THIAGO HENRIQUE SILVA BRUNO

SISTEMA AUTOMATIZADO PARA CONTROLE DE ACESSO DE VEÍCULOS

UNIFEV - CENTRO UNIVERSITÁRIO DE VOTUPORANGA

DEZEMBRO/2024

THIAGO HENRIQUE SILVA BRUNO

SISTEMA AUTOMATIZADO PARA CONTROLE DE ACESSO DE VEÍCULOS

Monografia apresentada à Unifev - Centro Universitário de Votuporanga - para a obtenção do grau de bacharel em Engenharia de Computação, sob a orientação do professor Renato Ferrari da Costa.

UNIFEV - CENTRO UNIVERSITÁRIO DE VOTUPORANGA

DEZEMBRO/2024

bruno, Thiago Henrique silva.
Sistema Automatizado para Controle de Acesso de Veículos. / Thiago Henrique silva bruno. - Votuporanga. Ed. do Autor, 2024.
66 p., 30cm.: il., tab.

Trabalho de Conclusão de Curso (Graduação - Bacharelado) - UNIFEV - Centro Universitário de Votuporanga, Curso de Engenharia de Computação, 2024.
Orientador: Prof. Esp. Renato Ferrari da Costa.

1.Automação de portão eletrônico. 2. Controle de acesso de veículos. 3. Processamento de imagem. 4. Pyton. 5. Reconhecimento de placas. I. Título.

Sistema de geração automática de fichas catalográficas da Unifev.
Dados fornecidos pelo autor(a).
Essa ficha não pode ser modificada.
Bibliotecária Responsável: Marcia Faria Cavalcante - CRB-8/ 10706

THIAGO HENRIQUE SILVA BRUNO

SISTEMA AUTOMATIZADO PARA CONTROLE DE ACESSO DE VEÍCULOS

Monografia apresentada à Unifev - Centro Universitário de Votuporanga - para a obtenção do grau de bacharel em Engenharia de Computação, sob a orientação do professor Renato Ferrari da Costa.

Aprovado:___/___/___

Primeiro Examinador
Nome:
Unifev – Centro Universitário de Votuporanga

Segundo Examinador
Nome:
Unifev – Centro Universitário de Votuporanga

Orientador
Renato Ferrari Costa
Unifev - Centro Universitário de Votuporanga

unifev

unifev nadoc
Núcleo de Acervo e Documentação

ATA DE REUNIÃO DE ANÁLISE E PARECER DEFINITIVOS DE TRABALHO DE CONCLUSÃO DO CURSO DE ENGENHARIA DA COMPUTAÇÃO

Aos dez dias do mês de dezembro de dois mil e vinte e quatro, às vinte horas e vinte minutos, realizou-se no local: LABORATORIO DE INFORMATICA I - CIDADE UNIVERSITÁRIA, do Centro Universitário de Votuporanga - Unifev, nas formas e termos regulamentais desta Instituição, a reunião de análise definitiva do Trabalho de Conclusão de Curso, intitulado SISTEMA AUTOMATIZADO PARA CONTROLE DE ACESSO DE VEÍCULOS de autoria de THIAGO HENRIQUE SILVA BRUNO. A sessão de defesa do trabalho deu-se sob o julgo da Banca Examinadora composta pelos docentes: Prof. Me. FERNANDO DATORRE, Prof. Dr. FERNANDO KENDY AOKI RIZZATTO E Prof. Especialista RENATO FERRARI DA COSTA, e presidida por Prof. Especialista RENATO FERRARI DA COSTA. Leram-se, inicialmente, as normas previstas no Regulamento de Trabalho de Conclusão de Curso, seguida da apresentação das alterações realizadas pelo autor(a), em conformidade com as recomendações previamente apontadas pela Banca Examinadora durante a sessão pública de defesa. Em seguida, os membros realizaram a análise das correções. Concluída a revisão e após deliberação reservada, o trabalho foi considerado **APROVADO**. O parecer, registrado de forma definitiva neste documento, tem complemento o que segue: Foi feito a correção conforme orientado pela banca, sendo considerados APROVADOS. Nada mais tendo a tratar, lavrou-se a presente ata que segue assinada pelos membros da banca e autores do trabalho.

Votuporanga, 10 de dezembro de 2024.

Assinaturas (4 pessoas)

Usuário	Função	Instituição	Lido em	Assinado em	IP
THIAGO HENRIQUE SILVA BRUNO	Aluno	UNIFEV	10/12/2024 20:11:15	10/12/2024 20:11:39	200.148.33.34
Fernando Datorre	Banca	UNIFEV	10/12/2024 20:11:09	10/12/2024 20:11:13	167.249.56.197
Fernando Kendy Aoki Rizzatto	Banca	UNIFEV	10/12/2024 20:32:13	10/12/2024 20:32:18	181.213.37.102
Renato Ferrari da Costa	Presidente	UNIFEV	10/12/2024 20:11:43	10/12/2024 20:12:35	191.26.151.248

***Banca realizada presencialmente e assinaturas coletadas via aceite eletrônico.**

Copyright © 2021-2024 - GEDEX UNIFEV - STI Unifev - Página 1 de 1

Ata criada digitalmente.
Para acessar aponte seu celular para o Código QR ao lado.

Dedico este trabalho a Tomori Kusunoki, cuja arte e determinação me deram forças para continuar um dia de cada vez. Em momentos de dificuldade sua dedicação e paixão foram uma fonte constante de inspiração, ajudando-me a superar os desafios e a manter o foco na realização deste projeto.

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema automatizado de controle de acesso de veículos, utilizando tecnologias de visão computacional, reconhecimento óptico de caracteres (OCR) e dispositivos IoT. O sistema foi concebido para otimizar o controle de entrada e saída de veículos, reduzindo a necessidade de intervenção manual e aprimorando a eficiência do processo. Os objetivos incluem a criação de um sistema capaz de detectar, processar e validar placas veiculares de forma automática.

A metodologia adotada abrangeu o uso de ferramentas como OpenCV e Tesseract OCR para processamento e reconhecimento de imagens, além da integração do dispositivo Sonoff PSF-B01 para automação do portão eletrônico. O projeto foi estruturado em etapas, desde a concepção da arquitetura até a implementação prática, com testes realizados em ambientes controlados.

Os resultados indicam que o sistema atendeu às expectativas, realizando o reconhecimento de placas com alto grau de precisão e demonstrando potencial para aplicação em locais como estacionamentos e condomínios. Conclui-se que a solução desenvolvida é funcional e escalável, sendo aplicável em cenários que demandam controle eficiente de acesso veicular.

Palavras chaves: Automação; Controle de acesso; IoT; OpenCV; Processamento de imagem; Python; Reconhecimento óptico de caracteres (OCR); Sonoff PSF-B01; Tesseract OCR; Visão computacional;

ABSTRACT

This paper presents the development of an automated vehicle access control system using computer vision, optical character recognition (OCR) and IoT devices. The system was designed to improve vehicle entry and exit control, requiring no manual intervention and improving process efficiency. The objectives include creating a system capable of automatically detecting, processing and validating vehicle license plates. The methodology adopted includes the use of tools such as OpenCV and Tesseract OCR for image processing and recognition, in addition to the integration of the Sonoff PSF-B01 device for electronic gate automation. The project was structured in stages, from the design of the architecture to practical implementation, with tests carried out in controlled environments. The results indicate that the system met expectations, performing license plate recognition with a high degree of accuracy and demonstrating potential for application in locations such as parking lots and condominiums. It is concluded that the developed solution is functional and scalable, being applicable in scenarios that require efficient vehicle access control.

Keywords: Automation; Access control; Computer vision; IoT; Image processing; OpenCV; Optical character recognition (OCR); Python; Sonoff PSF-B01; Tesseract OCR;

LISTA DE FIGURAS

Figura 1 - Diagrama de casos de uso 19
Figura 2 - Diagrama de casos especiais 20
Figura 3 - Um Sistema de Visão Artificial (SVA) e suas principais etapas. 24
Figura 4 - Especificações de Entradas Sonoff PSF-B01 30
Figura 5 - Linha de código aumento de contraste e brilho 34
Figura 6 - Linha de código conversão a escala de cinza 35
Figura 7 - Linha de código equalização do histograma 36
Figura 8 - Resultado da equalização do histograma 39
Figura 9 - Linha de código binarização 40
Figura 10 - Linha de código filtro de desfoque 43
Figura 11 - Resultado pré-processamento 44
Figura 12 - Linha de código deteccção de bordas 46
Figura 13 - Resultado detecção de bordas 47
Figura 14 - Linha de código extração de contornos 47
Figura 15 - Linha de código encontrar placa 48
Figura 16 - Resultado da busca pela placa 51
Figura 17 - Linha de código redimensionamento 52
Figura 18 - Linha de código limiarização 53
Figura 19 - Linha de código mediana 54
Figura 20 - Linha de código ajuste de contraste 54
Figura 21 - Resultado preparamento para leitura de caracteres 55
Figura 22 - Linha de código configuração OCR 56
Figura 23 - Linha de código limpeza de texto 56
Figura 24 - Linha de código conexão com o banco 57
Figura 25 - Linha de código consulta pela placa 57
Figura 26 - Linha de código extrair dados referente a placa 58
Figura 27 - Linha de código converter horário 58
Figura 28 - Linha de código verificar permissão 59
Figura 29 - Linha de código registrar eventos 59
Figura 30 - Linha de código instanciar webdriver 60
Figura 31 - Linha de código efetuar login 61
Figura 32 - Linha de código fechar notificação 62

Figura 33 - Linha de código acionar dispositivo 63
Figura 34 - Especificações de entradas da placa AGL 65
Figura 35 - Integração do Sonoff a placa AGL 66

LISTA DE TABELAS

Tabela 1 - Objetivos Funcionais 16
Tabela 2 - Objetivos Não Funcionais 17
Tabela 3 - Níveis Processamento de Imagem 23

LISTA DE ABREVIATURAS E SIGLAS

CDF	Função de Distribuição Acumulada (Cumulative Distribution Function)
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
OCR	Reconhecimento Óptico de Caracteres (Optical Character Recognition)
OpenCV	Open Source Computer Vision Library
PNG	Portable Network Graphics
RGB	Red, Green, Blue
SVC	Sistemas de Visão Computacional (Computer Vision Systems)
Tesseract	Software de Reconhecimento Óptico de Caracteres (OCR)
XML	Extensible Markup Language
IoT	Internet das Coisas (Internet of Things)

LISTA DE PALAVRAS ESTRANGEIRAS

CSV	Valores Separados por Vírgula (Comma Separated Values)
Git	Sistema de Controle de Versão Distribuído (Distributed Version Control System)
IoT	Internet das Coisas (Internet of Things)
JPEG	Grupo Conjunto de Especialistas em Imagens (Joint Photographic Experts Group)
JSON	Notação de Objetos JavaScript (JavaScript Object Notation)
OCR	Reconhecimento Óptico de Caracteres (Optical Character Recognition)
OpenCV	Biblioteca de Visão Computacional de Código Aberto (Open Source Computer Vision Library)
PNG	Gráfico de Rede Portátil (Portable Network Graphics)
RGB	Vermelho, Verde e Azul (Red, Green, Blue)
SQLite	Banco de Dados Relacional de Código Aberto (Structured Query Language Lite)
SVC	Sistemas de Visão Computacional (Computer Vision Systems)
XML	Linguagem de Marcação Extensível (Extensible Markup Language)

LISTA DE EQUAÇÕES

Equação 1 - Fórmula ajuste de contraste e brilho 33
Equação 2 - Fórmula conversão a escala de cinza 35
Equação 3 - Fórmula de equalização do histograma 37
Equação 4 - Fórmula distribuição acumulada 37
Equação 5 - Fórmula normalização 38
Equação 6 - Fórmula transformação nos pixels 38
Equação 7 - Fórmula média local 41
Equação 8 - Fórmula função gaussiana 41
Equação 9 - Fórmula media local 42
Equação 10 - Fórmula filtro gaussiano 45
Equação 11 - Fórmula derivadas sobel 45
Equação 12 - Fórmula de G e θ 45
Equação 13 - Condução de gradiente 46
Equação 14 - Fórmula perímetro do contorno 49
Equação 15 - Fórmula tolerância 49
Equação 16 - Fórmula aspect ratio 50
Equação 17 - Fórmula área do contorno 50
Equação 18 - Fórmula dimensão da nova imagem 51
Equação 19 - Fórmula da interpolação bicúbica 52
Equação 20 - Fórmula peso função de interpolação cúbica 53

SUMARIO

1 INTRODUÇÃO 12

1.1OBJETIVOS 13
1.1.1 OBJETIVOS GERAIS 13
1.1.2 OBJETIVOS ESPECÍFICOS 13
1.2 JUSTIFICATIVA 15
1.3 METODOLOGIA 15
1.3.1 ANÁLISE DE REQUISITOS FUNCIONAIS 16
1.3.2 ANÁLISE DE REQUISITOS NÃO FUNCIONAIS 17
1.3.3 DIAGRAMA DE CASOS DE USO 18
1.3.4 DIAGRAMA DE CASOS DE USO EXPANDIDO 19

2 REFERENCIAL TEÓRICO 21

2.1 VISÃO COMPUTACIONAL 21
2.2 IMAGENS DIGITAIS 22
2.3 PROCESSAMENTO DE IMAGENS 22
2.4 AUTOMAÇÃO 25

3 FERRAMENTAS UTILIZADAS 27

3.1 BIBLIOTECA OPENCV 27
3.2 BIBLIOTECA TESSERACT OCR 27
3.3 BIBLIOTECA SQLITE3 28
3.4 BIBLIOTECA TKINTER 29
3.5 SONOFF WI-FI RELÉ 29
3.6 BIBLIOTECA SELENIUM 30

4 DESENVOLVIMENTO 32

4.1 AQUISIÇÃO DAS IMAGENS 32
4.2 PRÉ-PROCESSAMENTO DAS IMAGENS 33
4.2.1 AJUSTE DE CONTRASTE E BRILHO 33
4.2.2 CONVERSÃO PARA ESCALA DE CINZA 34
4.2.3 EQUALIZAÇÃO DO HISTOGRAMA 36
4.3 BINARIZAÇÃO 40
4.4 FILTRO DE DESFOQUE 43
4.5 DETECÇÃO DE BORDAS 44
4.6 EXTRAÇÃO DE CONTORNOS 47
4.7 PREPARAR PARA LEITURA DE CARACTERES 51

4.8 LEITURA DE CARACTERES 55
4.9 VALIDAÇÃO DA PLACA 57
4.10 ABERTURA DO PORTÃO 60
4.11 AUTOMAÇÃO DO PORTÃO 64
5 CONCLUSÃO 67
REFERÊNCIAS 68

1 INTRODUÇÃO

Com o avanço tecnológico, a automação tem se tornado algo comum e quase indispensável na atualidade em diversos contextos. Seja na indústria, no comércio ou em residências. A automação oferece uma série de benefícios, como aumento da eficiência, redução de custos, maior agilidade e a independência de intervenção humana nos processos.

