INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO AMAZONAS
CAMPUS MANAUS DISTRITO INDUSTRIAL
ENGENHARIA DE CONTROLE E AUTOMAÇÃO

GIOVANE TAVEIRA DE SOUZA XAVIER

DESENVOLVIMENTO DE UM SISTEMA INTELIGENTE OCR UTILIZANDO VISÃO COMPUTACIONAL PARA LEITURA DE ETIQUETAS DE ROTEADOR

MANAUS - AM

2022

GIOVANE TAVEIRA DE SOUZA XAVIER

DESENVOLVIMENTO DE UM SISTEMA INTELIGENTE OCR UTILIZANDO VISÃO COMPUTACIONAL PARA LEITURA DE ETIQUETAS DE ROTEADOR

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Engenharia de Controle e Automação, do Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, Campus Manaus Distrito Industrial – IFAM/CMDI.

Orientador: Prof. Dr. Alyson de Jesus dos Santos

MANAUS - AM

2022

Dados Internacionais de Catalogação na Publicação (CIP)

X3d Xavier, Giovane Teixeira de Souza.

Desenvolvimento de um sistema inteligente OCR utilizando visão computacional para leitora de etiquetas de roteador. / Giovane Teixeira de Souza Xavier. – Manaus, 2022.
73f. : il. color

TCC (Graduação em engenharia de controle e automação) – Instituto Federal de Educação, Ciência e Tecnologia do Amazonas, *Campus* Manaus Distrito Industrial, 2022.
Orientador: Prof. Dr. Alyson de Jesus dos Santos

1. Visão computacional. 2. Open CV. 3. OCR. I. Santos, Alyson de Jesus (orient.) II. Instituto Federal de Educação, Ciência e Tecnologia do Amazonas. III. Título.

CDD 629.8

Elabora por Fc[a]. Amélia Frota, registro n.858 (CRB11)

ANEXO 7

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

Aos 29 dias do mês de Setembro, de 2022, às 15h, o(a) discente Giovane Taveira de Souza Xavier apresentou o seu Trabalho de Conclusão de Curso para avaliação da Banca Examinadora constituída pelos seguintes integrantes: Prof(a). Dr. Alyson de Jesus dos Santos (docente-orientador), Prof(a). Msc. Gabriel Pinheiro Compto (Membro 1) e Prof(a). Jaidson Brandão da Costa (Membro 2). A sessão publica de defesa foi aberta pelo(a) presidente da banca, que apresentou a Banca Examinadora e deu continuidade aos trabalhos, fazendo uma breve referência ao TCC, que tem como título projeto e desenvolvimento de uma câmera inteligente de baixo custo utilizando visão computacional ocr para leitura de etiquetas de roteador. Na sequencia, o(a) discente teve até 30 minutos para a comunicação oral de seu trabalho. Cada integrante da banca examinadora fez suas arguições após a defesa do mesmo. Ouvidas as explicações do(a) discente, a banca examinadora, reunida em caráter sigiloso, para proceder à avaliação final, deliberou e decidiu pela APROVAÇÃO com média final 8,0 (OITO) do referido trabalho.

Foi dada ciência ao(à) discente que a versão final do trabalho deverá ser entregue até o dia 30/10/2022, com as devidas alterações sugeridas pela banca. Nada mais havendo a tratar, a sessão foi encerrada às 16 h 20 min, sendo lavrada a presente ata, que, uma vez aprovada, foi assinada por todos os membros da Banca Examinadora e pelo(a) discente.

Prof.(a) Orientador(a)/Presidente: Alyson de Jesus dos Santos

Prof.(a) Avaliador 1: Gabriel Pinheiro Costa

Prof.(a) Avaliador 2: Jaidson Brandão da Costa

Discente: Giovane Taveira de Souza Xavier

AGRADECIMENTOS

Palavras não podem descrever o sentimento de gratidão perante a conclusão de mais uma jornada. Agradeço a minha família por todo o apoio e suporte em busca de meus sonhos e objetivos, sem eles nada seria possível, encontro em cada um, uma forma de inspiração.

Agradeço a Deus, pelo dom da vida, pela paciência e força me dada perante os momentos de aflição e angústia, a Ele entrego todos meus sonhos e propósitos, Ele me guia para todo o sempre.

Aos meus amigos de curso que somaram muito na minha formação, amigos esses que viraram irmãos.

Agradeço ao meu orientador, Prof. Dr. Alyson de Jesus dos Santos, pela sabedoria e paciência com que me orientou no pós-pandemia, sendo bastante acessível para finalizar a jornada, e por sempre estar disposto a me ajudar.

A Secretaria do Curso, pela cooperação.

Enfim, a todos os que de alguma forma contribuíram para a realização desta pesquisa e fim de mais esse ciclo.

‘‘Não temas, crê somente”.
(Marcos 5:36)

RESUMO

Este trabalho de conclusão de curso consiste no desenvolvimento de um sistema inteligente OCR utilizando visão computacional para leitura de etiquetas de roteadores, visando extrair as informações para inserir em programas que realizam os testes de roteadores. O sistema utiliza uma Raspberry Pi 4 como central de processamento do sistema, acoplada a ela está uma *webcam Logitech* C270, responsável pela captura das imagens das etiquetas. O sistema de visão utiliza do OpenCV para o tratamento da imagem com a finalidade de que o algoritmo OCR extraia da imagem o texto, após o programa armazena e mostra essas informações em telas simulando os *softwares* de teste.

Palavras-chave: Visão Computacional; OpenCV; OCR

ABSTRACT

This final paper consists of the development of an intelligent OCR system using computer vision to read router labels, to extract information to be inserted into programs that perform router tests. The system uses a Raspberry Pi 4 as the system's processing center, coupled to it is a Logitech C270 webcam, responsible for capturing the images of the labels. The vision system uses OpenCV to process the image so that the OCR algorithm extracts the text from the image, after which the program stores and displays this information on screens simulating the test software.

Keywords: Computer Vision. OpenCV. OCR.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O código de barras é considerado como uma das inovações mais importantes do século XX. Essa tecnologia permite que informações com maior complexidade sejam transferidas em uma alocação mínima de espaço, consequentemente, possuindo uma maior capacidade de respostas às necessidades das pessoas. Com sua segurança e baixo custo operacional, o código de barras é utilizado durante toda a cadeia de suprimento, desde o controle de estoque da fábrica, até a leitura de uma senha de roteador com o usuário final.

Construir algoritmos para que computadores possam enxergar o ambiente em que ele está inserido e extrair informações de objetos e locais, a princípio pode parecer tarefas complexas que necessitam de profundos conhecimentos de programação e de intensos conhecimentos matemáticos. Porém os sistemas de visão computacional estão cada vez mais acessíveis para todos os níveis de conhecimento, podendo ser usado em aplicações, construídos desde entusiastas até as gigantes da indústria.

Segundo os Autores Backes e Junior (2016), definem a visão computacional como a área de estudo que tenta repassar para máquinas a incrível capacidade de visão. A visão consiste em captar imagens, melhorá-las, separar as regiões ou objetos de interesse em cena, extrair várias informações da imagem analisada.

A visão computacional pode ser estudada em dois níveis de abstração: processamento de imagens (baixo nível) e análise de imagens (alto nível). As pesquisas sobre processamento digital de imagens apresentam técnicas de processamento de imagens com o objetivo de manipular informações representadas através de computadores e resultam em suas transformações, como, por exemplo, realçar bordas, remover ruídos ou ainda extrair características. Os estudos sobre reconhecimento de padrões são largamente utilizados para identificar e classificar os objetos representados nas imagens, proporcionando às máquinas a capacidade de reconhecer tais padrões em imagens.

1.1 JUSTIFICATIVA

Durante um período de trabalho em uma fábrica de roteadores foi verificado a ocorrência de diversos testes, sendo necessário a leitura dos códigos de barras para ligar as informações do teste com o número de série no servidor, no final da linha existe um posto que é necessário inserir manualmente todas as informações contidas na etiqueta.

A extração de texto através de imagens de etiquetas de roteadores foi realizada neste trabalho utilizando um processamento de reconhecimento de padrões chamado de Reconhecimento Ótico de caracteres (*Optical Character Recognition,* OCR). O OCR pode ser definido como uma técnica que lida com o problema de reconhecimento de caracteres oticamente processados, ou seja, obtidos por um escaneamento, escrita ou até uma fotografia (EIKVIL, 1993). A análise qualitativa, que consiste em tratar com precisão e legibilidade dos resultados da extração de dados foi feita com base na taxa de acerto dos campos definidos que o OCR conseguiu identificar comparando com os dados reais nas etiquetas.

Este trabalho tem como proposta capturar e analisar etiquetas de roteadores e extrair as informações em texto, necessárias para serem inseridas em programa de realização de testes em roteadores, utilizando técnicas de processamento de imagens e reconhecimento óptico de caracteres (OCR).

Neste contexto, a pergunta que norteou a pesquisa foi: “É possível projetar uma câmera de baixo custo para extrair as informações das etiquetas de roteadores?”

1.2 OBJETIVO

Diante disto, a realização desse Trabalho de Conclusão de Curso (TCC) ao estudar e apresentar uma alternativa metodológica para a leitura e extração das informações das etiquetas de forma automática e inserindo no programa de teste e, por consequência, reduzir o tempo de inserção das informações no teste, tornando se relevante para indústria, pois diminui o tempo necessário para o teste, aumentado a capacidade de produção da linha.

Neste contexto, o objetivo deste trabalho é projetar e desenvolver dispositivo de baixo custo capaz de reconhecer e extrair os caracteres de uma etiqueta de roteador e inserir em programa de teste.

1.3 OBJETIVO ESPECÍFICO

Para obter um resultado satisfatório listamos os seguintes objetivos específicos:

a) Modelar o JIG em ambiente 3D.

b) Construir o JIG para captura da foto

c) Identificar o posicionamento da etiqueta através de visão computacional

d) Realizar o tratamento necessário na imagem da região da etiqueta para que a biblioteca OCR realize o reconhecimento

e) Extrair e armazenar os caracteres da imagem identificada

f) Construir telas de teste para apresentação das informações

g) Apresentar as informações retiradas das etiquetas nas telas de teste.

1.4 METODOLOOGIA

A metodologia geral de pesquisa respeita as características da pesquisa aplicada, entendendo esse modelo como “[...] objetiva gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específicos”. (GERHARDT; SILVEIRA, 2009, p.35).

A essa pesquisa foi utilizada uma abordagem quantitativa, que esclarece Fonseca (2002, p.20):

> Diferentemente da pesquisa qualitativa, os resultados da pesquisa quantitativa podem ser quantificados. Como as amostras geralmente são grandes e consideradas representativas da população, os resultados são tomados como se constituíssem um retrato real de toda a população alvo da pesquisa. A pesquisa quantitativa se centra na objetividade. Influenciada pelo positivismo, considera que a realidade só pode ser compreendida com base na análise de dados brutos, recolhidos com o auxílio de instrumentos padronizados e neutros. A pesquisa quantitativa recorre à linguagem matemática para descrever as causas de um fenômeno, as relações entre variáveis etc. A utilização conjunta da pesquisa qualitativa e quantitativa permite recolher mais informações do que se poderia conseguir isoladamente.

Esse conceito se aplica ao TCC pois as aplicações do nosso trabalho podem ser quantificadas através da análise de tempo economizado utilizando o modo manual de inserção de informações e utilizando o modo automático.

Quanto à natureza da pesquisa aplicada, Gerhard e Silveira (2009, p.35) definem aquela cujo objetivo é o de “gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específicos. Envolve verdades e interesses locais.” O que define a nossa pesquisa, tendo em vista que a implantação do tema, é necessária uma aplicação prática para solucionar o problema proposto.

1.5 ESTRUTURA DO TCC

O restante deste TCC está estruturado da seguinte maneira. No Capítulo 2, é apresentada a fundamentação teórica com o propósito de mostrar uma explicação sobre a visão computacional e as técnicas utilizadas para captura das informações juntamente com a linguagem utilizada e o material. O Capítulo 3, apresenta os métodos e materiais utilizados a fim de mostrar as ferramentas computacionais utilizadas, descrição e implementação do trabalho proposto. No Capítulo 4 é apresentado os resultados e discussões. Por fim, no Capítulo 5 é apresentada a conclusão do trabalho.

2 REFERENCIAL TEÓRICO

Este capítulo tem por objetivo apresentar conceitos teóricos necessários para as discussões práticas deste TCC.

2.1 PYTHON

Segundo Borges (2010, p.10) define Python (Figura 1) como “uma linguagem de altíssimo nível (em inglês, *Very High Level Language*) orientada a objetos, de tipagem dinâmica e forte, interpretada e interativa” e Corrêa (2020, p.12) completa o pensamento afirmando:

> “Python é uma linguagem de programação de propósito geral, o que significa que ela pode ser empregada nos mais diferentes tipos de projetos, variando desde aplicações Web até sistemas de inteligência artificial.”

