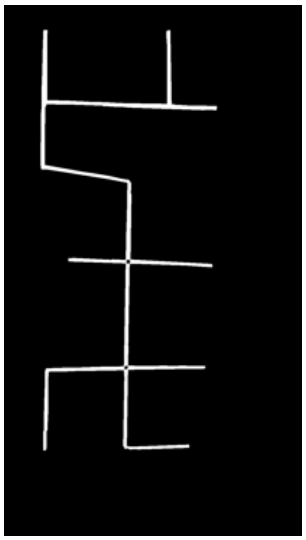

Figura 22: União das imagens na Figura 20 e na Figura 21.

implementação.

Tendo os pontos de interseção, é possível descobrir todos os pontos de destino, ou mesas, já que basta conferir se uma reta contém duas interseções ou não, pois um ponto final nunca será interseção. Portanto, caso uma reta só contenha um ponto de interseção ou outro ponto será um destino. Ao mesmo tempo que os pontos são encontrados, também é armazenado em que direção o ponto que o conecta está. O grafo funciona utilizando pontos cardeais como referência se algo está acima, abaixo, à esquerda ou à direita (Norte, Sul, Oeste, Leste, respectivamente).

Todos os pontos restantes também são conectados, estabelecendo uma coordenada entre pontos conectados por uma reta. E após isso, observa-se a quantidade de pontos

```python
def find_intersections(grouped_lines):
    intersections = {}
    for i in range(len(grouped_lines)):
        for j in range(len(grouped_lines)):
            if i == j:
                continue
            if type(grouped_lines[i]) == np.ndarray:
                _l = Line(grouped_lines[i][0][:2], grouped_lines[i][0][2:])
                grouped_lines[i] = _l
            if type(grouped_lines[j]) == np.ndarray:
                grouped_lines[j] = Line(grouped_lines[j][0][:2], grouped_lines[j][0][2:])

            l_i, l_j = grouped_lines[i], grouped_lines[j]
            if degrees(l_i.angle_between(l_j)) > 50:
                _inter = l_i.intersection(l_j)
                if len(_inter) == 0:
                    continue
                _inter = _inter[0]
                if _inter.distance(Segment(*l_i.points)) < 10 and _inter.distance(Segment(*l_j.points)) < 10:
                    if {i, j} not in [set(k) for k in intersections.keys()]:
                        intersections[(i, j)] = _inter
                        continue
    return intersections
```

Figura 23: Algoritmo que retorna todos os pontos de interseção entre as retas fornecidas na entrada.

conectados a um nó para separar aqueles que foram chamados de nós auxiliares de pontos auxiliares. A diferença entre eles é que um nó auxiliar é conectado a mais do que dois outros nós, sendo então relevante para o grafo, pois será um fator de decisão. Já um ponto auxiliar é nada mais do que um ponto que conecta duas retas sem outras bifurcações, não sendo importante para decisões de roteamento.

A Figura 25 demonstra todos os pontos encontrados a partir da imagem, ou seja, o resultado final do grafo. Os nós 0 a 7 são nós de destino (mesas), os nós 8 a 11 são nós auxiliares (conectados a mais do que dois outros pontos) e os nós 12 a 15 são pontos auxiliares (conectados a dois outros pontos).

```python
def build_destination_nodes(self, temporary_nodes):
    def find_closest_point(ref_point, ref_line):...

    for line in self._lines:
        possible_points = list(line.copy().points)
        for idx, inter_point in self._intersections.items():
            if self._lines.index(line) not in idx:
                continue
            d = np.array([inter_point.distance(_p) for _p in possible_points]) < 30
            node = np.where(d == True)[0]
            if len(node) == 0:
                continue
            del possible_points[node[0]]
            if len(possible_points) == 0:
                break
        if len(possible_points) > 0:
            for point in possible_points:
                _n = Node(point)
                closest = find_closest_point(point, line)
                c2p = _n.coordinate_to_point(closest)
                closest = self.find_node_by_coordinate(closest, temporary_nodes)
                setattr(_n, c2p, closest)
                setattr(closest, invert_coordinates(c2p), _n)
                self.destination_nodes.append(_n)
```

Figura 24: Algoritmo que encontra os pontos de destino a partir do conjunto de retas e interseções já obtidas.