De acordo com a ERPO (2023), o controle de acesso automatizado de veículos obedece à tendencia de inovação de processos. Em ambientes como estacionamentos, condomínios, empresas e áreas restritas, a monitorização e controle da entrada e saída de veículos são essenciais para garantir a segurança, a organização e a eficiência nesses ambientes.

Este trabalho surge no contexto desse avanço tecnológico, propondo o desenvolvimento de um sistema automatizado de controle de acesso de veículos, através da integração de técnicas de captação de imagem, reconhecimento de placas de veículos e registro em um sistema de banco de dados, busca-se criar uma solução eficiente e confiável para o gerenciamento de entrada e saída de veículos, permitindo ou restringindo o acesso a locais específicos.

O trabalho não se limita apenas à automação do controle de acesso, mas também busca contribuir para a melhoria da agilidade, otimização dos processos e redução dos custos operacionais associados à gestão de acesso de veículos em ambientes controlados.

1.1 Objetivos

O presente trabalho busca desenvolver um sistema automatizado de controle de acesso de veículos, utilizando técnicas de processamento de imagem e dispositivos de Internet das Coisas (IoT) para integrar identificação de placas veiculares, validação de acesso e acionamento automático de portões eletrônicos.

A estrutura do projeto foi organizada para atender a diferentes níveis de requisitos. Inicialmente, são estabelecidos os objetivos gerais, que definem a finalidade e o impacto esperado do sistema. Em seguida, os objetivos específicos detalham as etapas técnicas necessárias para o desenvolvimento e a implementação da solução proposta.

1.1.1 Objetivos gerais

A proposta deste trabalho é desenvolver um sistema automatizado de controle de acesso de veículos, utilizando reconhecimento de placas com uma integração com banco de dados, e integrando um dispositivo IoT (Sonoff PSF-B01) com a plataforma eWeLink, para que o portão seja aberto automaticamente quando um veículo autorizado for identificado, visando proporcionar uma solução eficiente para o gerenciamento de entrada e saída de veículos em ambientes controlados.

1.1.2 Objetivos específicos

Este projeto tem como objetivo principal o desenvolvimento de um sistema automatizado de controle de acesso de veículos, que visa otimizar o processo de entrada e saída em áreas controladas para melhorar a praticidade e agilidade desse processo. Para atingir esse objetivo, o projeto foi dividido em etapas técnicas que abrangem desde o reconhecimento de

placas de veículos até a integração do sistema com um portão eletrônico, assegurando a automação do processo de abertura com base na autorização da placa.

A primeira fase envolve a implementação do reconhecimento de placas, que utiliza técnicas de processamento de imagem para identificar as placas de veículos a partir de imagens capturadas. O sistema utiliza algoritmos de visão computacional, como o OpenCV, para segmentar as placas e aplicar o OCR (Reconhecimento Óptico de Caracteres) utilizando o Tesseract. O objetivo dessa etapa é garantir que o sistema seja capaz de identificar as placas de veículos em diversas condições, como variações de iluminação e diferentes tipos de placas. Para isso, o código foi otimizado para lidar com diferentes contrastes, cores de placas e condições de visibilidade, garantindo robustez no processo de detecção.

Após a identificação da placa, a próxima etapa é a integração com uma base de dados, na qual as informações sobre as placas de veículos autorizados são armazenadas. A verificação é feita ao comparar a placa detectada com as registradas na base de dados, utilizando uma consulta eficiente para garantir que o veículo tenha acesso permitido. Esta fase exige a implementação de um sistema de consulta de banco de dados que possibilite a rápida verificação da placa, de modo a garantir uma resposta quase em tempo real ao sistema.

A automação do portão eletrônico, que utiliza o dispositivo Sonoff PSF-B01, constitui uma parte crítica do sistema. Como o dispositivo depende da plataforma eWeLink para controle, e esta plataforma não oferece suporte direto a requisições HTTP, a solução encontrada foi automatizar a interação com a interface web da plataforma eWeLink. Para isso, foi utilizada a ferramenta Selenium, que permite simular as ações do usuário no navegador, como login na plataforma, navegação até o dispositivo Sonoff e a execução do comando para abrir o portão. O Selenium interage com a página da web por meio de comandos de clique e envio de dados, como o preenchimento de campos de login e validação de controle do dispositivo, realizando assim a ação de abrir o portão quando o veículo identificado for autorizado.

Por fim o projeto incluiu uma fase de validação e testes do sistema em condições reais de operação. O objetivo era verificar a eficácia do sistema em um cenário real, testando a precisão do reconhecimento das placas e o tempo de resposta do sistema para acionar o portão.

1.2 Justificativa

Diante da necessidade que existe em vários segmentos comerciais e até residências de se controlar o fluxo e o acesso de veículos a locais restritos, foram criados os métodos tradicionais, que, basicamente se constituem de ações manuais realizadas por pessoas.

Além de o trabalho manual ser suscetível a falhas, atrasos, imprecisão, vale destacar a possibilidade da má-conduta do indivíduo que opera o sistema além dos custos associados e outras possíveis falhas inerentes à atividade humana.

A automação do controle de acesso, embora também suscetível a eventuais falhas, apresenta vantagens significativas em termos de precisão, eficiência e custo-benefício.

Segundo a TOTVS (2024), sistemas automatizados de controle de entrada e saída não só melhoram a gestão, mas também minimizam impactos negativos na produtividade e reduzem custos operacionais.

De acordo com a SISPONTO (2024) essa eficiência operacional gerada pela automação do controle de acesso não só melhora a experiência dos usuários, como também otimiza a gestão do local, proporcionando um fluxo contínuo e organizado de veículos.

Portanto, a implementação de um sistema automatizado de controle de acesso não apenas resolve as fragilidades associadas aos métodos tradicionais, mas também oferece um meio sustentável de melhorar a experiência dos usuários, organizar fluxos de veículos e aumentar a confiabilidade do controle de áreas restritas.

1.3 Metodologia

Adotou-se uma abordagem estruturada e sequencial. Inicialmente, foi realizado um planejamento detalhado, com o levantamento dos requisitos funcionais e técnicos do sistema. Em seguida, definiu-se a arquitetura geral do projeto, dividindo-o em etapas claras, como processamento de imagens, controle de acesso e automação.

A metodologia seguiu uma lógica iterativa, permitindo ajustes ao longo do desenvolvimento conforme os testes realizados. Essa abordagem possibilitou integrar de forma coesa todas as etapas necessárias para alcançar os objetivos propostos, desde a captura das imagens até a validação do acesso e o registro de eventos no sistema.

1.3.1 Análise de requisitos funcionais

Os objetivos funcionais do sistema referem-se às funcionalidades essenciais que garantem o funcionamento do controle de acesso, como a captura de imagem, o processamento da placa e a abertura do portão. Esses objetivos estão detalhados na Tabela 1.

Tabela 1 - Objetivos Funcionais

Objetivo	Descrição
Captura da Imagem da Placa	O sistema deve ser capaz de capturar a imagem da placa de um veículo quando ele se aproxima do portão, utilizando uma câmera ou dispositivo de captura de imagem adequado.
Processamento e Reconhecimento da Placa	Após a captura da imagem, o sistema deve ser capaz de processá-la e extrair os caracteres da placa utilizando técnicas de visão computacional e OCR (Reconhecimento Óptico de Caracteres).
Validação da Placa no Banco de Dados	O sistema deve consultar um banco de dados contendo as placas autorizadas e validar se a placa identificada está registrada para permitir o acesso ao local. A validação deve ser feita em tempo real e retornar rapidamente se a placa é autorizada ou não.
Abertura Automática do Portão	Caso a placa seja autorizada, o sistema deve acionar automaticamente o mecanismo de abertura do portão eletrônico, permitindo o acesso do veículo ao local. Esse processo deve ser rápido e sem intervenção manual.
Registro de Entrada/Saída	O sistema deve registrar todos os eventos de entrada e saída dos veículos, incluindo a data, horário e identificação da placa, para garantir o controle e a segurança do fluxo de veículos.
Gerenciamento de Placas Autorizadas	O administrador deve ser capaz de adicionar, editar ou remover placas autorizadas no banco de dados por meio de uma interface administrativa. A modificação dos registros de placas deve ser fácil e segura.

Fonte: Autor (2024)

1.3.2 Análise de requisitos não funcionais

Por outro lado, os objetivos não funcionais tratam das características do sistema que influenciam sua performance, como desempenho, escalabilidade e confiabilidade. Esses aspectos são apresentados na Tabela 2.

Tabela 2 - Objetivos Não Funcionais

Objetivo	Descrição
Desempenho	O sistema deve ser capaz de realizar todas as operações, como captura de imagem, processamento de placas, validação no banco de dados e abertura do portão, em tempo real, com baixa latência, garantindo que o processo de acesso seja concluído rapidamente.
Usabilidade	A interface do administrador, que permite o gerenciamento das placas autorizadas, deve ser intuitiva e fácil de usar. O administrador deve ser capaz de realizar as operações de forma rápida e sem necessidade de treinamento extensivo.
Escalabilidade	O sistema deve ser capaz de lidar com uma grande quantidade de placas autorizadas no banco de dados, sem degradação de desempenho. Além disso, a arquitetura do sistema deve permitir futuras expansões, caso seja necessário incluir novos recursos ou aumentar a capacidade de processamento.
Confiabilidade	O sistema deve ser altamente confiável, com baixos índices de falhas. Deve ser capaz de operar de forma contínua e estável, com mecanismos de recuperação em caso de falhas no processamento ou no acesso ao banco de dados.
Manutenibilidade	O sistema deve ser projetado de maneira modular, para que seja fácil realizar atualizações, corrigir falhas e adicionar novos recursos. A documentação do código e da arquitetura do sistema deve ser clara, para facilitar a manutenção e o suporte técnico.
Eficiência Energética	O sistema deve ser eficiente no uso de recursos, como energia e processamento, para garantir que funcione de maneira sustentável e econômica, principalmente em ambientes de operação contínua.

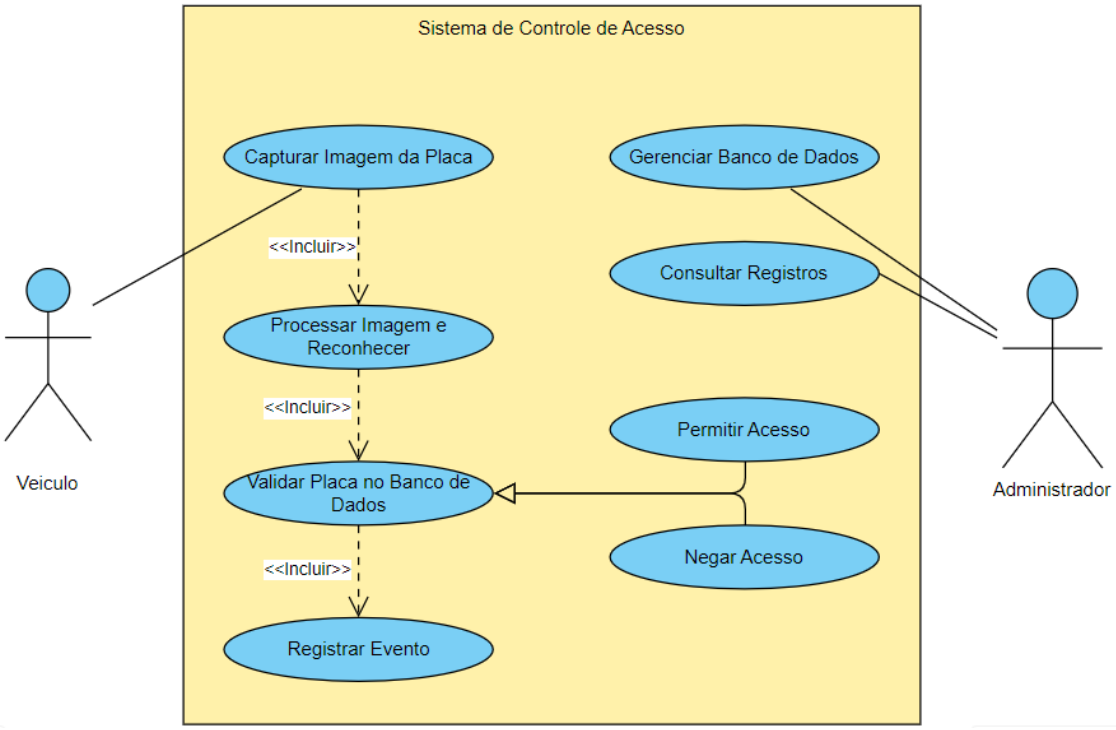
Fonte: Autor (2024)

1.3.3 Diagrama de casos de uso

O Diagrama de Casos de Uso apresentado na Figura 1 descreve os principais casos de uso do sistema de controle automatizado de acesso de veículos. O objetivo desse sistema é automatizar a entrada de veículos em uma área restrita, com base no reconhecimento da placa do veículo. O processo começa quando o veículo se aproxima do portão e o sistema captura a imagem da placa. A partir daí, o sistema processa e reconhece a placa, válida sua autorização no banco de dados e registra o evento no sistema. Dependendo da validação, o acesso é permitido ou negado.

O administrador é responsável por gerenciar o banco de dados das placas autorizadas e consultar os registros de acessos realizados, garantindo a manutenção e a segurança do sistema. O diagrama a seguir ilustra como os atores, veículos e administradores, interagem com os principais casos de uso dentro do sistema.