Figura 1 - Python Logo

Fonte: Python (2022)

O Python foi criado por Guido Van Rossum, em 1990. É uma linguagem interpretada que possui uma sintaxe mais clara e dinâmica comparada com as outras linguagens, tornando uma das linguagens mais produtivas e que abrange um público maior entre os mais experientes e iniciantes no mundo da programação. Sendo construído como um código aberto que pode ser modificado por qualquer usuário, é possível baixar e instalar o interpretador Python gratuitamente e sua licença compatível com a *General Public License* (Licença Pública Geral) acaba sendo bastante difundida entre grandes empresas como: Google, Microsoft, Yahoo e Disney. (CORREA, 2020)

A tipagem da linguagem é definida como sendo uma tipagem dinâmica não sendo necessário o usuário definir o definir o tipo de variáveis, apenas atribuir valor a ela que o interpretador se encarrega de definir o seu tipo a partir do que for atribuído a essa variável. Os passos que o interpretador realiza durante a execução de um código, são a análise do código, conversão para símbolos, armazena em *bytecode*,

um formato binário com instruções para o interpretador, e por fim, compila o código. Além disso o interpretador python pode ser usado de forma interativa, onde cada linha de código é imediatamente traduzida e executada, oferecendo aos programadores uma forma simples e eficiente para examinar no instante em que escrevem os resultados intermediários obtido em qualquer passo de um processo de análise de dados. (CORREA, 2020)

A linguagem Python pode ser facilmente expandida com a importação de diversos pacotes, atualmente existem milhares de pacotes disponíveis no repositório central do Python (Python Package Index – PyPI). Muitos deles voltados para a ciência de dados, tais como os famosos, 'NumPy' (manipulação de vetores e matrizes), 'SciPy' (rotinas numéricas para resolver problemas de integração, equações algébricas e cálculo numérico, entre outras coisas), 'pandas' (importação e transformação de bases de dados), 'Matplotlib' (geração de gráficos) e 'scikit-learn' (algoritmos de mineração de dados e aprendizado de máquina). No projeto será utilizado diversos pacotes que serão explicados mais à frente. (CORREA, 2020)

2.1.1 PYCHARM

O *PyCharm* (Figura 2) é um Ambiente de Desenvolvimento Integrado ou em inglês, *Integrated Development Environment* (IDE), desenvolvido pela *JetBranis* de plataforma cruzada que fornece experiência consistente nos sistemas operacionais *Windows*, *macOS* e *Linux*. *PyCharm* está disponível em três edições: *Professional*, *Community* e Edu. (*JETBRAINS,* 2021), para o projeto foi escolhida a versão *Community*, a versão livre, mas contendo os recursos necessários para o desenvolvimento. Para baixar basta visitar o site oficial como mostra a Figura 2 e escolher a versão *Community.*

Figura 2 - Pycharm

Fonte: JETBRAINS (2021)

Como ressalta Dos Reis (2016), O PyCharm é um IDE utilizado para programação e Python, que possui diversos recursos extremamente úteis e que facilitam deveras as tarefas de desenvolvimento de *software,* tais como:

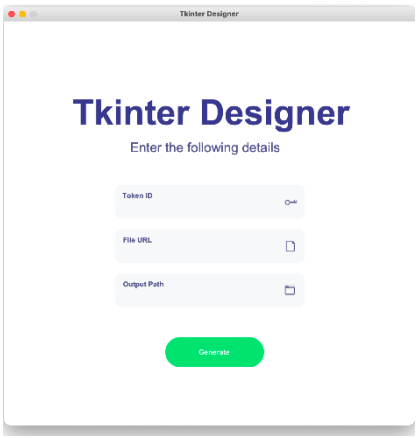
- Debugger gráfico;
- Unidade de testes integrada;
- Integração com sistemas de controle de versão, como Git, Mercurial e Subversion;
- Análise de código;
- *Code Completion;*
- Sintaxe e Erros destacados;
- Refatoração;
- Suporte a desenvolvimento com Django.

2.2 TKINTER

Para o desenvolvimento de uma interface gráfica é necessário usar bibliotecas que sejam capazes de gerar uma janela e interagir com o sistema. A biblioteca que foi utilizado no projeto foi a biblioteca Tkinter.

A Biblioteca tkinter (“interface Tk”) é a interface Python padrão para o kit de ferramentas de interface gráfica do usuário, em inglês, Graphical User Interface (GUI), Tcl/Tk como mostra a Figura 3. Ambos Tk e tkinter estão disponíveis na maioria das plataformas Unix, incluindo macOS, bem como em sistemas Windows. (PYTHON, 2021)

Figura 3 - Exemplo de Janela



Fonte: PYTHON (2021)

A execução a partir da linha de comando deve abrir uma janela demonstrando uma interface Tk simples, informando que está instalada corretamente em seu sistema. O Tkinter não é um invólucro fino, mas adiciona uma boa quantidade de sua própria lógica para tornar a experiência mais *pythonica.* (PYTHON, 2021)

Tcl / Tk não é uma única biblioteca, mas consiste em alguns módulos distintos, cada um com uma funcionalidade separada e sua própria documentação oficial. As versões binárias do Python também vêm com um módulo adicional junto com ele. (PYTHON, 2021)

Tcl é uma linguagem de programação interpretada dinâmica, assim como Python. Embora possa ser usado sozinho como uma linguagem de programação de propósito geral, é mais comumente embutido em aplicativos C como um mecanismo de script ou uma interface para o kit de ferramentas Tk. A biblioteca Tcl tem uma interface C para criar e gerenciar uma ou mais instâncias de um interpretador Tcl, executar comandos e scripts Tcl nessas instâncias e adicionar comandos personalizados implementados em Tcl ou C. Cada interpretador tem uma fila de eventos, e há instalações para enviar eventos e processá-los. Ao contrário do Python, o modelo de execução do Tcl é projetado em torno da multitarefa cooperativa e o Tkinter faz a ponte para essa. (PYTHON, 2021)

Tk é um pacote Tcl implementado em C que adiciona comandos personalizados para criar e manipular widgets GUI. Cada objeto TK incorpora sua própria instância de interpretador Tcl com Tk carregado nele. Os widgets do Tk são

muito personalizáveis, embora ao custo de uma aparência desatualizada. Tk usa a fila de eventos do Tcl para gerar e processar eventos GUI. (PYTHON, 2021)

Themed Tk (Ttk) é uma família mais recente de widgets Tk que fornecem uma aparência muito melhor em diferentes plataformas do que muitos dos widgets Tk clássicos. O Ttk é distribuído como parte do Tk, começando com o Tk versão 8.5. As ligações Python são fornecidas em um módulo separado tkinter.ttk. (Python, 2021)

2.3 VISÃO COMPUTACIONAL

Os sistemas de visão são extremamente importantes para se garantir a qualidade de um produto e a credibilidade da marca. A confiabilidade, competitividade e eficiência de toda a cadeia produtiva, não somente na inspeção acabam por crescerem com a inserção desses sistemas no ambiente fabril.

Um sistema de visão industrial é um conjunto de técnicas visuais de controle de qualidade, geralmente constituídos por uma câmera, iluminação, filtros e um software. Todo o conjunto desses equipamentos são utilizados para verificar se determinado produto está dentro dos padrões necessários para a aprovação e coletar dados da produção. Os sistemas de visão são excelentes para assegurar uma inspeção detalhada e com um alto padrão para controle de falhas e erros de fabricação. (BAUMGARTEN, 2018)

Os sistemas de visão coletam uma grande quantidade de dados de imagem, que podem orientar as operações e aumentar a produtividade de outras máquinas. Com uma quantidade tão grande de dados, torna-se muito importante no ambiente da Indústria 4.0, que envolve a coleta, interpretação, conexão e programação de dados da indústria. (BAUMGARTEN, 2018)

Conforme os recursos de análise de dados vão progredindo, as informações acessíveis por meio dos equipamentos de visão podem ser utilizadas para prevenir erros, GUI identificar com mais rapidez produtos defeituosos e permitir intervenções e respostas ágeis e eficazes. Antes que um produto apresente falha grave e gere paradas na linha de produção, um sistema de visão pode ter detectado uma mínima alteração desde o princípio, evitando muitos desperdícios. (BAUMGARTEN, 2018)

Um robô com sistema de visão, por exemplo, é capaz de realizar múltiplas inspeções por longos períodos, de forma muito mais confiável, eficiente e produtiva que os métodos tradicionais. (BAUMGARTEN, 2018)

Quando você precisa parar a linha de produção, jogar lotes inteiros fora, alocar vários funcionários para procurar a origem e resolução de uma falha, lidar com retorno de lotes ou descumprimento de requisitos, toda a atividade de produção se atrasa. Os prejuízos não são apenas financeiros, mas afetam também a credibilidade da empresa. (BAUMGARTEN, 2018)

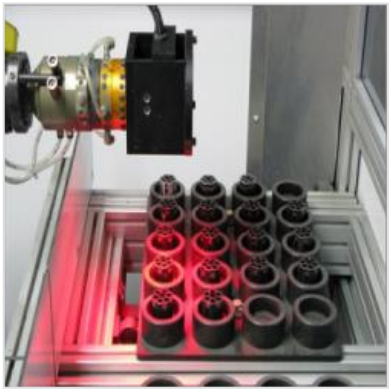
Os sistemas de visão também podem ser monitorados: eles podem medir o desempenho, diagnosticar e resolver problemas em tempo real, assim como muitos sistemas de visão de código de barras. Eles podem ler quase qualquer tipo de código, mesmo código corrompido, avaliá-lo e fornecer dados sobre todo o processo de instalação. (BAUMGARTEN, 2018)

Alguns sistemas de visão já possuem algoritmos integrados e não requerem um computador externo para processar os dados coletados. Eles são coletados e processados no próprio sistema. (BAUMGARTEN, 2018)

Os sistemas de visão industrial tornam a inspeção da qualidade do produto mais simples, rápida e confiável e podem capturar, conectar e processar grandes quantidades de dados. Esses dados podem ser usados para melhorar muitos processos industriais, desde a execução até a entrega. Os sistemas de visão são uma excelente solução para aumentar a produtividade e reduzir custos, integrando a empresa em um ambiente de indústria 4.0. (BAUMGARTEN, 2018)

Algumas aplicações de visão computacional na indústria podem ser vistas nas imagens a seguir: Na Figura 4 podemos ver a aplicação em fábricas de siderúrgicas, onde são realizadas a verificação e visualização de chapas metálicas em laminação (quente e frio) para inspeção de furos ou efeitos como “pitch” (TESLA, 2018).

Figura 4 - Sistema de visão em fábrica siderúrgica



Fonte: TESLA (2018)

Nas fábricas de farmacêutica os sistemas de visão computacional são utilizados para realizar a Inspeção de blisters, ampolas, rótulos, bisnagas; Verificação de lote, validade, cores, comprimidos e cápsulas, verificação de nível de envase. Em fábricas alimentícias são utilizados para inspeção de tampas e de montagem de tampas, inspeção de marca impressa, registro de cores, visualização de impressão, verificação de rótulos e etiquetas, como mostra a Figura 5. (TESLA, 2018).

Figura 5 - Sistema de visão em fábrica de embalagens

Fonte: TESLA (2018)

Também é utilizado em contagem de componentes, Inspeção de cor em alimentos assados para detecção de queimaduras ou elementos estranhos; Verificação de fechamento de embalagens e presença correta de itens na embalagem como mostra a Figura 6. (TESLA, 2018).

Figura 6 - Sistema de visão em fábrica alimentícia

Fonte: TESLA (2018)

Na área de autopeças e automobilística os sistemas de visão são utilizados na Metrologia em engrenagens e polias; Verificação de componentes como válvulas e molas; Identificação de partes para correta montagem, análise de parafusos e elementos de fixação. Verificação da montagem de componentes em chassi, como mostra a Figura 7, e blocos de motor, sistema de transmissão, inspeção em vidros silkscreen e terminais. (TESLA, 2018).

Figura 7 - Sistema de visão em fábrica automobilística

Fonte: TESLA (2018)

2.4 OPENCV

O OpenCV (open source Computer Vision) (Figura 8) é uma biblioteca de Visão Computacional de Código Aberto. Em 1999 Gary Bradski, trabalhando na Intel Corporation, lançou o OpenCV com a esperança de acelerar o desenvolvimento na área de visão computacional e de inteligência artificial, fornecendo uma sólida infraestrutura para qualquer pessoa interessada trabalhar nesse campo. A biblioteca é escrita em C e C++ e pode rodar em Linux, Windows e Mac OS X. Existe desenvolvimento ativo nas interfaces para Python, Java, MATLAB, entre outras linguagens incluindo a portabilidade da biblioteca para Android e IOS para aplicações móveis. (KAEHLER, BRADSKI, 2017)

Figura 8 - OpeCV Logo

Fonte: OPENCV.ORG(2014)

O OpenCV tem recebido bastante suporte através dos anos da Intel como também do Google, mas especialmente da Itseez (recentemente adquirida pela Intel), que fez grande parte do desenvolvimento inicial. Finalmente Arraiy se juntou para manter o Opencv.org gratuito. O OpenCV foi projetado para eficiência computacional e com um forte foco em aplicativos de tempo real. Ele é escrito em C++ otimizado e pode tirar proveito dos processadores Multicores.

Um dos objetivos do OpenCV é fornecer uma infraestrutura de visão computacional simples de usar que ajuda os desenvolvedores a criar aplicações de visão bastante sofisticadas rapidamente. A biblioteca do OpenCV contém atualmente mais de 500 funções que abrangem muitas das áreas de visão computacional, incluindo inspeção em produtos de fábrica, imagens médicas, segurança, interface de usuário, calibração de câmeras e robótica. Como a visão computacional e o aprendizado de máquina costumam andar de mãos dadas, o OpenCV também contém

uma biblioteca de aprendizado de máquina completa e de uso geral. (KAEHLER e BRADSKI, 2017)

Mesmo que existem outras tecnologias e bibliotecas para análise e reconhecimento de padrões, análise de movimento e rastreamento de objetos, o OpenCV ainda é a solução mais adequada para o pré-processamento e manipulação de imagens e por conta disso a comunidade mantém uma biblioteca atualizada. Essas informações podem ser verificadas por meio do repositório do OpenCV no GitHub. (KAEHLER e BRADSKI, 2017)

2.5 OCR

O reconhecimento óptico de caracteres ou OCR (Optical Character Recognition) é um processo utilizado para reconhecer caracteres a partir de um arquivo de imagem, podendo ser escaneado, impresso ou escrito à mão. Desta forma pode-se verificar que é possível converter diferentes tipos de arquivos digitalizados em documentos com dados pesquisáveis ou editáveis, ou seja, convertem imagens de texto em texto real (ISLAM et al., 2016).

Este tipo de reconhecimento analisa o documento e compara seus caracteres com fontes armazenadas em seu banco de dados e/ou reconhece características típicas de determinado caractere. Portanto, OCR é um programa que reconhece caracteres, extraindo o conteúdo de uma imagem e convertendo em texto (IFRS, 2018). Segundo Chaudhuri et al. (2017), em torno de 1870 surgiram os primeiros relatos sobre o reconhecimento óptico de caracteres, por meio da criação do scanner de retina desenvolvido por Charles R. Carey, de Boston, Massachusetts. Esse scanner era um sistema de transmissão de imagem que utilizava um mosaico de fotocélulas. Em 1890, Nipkow inventou o scanner sequencial, o qual analisava a imagem linha por linha, sistema semelhante ao usado na televisão moderna (CUNHA, 2018).