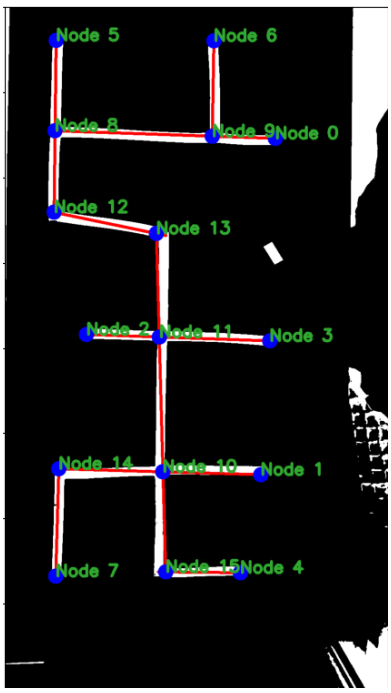

Figura 25: Todos os pontos encontrados a partir da imagem inicial.

Os números atribuídos para os nós seguem apenas uma regra: os nós de destino recebem de 0 a $n-1$, onde $n-1$ é a quantidade de nós de destino. Após isso os nós auxiliares recebem números n a $n+m-1$ com m representando a quantidade de nós auxiliares. Por último, os pontos auxiliares recebem número de $n+m$ a x para x igual à quantidade total de pontos encontrados.

Por último, para a transferência dos dados para o robô garçom, o grafo é transformado em uma matriz de adjacências [26] que contém a relação entre os nós em pontos cardeais, o formato utilizado pelo robô.

3.3 Transferência dos dados para o robô

Esta é a última etapa por parte do *software*. É estabelecida uma conexão serial através de *Bluetooth* utilizando a biblioteca *pySerial* [27], como ilustrado na Figura 26.

```
ser = Serial('COM3', 9600)
ser.timeout = 120
if not ser.is_open:
    ser.open()
```

Figura 26: Trecho do código de transmissão *Bluetooth* em que é estabelecida a conexão serial.

Após estabelecida a conexão, as informações da matriz de adjacência são transferidas uma por uma, em forma de relação entre nós. A transmissão é feita desta forma pois o robô garçom foi programado para receber todas as relações entre os nós em uma matriz de adjacência interna, com o intuito de também poder operar de forma independente do *software*. O tempo de transmissão das coordenadas varia de acordo com a quantidade de conexões a serem registradas. Para o caso de teste da Figura 17, vinte e duas relações entre nós foram cadastradas em 34 segundos. A Figura 27 demonstra uma parte do fluxo de cadastro, onde cada relação é informada, por exemplo, "0 9 W"indica que o caminho do nó 0 ao nó 9 é W, *West* (Oeste), e de forma automática é cadastrado que o caminho oposto, de 9 a 0 é o oposto a W, E (Leste). Após isso, é questionado se desejado inserir outra relação, então enquanto o software identificar que existem relações a serem cadastradas, ele irá mandar para o robô "y"de *Yes* (Sim) e cadastrará a próxima.

```
b'Enter node relation (node1 node2 direction12):\r\n'
0 9 W
b'0 9 W\r\n'
b'Add another node relation? (y/n)\r\n'
y
b'Enter node relation (node1 node2 direction12):\r\n'
1 10 W
b'1 10 W\r\n'
b'Add another node relation? (y/n)\r\n'
y
b'Enter node relation (node1 node2 direction12):\r\n'
2 11 E
b'2 11 E\r\n'
b'Add another node relation? (y/n)\r\n'
y
b'Enter node relation (node1 node2 direction12):\r\n'
```