Figura 2 - Diagrama de casos de uso

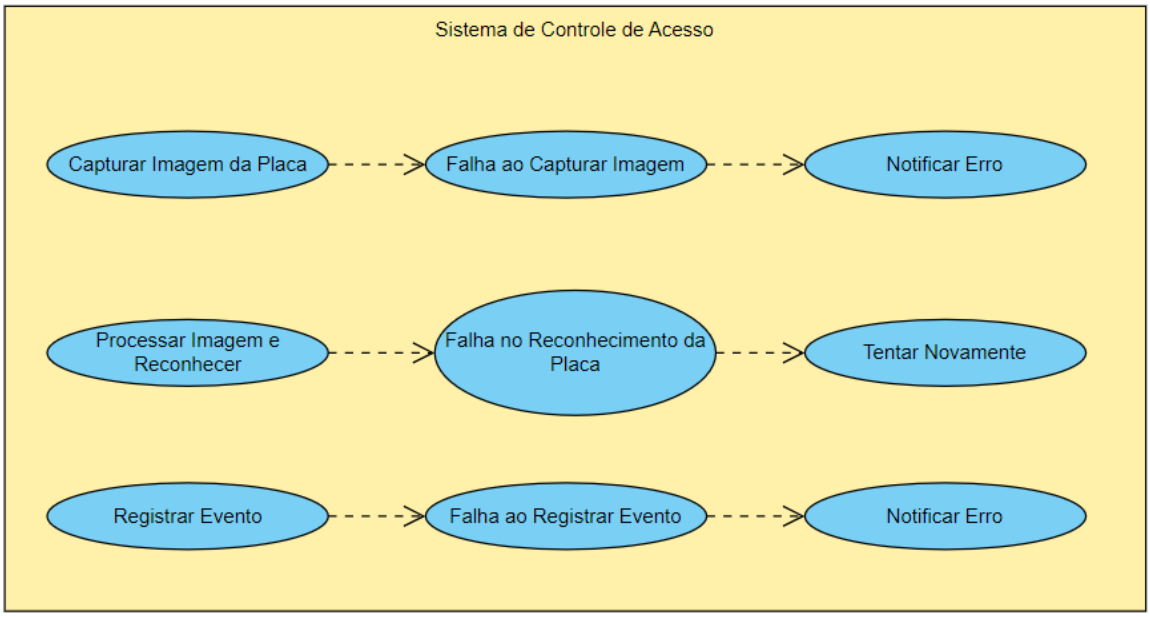


Fonte: Autor (2024)

1.3.4 Diagrama de Casos de Uso Expandido

Além dos casos de uso principais, também é importante considerar os casos de uso alternativos e excepcionais que podem ocorrer durante a operação do sistema de controle de acesso. Esses casos envolvem cenários onde o sistema precisa lidar com falhas, como a não captura da imagem, o erro de reconhecimento da placa ou a falta de correspondência no banco de dados conforme ilustrado na Figura 2.

Figura 3 - Diagrama de casos especiais



Fonte: Autor (2024)

2 REFERENCIAL TEÓRICO

O referencial teórico é uma das seções mais importantes de um trabalho acadêmico, pois oferece a base conceitual e científica necessária para a compreensão do tema estudado. Ele envolve a pesquisa e análise de estudos, teorias e conceitos existentes na área de interesse, fornecendo o embasamento para o desenvolvimento da pesquisa. Segundo Gil (2017), o referencial teórico tem como objetivo embasar teoricamente o estudo, orientando a definição de hipóteses e a escolha da metodologia mais adequada.

2.1 Visão Computacional

O conceito de SVC (Sistemas de Visão Computacional) surgiu na década de 1960, quando os primeiros experimentos começaram a explorar como as máquinas poderiam interpretar imagens, mas foi só quatro décadas depois em 2000 que o processo pode ser colocado em aplicações práticas como reconhecimento de objetos e sistemas de vigilância.

Esse progresso foi impulsionado pela disseminação da internet, que forneceu grandes volumes de imagens para análise, e pelos avanços no poder computacional, como indicado por Gonzalez e Woods (2008). Desde então, a área se expandiu rapidamente, com aplicações que vão desde o reconhecimento facial até a análise de imagens médicas e sistemas de veículos autônomos.

No contexto do projeto, a visão computacional é a nossa base para identificar a região da placa do veículo na imagem captada, extraí-la e prepará-la para o reconhecimento óptico dos caracteres.

2.2 Imagens Digitais

Imagens Digitais são representações de uma imagem bidimensional na forma de números binários ou seja 0 e 1. Elas são compostas por pixels, que são pequenas unidades de informação visual organizadas em uma matriz bidimensional de pontos f (x, y) de bits ou binários (0, 1). Cada pixel possui informações de cor e intensidade armazenadas em bits, que determinam a profundidade de cor da imagem. Quanto maior for a quantidade de bits na imagem mais fácil será representar um espectro mais amplo de cores ou níveis de cinza segundo (Gonzalez & Woods, 2008).

Para capturar imagens digitais, dispositivos como câmeras utilizam sensores que convertem a luz refletida pelos objetos em sinais elétricos, os quais são processados para gerar dados digitais. A qualidade da imagem depende da resolução que é medida em pixels por polegada, e do formato de compressão utilizado como JPEG (*Joint Photographic Experts Group*) ou PNG (*Portable Network Graphics*), que influenciam tanto a qualidade visual quanto o tamanho do arquivo conforme afirmado por (Jähne, 2005).

2.3 Processamento de Imagens

O processamento de imagens é uma subárea da ciência da computação que envolve a manipulação e análise de imagens digitais para extrair informações relevantes ou melhorar a qualidade visual.

Ele se aplica a uma ampla gama de áreas, como diagnóstico médico, segurança, inspeção industrial e visão computacional, sendo também uma base essencial para a confecção deste projeto.

De acordo com Gonzalez e Woods (2008) o processamento de imagens digitais, o conceito pode ser dividido em três níveis principais conforme mostra a Tabela 3.

Tabela 3 - Níveis Processamento de Imagem

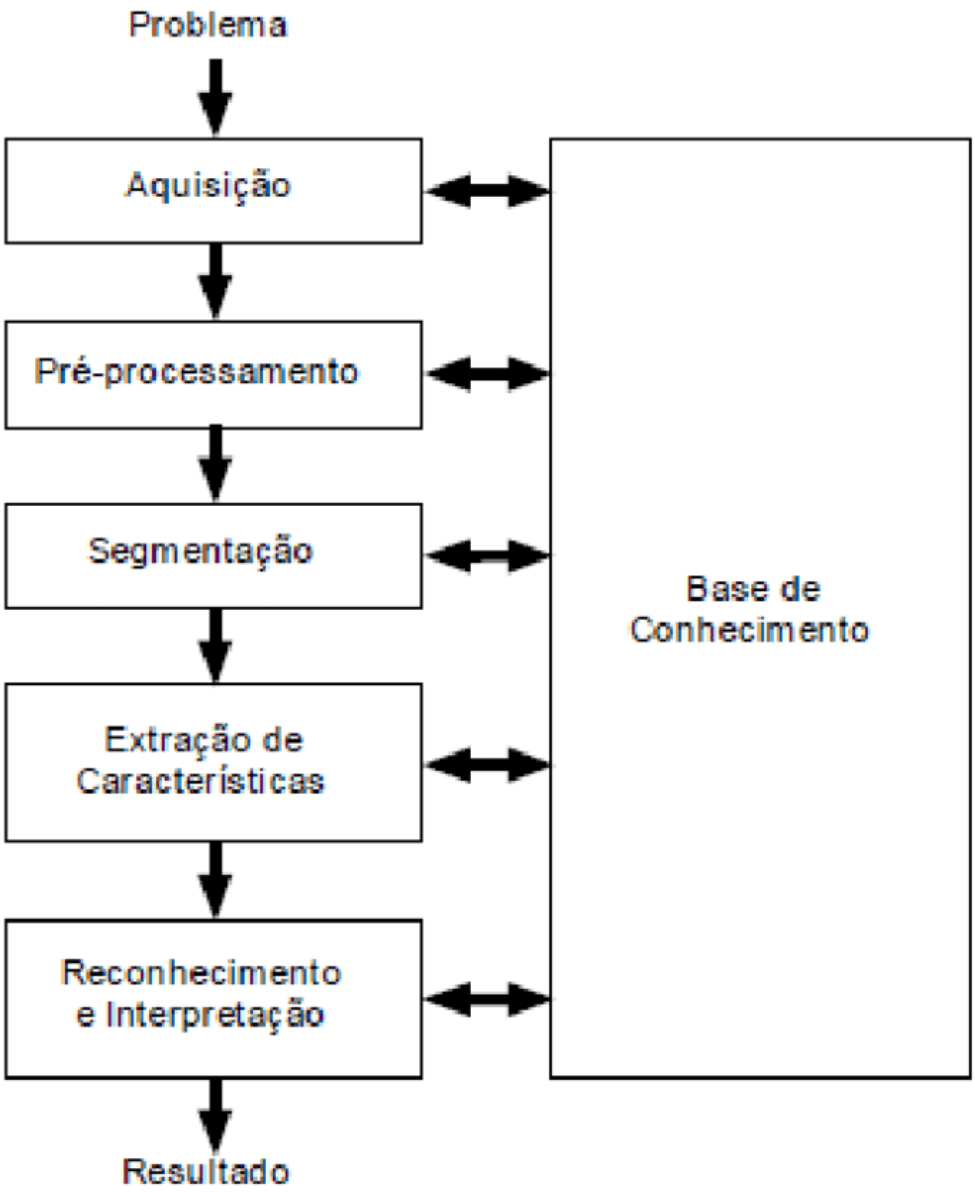
Nível	Descrição
Baixo nível	Realiza operações diretas sobre os pixels da imagem, como remoção de ruído, ajuste de contraste ou melhoria da nitidez, onde o processo de entrada e saída são sempre imagens.
Médio nível	Abrange tarefas como segmentação de regiões, detecção de bordas e extração de características específicas, permitindo que áreas ou objetos de interesse sejam destacados, onde a entrada é uma imagem e a saída um objeto extraído dessa imagem.
Alto nível:	O foco aqui está na interpretação dos dados extraídos para tomar decisões ou realizar ações específicas, como o reconhecimento de objetos ou rastreamento de movimento. Neste contexto os dados obtidos são exibidos na saída do sistema e representam características extraídas da imagem de sua entrada.

Fonte: Autor (2024)

No modelo de processamento de imagens digitais, cada etapa desempenha um papel crucial, compartilhando informações e resultados entre si. Essa interdependência cria um fluxo contínuo, onde os dados transformados em uma etapa servem de entrada para a próxima, garantindo um processo integrado que culmina na análise ou execução de ações específicas.

A Figura 3 ilustra o modelo de visão computacional empregado para a detecção e análise de imagens no sistema de reconhecimento de placas.

Figura 4 - Um Sistema de Visão Artificial (SVA) e suas principais etapas.



Fonte: (MARQUES FILHO, & VIEIRA NETO, 1999)

- Aquisição - A etapa inicial consiste na captura da imagem, onde dispositivos como câmeras convertem a luz refletida pelos objetos em sinais elétricos os quais são processados para gerar dados digitais. Isso envolve a amostragem e quantização, que transformam a imagem capturada em uma matriz de pixels representada numericamente.

- Pré-Processamento - O pré-processamento é responsável por preparar a imagem para análises mais avançadas, corrigindo ruídos ou imperfeições que possam prejudicar o desempenho nas etapas seguintes, utilizando para isso filtros como média e mediana para eliminar imperfeições, ajustes de contraste para melhor visibilidade e correções geométricas para ajustar distorções na imagem causadas por ângulos de captura ou deformações ópticas.
- Segmentação - Aqui, a imagem é dividida em regiões de interesse, separando objetos ou áreas específicas do fundo utilizando de limiarização que define um ponto de corte para destacar áreas escuras ou claras e, detecção de bordas para identificar contornos baseados em mudanças abruptas de intensidade.
- Extração de Características - Nesta etapa, informações relevantes como formas, texturas ou padrões específicos são retiradas das imagens segmentadas. Essa etapa é crítica para converter informações visuais em dados numéricos ou simbólicos que possam ser interpretados.
- Reconhecimento e Interpretação - Os dados extraídos são processados por algoritmos de reconhecimento para identificar objetos ou padrões. Técnicas de OCR (Reconhecimento Óptico de Caracteres) são aplicadas nesta etapa para transformar imagens de texto em dados digitais que possam ser processados e comparados.
- Pós-Processamento e Decisão - Após o reconhecimento, as informações são processadas para realizar uma ação ou armazenar os resultados.

Cada etapa é essencial para garantir a precisão no reconhecimento e a eficiência na execução das ações subsequentes conforme descrito por SCURI (2024).

2.4 Automação

A automação refere-se à aplicação de sistemas e tecnologias que permitem a execução de tarefas de maneira autônoma, sem a necessidade de intervenção humana, especialmente em processos repetitivos ou complexos. A automação tem sido uma parte integral da evolução

industrial e tecnológica, revolucionando diversos setores ao melhorar a eficiência e a precisão, ao mesmo tempo que reduz os custos operacionais.

O conceito de automação remonta à Revolução Industrial do século XVIII, com a introdução de máquinas a vapor que impulsionaram a produção em massa. Com o avanço da eletrônica e da computação no século XX, surgiram dispositivos mais sofisticados, como os Controladores Lógicos Programáveis (CLPs), que possibilitaram o controle de processos industriais de forma eficiente e mais precisa. No entanto, foi apenas nas últimas décadas, com o desenvolvimento da Internet das Coisas (IoT) e Inteligência Artificial (IA), que a automação atingiu novos patamares. Hoje, sistemas inteligentes são capazes de aprender e se adaptar em tempo real, automatizando tarefas em diversos contextos, como cidades inteligentes e automação residencial, como estacionamentos, condomínios e empresas, a automação oferece diversas vantagens. De acordo com SISPONTO (2024), ela proporciona maior eficiência operacional, melhorando a experiência dos usuários e a gestão do local. A automação do controle de acesso de veículos, por exemplo, facilita a organização do fluxo de veículos e aumenta a segurança, pois elimina a necessidade de intervenção humana, minimizando erros e fraudes.

A aplicação da automação no controle de acesso de veículos tem como foco a precisão, agilidade e confiabilidade. Utilizando tecnologias como reconhecimento de placas por meio de visão computacional, o sistema automatizado não só válida a autenticidade das placas, mas também registra a entrada e a saída dos veículos, otimizando os processos operacionais. O uso de dispositivos IoT, como o Sonoff PSF-B01, para controlar eletronicamente a abertura e o fechamento de portões, permite um controle mais ágil e sem falhas. Como ERPO (2023) aponta, a automação do controle de acesso tem se tornado cada vez mais essencial em ambientes com alta demanda de veículos, contribuindo significativamente para a segurança e organização.

Essa evolução tecnológica no campo da automação também reflete uma tendência crescente de integração de sistemas inteligentes.