Em 1900, o russo cientista Tyurin, por meio de experimentos bem-sucedidos realizados com OCR mostrou a aplicação para auxílio de deficientes visuais. Após diversos experimentos durante as primeiras décadas neste ramo se sucederam. Em torno de 1940 por meio do desenvolvimento de computadores digitais pode-se obter

a versão moderna de OCR. Uma década após esta tecnologia já 24 estava disponível comercialmente. (MIRANDA, 2021)

Os sistemas comerciais que surgiram entre 1960 e 1965 foram denominados como a primeira geração de OCR, esta versão era caracterizada pela leitura de um formato restrito de letras, o que com o decorrer do tempo foi aprimorado e pode reconhecer diversos tipos de formatos de letras. Em meados da década de 1960 e início da década de 1970 a segunda geração, o marco desta geração foi o reconhecimento de caracteres escritos à mão, porém com limitação a poucas letras. (MIRANDA, 2021)

Já a terceira geração surgiu em meados de 1970, nesta geração os principais desafios foram os documentos de baixa qualidade e grande conjunto de caracteres impressos e manuscritos. Os sistemas comerciais que começaram a ser disponibilizados na década de 1950 tiveram apenas o aumento das vendas após 1986. Este fato foi causado devido ao custo do hardware ter baixado ao longo desta época. Atualmente os sistemas OCR já estão incluídos em pacotes de software, o que gerou grande aumento na utilização desta tecnologia. (MIRANDA, 2021)

2.6 TESSERACT

Tesseract é um mecanismo de reconhecimento óptico de caracteres para vários sistemas operacionais. É um software livre , lançado sob a licença Apache . Originalmente desenvolvido pela Hewlett-Packard como software proprietário na década de 1980, foi lançado como código aberto em 2005 e o desenvolvimento foi patrocinado pelo Google desde 2006.(Vicent, 2006). Em 2006, o Tesseract foi considerado um dos mecanismos de OCR de código aberto mais precisos disponíveis. (UBUNTU, 2015)

O motor Tesseract foi originalmente desenvolvido como software proprietário nos laboratórios da Hewlett Packard em Bristol, Inglaterra e Greeley, Colorado entre 1985 e 1994, com mais alterações feitas em 1996 para a conversão para Windows e algumas migrações de C para C ++ em 1998. Muitos o código foi escrito em C e, em seguida, um pouco mais foi escrito em C++. Desde então, todo o código foi convertido para pelo menos compilar com um compilador C++. (Vicent, 2006) Muito pouco trabalho foi feito na década seguinte. Foi então lançado como código aberto em 2005

pela Hewlett Packard e pela Universidade de Nevada, Las Vegas (UNLV). O desenvolvimento do Tesseract foi patrocinado pelo Google desde 2006. (VICENT, 2006)

As versões iniciais do Tesseract só podiam reconhecer texto em inglês. O Tesseract v2 adicionou seis idiomas ocidentais adicionais (francês, italiano, alemão, espanhol, português do Brasil, holandês). A versão 3 estendeu o suporte a idiomas significativamente para incluir idiomas ideográficos (chinês e japonês) e da direita para a esquerda (por exemplo, árabe, hebraico), bem como muitos outros scripts. Os novos idiomas incluem árabe, búlgaro, catalão, chinês (simplificado e tradicional), croata, tcheco, dinamarquês, alemão (frakturscript), grego, finlandês, hebraico, hindi, húngaro, indonésio, japonês, coreano, letão, lituano, norueguês, polonês, português, romeno, russo, sérvio, eslovaco (script padrão e Fraktur), esloveno, sueco, tagalo, tâmil, Tailandês, turco, ucraniano e vietnamita. V3.04, lançado em julho de 2015, adicionou 39 combinações de idioma / script, elevando a contagem total de idiomas de suporte para mais de 100. (VICENT, 2006)

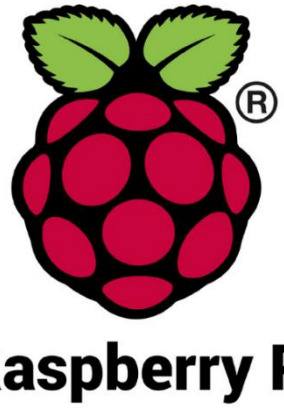
A saída do Tesseract terá qualidade muito baixa se as imagens de entrada não forem pré-processadas para se adequar a ele: As imagens (especialmente as capturas de tela) devem ser aumentadas de forma que o texto x-height seja de pelo menos 20 pixels, qualquer rotação ou inclinação deve ser corrigida ou nenhum texto será reconhecido, as mudanças de baixa frequência no brilho devem ser filtradas em alta frequência ou o estágio de binarização do Tesseract destruirá grande parte da página e as bordas escuras deverão ser removidas manualmente ou serão mal interpretadas como caracteres. (TESSERACT, 2014)

2.7 RASPBERRY

Raspberry Pi (Figura 9) é uma série de computadores de placa única de tamanho reduzido, que se conecta a um monitor de computador ou TV, e usa um teclado e um mouse padrão, desenvolvido no Reino Unido pela Fundação Raspberry Pi. Todo o hardware é integrado numa única placa. O principal objetivo é promover o ensino em Ciência da Computação básica em escolas, inclusão e empoderamento social, sendo multiplataforma, considerando as mais consagradas marcas de videogames do mundo é também como parte deste processo uma excelente

plataforma, tanto para a indústria quanto para as casas inteligentes e os IOT - Internet das Coisas. (RASPBERRY, 2012).

Figura 9 - Raspberry Logo

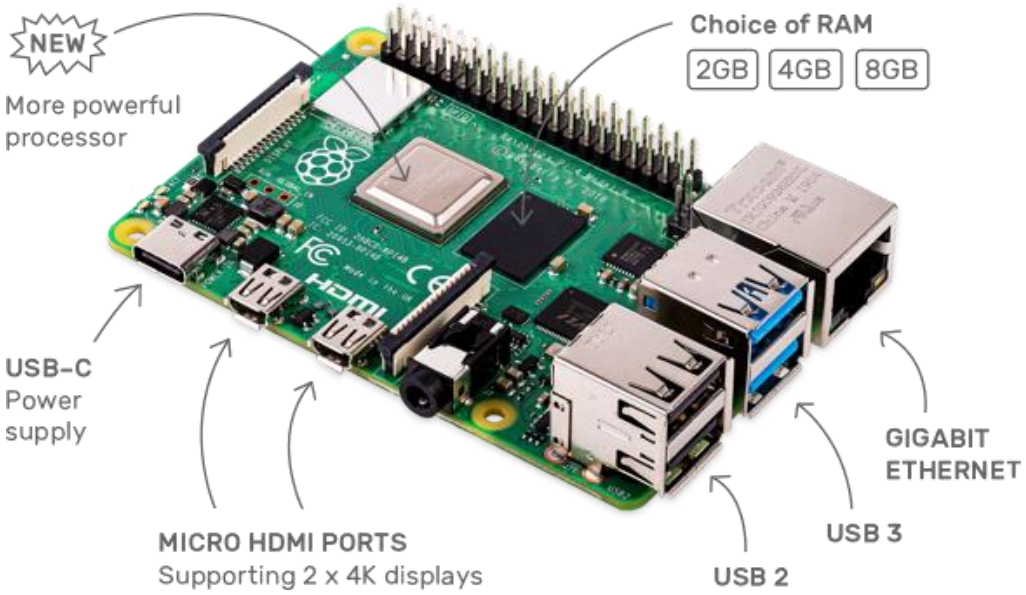


Fonte: RASPBERRY.ORG (2012)

Em 2006, os primeiros conceitos do Raspberry Pi foram baseados no microcontrolador Atmel ATmega644. Nesse tempo seus esquemas e layout de PCB foram disponíveis ao público. A Fundação *Trustee Eben Upton* reuniu professores, acadêmicos e admiradores da computação para criar um computador que motivasse as crianças a desenvolverem algo criativo. O computador é inspirado da Acorn BBC Micro de 1981 Modelo A, Modelo B e Modelo B+ são referências aos modelos originais de escolaridade do computador BBC Micro britânico, desenvolvido pela Acorn Computers. A primeira versão de arquitetura ARM do Raspberry Pi foi montada em um pacote do mesmo tamanho de uma memória stick USB.Tinha uma porta USB em uma extremidade e uma porta HDMI na outra. (MARIMOTO, 2012).

A Raspberry utilizada no projeto foi a Raspberry Pi 4 (Figura 10). Na parte da conectividade temos duas portas USB 2.0 e duas USB 3.0, enquanto a porta Ethernet USB 2.0 foi trocada por uma Gigabit Ethernet para altas velocidades de conexão. O conector de 3,5 mm permanece no lugar, permitindo ligar adaptadores de áudio e vídeo composto (para TVs antigas e outros projetos), enquanto ele agora suporta Bluetooth 5.0 e redes Wi-Fi, compatível com 802.11b/g/n/ac e frequências de 2,4 GHz a 5 GHz. (GOGONI, 2019)

Figura 10 - Rapberry Pi 4 e conectividade



Fonte: RASPBERRYPI.ORG (2021)

A porta HDMI padrão foi substituída por duas saídas micro-HDMI, permitindo usar dois monitores ao mesmo tempo; o usuário pode utilizar dois em 4K a 30 frames por segundo, ou um em 4K e 60 fps e outro na resolução Full HD. (GOGONI, 2019)

2.8 WEBCAM HD LOGITECH C270

A C270 HD Webcam oferece chamadas em conferência nítidas e suaves (720p/30 qps) em formato widescreen. A correção automática de luz mostra cores naturais e realistas. Possui iluminação com correção automática de luz para a melhor imagem possível. Na Figura 11 é mostrada a câmera usada no protótipo.

Figura 11 - Logitech C270

Fonte: LOGITECH (2020)

3 PROJETOS RELACIONADOS

Este capítulo apresenta projetos relacionados ao problema tratado neste trabalho. Desta forma são analisadas as características e a contribuição para o desenvolvimento do trabalho proposto. A análise é fundamentada em identificar e comparar o foco dos trabalhos, a tecnologia e a abordagem de desenvolvimento dos projetos.

3.1 RECONHECIMENTO DE PLACAS AUTOMOTIVAS

A técnicas utilizadas para reconhecer caracteres em documentos são as mesmas técnicas que podem ser aplicadas no reconhecimento de placas. Porém, como as placas de automóveis estão em ambiente aberto, com a iluminação incidente variável, bastante ruído, e muitas vezes temos que encontrar mais de uma placa em uma imagem que podem ter outros caracteres, precisamos de mais algumas operações. Em condições mais severas com potencial perda de dados, técnicas mais intensivas de computação, como aprendizado profundo com redes convolucionais, podem ser usadas. (PEIXOTO et al,2014)

O projeto de reconhecimento de placas de automóveis brasileiras foi desenvolvido por pesquisadores do Departamento de Ciência de Computação da Universidade Federal de Minas Gerais e da Federal de Ouro Preto. Visando estudar três abordagens de redes convolucionais diferentes para melhorar as representações de características dessas imagens de caracteres através de *deep learning*. Foram realizados experimentos em um conjunto de dados de imagens de placas brasileiras. (PEIXOTO et al,2014)

Neste trabalho, foram apresentadas três redes convolucionais que aplicam diferentes abordagens para aprender/gerar os pesos dos filtros da operação de convolução. Enquanto a primeira abordagem otimiza a arquitetura empregando filtros aleatórios com média zero e norma igual a 1, a segunda e a terceira utilizam um algoritmo supervisionado e um não supervisionado, respectivamente, para aprender os pesos dos filtros. (PEIXOTO et al,2014)

Os resultados mostram que as abordagens *deep learning* obtiveram acurácias muito superiores as que utilizam descritores *hand-designed* nos dois conjuntos de

dados avaliados. A abordagem de Otimização de Arquitetura (AO) com filtros aleatórios obteve a melhor acurácia no *dataset* Dígitos 99,81%, já no *dataset* Letras as três abordagens *deep learning* obtiveram a mesma taxa de erro 1,25%. A abordagem Aprendizado Supervisionado de Filtros (ASF) reduziu significativamente esse erro de classificação no *dataset* Letras aplicando a estratégia que adiciona camadas localmente conectadas a rede convolucional e a técnica *data augmentation*. Note que a abordagem OA apresentou outra vantagem sobre as demais: obteve a mesma acurácia no *dataset* Letras com apenas uma camada de convolução sem a operação de normalização e sem realizar pré-processamento nas imagens de entrada. (PEIXOTO et al, 2014)

Como uma importante conclusão do estudo, foi observado que a técnica *data augmentation* e a adição de camadas localmente conectadas aumentou significativamente as acurácias obtidas para ambos *datasets* na abordagem ASF. O resultado obtido no *dataset* Dígitos pela abordagem OA não foi alcançado pela abordagem ASF mesmo após a aplicação das técnicas investigadas. Isso sugere que a abordagem OA poderá obter resultados ainda melhores se combinada a tais estratégias. (PEIXOTO et al,2014)

3.2 LOCALIZAÇÃO DE ROBÔS POR RECONHECIMENTO ÓTICO DE CARACTERES DE PLACAS.

O projeto de localização de robôs por reconhecimento ótico de caracteres de placas foi desenvolvido por um aluno graduando da universidade federal de Santa Catarina construindo um sistema de localização baseado em visão capaz de detectar as placas de identificação de salas, reconhecer os caracteres e aplicá-lo para determinar sua posição em um mapa topológico do ambiente.