Figura 27: Parte das atividades registradas no terminal da *IDE* (*Integrated Development Environment* ou Ambiente de desenvolvimento integrado) onde as respostas são enviadas automaticamente pelo programa.

```
b'Add another node relation? (y/n)\r\n'
n
b'Connections list:\r\n'
b'0-9\r\n'
b'W\r\n'
b'1-10\r\n'
b'W\r\n'
b'2-11\r\n'
b'E\r\n'
b'3-11\r\n'
b'W\r\n'
b'4-10\r\n'
b'WN\r\n'
b'5-8\r\n'
b'S\r\n'
b'6-9\r\n'
b'S\r\n'
b'7-10\r\n'
b'NE\r\n'
b'8-5\r\n'
b'N\r\n'
b'8-9\r\n'
b'E\r\n'
b'8-11\r\n'
b'SES\r\n'
b'9-0\r\n'
```

Figura 28: Parte das atividades registradas no terminal da *IDE* quando o Pololu 3pi retorna as conexões estabelecidas.

Após cadastrar todas as relações, o robô retorna a lista de todas as conexões feitas, como na Figura 28. São indicados os nós em uma linha e abaixo dela o caminho para chegar do primeiro ao segundo.

Por último, é informada a posição do robô (mesa em que se encontra) e então o código entra em um *loop* em que irá fazer a requisição do ponto de destino desejado e enviar para o robô, esperando ele chegar e então perguntando novamente. Caso deseje finalizar o programa, basta o usuário digitar "q", de *Quit* (Sair), que o programa sairá do *loop* e finalizará.

3.4 Busca do menor caminho

A busca do menor caminho é feita utilizando um algoritmo modificado de BFS [28] (*Breadth-first search*, Busca em largura [29]).

Como é possível acompanhar pela Figura 29, dada uma origem e um destino (ambos representados como inteiros), é possível encontrar o menor caminho entre eles. A estrutura mais importante para isto é o vetor de inteiros *pred*, que armazena o ponto predecessor de um ponto i informado. Ele é preenchido quando a função BFS é chamada.

A estrutura mais importante da função BFS se encontra na Figura 30, é nela que é mapeado o caminho da origem para o destino e preenchido o vetor *pred*. São criadas duas estruturas: uma fila (uma estrutura *FIFO* [30]) que começa somente com o valor da origem e um vetor que registra a distância de um ponto i para a origem. Enquanto houver elementos na fila, o primeiro será retirado e ocorrerá a verificação: É iterado sobre todos os nós se existe um valor na matriz de adjacência entre ele e o valor retirado da fila, caso exista, a distância deste ponto à origem é atualizada como a do ponto anterior (ao qual ele está conectado) + 1, e então este ponto é adicionado à fila. Este processo se repete até que o ponto i, que tem conexão com o caminho, seja igual ao destino.

Com isto, é possível montar um vetor como o da Figura 29.

```
/**
    Finds shortest path from source to destination, in relation to nodes
*/
Vector<int> create_path_coordinates(int src, int dest)
{
  // Predecessor array stores predecessor of i and distance array stores
    // distance of i from s
  int pred[n_vertices];
  int dist[n_vertices];
  int a[n_vertices];
  Vector<int> path(a);

  // Source and destination are not connected
  if (BFS(src, dest, pred, dist) == false)
    return path;

  // Vector path stores the shortest path
  int crawl = dest;
  path.push_back(crawl);
  while (pred[crawl] != -1) {
    path.push_back(pred[crawl]);
    crawl = pred[crawl];
  }

  // Shortest path length is dist[dest];

  return path;
}
```

Figura 29: Algoritmo que retorna o menor caminho de um nó (*src*) a outro (*dest*).

```
// standard BFS algorithm
while (!queue.empty()) {
  int u = queue[0];
  queue.remove(0);
  for (int i = 0; i < n_vertices; i++) {
    if (visited[i] == false && adj[u][i][0]) {
      visited[i] = true;
      dist[i] = dist[u] + 1;
      pred[i] = u;
      queue.push_back(i);

      // BFS stops when destination is found.
      if (i == dest)
        return true;
    }
  }
}
```

Figura 30: Algoritmo que executa a Busca em Largura.