3 FERRAMENTAS UTILIZADAS

Este capítulo descreve as principais ferramentas empregadas no desenvolvimento do sistema automatizado de controle de acesso de veículos, abordando as tecnologias essenciais para o processamento de imagens, reconhecimento de caracteres e automação do processo. A escolha dessas ferramentas foi orientada pelas necessidades do projeto, visando eficiência e precisão.

3.1 Biblioteca OpenCV

De acordo com a documentação oficial do OpenCV para Python (OpenCV Documentation, 2024) o OpenCV (*Open Source Computer Vision Library*), é uma biblioteca utilizada no campo de visão computacional e processamento de imagens. Desenvolvido inicialmente pela Intel em 1999, o OpenCV foi projetado para oferecer ferramentas eficientes e de alto desempenho para tarefas como detecção, rastreamento e análise de imagens. Atualmente, é mantido como um projeto de código aberto disponível sob a licença BSD (Berkeley Software Distribution), permitindo seu uso em aplicações acadêmicas e comerciais.

No projeto, o OpenCV foi utilizado para efetuar todo o processo de pré-processamento das imagens obtidas para efetuarmos o reconhecimento da placa do veículo.

3.2 Biblioteca Tesseract OCR

Segundo a documentação oficial do Tesseract OCR (Tesseract OCR Documentation, 2024) o Tesseract é uma ferramenta de OCR (Reconhecimento Óptico de Caracteres) de código aberto utilizada para extrair texto de imagens. Desenvolvido originalmente pela Hewlett-Packard (HP) na década de 1980, o projeto foi posteriormente mantido pelo Google a partir de

2006, sendo hoje um dos motores de OCR mais populares devido à sua precisão e adaptabilidade.

No projeto, o Tesseract OCR é utilizado para reconhecer os caracteres presentes nas placas dos veículos previamente detectadas, recortadas e, pré-processadas pelo OpenCV.

3.3 Biblioteca SQLite3

A biblioteca SQLite3 é uma interface nativa do Python para o sistema de gerenciamento de banco de dados relacional SQLite (Structured Query Language Lite). Ela permite que desenvolvedores utilizem os recursos do SQLite diretamente em seus projetos Python de maneira simples e eficiente, sem a necessidade de instalar bibliotecas externas, já que o SQLite3 faz parte da biblioteca padrão do Python desde a versão 2.5.

De acordo com a documentação do SQLite (2024) o SQLite é um sistema de gerenciamento de banco de dados relacional RDBMS (Relational Database Management System) de código aberto, conhecido por sua simplicidade, eficiência e portabilidade. Ele foi projetado para fornecer uma solução compacta e leve, ideal para aplicações embarcadas, sistemas locais e projetos onde um servidor de banco de dados completo não é necessário. Desde sua introdução em 2000, o SQLite tem sido amplamente adotado em diversas áreas, incluindo dispositivos móveis, navegadores de internet, sistemas embarcados e ferramentas de automação.

O SQLite não segue o modelo tradicional de cliente-servidor. Em vez disso, ele opera diretamente no sistema de arquivos do dispositivo onde é executado. Cada banco de dados é armazenado em um único arquivo, que contém tabelas, índices e os dados propriamente ditos, além de metadados e o esquema do banco.

3.4 Biblioteca Tkinter

Na biblioteca oficial do Tinker (2024) o Tkinter é a biblioteca padrão do Python para a criação de interfaces gráficas de usuário. O Tkinter possibilita que os desenvolvedores construam aplicações com componentes gráficos como botões, caixas de texto, tabelas, menus e outros elementos interativos, permitindo a criação de interfaces intuitivas e funcionais.

Uma de suas principais vantagens é ser nativo do Python, eliminando a necessidade de instalação adicional e garantindo integração completa com a linguagem. Além disso, o Tkinter é multiplataforma, ou seja, permite a execução do mesmo código em diferentes sistemas operacionais (Windows, macOS e Linux) sem necessidade de modificações significativas.

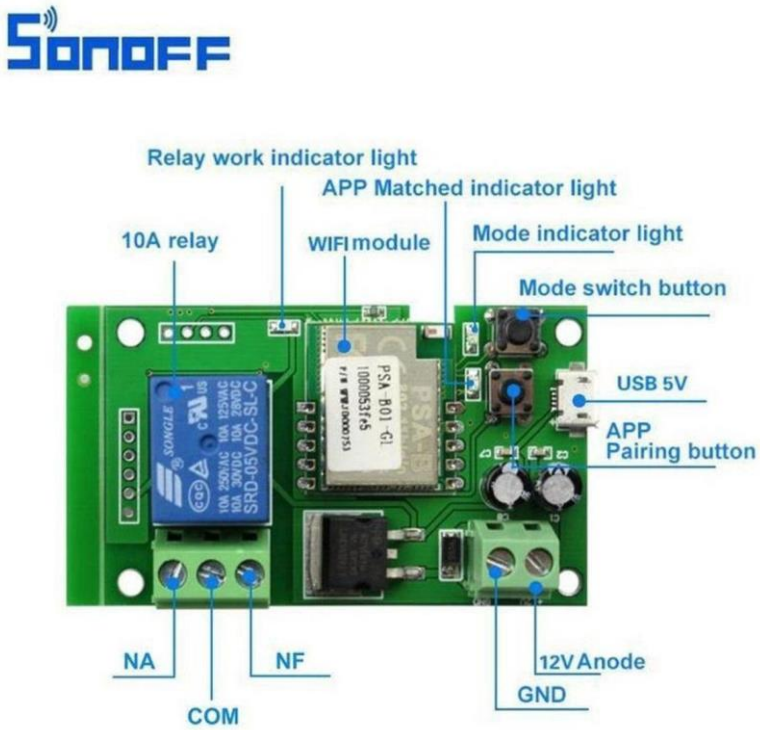
3.5 Sonoff Wi-Fi Relé

O Sonoff Wi-Fi Relé é um dispositivo inteligente projetado para controle remoto de sistemas elétricos. Fabricado pela empresa ITEAD, ele é utilizado em aplicações de automação residencial e industrial devido à sua facilidade de uso, custo acessível e compatibilidade com diversos sistemas de controle conforme descrito na documentação oficial do fabricante (Sonoff Documentation, 2024).

Utilizamos o modelo Sonoff PSF-B01 ilustrado na Figura 4, que oferece um relé de alta eficiência capaz de operar em diferentes tensões, variando geralmente entre 5V e 32V, e possui suporte para modos de pulso ou travado. Isso o torna adequado para aplicações que requerem controle momentâneo, como a abertura do portão.

O Sonoff foi configurado para emitir um pulso que aciona o motor do portão ao receber um comando via Wi-Fi. O envio do comando é condicionado à verificação de permissão da placa identificada pelo OCR.

Figura 5 - Especificações de Entradas Sonoff PSF-B01



MICROCWB. Placa Wi-Fi Sonoff PSF-B01 12V. Disponível em: https://www.microcwb.com.br/placa-wifi-sonoff-12v-automacao-portao-garagem. Acesso em: 4 nov. 2024.

3.6 Biblioteca Selenium

O Selenium é uma biblioteca projetada para automação de navegadores, permitindo que scripts interajam diretamente com a interface web de forma programática. Ele utiliza o protocolo WebDriver para estabelecer uma comunicação direta entre o navegador e a aplicação, replicando ações humanas como cliques, preenchimento de formulários, e navegação por elementos dinâmicos. Essa interação ocorre por meio de comandos que manipulam o DOM (*Document Object Model*) e acionam eventos nativos do navegador.

No contexto do sistema de automação implementado, o Selenium foi utilizado para acessar a interface web da plataforma eWeLink, necessária devido à limitação do modelo PSF-B01, que não aceita comandos diretos via rede.

4 DESENVOLVIMENTO

O desenvolvimento do sistema automatizado de controle de acesso de veículos foi estruturado em diversas etapas, com o objetivo de integrar tecnologias de processamento de imagem, reconhecimento óptico de caracteres (OCR) e automação. A seguir, serão detalhadas a arquitetura do sistema, as implementações realizadas em cada etapa do processo e a integração das ferramentas utilizadas.

4.1 Aquisição das Imagens

De acordo com Wang et al. (2019), no estudo intitulado *Vehicle License Plate Recognition Based on Convolutional Neural Networks*, uma resolução de 720p (1280x720 pixels) é frequentemente considerada suficiente para a legibilidade dos caracteres das placas de veículos, especialmente quando as câmeras estão posicionadas a uma distância padrão e sob ângulos convencionais de captura. O estudo analisa a relação entre a resolução das imagens e a precisão do OCR, concluindo que, para caracteres com uma altura mínima de 10 pixels, a resolução de 720p proporciona uma leitura eficiente.

As imagens utilizadas no desenvolvimento deste trabalho foram capturadas utilizando a câmera integrada do dispositivo móvel Samsung Galaxy A50 (modelo SM-A505), que conta com um sensor de 25 megapixels e capacidade de resolução de 4032 x 3024 pixels. A escolha deste equipamento baseou-se na sua disponibilidade imediata e na capacidade técnica suficiente para atender às necessidades do projeto, fornecendo imagens de alta qualidade para o reconhecimento automático das placas veiculares.

As capturas foram realizadas de forma direta, sem modificações ou compressão no momento da aquisição, a fim de manter a integridade dos dados visuais para o processamento.

4.2 Pré-processamento das Imagens

Segundo Gonzalez e Woods (2008) o pré-processamento das imagens é uma etapa crucial no reconhecimento de placas de veículos, pois visa preparar as imagens capturadas para as fases subsequentes de segmentação e extração de características. Durante essa fase, são aplicadas técnicas que melhoram a qualidade da imagem, removem ruídos e ajustam o contraste, garantindo que os dados extraídos sejam precisos e relevantes.

4.2.1 Ajuste de Contraste e Brilho

No processamento digital de imagens, o contraste define a diferença de intensidade entre os pixels mais claros e mais escuros de uma imagem, enquanto o brilho controla a intensidade geral dos pixels. Para controlar esses valores utilizamos a função cv2.convertScaleAbs() conforme ilustrado na Figura 5, que aplica uma transformação linear aos valores de intensidade de cada pixel da imagem aplicando a fórmula presente na Equação 1.

Equação 1 - Fórmula ajuste de contraste e brilho

$$\text{Saída}(x,y)=\alpha\cdot\text{Entrada}(x,y)+\beta$$

Fonte: Autor (2024)

- α (alpha): É o coeficiente de ganho que controla o contraste. Valores maiores ampliam a diferença relativa entre as intensidades dos pixels.
- β (beta): É o deslocamento que ajusta o brilho global da imagem. Um valor positivo aumenta a intensidade média, enquanto um negativo reduz.

Figura 5 - Linha de código aumento de contraste e brilho

```
# Aumentar o contraste e brilho para destacar a placa
img_contraste = cv2.convertScaleAbs(img, alpha=1.5, beta=0)
```

Fonte: Autor (2024)

Cada pixel em uma imagem tem um valor de intensidade que varia entre 0 (preto) e 255 (branco). O ajuste de contraste redistribui esses valores, ampliando as diferenças entre regiões escuras e claras tornando mais fácil distinguir a placa do veículo do restante da imagem.

Se um pixel tinha valor 100 e $\alpha=1.5$, o novo valor será $100 \cdot 1.5=150$. Caso o valor exceder 255 (o limite máximo), ele será truncado para 255, garantindo que a imagem resultante continue dentro do intervalo válido de intensidades evitando saturação.

Utilizamos o aumento do contraste para realçar a diferenças entre áreas escuras como as bordas da placa ou seus caracteres das partes mais claras como o fundo da placa.

Em placas que estão sob luz direta ou em sombras, o contraste aumentado compensa a perda de detalhes visíveis causada por reflexos ou baixa luminosidade.

O brilho não é ajustado para evitar o clareamento excessivo da imagem que poderia ocultar detalhes importantes.

4.2.2 Conversão para Escala de Cinza

No processamento digital de imagens, a conversão para escala de cinza é uma etapa crucial quando se busca simplificar e destacar informações visuais em imagens, especialmente em sistemas que utilizam visão computacional, como o projeto em questão. Esta técnica transforma uma imagem colorida (RGB – Red, Green, Blue) em uma imagem monocromática, representando apenas a intensidade luminosa dos pixels, eliminando a informação de cor.

Uma imagem colorida é composta por três canais de cor: vermelho (R - Red), verde (G - Green) e azul (B - Blue). Cada pixel da imagem contém uma combinação de valores para esses canais, variando de 0 a 255, formando as cores percebidas. A conversão para escala de cinza

utiliza uma fórmula ponderada para combinar os valores desses canais em um único valor de intensidade luminosa.

No OpenCV, a função cv2.cvtColor() é usada para realizar essa transformação conforme mostra a Figura 6.

Figura 6 - Linha de código conversão a escala de cinza

```
# Converter para escala de cinza
cinza = cv2.cvtColor(img_contraste, cv2.COLOR_BGR2GRAY)
```

Fonte: Autor (2024)

Na Figura 6 cv2.COLOR_BGR2GRAY especifica que a imagem de entrada está no formato BGR (Blue, Green, Red), e que ela deve ser convertida para escala de cinza.

O OpenCV utiliza a fórmula ponderada presente na Equação 2 para realizar a conversão.

Equação 2 - Fórmula conversão a escala de cinza

$$\text{Cinza}=0.299\cdot R+0.587\cdot G+0.114\cdot B$$

Fonte: Autor (2024)

- 0.299: Peso atribuído ao canal vermelho, refletindo a maior sensibilidade do olho humano à luz vermelha.
- 0.587: Peso do canal verde, que contribui mais para a percepção de brilho devido à sensibilidade média.
- 0.114: Peso do canal azul, o menor, pois o olho humano é menos sensível a ele.

Com a remoção dessas informações de cor, a análise da imagem se torna mais eficiente, uma vez que elimina artefatos de cor que poderiam introduzir ruídos e interferir na detecção de padrões importantes.

A escala de cinza foca apenas na luminosidade da imagem, destacando as diferenças de brilho que são essenciais para identificar formas, bordas e texturas relevantes. Esse enfoque

permite que as etapas subsequentes, como binarização e detecção de bordas, funcionem de maneira mais eficaz, pois os algoritmos podem concentrar-se nas características estruturais da imagem, sem a distração das variações de cor.