A aplicação deste sistema destina-se a ambientes internos estruturados. Os marcadores utilizados para localização são placas de identificação padronizadas, com crachás retangulares compostas por cores contrastantes e a fonte utilizada não é estilizada. O sistema é consiste em um único software integrado desenvolvido em linguagem C++. Sua arquitetura é composta por: um processo de identificação de caracteres em potencial, uma interface com um motor de OCR de código aberto para reconhecimento de caracteres e um processo que combina as informações obtidas

dos caracteres com um mapa do ambiente de trabalho do robô para determinar a sua localização. (JUNIOR, 2018)

Cada uma das partes do sistema foi desenvolvida para iterar sobre um vetor de entradas e retornar um vetor de resultados. Desta forma, se alguma situação, como nenhuma possível placa ser detectada na imagem, levar algum dos processos a gerar um vetor de resultados vazio, o funcionamento do sistema não é comprometido. Além disso, caso um processo gere um vetor de resultados vazio, os processos seguintes não desperdiçam tempo e o sistema pode rapidamente retornar para a etapa de detecção. (JUNIOR, 2018)

Por fim a localização apresentou bons resultados. De todas as 57 leituras aceitas para o processo de localização, 52 delas resultaram e localizações corretas. Porém, como apenas uma parcela de todas as leituras foi aceita, o sistema só foi capaz de determinar sua localização em 54,54% das imagens. Este método também não é viável para mapeamento do ambiente, visto que as condições heurísticas dependem de conhecimento prévio dos marcadores que podem ser encontrados pelo robô. (JUNIOR, 2018)

Em geral o sistema cumpriu com os objetivos do trabalho, porém com muitas melhorias a serem implementadas como métodos mais sofisticados e robustos para detecção de placas e seus cantos. A estruturação completa do mapa do ambiente e a integração do sistema de localização com estratégias de navegação para aplicação em um robô real permitiria o teste do comportamento do sistema em uma aplicação real. (JUNIOR, 2018)

4 MÉTODO E MATERIAIS UTILIZADOS

Este capítulo tem como objetivo apresentar os materiais necessários para a construção do JIG de teste, da programação do software e a construção das telas que simulará programas de teste. Explanando os métodos utilizados para encontrar as soluções propostas na introdução deste trabalho.

4.1 COMPONENTES

A montagem do projeto se deu a partir da utilização dos componentes citados na tabela 1 abaixo:

Tabela 1 - Lista de Componentes

COMPONENTES	QUANTIDADE
Raspberry Pi 4	1
Logitech C270	1
Módulo Relé	1
Conjunto de Leds	4

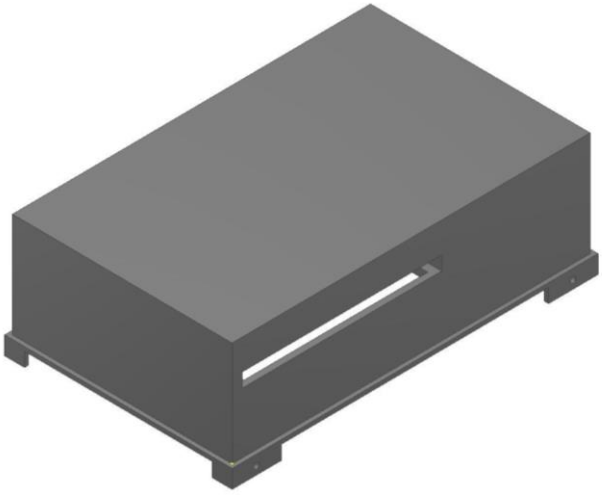
Fonte: O Próprio Autor (2021)

4.2 JIG DE TESTE

O protótipo para aquisição de imagens foi modelado no Software CAD de modelagem. Composto por duas partes: A Tampa e a Base.

Na Figura 12 mostra a Tampa do JIG, tendo uma abertura para inserção da etiqueta do roteador, e mais quatro suportes para encaixe na parte inferior. Com dimensões de 181 mm de comprimento, 111 mm de largura e 54 mm de altura.

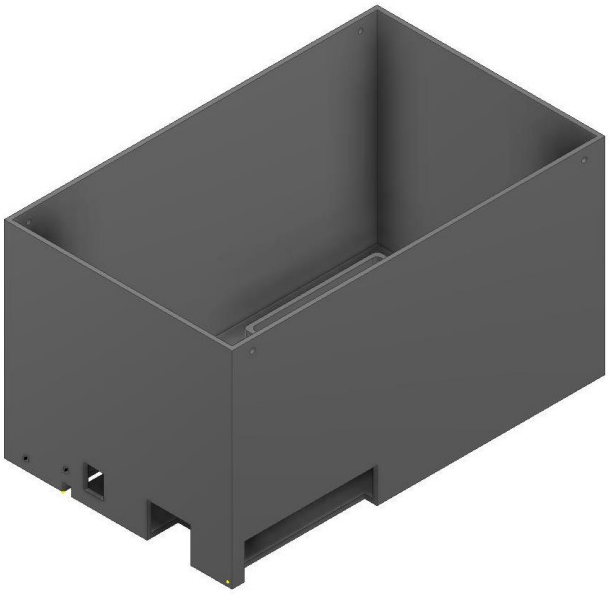
Figura 12 - Tampa do JIG



Fonte: O Próprio Autor (2022)

Na Figura 13 mostra a Base do JIG, com dimensões de 180 mm de comprimento, 110 mm de largura e 100 mm de altura. Com rasgos para acesso ao cartão SD da Raspberry e as portas HDMI e de energia.

Figura 13 - Base do JIG



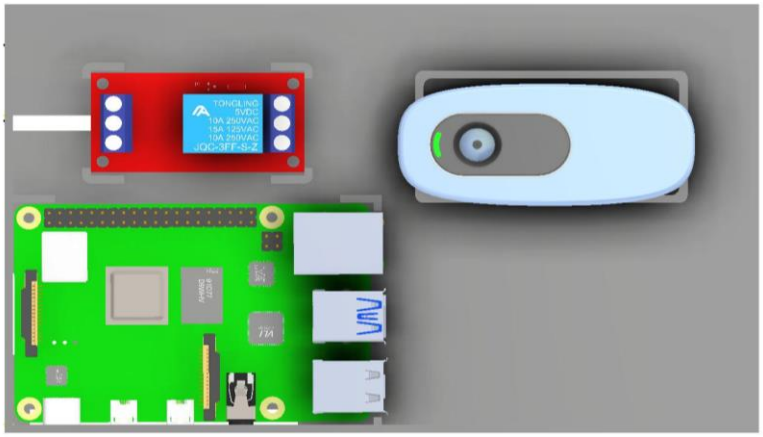
Fonte: O Próprio Autor (2022)

Na Figura 14 mostra os suportes para a câmera, módulo relé e para a Raspberry, fazendo com que os equipamentos fiquem nos locais predeterminados e na Figura 15 mostra todos os equipamentos nos seus locais.

Figura 14 - Visão superior da base

Fonte: O Próprio Autor (2022)

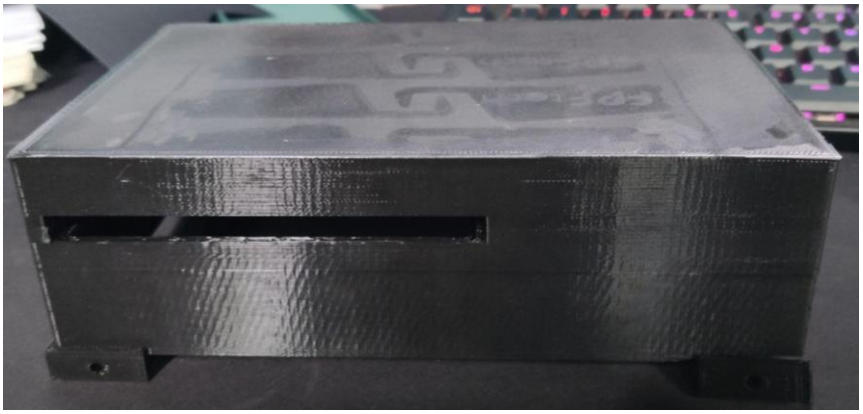
Figura 15 - Visão superior da base com os equipamentos



Fonte: O Próprio Autor (2022)

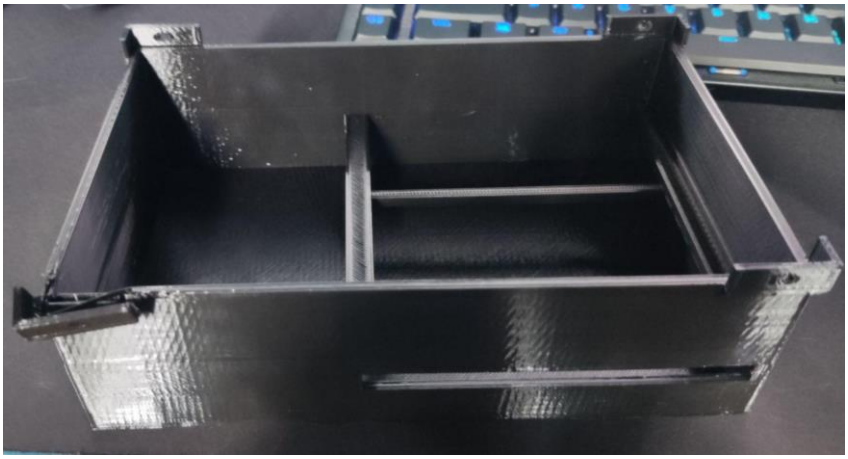
Após a finalização da impressão e colocação dos equipamentos as peças ficaram da seguinte forma: Tampa (Figura 16 e Figura 17) e a Base (Figura 18).

Figura 16 - Tampa superior impressa



Fonte: O Próprio Autor (2022)

Figura 17 - Parte interna da tampa



Fonte: O Próprio Autor (2022)

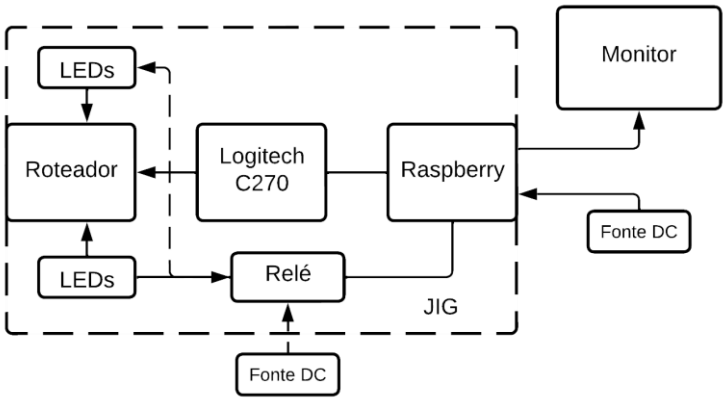
Figura 18 - Componentes Montados

Fonte: O Próprio Autor (2022)

4.3 DIGRAMA DE FUNCIONAMENTO

O protótipo para aquisição de imagens proposto neste TCC apresenta a composição mostrada na Figura 19. O sistema é capaz de iluminar o roteador com o conjunto de leds, capturando a imagem através da câmera enviando ao Raspberry, que realiza o processamento e apresenta os resultados no monitor externo.

Figura 19 - Diagrama de montagem e funcionamento



Fonte: O Próprio Autor (2022)

No Diagrama estão presentes os seguintes componentes:

- **Fonte DC:** A fonte DC é um dispositivo capaz de converter a corrente alternada em corrente contínua fornecendo a tensão elétrica com boa capacidade de corrente necessária para o funcionamento do Raspberry e do conjunto de Leds.
- **Relé:** O relé é um interruptor eletromecânico com inúmeras aplicações possíveis em comutação de contatos elétricos, servindo para ligar ou desligar dispositivos.
- **LEDs:** O led é usado para a emissão de luz em locais e instrumentos onde se torna mais conveniente a sua utilização no lugar de uma lâmpada, o nosso projeto é utilizado um conjunto de leds
- **Raspberry:** É uma série de computadores de placa única do tamanho reduzido, que se conecta a um monitor de computador ou TV, e usa um teclado e um mouse padrão, o modelo utilizado no projeto é o Raspberry Pi 4.
- **Logitech C270:** A Webcam possui resolução máxima de 720p, com sensor de 0,9 Megapixels, com campo de visão diagonal de 55°.

- **Monitor:** Um monitor é um dispositivo de saída do computador, cuja função é transmitir informação ao utilizador através da imagem.

4.4 SOFTWARE

Este tópico contém informações relacionadas a construção do software usado no tratamento da imagem, leitura das etiquetas e apresentação das informações nas telas de teste.

4.4.1 BIBLIOTECAS

Biblioteca é uma coleção de subprogramas utilizados no desenvolvimento de software. Bibliotecas contém código e dados auxiliares, que provém serviços a programas independentes, o que permite o compartilhamento e a alteração de código e dados de forma modular.

4.4.1.1CV2

OpenCV é uma biblioteca de programação, de código aberto possuindo mais de 500 funções, é usada para diversos tipos de análise em imagens e vídeos, reconhecimento facial, detecção e análise de textos entre outros. (CEDRO TECHNOLOGIES, 2018)

4.4.1.2 NumPy

A biblioteca NumPy facilita trabalhar com arranjos, vetores e matrizes. Como vamos trabalhar com imagens digitais, que são representadas como matrizes de pixels, esse pacote é indispensável para aumentar a produtividade. (BARELLI, 2018)

4.4.1.3 TkInter

Tkinter é o pacote GUI padrão de fato do Python. É uma fina camada orientada a objetos. Tkinter não é o único kit de ferramentas de GUI para Python. No entanto, é o mais usado. (TKINTER, 2022).

4.4.1.4Pytesseract

Python-tesseract é uma ferramenta óptica de reconhecimento de caracteres (OCR) para python. Sendo um invólucro para o Motor Tesseract-OCR do Google. Também é útil como um script de invocação autônomo para Tesseract. (PYPI, 2021)

4.4.1.5 Pil

A PIL *(Python Imaging Library)* adiciona recursos de processamento de imagens ao seu interpretador Python. Esta biblioteca fornece um amplo suporte e recursos de processamento de imagem bastante poderosos. (PILLOW, 2021)

4.4.2 ALGORITMO

Para a aquisição da imagem na Raspberry foram criadas funções que permitiam o acesso a câmera e realizar as tarefas necessárias, as funções são uma sequência de comandos que executa alguma tarefa e que tem um nome. A sua principal finalidade é nos ajudar a organizar programas em pedaços que correspondam a como imaginamos uma solução do problema. (Elkner et al.,2016)

No projeto as funções são utilizadas para modularizarmos o software, assim permitindo trabalhar em partes diferentes do código sem a necessidade de todo o conjunto esteja em perfeito funcionamento. Aqui serão apresentadas as funções que criamos para o projeto.