4 TESTES

Devido a restrições de tempo, os testes listados a seguir foram realizados utilizando os caminhos obtidos a partir da Figura 17:

- Teste da criação do grafo a partir de imagem: O algoritmo de criação de grafo foi executado várias vezes (e com diferentes parâmetros de ângulo mínimo e distância mínima entre retas a serem combinadas) para garantir que não havia instabilidade no resultado obtido;

- Teste da transmissão do grafo para o robô: Em sequência ao teste da criação do grafo, era testado se a conexão serial tinha sido estabelecida com sucesso e se todas as informações relevantes eram repassadas;

- Teste de impressão do caminho entre os nós: a matriz de adjacência foi cadastrada e foi utilizada a função que retorna o menor caminho para todos os nós presentes na matriz e observado se o resultado retornado estava de acordo com o esperado;

- Teste de navegação do robô entre os nós: Semelhante ao teste de retorno, o robô foi ordenado a ir para todos os pontos possíveis e foi verificado se o mesmo chegou ao ponto de forma correta (vários projetos de robôs seguidores de linha têm dificuldade em alguns pontos que o robô não navega de forma correta, se perdendo da linha ou rotacionando no próprio eixo);

- Teste de ajuste fino na movimentação do robô: Observado ao longo dos testes de navegação, este teste tinha o intuito de verificar se o robô corrigia sua posição mesmo sob leves mudanças na direção.

5 DIFICULDADES ENCONTRADAS

As principais dificuldades encontradas ao longo do desenvolvimento deste trabalho foram:

- Informações e aplicações utilizando a plataforma Pololu 3pi são muito mais restritas que outros microcontroladores comuns em projetos como este (como Arduino ou *Raspberri Pi*), dificultando casos de dúvida relacionados à operação ou estrutura do robô. Devido ao microcontrolador utilizado, algumas dúvidas relacionadas ao Pololu 3pi puderam ser tiradas traçando paralelos com *Arduino*.

- Acesso a componentes de *hardware*. Como este trabalho foi conduzido à distância, a quantidade de ferramentas para algumas necessidades foi restrita. Por exemplo, a solda dos cabos utilizados para a conexão do módulo *Bluetooth* com o robô, ou medir a corrente passando na conexão. Esta dificuldade pôde ser resolvida utilizando equipamentos disponíveis nos laboratórios do departamento de Engenharia Eletrônica.

- Durante a implementação do algoritmo de busca do menor caminho, para facilitar e agilizar testes, o algoritmo foi desenvolvido na linguagem de programação *C++*, por causa de sua semelhança com *Arduino*. No entanto, após os testes serem concluídos, foi gasto muito tempo adaptando o código para *Arduino*, sendo mais interessante ter montado uma estrutura de testes com a linguagem final desde o início.

6 CONCLUSÃO E TRABALHOS FUTUROS

Como descrito no Capítulo 3, a solução proposta consistia na criação de um protótipo de robô garçom que receberia as informações de navegação a partir de um *software* utilizando visão computacional e encontraria o caminho mais curto do ponto em que se encontrava para o ponto de destino.

Pelos resultados obtidos, é possível concluir que o objetivo deste trabalho foi atingido, já que o sistema desenvolvido realizou todas as atividades propostas, sem apresentar falhas notáveis em seus comportamentos. Foi gravado um vídeo de demonstração do funcionamento do robô em [31].

Vale mencionar que, como abordado no capítulo 4, não foi possível testar exaustivamente o sistema, não sendo possível garantir a estabilidade do mesmo para qualquer caso.