4.2.3 Equalização do Histograma

A equalização do histograma tem como objetivo melhorar o contraste de uma imagem redistribuindo os valores de intensidade dos pixels de forma mais uniforme ao longo do espectro disponível através da função cv2.equalizeHist() conforme ilustrado na Figura 7.

Figura 7 - Linha de código equalização do histograma

```
# Equalizar o histograma para melhorar o contraste
cinza_eq = cv2.equalizeHist(cinza)
```

Fonte: Autor (2024)

De acordo com (JAIN, 1989) esse processo é baseado no cálculo da função de distribuição acumulada (CDF) e na aplicação de uma transformação que ajusta os valores de intensidade dos pixels para obter uma distribuição mais equilibrada.

O histograma de uma imagem é um gráfico que mostra quantos pixels existem para cada valor de intensidade na imagem. No caso de uma imagem em escala de cinza de 8 bits, os valores de intensidade variam de 0 (preto) a 255 (branco). O histograma é computado para cada nível de intensidade (i), sendo que a quantidade de pixels em cada valor de intensidade é dada pela fórmula presente na Equação 3.

Equação 3 - Fórmula de equalização do histograma

$$H(i) = \sum_{x=1}^{M} \sum_{y=1}^{N} img(x,y)$$

Fonte: Autor (2024)

- H(i) é o número de pixels com intensidade i.
- img(x, y) é o valor da intensidade do pixel na posição (x, y) da imagem.
- M e N são as dimensões da imagem (número de linhas e colunas).

Esse histograma mostra como os pixels estão distribuídos entre os diferentes valores de intensidade.

A função de distribuição acumulada (CDF) calcula a soma acumulada dos valores do histograma, representando a quantidade total de pixels com intensidade menor ou igual a um valor específico. A CDF é calculada da seguinte forma presente na Equação 4.

Equação 4 - Fórmula distribuição acumulada

$$CDF(i) = \sum_{j=0}^{i} H(j)$$

Fonte: Autor (2024)

- CDF(i) é a quantidade acumulada de pixels com intensidade até o valor i.
- H(j) é o valor do histograma para a intensidade j.

A CDF mostra a distribuição acumulada das intensidades da imagem e ajuda a determinar como os valores de intensidade devem ser redistribuídos.

Para que a CDF tenha uma distribuição adequada para a transformação dos valores de intensidade, ela precisa ser normalizada. A normalização ajusta a CDF para que os valores resultantes fiquem dentro do intervalo de intensidade permitido para uma imagem de 8 bits (0 a 255). A normalização é realizada pela seguinte fórmula presente na Equação 5.

Equação 5 - Fórmula normalização

$$CDF_{norm}(i)=\frac{CDF(i) - CDF_{min}}{(M \times N) - CDF_{min}} \times 255$$

Fonte: Autor (2024)

- CDF(i) é o valor da CDF para o nível de intensidade i.
- CDF_min é o menor valor não nulo da CDF.
- M x N é o número total de pixels na imagem.
- CDFnorm(i) é a CDF normalizada, ajustada para o intervalo de 0 a 255.

A normalização permite que a CDF tenha valores entre 0 e 255, o que facilita a aplicação de uma transformação nos pixels.

Após calcular e normalizar a CDF, a transformação é aplicada aos valores de intensidade dos pixels da imagem. Cada pixel da imagem original, com intensidade i, é mapeado para um novo valor de intensidade i' baseado na CDF normalizada seguindo a formula presente na Equação 6.

Equação 6 - Fórmula transformação nos pixels

$$img_{eq}(x,y)=CDF_{norm}(img(x,y))$$

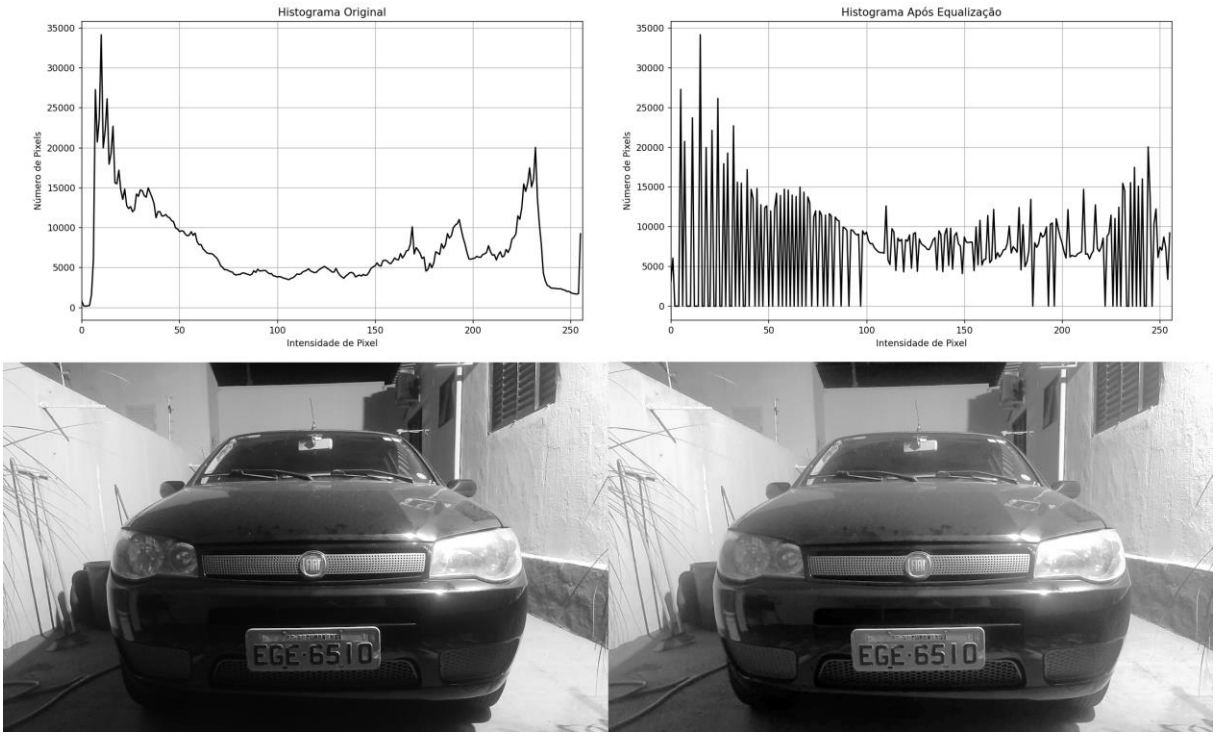
Fonte: Autor (2024)

- img(x, y) é o valor de intensidade do pixel na posição (x, y) da imagem original.
- img_eq(x, y) é o novo valor de intensidade para o pixel na imagem equalizada.

Essa transformação ajusta os valores de intensidade dos pixels, espalhando-os de maneira mais uniforme entre os valores de 0 a 255. Como resultado, a imagem possui um histograma mais equilibrado, com uma distribuição mais homogênea de intensidades, o que leva a um contraste visual melhorado.

Ao redistribuir os níveis de intensidade de forma mais equilibrada, ela prepara melhor as imagens para tarefas seguintes de análise como pode se observar na Figura 8.

Figura 8 - Resultado da equalização do histograma



Fonte: Autor (2024)

4.3 Binarização

A binarização ou limiarização é uma técnica de processamento de imagens cujo objetivo é transformar uma imagem de múltiplos tons em uma imagem composta apenas por dois valores de intensidade: preto (valor 0) e branco (valor 255).

Com a binarização, conseguimos facilitar as etapas subsequentes, como a detecção de bordas e o reconhecimento de padrões, essencial para a identificação de objetos em imagens.

O processo de binarização consiste em atribuir um valor binário a cada pixel da imagem com base em sua intensidade original. Esse valor binário é determinado por um limiar, que define o ponto de corte entre os valores considerados como "escuro" (0) e "claro" (255). Quando a intensidade de um pixel é superior ao limiar, ele é definido como branco (255); caso contrário, é definido como preto (0).

No entanto, a escolha de um único limiar para toda a imagem nem sempre é eficiente, especialmente em situações de imagens com variações de iluminação em ambientes externos ou à noite.

É nesse contexto que o threshold adaptativo se torna importante. O threshold adaptativo, como o nome sugere, ajusta dinamicamente o limiar de binarização de acordo com as características locais da imagem, levando em consideração a intensidade dos pixels ao redor de cada pixel.

O processo é feito através da função cv2.adptiveThreshold() conforme ilustrado na Figura 9.

Figura 9 - Linha de código binarização

```
# Aplicar threshold adaptativo para imagens com baixo contraste
bin = cv2.adaptiveThreshold(cinza_eq, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
```

Fonte: Autor (2024)

O primeiro passo para o threshold adaptativo é o cálculo da média local para o método gaussiano que é dada pela fórmula presente na Equação 7.

Equação 7 - Fórmula média local

$$\text{Media Local}(x,y) = \frac{1}{Z} \sum_{i=-\frac{k}{2}}^{\frac{k}{2}} \sum_{j=-\frac{k}{2}}^{\frac{k}{2}} w(i,j) \cdot I(x+i,y+j)$$

Fonte: Autor (2024)

- I(x+i,y+j) é a intensidade do pixel na posição (x+i,y+j) dentro da janela de tamanho k×k.
- w(i,j) é o peso da função Gaussiana, que pondera a contribuição de cada pixel vizinho com base na sua proximidade ao pixel central (x,y).
- Z é o fator de normalização para garantir que a soma dos pesos w(i,j) seja 1.

A função gaussiana w(i,j) é dada pela Equação 8.

Equação 8 - Fórmula função gaussiana

$$w(i,j) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{i^2 + i^2}{2\sigma^2}\right)$$

Fonte: Autor (2024)

- σ é o desvio padrão da distribuição Gaussiana. Ele controla o quanto a média local deve ser suavizada. Quanto maior σ, maior a suavização, afetando uma área maior ao redor de (x,y).

Depois precisamos calcular o limiar local (T(x, y)). Para cada pixel (x,y), o limiar T(x,y) é calculado utilizando uma média ponderada dos valores dos pixels em uma janela ao redor do pixel central seguindo a fórmula presente na Equação 9.

Equação 9 - Fórmula media local

$$T(x,y) = \text{Media Local}(x,y) - C$$

Fonte: Autor (2024)

- Media Local(x, y) é a média ponderada dos valores dos pixels na janela ao redor de (x,y), que é calculada com a função Gaussiana.
- C é o valor constante subtraído da média local. Este valor ajusta o limiar, permitindo uma diferença entre o limiar local e a intensidade dos pixels vizinhos. Geralmente, C é pequeno e pode ser ajustado para modificar o efeito da binarização.

Por fim depois de calcular o limiar local T(x,y) para cada pixel, a binarização é realizada através da seguinte fórmula presente na Equação 10.

Equação 10 - Fórmula binarização

$$I'(x,y) = \begin{cases} 255, & \text{se } I(x,y) > T(x,y) \\ 0, & \text{se } I(x,y) \leq T(x,y) \end{cases}$$

Fonte: Autor (2024)

- I′(x,y) é o valor binarizado do pixel na posição (x,y).
- I(x,y) é a intensidade do pixel original.
- T(x,y) é o limiar calculado com base na média local.

Se o valor do pixel I(x,y) for maior que o limiar T(x,y), o pixel será definido como 255 (branco). Caso contrário, será definido como 0 (preto).

Esse processo permite que a binarização se ajuste a diferentes condições de iluminação e contraste local, proporcionando melhores resultados, especialmente em imagens com iluminação desigual.

4.4 Filtro de Desfoque

Mesmo após a binarização, pequenas irregularidades ou artefatos de intensidade podem permanecer na imagem. O desfoque suaviza essas variações, eliminando detalhes irrelevantes.

A ação é realizada pela função cv2.GaussianBlur() conforme ilustrado na Figura 10.

Figura 6 - Linha de código filtro de desfoque

```
# Aplicar filtro de desfoque para reduzir ruídos
desfoque = cv2.GaussianBlur(bin, (7, 7), 0)  # Ajuste o tamanho do kernel
```

Fonte: Autor (2024)

Essa operação aplica um filtro de desfoque utilizando um kernel de 7×7 pixels e um desvio padrão (σ) determinado automaticamente pelo OpenCV.

A formula utilizada aqui é bastante semelhante a que usamos na binarização, mas o foco aqui é reduzir ruídos, suavizando gradientes e áreas de intensidade variável.

Essa etapa melhora a qualidade da imagem ao reduzir interferências que poderiam comprometer a detecção de contornos, assegurando que apenas as formas significativas sejam realçadas. O desfoque otimiza o resultado final, tornando as etapas subsequentes mais precisas conforme ilustra a Figura 11.

Figura 7 - Resultado pré-processamento

Fonte: Autor (2024)

4.5 Detecção de Bordas

O algoritmo de Canny é uma técnica de detecção de bordas que utiliza operações sequenciais para identificar transições de intensidade em uma imagem, com ênfase em bordas contínuas e bem definidas.

O processo começa com a suavização da imagem para reduzir ruídos que poderiam ser interpretados como bordas. Essa suavização é feita aplicando um filtro gaussiano, descrito pela seguinte fórmula presente na Equação 10.

Equação 10 - Fórmula filtro gaussiano

$$G(x,y) = \frac{1}{2\pi\sigma} e^{-\frac{x^2 + y^2}{2\pi\sigma}}$$

Fonte: Autor (2024)

- x,y: coordenadas do pixel.
- σ: desvio padrão, que controla o grau de suavização.
- e: base do logaritmo natural, um número irracional com valor aproximado de e≈2.71828.

A próxima etapa envolve o cálculo da magnitude (G) e da direção (θ) do gradiente de intensidade para cada pixel. Isso é feito usando derivadas sobel, que avaliam mudanças de intensidade nas direções horizontal (Gx) e vertical (Gy) conforme ilustra a Equação 11.

Equação 11 - Fórmula derivadas sobel

$$Gx = \frac{\partial I}{\partial x} \qquad Gy = \frac{\partial I}{\partial y}$$

Fonte: Autor (2024)

Para o calculo de G e θ temos as fórmulas ilustradas na Equação 12.

Equação 12 - Fórmula de G e θ

$$G = \sqrt{G_x^2 + G_y^2} \qquad \theta = \arctan \frac{Gy}{Gx}$$

Fonte: Autor (2024)

A magnitude do gradiente indica a força da transição de intensidade, enquanto a direção determina o ângulo da borda.

Após o cálculo do gradiente, a supressão de não-máximos refina as bordas ao manter apenas os pixels que representam máximos locais ao longo da direção do gradiente. Isso é feito verificando os pixels vizinhos na direção de θ\ e descartando aqueles que não são picos.