Utilizamos duas grandes funções no programa, chamadas de “ler_etiqueta” e a função “chamar_telas”, que explicaremos adiante.

4.4.2.1 Ler_etiqueta

Primeiramente, é necessário realizar a captura da foto, utilizamos o comando de captura de foto na Raspberry, para garantir que a câmera esteja com o melhor foco, realizamos a captura de 30 fotos, mas somente guardando a última foto como que é mostrado no código na Figura 20.

Figura 20 - Captura da foto

```
imagemOriginal = cv2.imread("etiqueta3.jpg")
```

Fonte: O Próprio Autor (2021)

Para garantir que a imagem tirada esteja no tamanho ideal para o processamento, utilizamos a função *"resize()"* do Python.

4.4.2.2 Resize()

Nós o redimensionamos com a função *resize()*. Um aspecto importante aqui é o parâmetro *interpolation*, que essencialmente diz como redimensionar uma imagem. Utilizamos o método de interpolação *"INTER_AREA"* como é mostrado na Figura 21, esse método utiliza a relação da área de pixels para o redimensionamento.

Devido ao tamanho da resolução da imagem ser alta, é necessário realizar o redimensionamento da imagem, foi escolhida a redução da imagem em 30 por cento do tamanho inicial, assim utilizamos duas variáveis, para realizar o redimensionamento de altura (*height*) e largura (*width*), por fim enviamos essas informações para a função "*resized*".

Figura 21 - Função de redimensionamento

```
scale_percent = 30
width = int(imagemOriginal.shape[1] * scale_percent / 100)
height = int(imagemOriginal.shape[0] * scale_percent / 100)
dim = (width, height)
resized = cv2.resize(imagemOriginal, dim, interpolation=cv2.INTER_AREA)
```

Fonte: O Próprio Autor (2021)

4.4.2.3 Escala de cinza

Para melhorar a precisão do software é necessário a aplicação de filtros e correções de imagens, sendo o sistema controlado sem a entrada de iluminação externa e com iluminação apropriada, não houve necessidade de aplicar esses filtros na captura da imagem, fizemos somente a conversão da escala de cores para a escala de cinza (Figura 22).

Figura 22 - Conversão para escala de cinza

```
imagemTratada = cv2.cvtColor(imagemOriginal, cv2.COLOR_BGR2GRAY)
```

Fonte: O Próprio Autor (2021)

4.4.2.4 Cortar_imagem()

Como o nosso projeto é especificado para um determinado tipo de roteador com um determinado tipo de etiqueta, fizemos o mapeamento do local da etiqueta para a maior precisão do local, como o tamanho da imagem foi redimensionado, através de tentativa e erro encontramos o local exato dos contornos da etiqueta, então foi criado a função “cortar_imagem()” (Figura 23) para realizar os cortes necessários nas imagens.

Figura 23 - Função cortar imagem

```
def cortar_imagem(self,imagem,x,w,y,h):
    imagemMod = imagem[y:y + h, x:x + w]
    return (imagemMod)
```

Fonte: O Próprio Autor (2021)

4.4.2.5 Ler_SERIAL()

Agora que já realizamos o tratamento da imagem, o redimensionamento e corte, iniciamos o início da leitura dos caracteres através da tecnologia OCR. Na Figura 24 exemplifica uma etiqueta.

Figura 24 - Exemplo de etiqueta

Fonte: O Próprio Autor (2021)

É possível ver algumas informações que são de nosso interesse, são elas:

- S/N
- CM MAC
- EMTA MAC
- WIFI MAC 2.4GHz
- WIFI MAC 5GHz
- REDE WIFI MAC 2.4GHz
- REDE WIFI MAC 5GHz
- USUÁRIO
- SENHA.

Como já realizamos o mapeamento da etiqueta, sabemos o local de todas as informações, iniciamos o processo como mostra a Figura 25, realizamos o corte na imagem da etiqueta, selecionando somente os locais da informação que queremos, após utilizamos a função *"image_to_string()"* do *Pytesseract* para realizar a transcrição dos caracteres presentes na imagem, armazenando as informações na variável *"text",* na leitura também é extraído os caracteres do início onde indica de qual informação se referem os números, por isso realizamos uma nova armazenagem dos valores do número de série, para os retirar, foi realizado o mapeamento e salvo somente a informação necessária.

Figura 25 - Função ler_serial

```
def ler_serial(self, imagemModificada):
    global SN
    IMGserial = self.cortar_imagem(imagemModificada, 40, 212, 78, 60)
    text = tess.image_to_string(IMGserial)
    tam = len(text)
    SN = text[10:tam - 1]
```

Fonte: O Próprio Autor (2021)

4.4.2.6Ler_CM_MAC()

A Figura 26 mostra como foi realizado a leitura do CM_MAC.

Figura 26 - Função Ler_CM_MAC

```
def ler_CM_MAC(self, imagemModificada):
    global CM_MAC
    IMGcm_mac = self.cortar_imagem(imagemModificada, 40, 222, 150, 50)
    text = tess.image_to_string(IMGcm_mac)
    tam = len(text)
    CM_MAC = text[13:tam - 1]
    print(CM_MAC)
```

Fonte: O Próprio Autor (2021)

4.4.2.7 Ler_EMTA_MAC()

A Figura 27 mostra como foi realizado a leitura do EMTA_MAC.

Figura 27 - Funçao Ler_EMTA_MAC

```
def ler_EMTA_MAC(self, imagemModificada):
    global EMTA_MAC
    IMGemta_mac = self.cortar_imagem(imagemModificada, 60, 222, 220, 40)
    text = tess.image_to_string(IMGemta_mac)
    tam = len(text)
    EMTA_MAC = text[22:tam - 1]
```

Fonte: O Próprio Autor (2021)

4.4.2.8Ler_WIFI_MAC_24G

A Figura 28 mostra como foi realizado a leitura do WIFI MAC 2.4G.

Figura 28 - Função Ler_WIFI_MAC_24G

```
def ler_WIFI_MAC_24G(self, imagemModificada):
    global WIFI_MAC_24G
    IMGwifi_mac_24g = self.cortar_imagem(imagemModificada, 100, 290, 270, 50)
    text = tess.image_to_string(IMGwifi_mac_24g)
    tam = len(text)
    WIFI_MAC_24G = text[5:tam - 1]
```

Fonte: O Próprio Autor (2021)

4.4.2.9Ler_WIFI_MAC_5G()

A Figura 29 mostra como foi realizado a leitura do WIFI MAC 5G.

Figura 29 - Função Ler_WIFI_MAC_5G

```
def ler_WIFI_MAC_5G(self, imagemModificada):
    global WIFI_MAC_5G
    IMGwifi_mac_5g = self.cortar_imagem(imagemModificada, 100, 290, 330, 45)
    text = tess.image_to_string(IMGwifi_mac_5g)
    tam = len(text)
    WIFI_MAC_5G = text[17:tam - 1]
```

Fonte: O Próprio Autor (2021)

4.4.2.10 Ler_REDE_WIFI_2.4G()

A Figura 30 mostra como foi realizado a leitura da REDE WIFI 2.4G.

Figura 30 - Função Ler_REDE_WIFI_2.4G

```
def ler_REDE_WIFI_24G(self, imagemModificada):
    global REDE_WIFI_24G
    IMGrede_wifi_24g = self.cortar_imagem(imagemModificada, 100, 250, 360, 40)
    text = tess.image_to_string(IMGrede_wifi_24g)
    tam = len(text)
    REDE_WIFI_24G = text[14:tam - 1]
```

Fonte: O Próprio Autor (2021)

4.4.2.11 Ler_SENHA_REDE_WIFI_2.4G()

A Figura 31 mostra como foi realizado a leitura da SENHA DA REDE WIFI 2.4G.

Figura 31 - Função Ler_SENHA_REDE_WIFI_2.4G

```
def ler_SENHA_REDE_WIFI_24G(self, imagemModificada):
    global SENHA_REDE_WIFI_24G
    IMGsenha_rede_wifi_24g = self.cortar_imagem(imagemModificada, 100, 250, 390, 40)
    text = tess.image_to_string(IMGsenha_rede_wifi_24g)
    tam = len(text)
    SENHA_REDE_WIFI_24G = text[15:tam - 1]
```

Fonte: O Próprio Autor (2021)

4.4.2.12 Ler_REDE_WIFI_5G()

A Figura 32 mostra como foi realizado a leitura do REDE WIFI 5G.

Figura 32 - - Função Ler_REDE_WIFI_5G

```
def ler_REDE_WIFI_5G(self, imagemModificada):
    global REDE_WIFI_5G
    IMGrede_wifi_5g = self.cortar_imagem(imagemModificada, 390, 250, 360,35)
    text = tess.image_to_string(IMGrede_wifi_5g)
    tam = len(text)
    REDE_WIFI_5G = text[14:tam - 1]
```

Fonte: O Próprio Autor (2021)

4.4.2.13 Ler_SENHA_REDE_WIFI_5G()

A Figura 33 mostra como foi realizado a leitura da SENHA DA REDE WIFI 5G.

Figura 33 - Função Ler_SENHA_REDE_WIFI_5G

```
def ler_SENHA_REDE_WIFI_5G(self, imagemModificada):
    global SENHA_REDE_WIFI_5G
    IMGsenha_rede_wifi_5g = self.cortar_imagem(imagemModificada, 390, 250, 390, 40)
    text = tess.image_to_string(IMGsenha_rede_wifi_5g)
    tam = len(text)
    SENHA_REDE_WIFI_5G = text[15:tam - 1]
```

Fonte: O Próprio Autor (2021)

4.4.2.14 Ler_USUARIO()

A Figura 34 mostra como foi realizado a leitura do USUÁRIO.

Figura 34 - Função Ler_usuário

```
def ler_USUARIO(self, imagemModificada):
    global USUARIO
    IMGusuario = self.cortar_imagem(imagemModificada, 490, 150, 85,20)
    text = tess.image_to_string(IMGusuario)
    tam = len(text)
    USUARIO = text[3:tam - 1]
```

Fonte: O Próprio Autor (2021)

4.4.2.15 Ler_SENHA()

Na Figura 35 mostra como foi realizada a leitura da senha, mas como a senha consiste na mesma sequência de caracteres do CM_MAC, nós só chamamos as variáveis globalmente, e atribuímos para a variável SENHA o mesmo valor do CM_MAC.

Figura 35 - Função Ler_SENHA

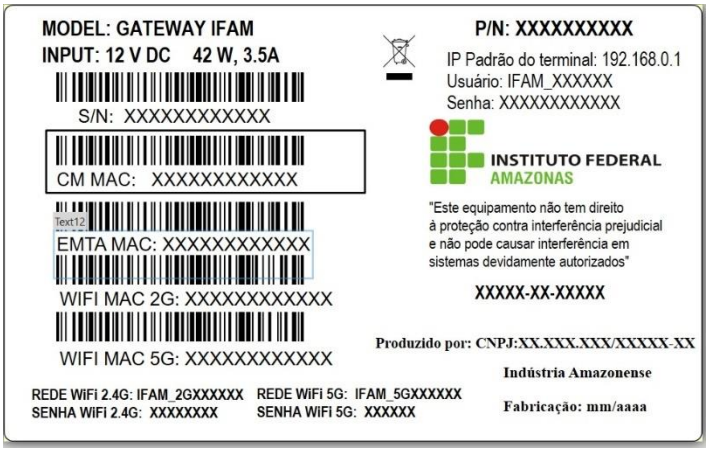
```
def ler_SENHA(self):
    global SENHA
    global CM_MAC
    SENHA = CM_MAC
```

Fonte: O Próprio Autor (2021)

4.4.2.16 Chamar_telas()

Finalizando a parte de leitura das informações das etiquetas, o próximo passo é a apresentação dessas informações nas telas de teste. Para apresentação das telas foi criado um modelo de etiqueta da tela como mostra a Figura 36.

Figura 36 - Etiqueta de exemplo



Fonte: O Próprio Autor (2021)

Para a construção das telas, utilizamos a TkInter, o pacote padrão de GUI, como todas as telas são com a mesma estrutura, fizemos uma função como mostra a Figura 37, a função recebe por parâmetro o texto que é apresentado na tela, o nome da imagem utilizada para a apresentação, o título que é apresentado na tela de teste, e por fim a variável que armazena os caracteres extraídos da etiqueta.

Figura 37 - Função chamar_telas

```
def chamar_telas(self):
    self.tela_teste("Insira o SN:", "IFAM_SN.png","SN",SN)
    self.tela_teste("Insira o CM MAC:", "IFAM_CMMAC.png", "CM MAC",CM_MAC)
    self.tela_teste("Insira o EMTA MAC:","IFAM_EMTAMAC.png","EMTA MAC",EMTA_MAC)
    self.tela_teste("Insira o WIFI MAC 24G:", "IFAM_WIFIMAC24G.png", "WIFI MAC 24G",WIFI_MAC_24G)
    self.tela_teste("Insira o WIFI MAC 5G:", "IFAM_WIFIMAC5G.png", "WIFI MAC 5G",WIFI_MAC_5G)
    self.tela_teste("Insira o nome da rede WIFI 2.4G:", "IFAM_REDE24G.png", "REDE WIFI 2.4G",REDE_WIFI_24G)
    self.tela_teste("Insira a senha da rede WIFI 2.4G:", "IFAM_SENHA24G.png", "SENHA REDE WIFI 2.4G",SENHA_REDE_WIFI_24G)
    self.tela_teste("Insira o nome da rede WIFI 5G:", "IFAM_REDE5G.png", "REDE WIFI 5G",REDE_WIFI_5G)
    self.tela_teste("Insira a senha da rede WIFI 5G:", "IFAM_SENHA5G.png", "SENHA REDE WIFI 5G",SENHA_REDE_WIFI_5G)
    self.tela_teste("Insira o nome do Usuario:", "IFAM_USUARIO.png", "USUARIO",USUARIO)
    self.tela_teste("Insira a senha de Usuario:", "IFAM_SENHA.png", "SENHA",SENHA)
```

Fonte: O Próprio Autor (2021)

4.4.2.17 Tela_testes()

Esta função apresenta os comandos necessários para a construção da tela de teste. (Figura 38), primeiro definimos outros função bt_ok(), que será chamada quando o botão “OK” for acionado, fazendo com que a janela aberta seja destruída. Depois iniciamos “janela” como um objeto de “Tk()”. A seguir serão apresentadas as funções usadas para a construção das telas.