Além disso, outras atividades podem ser mapeadas como trabalhos futuros, entre elas:

- Testar projeto com mais casos/configurações: Configurações diferentes de caminhos podem acarretar comportamentos inusitados. Durante os testes de *Bluetooth*, houve ocasiões pontuais em que a conexão não foi estabelecida com sucesso. Um caso de testes muito importante é em um ambiente "real", porém controlado;

- Otimizar códigos em Python e Arduino: Houve algumas redundâncias ao longo dos códigos e principalmente pelo lado do *firmware*, seria interessante otimizar o uso de memória, pois para a solução atual há um número máximo de nós que podem existir;

- A movimentação que o robô faz enquanto calibra seus sensores pode deixá-lo um pouco deslocado da linha em algumas situações;

- Adição do sensor ultrassônico (como o modelo HC-SR04 [32]) para evitar colisões;

- Adicionar suporte físico para o módulo *Bluetooth*;

- Construção de suporte para fixar objetos sobre o robô ou troca da plataforma para uma mais robusta com maior capacidade de transporte de carga.

REFERÊNCIAS

[1] MACHADO, D. P.; MOURA, E. R. V. D. *DESENVOLVIMENTO DE UM ROBÔ SEGUIDOR DE LINHA ALIMENTADO POR FONTE DE ENERGIA ALTERNATIVA*. [S.l.], 2019.

[2] POLOLU. *Pololu 3pi Robot*. Available at: <https://www.pololu.com/product/975>. Accessed in: 08/01/2023.

[3] POLOLU. *Pololu 3pi Robot User's Guide*. Available at: <https://www.pololu.com/docs/pdf/0J21/3pi.pdf>. Accessed in: 08/01/2023.

[4] SAIPOS. *Sistema de Automação Comercial para restaurantes: confira os benefícios dessa tecnologia*. Available at: <https://saipos.com/gestao-de-restaurante/sistema-de-automacao-comercial-para-restaurantes>. Accessed in: 07/01/2023.

[5] MINATOGAWA, H.; TAKAHASHI, J. *ESTEIRA COM SUSHI*. Available at: <https://madeinjapan.com.br/2015/09/04/esteira-com-sushi/#:~:text=A%20ideia%20foi%20do%20sushiman,kaitenzushi%20em%201958%2C%20em%20Osaka>. Accessed in: 07/01/2023.

[6] HOME, A. R. *AGV Applications - Where are Automated Guided Vehicles Used?* Available at: <https://www.agvnetwork.com/agv-applications>. Accessed in: 07/01/2023.

[7] RECORDS, G. W. *First restaurant with robot waiting staff*. Available at: <https://www.guinnessworldrecords.com/world-records/first-restaurant-with-robot-waiting-staff#:~:text=Owned%20by%20Shayne%20Hayashi%2C%20the,Chinese%20food%20to%20its%20customers>. Accessed in: 07/01/2023.

[8] UOL, N. *Restaurante na Flórida recorre a robôs por falta de garçons na pandemia*. Available at: <https://www.uol.com.br/nossa/noticias/redacao/2021/10/22/restaurante-na-florida-recorre-a-robos-por-falta-de-garcons-na-pandemia.htm>. Accessed in: 07/01/2023.

[9] DIGITAL, O. *Restaurante na Holanda usa robô-garçom*. Available at: <https://olhardigital.com.br/2020/06/02/videos/restaurante-na-holanda-usa-robo-garcom/>. Accessed in: 07/01/2023.

[10] FINANçAS, Y. *Restaurantes na Malásia empregam robôs como garçons durante a pandemia*. Available at: <https://esportes.yahoo.com/noticias/restaurantes-na-malasia-empregam-robos-como-garcons-durante-a-pandemia-111654089.html>. Accessed in: 07/01/2023.

[11] HOUSE, B. of. *HOW DO ROBOTIC WAITERS WORK AND ARE THEY RIGHT FOR YOUR RESTAURANT?* Available at: <https://backofhouse.io/stories/how-do-robotic-waiters-work-and-are-they-right-for-your-restaurant>. Accessed in: 07/01/2023.