Matematicamente, um pixel (x,y) com gradiente G(x,y) será mantido se atender as condições ilustradas na Equação 13.

Equação 13 - Condução de gradiente

$$G(x,y) > G(x+\Delta x, y+\Delta y) \quad e \quad G(x,y) > G(x-\Delta x, y-\Delta y)$$

Fonte: Autor (2024)

Onde Δx e Δy dependem da direção θ. Essa etapa resulta em bordas finas e bem definidas.

Após o calculo da supressão de não-máximos passamos para o Limiar Duplo.

Nesta etapa, dois limiares, T{low} e T{high}, são aplicados para classificar os pixels. Os critérios são:

- Bordas fortes: pixels com G > T{high}.
- Bordas fracas: pixels com T{low} ≤ G ≤ T{high}.
- Não bordas: pixels com G < T{low}.

Figura 12 - Linha de código deteccção de bordas

```
# Detecção de bordas com Canny
bordas = cv2.Canny(desfoque, 50, 150)
```

Fonte: Autor (2024)

Os limiares são definidos no código como T{low} = 50 e T{high} = 150 como ilustra a Figura 12.

A última etapa conecta bordas fracas a bordas fortes se estiverem em uma mesma sequência. Pixels de bordas fracas que não estão conectados a bordas fortes são descartados. Isso assegura que apenas bordas consistentes sejam preservadas, eliminando ruídos remanescentes como pode-se observar na Figura 13.

Figura 13 - Resultado detecção de bordas

Fonte: Autor (2024)

4.6 Extração de Contornos

Agora com todos os contornos mapeados na imagem precisamos realizar a extração desses contornos para localizamos a placa na imagem. Esse processo é realizado por meio da função presente na figura 14.

Figura 14 - Linha de código extração de contornos

```
# Encontrar contornos na imagem de bordas
contornos, hier = cv2.findContours(bordas, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
```

Fonte: Autor (2024)

A imagem de entrada para essa função é o resultado do detector de bordas canny, que gera uma imagem binária com pixels brancos (255) representando bordas e pixels pretos (0) representando o fundo. A função busca identificar pixels brancos conectados, analisando a conectividade de 8 pixels (horizontal, vertical e diagonal), e retorna uma lista de arrays, onde cada array representa um contorno formado por uma sequência de coordenadas (x,y) dos pixels que compõem a borda.

Utilizamos cv2.RETR_TREE como o modo de recuperação. Esse método detecta todos os contornos presentes na imagem e organiza suas relações hierárquicas, como pai, filho e irmãos. Para a aproximação dos contornos, usamos o método cv2.CHAIN_APPROX_SIMPLE, que remove pontos redundantes, armazenando apenas os necessários para descrever as bordas. Isso reduz a quantidade de dados processados sem perder a integridade das formas detectadas.

Com os contornos detectados, o próximo passo é passar cada um por uma análise geométrica para determinar se corresponde a placa do veículo conforme ilustra a Figura 15.

Figura 15 - Linha de código encontrar placa

```
for c in contornos:
    perimetro = cv2.arcLength(c, True)
    # Aproximar contorno para verificar se possui 4 lados
    aprox = cv2.approxPolyDP(c, 0.05 * perimetro, True)
    # Verificar se o contorno tem 4 lados (potencial placa)
    if len(aprox) == 4:
        # Determinar o retângulo delimitador
        (x, y, lar, alt) = cv2.boundingRect(c)
        # Verificar a proporção (aspect ratio) entre a largura e a altura
        aspect_ratio = lar / alt
        if 2 < aspect_ratio < 5:  # A razão largura/altura típica para placas
            # Calcular a área do contorno
            area = cv2.contourArea(c)
            # Descartar contornos com área muito pequena ou muito grande
            if 1100 < area:  # Ajuste a área mínima e máxima conforme necessário
                # Ajustar coordenadas para a imagem original
                y = y + altura // 2
                # Desenhar o retângulo na imagem original
                cv2.rectangle(img, (x, y), (x + lar, y + alt), (0, 255, 0), 2)
                # Recortar e salvar a região da placa detectada
                roi = img[y:y + alt, x:x + lar]
                cv2.imwrite("C:/Trabalho/img/placa.jpg", roi)
                print("Placa detectada e salva como 'placa.jpg'")
                break
```

Fonte: Autor (2024)

Inicialmente, o perímetro do contorno é calculado utilizando a função cv2.arcLength seguindo a formula presente da Equação 14.

Equação 14 - Fórmula perímetro do contorno

$$P = \sum_{i=1}^{n} \sqrt{(Xi - Xi - 1)^2 + (Yi - Yi - 1)^2}$$

Fonte: Autor (2024)

- Onde n é o número de pontos no contorno, e (Xi,Yi) são as coordenadas dos pontos consecutivos.

Esse valor é utilizado em seguida no método cv2.approxPolyDP, que simplifica o contorno utilizando o algoritmo de Douglas-Peucker. Esse algoritmo aproxima o contorno ao substituir grupos de pontos por segmentos retos, de acordo com um limite de tolerância (ϵ) proporcional ao perímetro seguindo a fórmula presente da Equação 15.

Equação 15 - Fórmula tolerância

$$\epsilon = \alpha \cdot P$$

Fonte: Autor (2024)

Aqui, α é um fator ajustável definido como 0.05. O algoritmo funciona iterativamente, descartando pontos intermediários que estejam dentro de uma distância menor que ϵ de uma linha reta entre dois pontos extremos. Contornos aproximados com 4 vértices são considerados candidatos por representarem formas retangulares.

Uma vez identificado como um retângulo, o contorno é avaliado por meio de seu bounding box, ou retângulo delimitador, calculado pela função cv2.boundingRect.

Esse bounding box fornece as coordenadas do canto superior esquerdo e as dimensões (largura e altura) do retângulo. A proporção entre largura e altura (aspect ratio) é calculada para verificar se está dentro do intervalo esperado para a placa do veículo seguindo a fórmula presente da Equação 16. Apenas contornos com proporções entre 2 e 5 são considerados válidos, refletindo as dimensões típicas de placas.

Equação 16 - Fórmula aspect ratio

$$AR = \frac{lar}{alt}$$

Fonte: Autor (2024)

Além disso, é calculada a área do contorno, com a função cv2.contourArea, e contornos com áreas muito pequenas ou muito grandes são descartados, evitando falsos positivos causados por ruídos ou objetos irrelevantes conforme ilustra a Equação 17.

Equação 17 - Fórmula área do contorno

$$A = \frac{1}{2}\left|\sum_{i=1}^{n}(Xi \cdot Yi + 1 - Yi \cdot Xi + 1)\right|$$

Fonte: Autor (2024)

Uma vez identificado um contorno válido, ele é destacado na imagem original com a função cv2.rectangle, que desenha um retângulo verde ao redor do contorno. Em seguida, a região correspondente à placa é recortada e salva como uma nova imagem utilizando as coordenadas do bounding box seguindo a fórmula presente na Equação 18.

Equação 18 - Fórmula dimensão da nova imagem

$$ROI=img[y:y+alt,x:x+lar]$$

Fonte: Autor (2024)

O processo é interrompido assim que uma placa válida é encontrada, melhorando a eficiência ao evitar o processamento desnecessário de outros contornos como mostra a Figura 16.

Figura 16 - Resultado da busca pela placa

Fonte: Autor (2024)

Essa imagem será utilizada em etapas posteriores, para que possamos realizar a leitura dos caracteres presentes nela.

4.7 Preparar para Leitura de Caracteres

Para garantir a acurácia e eficiência no reconhecimento óptico de caracteres (OCR), é imprescindível realizar um pré-processamento adequado da imagem extraída. Essa etapa prepara a imagem para que os caracteres possam ser identificados de maneira clara e precisa,

mesmo diante de desafios como ruídos, baixa resolução ou contraste inadequado. O pré-processamento busca eliminar artefatos indesejados, otimizar a visibilidade das informações relevantes e estruturar a imagem para as etapas subsequentes do sistema de reconhecimento. Para isso usaremos técnicas similares as que utilizamos nos passos anteriores.

Garantir que os caracteres na imagem da placa estejam claramente definidos é um requisito indispensável para o sucesso do OCR, imagens pequenas ou de baixa resolução frequentemente apresentam detalhes pouco nítidos, o que pode comprometer a identificação precisa dos caracteres. Para contornar essa limitação, utiliza-se a técnica de ampliação da imagem, onde o tamanho é aumentado mantendo sua proporção e com interpolação bicúbica como ilustra a Figura 17.

Figura 17 - Linha de código redimensionamento

```
# Aumentar a imagem mantendo a proporção com os fatores fx=5 e fy=5
img_resize = cv2.resize(img_placa, None, fx=5, fy=5, interpolation=cv2.INTER_CUBIC)
```

Fonte: Autor (2024)

Essa abordagem é especialmente eficaz porque considera os valores de intensidade dos 16 pixels vizinhos mais próximos para criar uma transição suave entre os novos pixels gerados, preservando os contornos dos caracteres sem introduzir ruídos visuais significativos seguindo a Equação 19.

Equação 19 - Fórmula da interpolação bicúbica

$$F(x,y) = \sum_{i=-1}^{2} \sum_{j=-1}^{2} w(i,j) \cdot I(x+i,y+j)$$

Fonte: Autor (2024)

- F(x,y): Valor do novo pixel gerado.
- I(x+i,y+j): Intensidade dos pixels vizinhos em um espaço de 4x4 ao redor do novo pixel.

- w(i,j): Peso aplicado a cada pixel vizinho com base em sua distância do novo pixel.

Os pesos usados na Equação 19 são calculados com a formula presente na Equação 20.

Equação 20 - Fórmula peso função de interpolação cúbica

$$w(d) = \begin{cases} (1.5|d|^3 - 2.5|d|^2+1) & \text{se } |d|\leq 1 \\ (-0.5|d|^3 + 2.5|d|^2 - 4|d|+2) & \text{se } 1<|d|\leq 2 \\ 0 & \text{se } |d|>2 \end{cases}$$

Fonte: Autor (2024)

- d: Distância entre o pixel original e o novo pixel no eixo horizontal ou vertical.

Esse aprimoramento é decisivo para evitar erros de reconhecimento, especialmente em imagens obtidas em condições desfavoráveis, como baixa iluminação ou ruído visual.

Em seguida, a imagem é convertida para escala de cinza utilizando a função cv2.cvtColor. Essa conversão, abordada anteriormente, reduz a dimensionalidade dos dados ao eliminar os canais de cor, preservando informações essenciais de luminosidade, o que simplifica os cálculos nas etapas subsequentes conforme ilustra a Figura 18.

Figura 18 - Linha de código limiarização

```
# Aplique limiarização simples para melhorar o contraste
_, img_bin = cv2.threshold(img_cinza, 150, 255, cv2.THRESH_BINARY)
```

Fonte: Autor (2024)

Após a conversão, é aplicada a técnica de limiarização simples por meio da função cv2.threshold. Essa técnica transforma os pixels da imagem em valores binários, sendo 0 (preto) ou 255 (branco), com base em um valor de limiar fixo, neste caso, 150. Valores de intensidade abaixo do limiar são atribuídos a 0, enquanto valores iguais ou superiores são atribuídos a 255.

Essa binarização realça os caracteres em contraste com o fundo, destacando as informações que procuramos como mostra a Figura 19.

Figura 19 - Linha de código mediana

```
# Aplique um filtro de mediana para remover ruídos
img_filtrada = cv2.medianBlur(img_bin, 3)
```

Fonte: Autor (2024)

Para lidar com ruídos residuais, é utilizado o filtro de mediana com a função cv2.medianBlur, utilizando uma janela de tamanho 3x3. Esse filtro substitui o valor de cada pixel pela mediana dos valores dos pixels vizinhos, o que permite suavizar pequenos ruídos sem comprometer as bordas dos caracteres. Essa característica é particularmente importante em imagens contendo caracteres finos, onde outros tipos de desfoque poderiam distorcer informações relevantes.

Por fim, é aplicado um ajuste de contraste utilizando a função cv2.convertScaleAbs conforme mostra a Figura 20. Com parâmetros alpha=1.5 e beta=0 para reforçar a visibilidade dos caracteres em imagens onde o contraste inicial é insuficiente, garantindo que as informações sejam facilmente distinguíveis. A intensidade de cada pixel é escalada de forma linear.

Figura 20 - Linha de código ajuste de contraste

```
# Ajuste de contraste
img_contraste = cv2.convertScaleAbs(img_filtrada, alpha=1.5, beta=0)
```

Fonte: Autor (2024)

Como resultado, a imagem processada é salva para que possa ser utilizada na próxima etapa conforme mostra a Figura 21.

Figura 21 - Resultado preparamento para leitura de caracteres

Fonte: Autor (2024)

4.8 Leitura de caracteres

Ao carregar a imagem pré-processada, a função pytesseract.image_to_string realiza a leitura, que é um processo complexo baseado em várias etapas de análise da imagem. O Tesseract começa seu trabalho convertendo a imagem em uma matriz de pixels, onde ele examina as estruturas de cada região da imagem, buscando padrões que correspondem a caracteres. Cada caractere é reconhecido como uma sequência de formas geométricas, com base nas características visuais dos pixels ao redor. O modelo de OCR compara essas formas com um banco de dados interno que contém padrões de letras e números treinados previamente. Isso permite que o OCR identifique as letras e números presentes na placa, mesmo que estes estejam ligeiramente deformados ou em ângulos diferentes.

A chave para o sucesso do OCR nesse processo está na sua capacidade de segmentação. O Tesseract realiza uma segmentação de texto baseada em dois métodos principais: a segmentação por linhas e a segmentação por palavras. Primeiramente, o OCR detecta a orientação do texto e divide a imagem em regiões de texto, identificando linhas e, em seguida,

palavras dentro dessas linhas. Uma vez que o texto foi segmentado, o Tesseract passa a comparar essas formas segmentadas com seus padrões internos, utilizando um modelo de rede neural.

O treinamento do modelo de OCR permite que ele aprenda variações de fontes, estilos e tamanhos, ajustando-se às condições de iluminação e distorção presentes na imagem da placa. Isso é realizado por meio de uma técnica chamada "aprendizado supervisionado", onde o modelo é treinado para reconhecer caracteres em uma ampla variedade de condições de imagem.