- Janela.title(título): Função usada para definir o nome que será apresentado na barra superior da tela.
- Janela.geometry(‘1050x800’): Função usada para definir o tamanho da janela em pixels.
- Label(janela, text=texto_inicial): Implementa uma caixa de exibição onde pode se colocar texto ou imagens. O texto exibido pode ser atualizado a qualquer momento.
- texto_inicio.pack(): Este gerenciador de geometria organiza widgets blocos antes de colocá-los no widget pai.
- Text(janela, height=1, width=100): Os widgets de texto fornecem recursos avançados que permitem editar um texto multiline e formatar a maneira como ele deve ser exibido, como alterar sua cor e fonte.
- caixa_texto.insert(INSERT, leitura): Este método insere strings no local do índice especificado.
- ImageTk: O módulo ImageTk contém suporte para criar e modificar objetos Tkinter BitmapImage e PhotoImage a partir de imagens PIL.

- PhotoImage(Image.open(nome_imagem)): Abre o arquivo indicado e inicia um novo objeto.
- Button(janela, text="OK", command=bt_ok): O widget Button é usado para adicionar botões em um aplicativo Python. Esses botões podem exibir texto ou imagens que transmitam o propósito dos botões. A função “bt_ok” é anexada, chamando automaticamente quando o botão é clicado.
- Janela.mainloop(): É um loop infinito da janela do aplicativo que funciona para sempre para que possamos ver a tela parada.

Figura 38 - Função tela_teste

```
def tela_teste(self, texto_inicial, nome_imagem, titulo, leitura):
    def bt_ok():
        janela.destroy()

    janela = Tk()
    janela.title(titulo)
    janela.geometry('1050x800')

    texto_inicio = Label(janela, text=texto_inicial)
    texto_inicio.pack(padx=0, pady=10)

    caixa_texto = Text(janela, height=1, width=100)
    caixa_texto.insert(INSERT, leitura)
    caixa_texto.pack(padx=0, pady=10)

    img = ImageTk.PhotoImage(Image.open(nome_imagem))
    etiqueta = Label(janela, image=img)
    etiqueta.pack(padx=0, pady=10)

    botao = Button(janela, text="   OK   ", command=bt_ok)
    botao.pack(padx=0, pady=10, side = 'top')

    janela.mainloop()
```

Fonte: O Próprio Autor (2021)

5 RESULTADOS E DISCUSSÕES

O objetivo do TCC foi a modelagem e construção de um JIG de teste, construído em impressora 3D específica para o roteador, devido a pouca disponibilidade de roteadores para testes, foram usados exemplos de etiquetas coladas em um suporte para serem inseridas no JIG (Figura 39).

Figura 39 - Suporte das etiquetas

Fonte: O Próprio Autor (2022)

5.1 PROCESSAMENTO DE IMAGEM

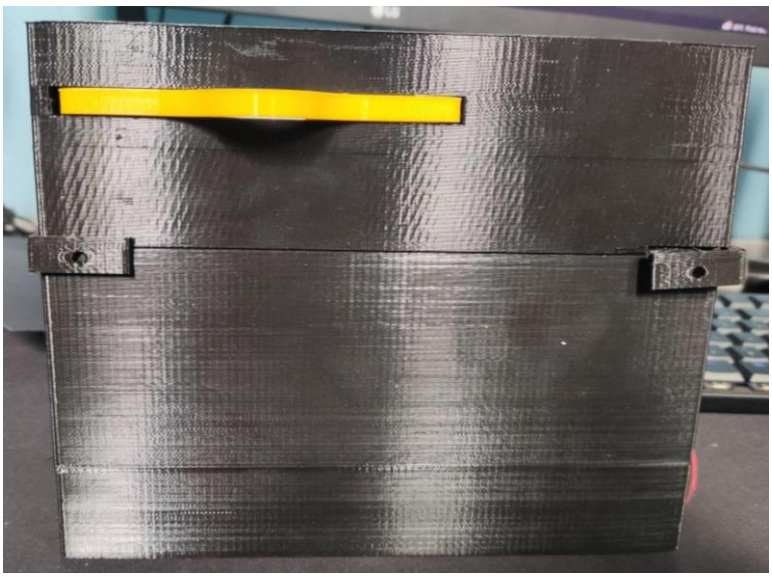
Como primeiro passo, a etiqueta é colocada na moldura (Figura 40).

Figura 40 - Moldura da etiqueta

Fonte: O Próprio Autor (2022)

Para ser inserida no recorte do JIG, como mostrado na Figura 41, o processo de iluminação inicia e são capturadas as fotos.

Figura 41 - JIG montado



Fonte: O Próprio Autor (2022)

Na Figura 42 é mostrada o resultado da captura da foto pela câmera.

Figura 42 - Foto capturada pela câmera

Fonte: O Próprio Autor (2021)

Na Figura 43 é evidenciado as melhorias causadas pela conversão da imagem para escala de cinza, corrigindo as imperfeições de cores na imagem.

Figura 43 - Imagem convertida para escala de cinza

Fonte: O Próprio Autor (2021)

Após o tratamento da imagem, é realizado o corte como mostra a Figura 44, ficando somente a imagem da área de interesse.

Figura 44 - Corte na área da etiqueta

Fonte: O Próprio Autor (2021)

Finalizado a parte de aquisição da imagem, é iniciado o processo de corte para as informações desejadas. A primeira é referente ao número de série, como mostrado na Figura 45.

Figura 45 - Número de série

Fonte: O Próprio Autor (2021)

A Figura 46 mostra a região de interesse do CM MAC.

Figura 46 - CM MAC

Fonte: O Próprio Autor (2021)

A Figura 47 mostra a região de interesse do EMTA MAC.

Figura 47 - EMTA MAC

Fonte: O Próprio Autor (2021)

A Figura 48 mostra a região de interesse do WIFI MAC 2.4G.

Figura 48 - WIFI MAC 2.4G

Fonte: O Próprio Autor (2021)

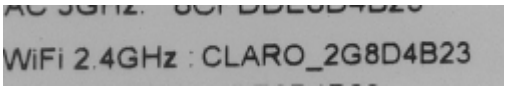
A Figura 49 mostra a região de interesse do WIFI MAC 5G.

Figura 49 - WIFI MAC 5G

Fonte: O Próprio Autor (2021)

A Figura 50 mostra a região de interesse da REDE WIFI 2.4G.

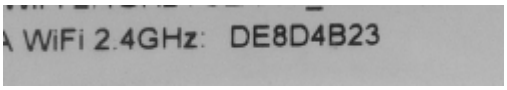
Figura 50 - REDE WIFI 2.4G



Fonte: O Próprio Autor (2021)

A Figura 51 mostra a região de interesse da SENHA DA REDE WIFI 2.4G.

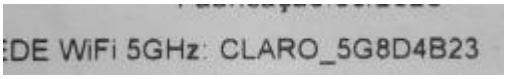
Figura 51 - SENHA DA REDE WIFI 2.4G



Fonte: O Próprio Autor (2021)

A Figura 52 mostra a região de interesse da REDE WIFI 5G.

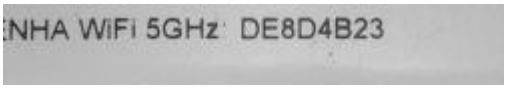
Figura 52 - REDE WIFI 5G



Fonte: O Próprio Autor (2021)

A Figura 53 mostra a região de interesse da SENHA DA REDE WIFI 5G.

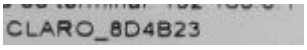
Figura 53 - SENHA DA REDE WIFI 5G



Fonte: O Próprio Autor (2021)

A Figura 54 mostra a região de interesse do USUÁRIO.

Figura 54 - USUÁRIO



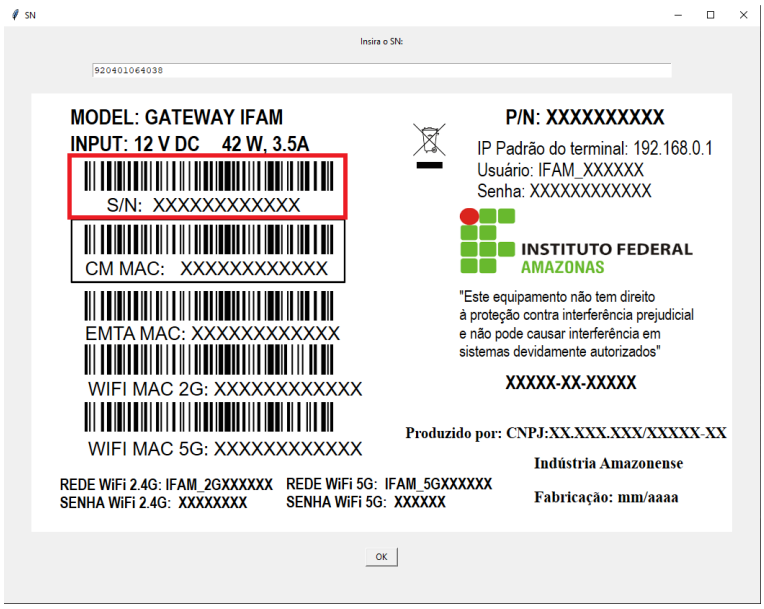
Fonte: O Próprio Autor (2021)

5.2 TELAS DO TESTE

Nesta seção serão apresentados os resultados das telas de testes com os resultados da leitura da etiqueta que poderão ser comparadas com a Figura 45.

A primeira tela (Figura 55) apresenta a leitura dos caracteres do número de série da etiqueta.

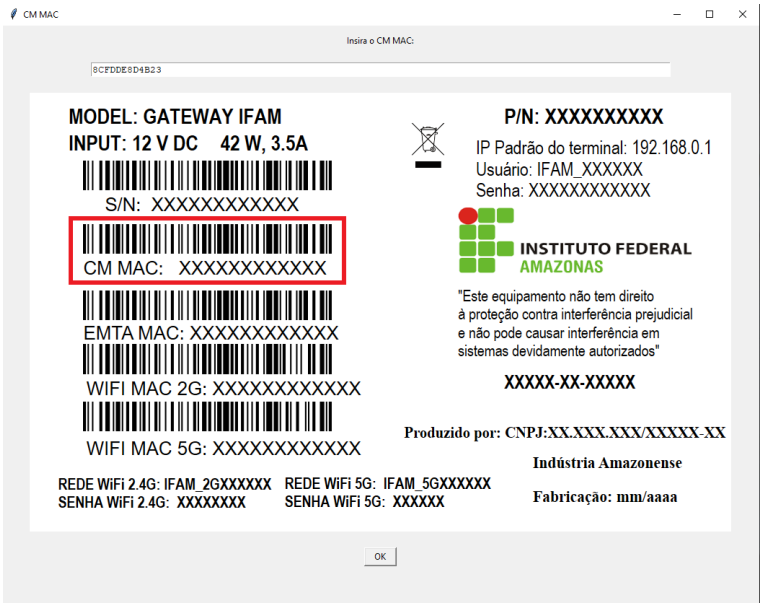
Figura 55 - Tela Número de Série



Fonte: O Próprio Autor (2021)

A tela da Figura 56 apresenta a leitura dos caracteres CM MAC da etiqueta.

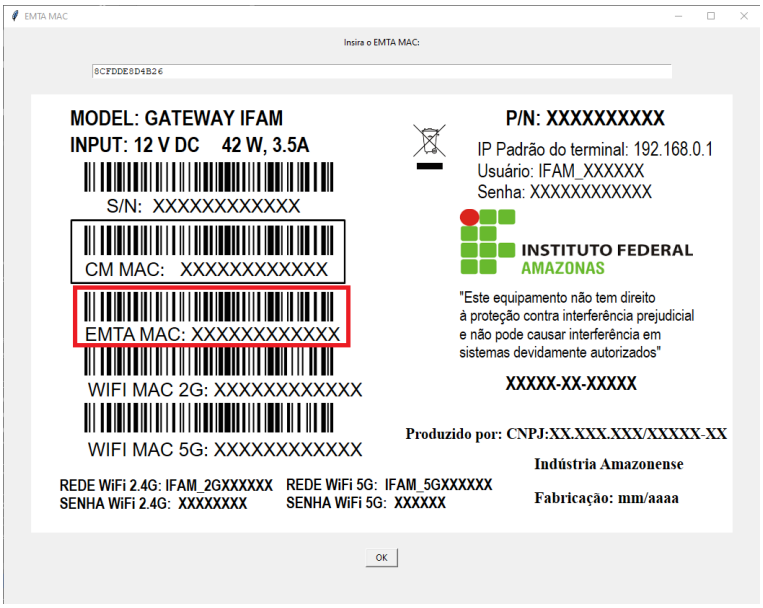
Figura 56 - Tela CM MAC



Fonte: O Próprio Autor (2021)

A tela da Figura 57 apresenta a leitura dos caracteres EMTA MAC da etiqueta.