[12] TELES, B. *Apresentação v0 Robô Garçom*. Available at: <https://youtu.be/CMcLYMOm5gI>. Accessed in: 08/01/2023.

[13] FIEL, E. G. et al. *Robçom, o Robô Garçom*. [S.l.], 2018.

[14] OPENCV. *About*. Available at: <https://opencv.org/about/>. Accessed in: 08/01/2023.

[15] PYTHON. *About*. Available at: <https://www.python.org/about/>. Accessed in: 08/01/2023.

[16] TELES, B. *Projeto-Micro-PDI*. Available at: <https://github.com/brunoctt/Projeto-Micro-PDI>. Accessed in: 08/01/2023.

[17] WIKIPEDIA. *Bluetooth*. Available at: <https://en.wikipedia.org/wiki/Bluetooth>. Accessed in: 08/01/2023.

[18] USINAINFO. *Módulo Bluetooth HC-06 Arduino - Slave*. Available at: <https://www.usinainfo.com.br/modulo-bluetooth-arduino/modulo-bluetooth-hc-06-arduino-slave-2826.html>. Accessed in: 08/01/2023.

[19] POLOLU. *9. Pin Assignment Tables*. Available at: <https://www.pololu.com/docs/0J21/9>. Accessed in: 08/01/2023.

[20] POLOLU. *3pi+ 32U4 Robot - Standard Edition (30:1 MP Motors), Assembled*. Available at: <https://www.pololu.com/product/3737>. Accessed in: 08/01/2023.

[21] AMAZON. *Pilha Recarregável AAA Blister Com 4 Pilhas, Elgin, Baterias, 1000mAh*. Available at: <https://www.amazon.com.br/Recarreg%C3%A1vel-AAA-1000Mah-Blister-Elgin-Baterias/dp/B0754KMGFN/ref=asc_df_B0754KMGFN/?tag=googleshopp00-20&linkCode=df0&hvadid=379698982510&hvpos=&hvnetw=g&hvrand=5250493011744389590&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=1001625&hvtargid=pla-593310443411&th=1>. Accessed in: 08/01/2023.

[22] OPENCV. *Contours : Getting Started*. Available at: <https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html>. Accessed in: 08/01/2023.

[23] OPENCV. *HoughLinesP()*. Available at: <https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga8618180a5948286384e3b7ca02f6feeb>. Accessed in: 08/01/2023.

[24] RACKWITZ, C. *How to merge lines after HoughLinesP?* Available at: <https://stackoverflow.com/a/70318827>. Accessed in: 08/01/2023.

[25] SYMPY. *Welcome to SymPy's documentation!* Available at: <https://docs.sympy.org/latest/index.html>. Accessed in: 08/01/2023.

[26] WIKIPEDIA. *Matriz de adjacência*. Available at: <https://pt.wikipedia.org/wiki/Matriz_de_adjac%C3%AAncial>. Accessed in: 08/01/2023.

[27] PYSERIAL. *pySerial*. Available at: <https://pyserial.readthedocs.io/en/latest/pyserial.html#>. Accessed in: 08/01/2023.

[28] GEEKS, G. for. *Shortest path in an unweighted graph*. Available at: <https://www.geeksforgeeks.org/shortest-path-unweighted-graph/>. Accessed in: 08/01/2023.

[29] WIKIPEDIA. *Busca em largura*. Available at: <https://pt.wikipedia.org/wiki/Busca_em_largura>. Accessed in: 08/01/2023.

[30] WIKIPEDIA. *FIFO*. Available at: <https://pt.wikipedia.org/wiki/FIFO>. Accessed in: 08/01/2023.

[31] TELES, B. *Apresentação Robô Garçom TCC*. Available at: <https://youtu.be/cEYOQbmWXi8>. Accessed in: 08/01/2023.

[32] FILIPEFLOP. *Sensor de Distância Ultrassônico HC-SR04*. Available at: <https://www.filipeflop.com/produto/sensor-de-distancia-ultrassonico-hc-sr04/>. Accessed in: 08/01/2023.