Para garantir uma leitura mais precisa utilizamos uma configuração de tessedit_char_whitelist para restringir o reconhecimento apenas a letras maiúsculas e números minimizando os erros de reconhecimento causados por símbolos ou caracteres não alfanuméricos que poderiam ser confundidos com os caracteres da placa conforme mostra a Figura 22.

Figura 22 - Linha de código configuração OCR

```
# Configuração do OCR
config = (
    r'-c tessedit_char_whitelist=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 ')
```

Fonte: Autor (2024)

Após a detecção inicial do texto, é comum que o OCR retorne uma string com caracteres indesejados, como espaços extras ou erros de reconhecimento causados por ruídos na imagem. Para corrigir isso, o texto extraído é processado com uma função que remove caracteres não alfanuméricos e espaços em branco, garantindo que apenas os caracteres válidos permaneçam na string final como mostra a Figura 23.

Figura 23 - Linha de código limpeza de texto

```
# Limpeza do texto para retirar espaços extras e garantir que temos apenas alfanuméricos
saida = ''.join([c if c.isalnum() else '' for c in saida]).strip()
```

Fonte: Autor (2024)

Este processo de leitura e filtragem dos caracteres garante que o OCR consiga produzir um texto preciso, pronto para ser utilizado na etapa seguinte de validação das informações extraídas.

4.9 Validação da placa

O primeiro passo na validação do texto extraído pelo OCR é conectar-se ao banco de dados SQLite, onde as informações dos veículos, como a placa e horários que estão permitidos acessar o local são armazenadas como ilustrado na Figura 24.

Figura 24 - Linha de código conexão com o banco

```
# Conectar ao banco de dados SQLite
conn = sqlite3.connect('C:/Trabalho/retorno/banco.db')
cursor = conn.cursor()
```

Fonte: Autor (2024)

Uma vez estabelecida a conexão com o banco de dados, a função executa uma consulta SQL para procurar pela placa lida pelo OCR na tabela veículos conforme mostra a Figura 25.

Figura 25 - Linha de código consulta pela placa

```
# Consultar o banco para verificar se a placa está registrada
cursor.execute("SELECT * FROM veiculos WHERE placa = ?", (saida,))
veiculo = cursor.fetchone()
```

Fonte: Autor (2024)

Se o banco de dados retorna um registro para a placa consultada, o código então extrai os dados relativos ao veículo, incluindo o nome do condutor, a placa, os horários de entrada e

saída permitidos. A partir dessas informações, o código verifica se a hora atual está dentro do intervalo de horário de entrada e saída permitido para o veículo como mostra a Figura 26.

Figura 26 - Linha de código extrair dados referente a placa

```
if veiculo:
    condutor = veiculo[1]
    placa = veiculo[2]
    horario_entrada = veiculo[3]
    horario_saida = veiculo[4]

    print(f"Horário Entrada: {horario_entrada}, Horário Saída: {horario_saida}")  # Depuração
```

Fonte: Autor (2024)

Para garantir que os valores de horários extraídos do banco estejam no formato correto e possam ser comparados de maneira eficiente, é necessário realizar a conversão dos horários em formato de texto (string) para objetos do tipo datetime.time. Esse procedimento é necessário para realizar comparações precisas entre os horários armazenados no banco de dados e o horário atual.

O horário atual é obtido usando datetime.datetime.now().time(), que retorna a hora atual no formato datetime.time como ilustra a Figura 27.

Figura 27 - Linha de código converter horário

```
try:
    # Tentar converter os horários para objetos de hora
    horario_entrada = datetime.datetime.strptime(horario_entrada, "%H:%M:%S").time()
    horario_saida = datetime.datetime.strptime(horario_saida, "%H:%M:%S").time()

    print(f"Hora de entrada convertida: {horario_entrada}, Hora de saída convertida: {horario_saida}")  # Depuração

    # Hora atual
    horario_atual = datetime.datetime.now().time()
    print(f"Horário atual: {horario_atual}")
```

Fonte: Autor (2024)

A comparação entre os horários de entrada, saída e o horário atual é feita utilizando operadores lógicos simples. Se o horário atual estiver dentro do intervalo permitido, o acesso é considerado liberado; caso contrário, o acesso é negado conforme mostra a Figura 28.

Figura 28 - Linha de código verificar permissão

```
            # Verificar se a entrada e saída estão dentro do horário permitido
            if horario_entrada <= horario_atual <= horario_saida:
                print("O carro está autorizado a acessar o local.")
                acesso = "Liberado"
            else:
                print("O carro não está autorizado a acessar o local.")
                acesso = "Negado"
        except ValueError as e:
            print(f"Erro ao processar os horários: {e}")
            acesso = "Negado"
    else:
        print("Placa não encontrada no banco de dados.")
        acesso = "Negado"
        condutor = "Desconhecido"
```

Fonte: Autor (2024)

Independentemente de o acesso ser liberado ou negado, o evento é registrado no banco de dados.

Para inserir um evento, a função captura a data e o horário atuais e executa um comando SQL para inserir um novo registro na tabela eventos. Os parâmetros passados para o comando incluem a placa do veículo, o status de acesso, a data e o horário atual, bem como o nome do condutor caso a placa esteja registrada conforme mostra a Figura 29.

Figura 29 - Linha de código registrar eventos

```
def registrar_evento(placa, acesso, condutor):
    # Conectar ao banco de dados SQLite
    conn = sqlite3.connect('C:/Trabalho/retorno/banco.db')
    cursor = conn.cursor()

    # Definir a data e hora atual
    data_atual = datetime.datetime.now().strftime("%Y-%m-%d")
    horario_atual = datetime.datetime.now().strftime("%H:%M:%S")

    # Inserir o evento no banco
    cursor.execute("INSERT INTO eventos (placa, acesso, data, horario, condutor) VALUES (?, ?, ?, ?, ?)",
                   (placa, acesso, data_atual, horario_atual, condutor))
    conn.commit()

    print(f"Evento registrado: {placa} - Acesso: {acesso} - Condutor: {condutor}")
```

Fonte: Autor (2024)

Após a execução do comando SQL, o método conn.commit() é chamado para garantir que a transação seja confirmada e os dados sejam efetivamente persistidos no banco de dados. Caso o commit não seja chamado, as alterações feitas na base de dados seriam descartadas, não sendo refletidas no armazenamento permanente.

4.10 Abertura do portão

A abertura do portão ocorre de maneira integralmente controlada através da plataforma eWeLink, que gerencia dispositivos como o Sonoff PSF-B01.

No caso do Sonoff PSF-B01, o obstáculo surge devido à incapacidade do drive do dispositivo de aceitar requisições HTTP diretas, restringindo a possibilidade de comunicação com o dispositivo de forma convencional.

A solução encontrada para esse problema envolve a automação do navegador web, utilizando ferramentas como o Selenium, para simular as ações necessárias dentro da interface da plataforma. Ao emular a interação do usuário, o sistema consegue acessar a conta do usuário na eWeLink, navegar até o dispositivo Sonoff, realizar a validação da placa do veículo e, com isso, acionar o comando para abrir o portão.

A implementação começa com a criação de uma instância do WebDriver, a ferramenta que permite o controle remoto do navegador. Para isso vamos instanciar o Chrome WebDriver através do webdriver.Chrome() utilizando a biblioteca ChromeDriverManager, que gerencia automaticamente a versão do driver do navegador instalado no sistema, evitando erros que poderiam interromper a execução do script conforme mostra a Figura 30.

Figura 30 - Linha de código instanciar webdriver

```
def controle_dispositivo():
    global placa_autorizada
    driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
```

Fonte: Autor (2024)

Com o navegador aberto, o Selenium utiliza o método driver.get() para acessar a URL da plataforma eWeLink, abrindo a página de login. O WebDriverWait foi utilizado para aguardar que os elementos da interface carregassem antes de realizar qualquer interação. Esse método garante que o script não falhe devido ao carregamento lento de elementos da página. Através do WebDriverWait e do comando EC.element_to_be_clickable(), o script aguarda o momento exato para interagir com a interface, garantindo que os elementos como botões e campos de texto estejam prontos para receber comandos.

Após o carregamento da página de login, o Selenium localiza os campos de entrada (telefone, email e senha) utilizando seletores como By.ID ou By.XPATH. Os dados são inseridos automaticamente utilizando o método send_keys(), simulando a digitação do usuário. O preenchimento desses campos é seguido do clique no botão de login, que é também identificado utilizando WebDriverWait para garantir que o botão esteja disponível antes de ser clicado como ilustrado na Figura 31.

Figura 31 - Linha de código efetuar login

```
# Acessa a URL
    driver.get("https://web.ewelink.cc/#/login")
# Aguarda a página carregar e clica na aba "Conta"
    WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.XPATH, '//*[@id="rc-tabs-0-tab-2"]'))  # XPath para a aba "Conta"
).click()
# Aguarda o campo de região ser visível
    WebDriverWait(driver, 10).until(
    EC.visibility_of_element_located((By.ID, "form_item_tel-region"))
)
# Clica no campo de região (input de telefone)
    region_input = driver.find_element(By.ID, "form_item_tel-region")
    region_input.click()
# Digita "+55" para selecionar o Brasil
    region_input.send_keys("+55")
    region_input.send_keys(Keys.ENTER)
# Aguarda o campo de email aparecer e preenche com suas informações
    email_input = WebDriverWait(driver, 10).until(
    EC.visibility_of_element_located((By.ID, "form_item_username"))
)
    email_input.send_keys("             @hotmail.com")
# Preenche o campo de senha
    password_input = driver.find_element(By.ID, "form_item_password")
    password_input.send_keys("            ")
# Aguarda e clica no botão de login (submeter)
    login_button = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.XPATH, '//*[@type="submit"]'))  # XPath para o botão de login
)
    login_button.click()
```

Fonte: Autor (2024)

Uma vez logado, a interface da eWeLink pode apresentar notificações ou pop-ups que precisam ser fechados antes de continuar a automação. O Selenium foi configurado para esperar que essas notificações apareçam e, assim que visíveis, realiza o clique no botão de fechar para removê-las da tela, utilizando novamente o WebDriverWait para aguardar a visibilidade do elemento conforme mostra a Figura 32.

Figura 32 - Linha de código fechar notificação

```
    notification_img = WebDriverWait(driver, 10).until(
    EC.element_to_be_clickable((By.XPATH, "//img[@src='data:image/png;base64

# Clicar na imagem para fechar a notificação
    notification_img.click()
```

Fonte: Autor (2024)

O próximo passo foi a validação da placa do veículo, o que é feito por meio da variável placa_autorizada. O script aguarda a confirmação da placa antes de prosseguir com o controle do dispositivo. Assim que a placa é validada, o Selenium localiza o botão de controle do Sonoff PSF-B01, utilizando o WebDriverWait para garantir que o botão de comando esteja disponível conforme ilustra a Figura 33.

Figura 33 - Linha de código acionar dispositivo

```
device_img = WebDriverWait(driver, 10).until(
EC.element_to_be_clickable((By.XPATH, "//img[@src='/assets/socket_1_off-a0e50de4.png' and @alt='device']"))
)
device_img.click()

# Aqui aguardamos a validação da placa antes de clicar no botão "Off"
print("Aguardando validação da placa...")

while not placa_autorizada:
    time.sleep(1)  # Aguarda até que a placa seja autorizada

print("Placa autorizada! Controlando dispositivo...")

# Aguarda o botão "Off" estar disponível
off_button = WebDriverWait(driver, 30).until(
    EC.element_to_be_clickable((By.XPATH, "//div[@data-v-f8607bac='' and contains(text(), 'Off')]"))
)
off_button.click()
time.sleep(25)
off_button.click()
```

Fonte: Autor (2024)

Após o acionamento do botão o programa aguarda 25 segundos para garantir que o portão tenha tempo suficiente para abrir e permitir que o veículo entre no local sem problemas.

Após o tempo de espera, o programa localiza novamente o botão de controle e aciona-o para fechar o portão, garantindo que o local seja novamente fechado e o processo de espera pela validação da próxima placa possa ser iniciado novamente antes que o programa realize qualquer outro comando.

Embora esse método seja uma solução alternativa, ele contorna eficientemente a limitação do PSF-B01.

4.11 Automação do Portão

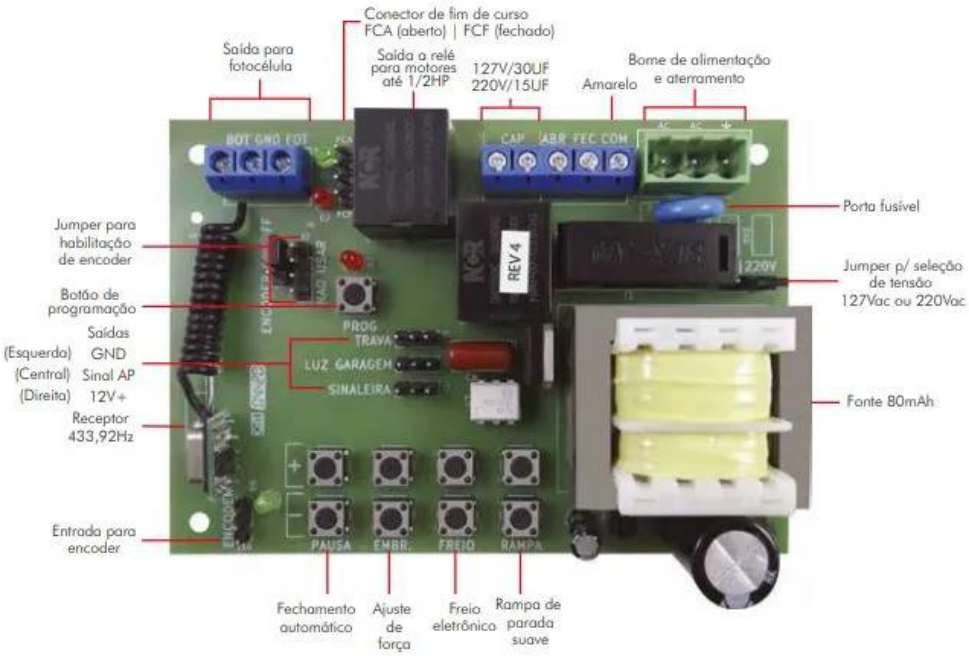
Após a conclusão das etapas de programação e automação do sistema de controle de acesso, o passo final foi a integração física do dispositivo Sonoff PSF-B01 à placa de controle AGL do portão automático.

Este processo exigiu o estudo das especificações técnicas do hardware envolvido e a execução de conexões elétricas para garantir o funcionamento correto e seguro do sistema.