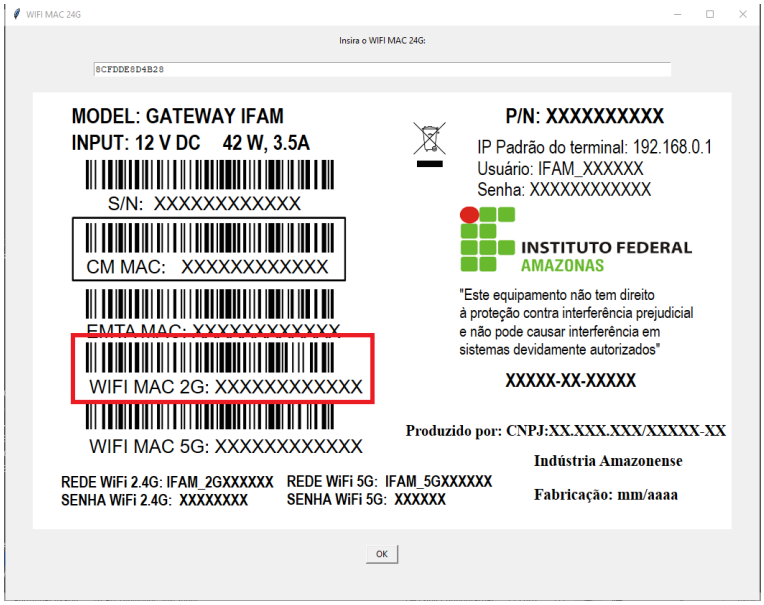
Figura 57 - Tela EMTA MAC



Fonte: O Próprio Autor (2021)

A tela da Figura 58 apresenta a leitura dos caracteres WIFI MAC 2.4G da etiqueta.

Figura 58 - Tela WIFI MAC 2.4G



Fonte: O Próprio Autor (2021)

A tela da Figura 59 apresenta a leitura dos caracteres WIFI MAC 5G da etiqueta.

Figura 59 - Tela WIFI MAC 5G

Fonte: O Próprio Autor (2021)

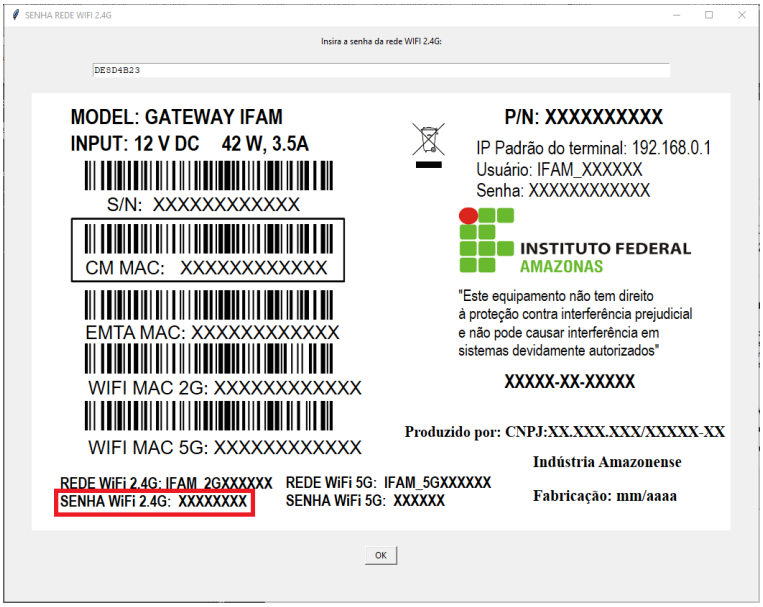
A tela da Figura 60 apresenta a leitura dos caracteres REDE WIFI 2.4G da etiqueta.

Figura 60 - Tela REDE WIFI 2.4G

Fonte: O Próprio Autor (2021)

A tela da Figura 61 apresenta a leitura dos caracteres SENHA WIFI 2.4G da etiqueta.

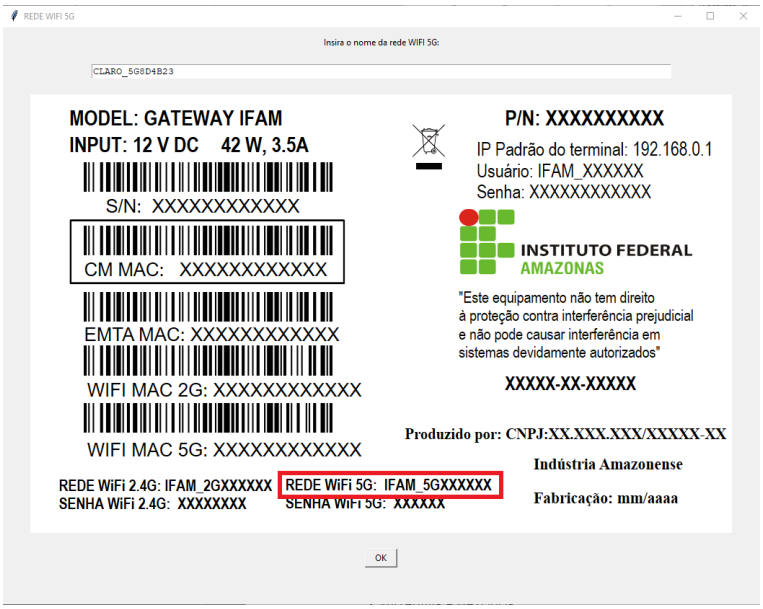
Figura 61 - Tela SENHA WIFI 2.4G



Fonte: O Próprio Autor (2021)

A tela da Figura 62 apresenta a leitura dos caracteres WIFI 5G da etiqueta.

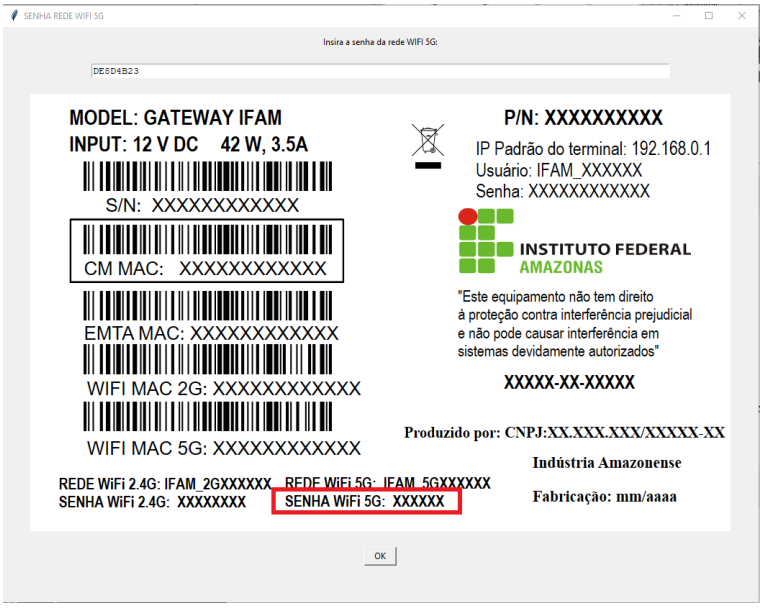
Figura 62 - Tela WIFI 5G



Fonte: O Próprio Autor (2021)

A tela da Figura 63 apresenta a leitura dos caracteres SENHA WIFI 5G da etiqueta.

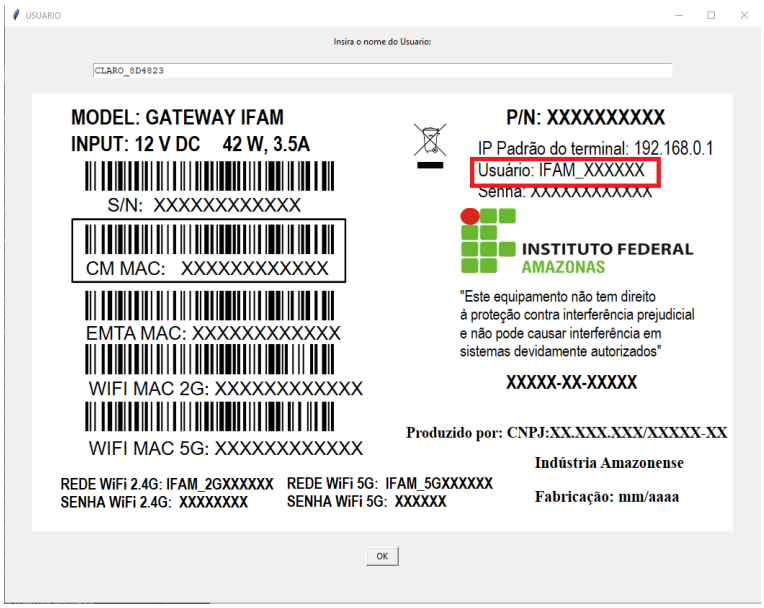
Figura 63 - Tela SENHA WIFI 5G



Fonte: O Próprio Autor (2021)

A tela da Figura 64 apresenta a leitura dos caracteres do USUÁRIO.

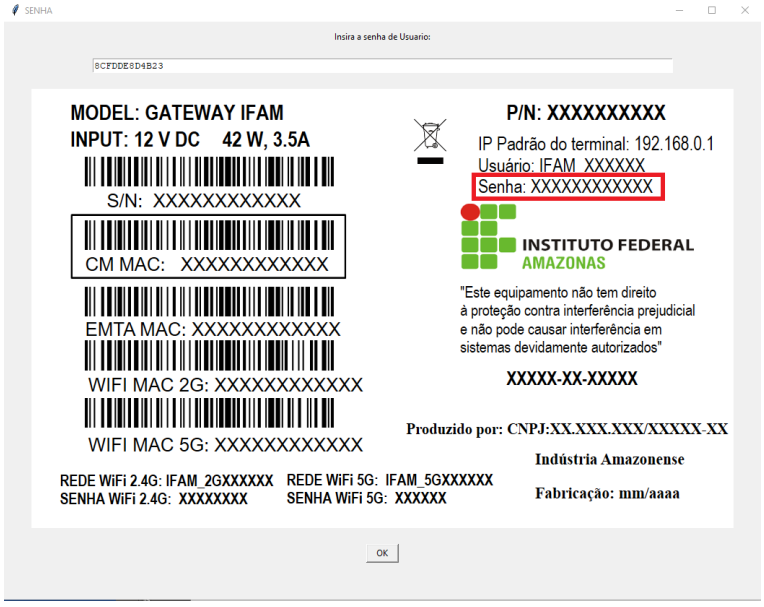
Figura 64 - Tela USUÁRIO



Fonte: O Próprio Autor (2021)

A tela da Figura 65 apresenta a leitura dos caracteres do SENHA da etiqueta.

Figura 65 - Tela SENHA



Fonte: O Próprio Autor (2021)

E após a finalização o teste reinicia podendo realizar novos testes.

6 CONCLUSÕES FINAIS

Este TCC teve como objetivo desenvolver um protótipo de um sistema inteligente para leitura de etiquetas e inserção em programas de testes. Para isso utilizou-se equipamentos disponíveis de baixo custo no mercado.

Foram feitos os testes com etiquetas de testes, tivemos diversos resultados positivos, porém como o mapeamento da leitura é no local definido, se o posicionamento estiver errado poderá diminuir a precisão da leitura. Outro ponto são as imprecisões na leitura, como as trocas da letra “O” pelo número “0” ou vice-versa.

Uma solução para diminuir esses erros, seria a utilização de uma câmera com maior resolução para quando realizar os cortes e a ampliação da imagem a ocorrência desses erros serem menores, pois a quantidade de pixel na imagem é maior, aumentando a fidelidade da imagem.

Fazendo a cronometragem do tempo de conclusão da tarefa manualmente como é feito hoje em dia na fábrica, temos por volta de dois minutos de trabalho de inserir os dados e de teste completo um tempo 10 minutos, como existem quatro postos de testes, um ciclo completo nos quatro postos leva em torno de 16 minutos, já com o sistema automático leva por volta de cinco segundos para fazer a captura e o processamento de dados. Levando mais 25 segundos para comparação dos dados mostrados e com a etiqueta do roteador, assim com um ganho de 75% no tempo da tarefa, o que seria possível iniciar 4 testes em postos diferentes durante os dois minutos iniciais. O que tornaria possível um ganho de 4 minutos em um ciclo de teste nos quatro postos.

O protótipo apresentando mostrou-se uma ótima alternativa para a diminuição do tempo necessário para completar a tarefa, assim realizando atualizações no código e melhorias no equipamento, pode até vir a se tornar um projeto aplicável na indústria.

APÊNDICE

ANEXO 1 – SCRIPT MAIN

```
from TCC import Projeto

def main():
    Projeto()
if __name__ == '__main__':
    main()
```

ANEXO 2 – SCRIPT PROJETO

```
import cv2
from tkinter import *
import PIL
import os
import numpy as np
import time
import random as rng
import pytesseract as tess
from PIL import ImageTk, Image
from time import sleep
import RPi.GPIO as GPIO

LED = 18
BOTAO = 17
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(BOTAO, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)
GPIO.setup(LED, GPIO.OUT)

video = cv2.VideoCapture(0)
video.set(cv2.CAP_PROP_FPS, 30.0)
video.set(cv2.CAP_PROP_FRAME_WIDTH, 1000)
video.set(cv2.CAP_PROP_FRAME_HEIGHT, 1000)

class Projeto:

    def __init__(self):

        self.chamar_telas_inicial()
        self.ler_etiqueta()
        self.chamar_telas()
        self.chamar_telas_final()
```

```
    def ligar_led(self):

        GPIO.output(LED, GPIO.LOW)

    def desligar_led(self):

        GPIO.output(LED, GPIO.HIGH)

    def chamar_telas_inicial(self):

        self.init()
        self.tela_teste_inicial("CLIQUE PARA INICIAR E CAPTURAR FOTO:",
"INÍCIO")

    def chamar_telas_final(self):

        self.tela_teste_inicial("TESTE FINALIZADO!", "FINALIZADO")

    def chamar_telas(self):

        self.tela_teste("Insira o SN:", "IFAM_SN.png","SN",SN)
        self.tela_teste("Insira o CM MAC:", "IFAM_CMMAC.png", "CM MAC",CM_MAC)
        self.tela_teste("Insira o EMTA MAC:","IFAM_EMTAMAC.png","EMTA
MAC",EMTA_MAC)
        self.tela_teste("Insira o WIFI MAC 24G:", "IFAM_WIFIMAC24G.png", "WIFI
MAC 24G",WIFI_MAC_24G)
        self.tela_teste("Insira o WIFI MAC 5G:", "IFAM_WIFIMAC5G.png", "WIFI
MAC 5G",WIFI_MAC_5G)
        self.tela_teste("Insira o nome da rede WIFI 2.4G:",
"IFAM_REDE24G.png", "REDE WIFI 2.4G",REDE_WIFI_24G)
        self.tela_teste("Insira a senha da rede WIFI 2.4G:",
"IFAM_SENHA24G.png", "SENHA REDE WIFI 2.4G",SENHA_REDE_WIFI_24G)
        self.tela_teste("Insira o nome da rede WIFI 5G:", "IFAM_REDE5G.png",
"REDE WIFI 5G",REDE_WIFI_5G)
        self.tela_teste("Insira a senha da rede WIFI 5G:", "IFAM_SENHA5G.png",
"SENHA REDE WIFI 5G",SENHA_REDE_WIFI_5G)
        self.tela_teste("Insira o nome do Usuario:", "IFAM_USUARIO.png",
"USUARIO",USUARIO)
        self.tela_teste("Insira a senha de Usuario:", "IFAM_SENHA.png",
"SENHA",SENHA)

    def ler_etiqueta(self):

        imagemModificada = cv2.imread('etiqueta_cortada.jpg')
        self.ler_serial(imagemModificada)
        self.ler_CM_MAC(imagemModificada)
        self.ler_EMTA_MAC(imagemModificada)
        self.ler_WIFI_MAC_24G(imagemModificada)
        self.ler_WIFI_MAC_5G(imagemModificada)
```

```
        self.ler_REDE_WIFI_24G(imagemModificada)
        self.ler_SENHA_REDE_WIFI_24G(imagemModificada)
        self.ler_REDE_WIFI_5G(imagemModificada)
        self.ler_SENHA_REDE_WIFI_5G(imagemModificada)
        self.ler_USUARIO(imagemModificada)
        self.ler_SENHA(imagemModificada)

    def tirar_foto(self):

        while(True):
            conectado, frame = video.read()
            cv2.imshow('Img',frame)
            if cv2.waitKey(1) & (GPIO.input(BOTAO) == 1):
                break
        cv2.imwrite('etiqueta.jpg', frame)
        self.desligar_led()
        video.release()
        cv2.destroyAllWindows()

    def cortar_etiqueta(self):

        imagemOriginal = cv2.imread("etiqueta.jpg")
        dim = (1000, 500)
        imagemModificada = cv2.resize(imagemOriginal, dim,
interpolation=cv2.INTER_AREA)
        imagemModificada = self.cortar_imagem(imagemModificada, 80, 880, 75,
385)
        cv2.imwrite('etiqueta_cortada.jpg', imagemModificada)

    def cortar_imagem(self,imagem,x,w,y,h):

        imagemMod = imagem[y:y + h, x:x + w]
        return (imagemMod)

    def init(self):

        self.desligar_led()
        while (GPIO.input(BOTAO) == 0):
            time.sleep(1)

        while True:
            if(GPIO.input(BOTAO) == 1):
                self.ligar_led()
                self.tirar_foto()
                self.cortar_etiqueta()
                self.desligar_led()
                break
            time.sleep(1)
        self.desligar_led()
```

```
        GPIO.cleanup()

    def ler_serial(self, imagemModificada):

        global SN

        IMGserial = self.cortar_imagem(imagemModificada, 75, 150, 85, 20) # x
+ w, y + h
        text = tess.image_to_string(IMGserial)
        tam = len(text)
        SN = text[tam-14:tam-2]

    def ler_CM_MAC(self, imagemModificada):

        global CM_MAC

        IMGcm_mac = self.cortar_imagem(imagemModificada, 120, 170, 135, 20)
        text = tess.image_to_string(IMGcm_mac)
        tam = len(text)
        CM_MAC = text[tam-14:tam-1]

    def ler_EMTA_MAC(self, imagemModificada):

        global EMTA_MAC

        IMGemta_mac = self.cortar_imagem(imagemModificada, 150, 170, 193, 15)
        text = tess.image_to_string(IMGemta_mac)
        tam = len(text)
        EMTA_MAC = text[tam-14:tam - 2]

    def ler_WIFI_MAC_24G(self, imagemModificada):

        global WIFI_MAC_24G

        IMGwifi_mac_24g = self.cortar_imagem(imagemModificada,210,160,240, 20)
        text = tess.image_to_string(IMGwifi_mac_24g)
        tam = len(text)
        WIFI_MAC_24G = text[tam-14:tam - 2]

    def ler_WIFI_MAC_5G(self, imagemModificada):

        global WIFI_MAC_5G

        IMGwifi_mac_5g = self.cortar_imagem(imagemModificada,207,160,295, 20)
        text = tess.image_to_string(IMGwifi_mac_5g)
        tam = len(text)
        WIFI_MAC_5G = text[tam-14:tam - 1]

    def ler_REDE_WIFI_24G(self, imagemModificada):
```

```
        global REDE_WIFI_24G

        IMGrede_wifi_24g = self.cortar_imagem(imagemModificada,220,195,315,20)
        text = tess.image_to_string(IMGrede_wifi_24g)
        tam = len(text)
        REDE_WIFI_24G = text[0:tam - 1]

    def ler_SENHA_REDE_WIFI_24G(self, imagemModificada):

        global SENHA_REDE_WIFI_24G

        IMGsenha_rede_wifi_24g = self.cortar_imagem(imagemModificada, 240,
110, 335, 20)  # x + w, y + h
        text = tess.image_to_string(IMGsenha_rede_wifi_24g)
        tam = len(text)
        SENHA_REDE_WIFI_24G = text[0:tam - 1]

    def ler_REDE_WIFI_5G(self, imagemModificada):

        global REDE_WIFI_5G

        IMGrede_wifi_5g = self.cortar_imagem(imagemModificada, 635, 195, 320,
20)  # x + w, y + h
        text = tess.image_to_string(IMGrede_wifi_5g)
        tam = len(text)
        REDE_WIFI_5G = text[0:tam - 1]

    def ler_SENHA_REDE_WIFI_5G(self, imagemModificada):

        global SENHA_REDE_WIFI_5G

        IMGsenha_rede_wifi_5g = self.cortar_imagem(imagemModificada,650, 110,
335, 20)
        text = tess.image_to_string(IMGsenha_rede_wifi_5g)
        tam = len(text)
        SENHA_REDE_WIFI_5G = text[0:tam - 1]

    def ler_USUARIO(self, imagemModificada):

        global USUARIO
        IMGusuario = self.cortar_imagem(imagemModificada, 630, 150, 55,20)
        text = tess.image_to_string(IMGusuario)
        tam = len(text)
        USUARIO = text[0:tam - 1]

    def ler_SENHA(self, imagemModificada):

        global SENHA
```

```
        global CM_MAC
        SENHA = CM_MAC

    def tela_teste(self, texto_inicial, nome_imagem, titulo, leitura):

        def bt_ok():
            janela.destroy()
        janela = Tk()
        janela.title(titulo)
        janela.geometry('1050x800')
        texto_inicio = Label(janela, text=texto_inicial)
        texto_inicio.pack(padx=0, pady=10)
        caixa_texto = Text(janela, height=1, width=100)
        caixa_texto.insert(INSERT, leitura)
        caixa_texto.pack(padx=0, pady=10)
        img = ImageTk.PhotoImage(Image.open(nome_imagem))
        etiqueta = Label(janela, image=img)
        etiqueta.pack(padx=0, pady=10)
        botao = Button(janela, text="   OK   ", command=bt_ok)
        botao.pack(padx=0, pady=10, side = 'top')
        janela.mainloop()

    def tela_teste_inicial(self, texto_inicial, titulo):

        def bt_ok():
            janela.destroy()
        janela = Tk()
        janela.title(titulo)
        janela.geometry('400x100')
        texto_inicio = Label(janela, text=texto_inicial)
        texto_inicio.pack(padx=0, pady=10)

        botao = Button(janela, text="   OK   ", command=bt_ok)
        botao.pack(padx=0, pady=10, side='top')
        janela.mainloop()

    def tela_teste_final(self, texto_inicial, titulo):

        def bt_ok():
            janela.destroy()

        janela = Tk()
        janela.title(titulo)
        janela.geometry('400x100')
        texto_inicio = Label(janela, text=texto_inicial)
        texto_inicio.pack(padx=0, pady=10)
        botao = Button(janela, text="   OK   ", command=bt_ok)
        botao.pack(padx=0, pady=10, side='top')
        janela.mainloop()
```

REFERÊNCIAS BIBLIOGRÁFICAS

JETBRAINS. ***Install PyCharm.*** Jetbrains, 22 maio 2021. Disponível em: <https://www.jetbrains.com/help/pycharm/installation-guide.html> . Acesso em 25 out. 2021.

BAUMGARTEN, Gustavo. **Sistema de visão industrial: descubra tudo o que eles podem fazer por você.** Pollux. 9 out. 2018. Disponível em : <https://pollux.com.br/blog/sistemas-de-visao-industriais-descubra-tudo-o-que-eles-podem-fazer-por-voce/>. Acesso em 30 out. 2021.

TESLA. **Sistema de Visão Industrial.** Tesla. Disponível em: <https://www.teslaautomacao.com.br/sistemas-de-visao-industrial.php>. Acesso em 5 nov. 2021.

VINCENT, Luc. ***Announcing Tesseract OCR.*** 30 ago. 2006. Disponível em: <http://googlecode.blogspot.com/2006/08/announcing-tesseract-ocr.html>. Acesso em: 5 nov. 2021.

UBUNTU. ***OCR - Optical Character Recognition.*** 31 mar. 2015.. Disponível em: <https://help.ubuntu.com/community/OCR> Acesso em 7 nov. 2021.

MORITOMO, Carlos. **A revolução do Raspberry Pi.** 16 Abr. 2012. Disponível em: <https://www.hardware.com.br/artigos/raspberrypi/> Acesso em 10 nov. 2021.

GOGONI, Ronaldo. **Raspberry Pi 4 Model B, tras 4k e 4GB de ram por US$55.** 2019. Disponível em <https://tecnoblog.net/meiobit/402892/raspberry-pi-4-model-b-ficha-tecnica/> Acesso em 25 nov. 2021.

IFRS CTA. **Ferramentas OCR – entenda o que são e sua relação com a acessibilidade**. 17 dez. 2018. Disponível em: <Ferramentas OCR – entenda o que são e sua relação com a acessibilidade - Centro Tecnológico de Acessibilidade do IFRS>. Acesso em: 27 de Dezembro de 2021.

SLAM, Norman; ISLAM, Zeeshan. NOOR, Nazia. ***A Survey on Optical Character Recognition System. ournal of Information & Communication Technology***, Nova York, volume X, ed.2, pág. 1-2, dezembro de 2016.

PYTHON, **HOME**. 2022. Disponível em: < https://www.python.org> Acesso em: 10 jan. 2022.

PYTHON, **tkinter — Interface Python para Tcl/Tk**. 2021. Disponível em: < https://docs.python.org/pt-br/dev/library/tkinter.html> Acesso em: 15 jan. 2022.

LOGITECH, **C270 HD WEBCAM.** Disponível em: https://www.logitech.com/pt-br/products/webcams/c270-hd-webcam.960-000694.html Acesso em: 22 maio 2022.

CHAUDHURI, Arindam et al. **Optical Character Recognition Systems for different languages with soft computing** – 1ª ed. Nova York: Springer International Publishing, 2017.

CUNHA, Renata. **Estudo comparativo sobre ferramentas de reconhecimento óptico de caracteres**. Dissertação (Graduação), Instituto Federal de Minas Gerais – Curso de Ciência da Computação, Minas Gerais, 2018.

MIRANDA, Ligia. **Estudo da viabilidade da construção de dispositivo de baixo custo para o reconhecimento de placas veiculares.** Dissertação (Graduação), Universidade Estadual Do Rio Grande Do Sul – Curso Superior De Tecnologia Em Automação Industrial, Porto Alegre, 2021.

JÚNIOR, Rômulo Pacher. **Localização de robôs por reconhecimento ótico de caracteres de placas.** Dissertação (Graduação), Universidade Federal de Santa Catarina, Santa Catarina, 2018.

PYPI. **pytesseract 0.3.10– Project description.** 2021. Disponível em:<https://pypi.org/project/pytesseract/>. Acesso em: 01 Jul 2021.

PILLOW. **Pillow 9.2.0.** 2021. Disponível em:< https://pypi.org/project/Pillow/>. Acesso em 01 Jul 2021.

BACKES, André Ricardo; JUNIOR, Jarbas Joaci de Mesquita Sá. **Introdução à visão computacional usando Matlab**. Rio de Janeiro: Alta Books Editora, 2019.

FONSECA, J. J. S. **Metodologia da pesquisa científica.** Fortaleza: UEC, 2002. Apostila.

BORGES, Luiz Eduardo. **Python para desenvolvedores.** 2° ed.

CORREA, Eduardo. **Meu Primeiro livro de Python.** 2° ed.

GERHARDT, Tatiana Engel; SILVEIRA, Denise Tolfo **Métodos de pesquisa**. – Porto Alegre: Editora da UFRGS, 2009 (GERHARDT; SILVEIRA, 2009, p.35).

CEDRO TECHNOLOGIES. **OpenCV: Uma breve introdução à visão computacional com python.** 3 out. 2018.

BARELLI, Felipe. Introdução à visão computacional. Casa do Código, 2018. 256 p.

ELKNER, Jeffrey; DOWNEY, Allen; MEYERS, Chris. ***How to Think Like a Computer Scientist: Interactive Edition.*** 2016. Disponível em: <https://runestone.academy/ns/books/published/thinkcspy/index.html>. Acesso em 1 set. 2022.

PEIXOTO,S. P et al. **Brazilian License Plate Character Recognition using Deep Learning**. Universidade Federal de Ouro Preto Ouro Preto, MG, Brazil. 2014.