A placa controladora AGL é responsável por gerenciar os comandos de abertura e fechamento do portão, dispondo de terminais específicos para controle externo. Após análise do circuito e da documentação técnica, identificaram-se dois pontos de conexão que vamos utilizar conforme mostra a Figura 34.

- GND (terra): Terminal que serve como referência elétrica para o circuito.
- BOT (botão de acionamento): Terminal que ativa o motor do portão ao ser momentaneamente conectado ao terra, simulando a ação de um botão físico.

Figura 34 - Especificações de entradas da placa AGL



Fonte: AGL BRASIL (2024)

O Sonoff PSF-B01 foi configurado no modo relé de contato seco, permitindo a conexão elétrica sem a transmissão de energia diretamente para a placa controladora.

A integração entre o Sonoff e a placa AGL foi realizada com os seguintes passos:

- Terminal COM do Sonoff conectado ao GND da placa AGL: Essa conexão estabeleceu um ponto comum de referência elétrica entre os dispositivos.
- Terminal NO (normalmente aberto) do Sonoff conectado ao BOT da placa AGL: Essa conexão foi configurada para fechar momentaneamente o circuito quando o relé do Sonoff fosse acionado. Assim, simulava-se o acionamento manual do botão do portão.

A alimentação do Sonoff foi feita por meio de um adaptador de carregador de celular, permitindo que o dispositivo receba a tensão necessária para operar.

Figura 35 - Integração do Sonoff a placa AGL

Fonte: Autor (2024)

Após a integração elétrica ilustrada na Figura 35, foi realizado um conjunto de testes práticos para verificar o funcionamento do sistema. A partir do envio de comandos por meio do software desenvolvido neste trabalho, o relé do Sonoff foi acionado corretamente, resultando na abertura e fechamento do portão. O sistema demonstrou boa funcionabilidade em condições reais, tanto em comandos diretos quanto integrados ao fluxo de automação.

5 CONCLUSÃO

O sistema automatizado para controle de acesso de veículos desenvolvido neste trabalho foi capaz de atender aos objetivos propostos, integrando de forma eficiente técnicas de visão computacional, reconhecimento óptico de caracteres (OCR) e automação de dispositivos IoT.

Através da implementação do reconhecimento de placas de veículos com a utilização de bibliotecas como OpenCV e Tesseract, o sistema foi capaz de detectar e identificar placas em diversas condições de imagem, garantindo a precisão na validação do acesso.

A integração com um banco de dados e a automação do controle de portões utilizando o dispositivo Sonoff PSF-B01 permitiram que o sistema operasse de forma autônoma, realizando a abertura e fechamento do portão sem a necessidade de intervenção manual, o que proporciona maior agilidade no controle de acesso a áreas restritas.

Como próximo passo, o sistema pode ser aprimorado com a integração de um módulo de reconhecimento do veículo, utilizando técnicas de captura e análise de imagens para identificar características físicas do veículo, como a forma, cor e modelo. Esse aprimoramento visa fornecer uma camada adicional de segurança, evitando falsos positivos causados por clonagem de placas e aumentando a confiabilidade do processo de autorização de entrada. A modularidade do sistema permitirá a adição dessa funcionalidade sem comprometer a estrutura existente, oferecendo flexibilidade para futuras atualizações que atendam a necessidades emergentes de segurança e controle de acesso.

Portanto, este trabalho apresenta uma solução inovadora e viável para a automação de controle de acesso de veículos, com grande potencial de aplicação em ambientes comerciais e residenciais, trazendo benefícios tanto em termos de praticidade quanto de eficiência operacional.

REFERÊNCIAS

BACKES, Alexandre. *Aula 04 - Processamento de Histogramas.* UFU, 2024. Disponível em: https://www.facom.ufu.br/~backes/gsi058/Aula04-ProcessamentoHistogramas.pdf. Acesso em: 21 outubro. 2024.

BRADSKI, Gary; KAEHLER, Adrian. *Learning OpenCV: Computer Vision with the OpenCV Library.* O'Reilly Media, 2008.

COVAP UTFPR. *Introdução ao Processamento Digital de Imagens.* Disponível em: https://covap-utfpr.github.io/pdi/intro.html. Acesso em: 27 junho. 2024.

DIGITALIZA PRO. *Visão computacional: entenda o conceito e aplicações.* Disponível em: https://digitalizapro.com.br/glossario/visao-computacional-entenda-o-conceito-e-aplicacoes/. Acesso em: 19 nov. 2024.

ERPO. *Monitoramento de entrada e saída de veículos*. Disponível em: https://erpo.com.br/monitoramento-entrada-saida-veiculos/. Acesso em: 9 outubro. 2024.

GIL, Antonio Carlos. Como elaborar projetos de pesquisa. 6. ed. São Paulo: Atlas, 2017.

GONZALEZ, Rafael C.; WOODS, Richard E. *Digital Image Processing.* 3. ed. Pearson Education, 2008.

JÄHNE, Bernd. Digital image processing: concepts, algorithms, and applications. Springer-Verlag, 2005.

JAIN, Anil K. *Fundamentals of Digital Image Processing.* Prentice-Hall, 1989.

OPENCV. *OpenCV Documentation.* Disponível em: https://docs.opencv.org/. Acesso em: 12 junho. 2024.

PYTHON. *Documentação oficial do SQLite.* Disponível em: https://docs.python.org/3/library/sqlite3.html. Acesso em: 11 nov. 2024.

SAS. *Computer Vision: Como transformar imagens em insights.* Disponível em: https://www.sas.com/pt_br/insights/analytics/computer-vision.html. Acesso em: 23 nov. 2024.

SCURI, Marco. *Fundamentos do Processamento Digital de Imagens.* PUC-Rio, 2024. Disponível em: https://web.tecgraf.puc-rio.br/~scuri/download/fid.pdf. Acesso em: 9 abril 2024.

SMITH, Ray. *An Overview of the Tesseract OCR Engine.* IEEE Xplore, 2007. Disponível em: https://ieeexplore.ieee.org/. Acesso em: 19 nov. 2024.

TESSERACT. *Documentação oficial do Tesseract OCR.* Disponível em: https://github.com/tesseract-ocr/tesseract. Acesso em: 10 agosto. 2024.

TOTVS**.** Acesso e segurança by Suricato: Ficha técnica. Disponível em: https://produtos.totvs.com/ficha-tecnica/acesso-e-seguranca-by-suricato/. Acesso em: 14 julho 2024.

VISUAREA. *As imagens digitais: uma introdução.* Disponível em: https://www.visuarea.com.br/artigos/as-imagens-digitais-uma-introducao. Acesso em: 12 setembro 2024.

WANG, Jian; ZHANG, Zhi; LIU, Xue; CHEN, Xiaohui. *Vehicle License Plate Recognition Based on Edge Detection.* 2019. Disponível em: https://www.researchgate.net/publication/341237777_Vehicle_License_Plate_Recognition_Based_on_Edge_Detection. Acesso em: 3 setembro 2024.

SCURI, Marco. *Fundamentos do Processamento Digital de Imagens*. PUC-Rio, 2024. Disponível em: https://web.tecgraf.puc-rio.br/~scuri/download/fid.pdf. Acesso em: 5 julho 2024.

SISPONTO. 3 benefícios do controle de acesso para veículos: confira. Disponível em: https://www.sisponto.com.br/3-beneficios-do-controle-de-acesso-para-veiculos-confira. Acesso em: 15 agosto 2024.

SONOFF. *Documentação oficial do Sonoff*. Disponível em: https://sonoff.tech. Acesso em: 16 maio 2024.

unifev

TERMO DE AUTORIZAÇÃO DE PUBLICAÇÃO

Eu, Thiago Henrique Silva Bruno, nacionalidade Brasileira, estado civil Solteiro, portador da Cédula de identidade RG nº. 59.032.933-9, inscrito no CPF/MF sob nº 443.705.918-21, residente à Av./Rua Bahia, nº. 2127, município de Votuporanga, Estado de SP, **AUTORIZO** a Unifev – Centro Universitário de Votuporanga, a disponibilizar publicamente o Trabalho de Conclusão de Curso (TCC) de minha autoria pela Internet, bem como de preservar a obra integralmente em seu Repositório Institucional.

Por esta ser a expressão da minha vontade, **DECLARO** o uso acima descrito sem que nada haja a ser reclamado a título de direitos conexos à minha imagem ou a qualquer outro e assino a presente autorização.

Votuporanga, dia 05 de dezembro de 2024.

Thiago Henrique S. Bruno

(Assinatura)

Nome: Thiago Henrique Silva Bruno

Telefone p/ contato: (55) 17 99643-0598

unifev

TERMO DE CONSENTIMENTO PARA TRATAMENTO DE DADOS

Este documento visa a registrar a manifestação livre, informada e inequívoca pela qual o Titular concorda com o tratamento de seus dados pessoais para finalidade específica, em conformidade com a Lei nº 13.709/2018 – Lei Geral de Proteção de Dados Pessoais (LGPD).

Ao manifestar sua aceitação com o presente termo, o TITULAR consente e concorda que a Unifev – Centro Universitário de Votuporanga, doravante denominada CONTROLADORA, tome decisões referentes ao tratamento dos dados pessoais do TITULAR ou dados necessários ao usufruto de serviços ofertados por esta instituição de ensino, bem como realize o tratamento de tais dados, envolvendo operações como as que se referem a coleta, produção, recepção, classificação, utilização, acesso, reprodução, transmissão, distribuição, processamento, arquivamento, armazenamento, eliminação, avaliação ou controle da informação, modificação, comunicação, transferência, difusão ou extração.

Dados Pessoais: A Controladora fica autorizada a realizar e a tomar decisões referentes ao tratamento dos seguintes dados pessoais do TITULAR: Nome completo; Nome empresarial; Data de nascimento; Número e imagem da Carteira de Identidade (RG); Número e imagem do Cadastro de Pessoas Físicas (CPF); Número e imagem da Carteira Nacional de Habilitação (CNH); Número do Cadastro Nacional de Pessoas Jurídicas (CNPJ); Estado civil; Nível de instrução ou escolaridade; Endereço completo; Números de telefone, WhatsApp e endereços de e-mail; Banco, agência e número de contas bancárias; Comunicação, verbal e escrita, mantida entre o Titular e o Controlador; Certidão de Nascimento e/ou de Casamento; Dados referentes ao local de trabalho; Comprovantes de renda; Comprovante de endereço completo; Dados de saúde.

Finalidades do Tratamento dos Dados: O tratamento dos dados pessoais listados neste termo tem as seguintes finalidades: - Possibilitar que a Controladora identifique e entre em contato com os Titulares para fins de esclarecimentos relativos aos editais. - Possibilitar que a Controladora utilize tais dados na elaboração de relatórios; - Possibilitar que a Controladora utilize tais dados em documentos financeiros; - Possibilitar que a Controladora utilize tais dados para a execução de editais e auditorias; - Possibilitar que a Controladora utilize o nome completo dos Titulares nas publicações de resultados de editais, chamadas de lista de espera de editais, relações de alunos aptos a recebimento do auxílio, dentre outras publicações relacionadas à transparência da execução dos editais.

Compartilhamento de Dados: A Controladora fica autorizada a compartilhar os dados pessoais dos Titulares com outros agentes de tratamento de dados, caso seja necessário para as finalidades listadas neste termo, observados os princípios e as garantias estabelecidas pela Lei nº 13.709/2018.

Segurança dos Dados: A Controladora responsabiliza-se pela manutenção de medidas de segurança, técnicas e administrativas aptas a proteger os dados pessoais de acessos não autorizados e de situações acidentais ou ilícitas de destruição, perda, alteração, comunicação ou qualquer forma de tratamento inadequado ou ilícito.

Em conformidade ao art. 48 da Lei nº 13.709/2018, a Controladora comunicará aos Titulares e à Autoridade Nacional de Proteção de Dados (ANPD) a ocorrência de incidente de segurança que possa acarretar risco ou dano relevante ao Titular.

unifev

Término do Tratamento dos Dados: A Controladora poderá manter e tratar os dados pessoais do Titular durante todo o período em que forem pertinentes ao alcance das finalidades listadas neste termo. Dados pessoais anonimizados, sem possibilidade de associação ao indivíduo, poderão ser mantidos por período indefinido.

O Titular poderá solicitar via e-mail ou correspondência à Controladora, a qualquer momento, que sejam eliminados os dados pessoais não anonimizados do Titular. O Titular fica ciente de que poderá ser inviável à Controladora continuar o fornecimento de serviços e programas ao Titular a partir da eliminação dos dados pessoais.

Direitos do Titular: O Titular tem direito a obter da Controladora, em relação aos dados por ela tratados, a qualquer momento e mediante requisição: I - confirmação da existência de tratamento; II - acesso aos dados; III - correção de dados incompletos, inexatos ou desatualizados; IV - anonimização, bloqueio ou eliminação de dados desnecessários, excessivos ou tratados em desconformidade com o disposto na Lei nº 13.709/2018; V - eliminação dos dados pessoais tratados com o consentimento do titular, exceto nas hipóteses previstas no art. 16 da Lei nº 13.709/2018; VI - informação das entidades públicas e privadas com as quais a controladora realizou uso compartilhado de dados; VII - informação sobre a possibilidade de não fornecer consentimento e sobre as consequências da negativa; VIII - revogação do consentimento, nos termos do § 5º do art. 8º da Lei nº 13.709/2018.

Direito de Revogação do Consentimento: Este consentimento poderá ser revogado pelo Titular, a qualquer momento, mediante solicitação via e-mail ou correspondência à Controladora.

Votuporanga, 05 de dezembro de 2024.

Thiago Henrique D. Pereira

Assinatura do aluno

CPF: 44370591821

RG: 590329339

unifev

TERMO DE ISENÇÃO DE RESPONSABILIDADE

Eu, Thiago Henrique Silva Bruno, RA 104203, aluno(a) do curso de Engenharia de Computação da Unifev – Centro Universitário de Votuporanga, declaro, para todos os fins que se fizerem necessários, que assumo total responsabilidade pelo aporte ideológico e referencial conferido ao Trabalho de Conclusão de Curso (TCC), isentando a Instituição, o coordenador, o responsável pela disciplina, o orientador e a banca examinadora de todo e qualquer reflexo acerca da pesquisa apresentada.

Estou ciente de que poderei responder administrativa, civil e criminalmente em caso de plágio comprovado.

Votuporanga-SP., 05 de dezembro de 2024.

Thiago Henrique Silva Bruno
Nome completo do(a) aluno(a)