UNIVERSIDADE FEDERAL DE SANTA MARIA

CENTRO DE TECNOLOGIA

CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

André do Amaral Chagas Júnior

DESENVOLVIMENTO DE SISTEMA AUTOMATIZADO PARA ENTRADA DE VEÍCULOS EM LOCAIS PRIVADOS COM VISÃO COMPUTACIONAL

Santa Maria, RS

2023

André do Amaral Chagas Júnior

DESENVOLVIMENTO DE SISTEMA AUTOMATIZADO PARA ENTRADA DE VEÍCULOS EM LOCAIS PRIVADOS COM VISÃO COMPUTACIONAL

Trabalho de conclusão de curso apresentado ao curso de Bacharelado em Engenharia de Controle e Automação da Universidade federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Controle e Automação

Orientador: Prof. Dr. Daniel Fernando Tello Gamarra

Santa Maria, RS

2023

André do Amaral Chagas Júnior

DESENVOLVIMENTO DE SISTEMA AUTOMATIZADO PARA ENTRADA DE VEÍCULOS EM LOCAIS PRIVADOS COM VISÃO COMPUTACIONAL

Trabalho de conclusão de curso apresentado ao curso de Bacharelado em Engenharia de Controle e Automação da Universidade federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Controle e Automação

Aprovado em 14 de julho de 2023:

_____

Daniel Fernando Tello Gamarra, Prof. Dr. (UFSM)
(Presidente/Orientador)

_____

Frederico Menine Schaf, Prof. Dr. (UFSM)

_____

Candice Muller, Profª. Dr. (UFSM)

Santa Maria, RS

2023

RESUMO

DESENVOLVIMENTO DE SISTEMA AUTOMATIZADO PARA ENTRADA DE VEÍCULOS EM LOCAIS PRIVADOS COM VISÃO COMPUTACIONAL

AUTOR: ANDRÉ DO AMARAL CHAGAS JUNIOR

ORIENTADOR: DANIEL FERNANDO TELLO GAMARRA

Este trabalho consiste na implementação de um sistema para automatização de acessos a locais fechados, onde o sistema é capaz de detectar a presença do veículo e realizar a leitura de sua placa e o controle e a supervisão são operados de forma remota, os dispositivos estão conectados através da Internet, utilizando uma VPN (Virtual Private Network). Foi utilizado uma câmera IP que é a responsável por detectar o movimento do veículo e enviar a imagem para o processamento através do protocolo FTP (File Transport Protocol) até o servidor remoto o qual fará o processamento da imagem. Neste computador foi desenvolvido o software que verifica se existe um veículo na posição correta de processamento, utilizando o algoritmo de detecção de objetos YOLO (You Only Look Once), após utilizou-se o software Openalpr e o Google OCR (Optical Character Recognition) para extração dos caracteres das placas. O algoritmo verifica se a placa está cadastrada, e caso a resposta seja afirmativa, envia o comando de abertura do portão via protocolo UDP (User Datagram Protocol) para o controlador, tarefa essa executada pelo ESP8266. Sendo os dados do veículo armazenados em um banco de dados para consultas futuras. O sistema mostrou-se funcional tendo um desempenho apropriado para a aplicação, permitindo desta maneira o controle de acesso automatizado e controlado em locais privados.

Palavras chaves: Automatização, Processamento de imagens, OCR, Supervisão remota.

ABSTRACT

DEVELOPMENT OF AN AUTOMATED SYSTEM FOR VEHICLE ENTRY IN PRIVATE LOCATIONS WITH COMPUTATIONAL VISION

AUTHOR: ANDRÉ DO AMARAL CHAGAS JUNIOR

ADVISOR: DANIEL FERNANDO TELLO GAMARRA

This work consists in the implementation of a system for automating access to closed places, where the system is capable of detecting the presence of the vehicle and reading its license plate and the control and supervision are operated remotely, the devices are connected over the Internet using a VPN (Virtual Private Network). An IP camera was used, which is responsible for detecting the movement of the vehicle and sending the image for processing through the FTP (File Transport Protocol) to the remote server which will process the image. On this computer, the software was developed to check if there is a vehicle in the correct processing position, using the YOLO object detection algorithm (You Only Look Once), after which Openalpr software and Google OCR (Optical Character Recognition) were used to extract the characters from the license plates. The algorithm checks if the card is registered, and if the answer is affirmative, it sends the command to open the gate via the UDP (User Datagram Protocol) to the controller, a task performed by the ESP8266. The vehicle data being stored in a database for future consultations. The system proved to be functional with an appropriate performance for the application, thus allowing automated and controlled access control in private places.

Keywords: Automation. Image processing. OCR. Remote supervision.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
FTP	*File Transfer Protocol*
GPU	*Graphics Processing Unit*
IoT	*Internet of Things*
IP	*Internet Protocol*
ISO	***I**nternational **O**rganization for **S**tandardization*
KiB	*KibiByte*
LSTM	*Long Short Term Memory*
NVMe	*Non-Volatile Memory Express*
OCR	*Reconhecimento de Caracteres Óptico*
OSI	*Open Systems Interconnection*
RAM	*Random Access Memory*
RNC	*Rede Neural Convolucional*
SFTP	Secure *File Transfer Protocol*
SQL	*Structured Query Language*
TCP	*Tranport Control Protocol*
UDP	*User Datagram Protocol*
VOIP	*Voice over Internet Protocol*
VPN	*Virtual Private Network*
YOLO	*You Only Look Once*

SUMÁRIO

1. INTRODUÇÃO

Segundo IBGE, o Brasil possuía no ano de 2021 uma frota de 111.446.870 veículos emplacados considerando todo o território nacional, veículos estes que circulam pelas vias abertas todos os dias. Desta maneira existe a necessidade de um melhor acompanhamento e supervisão destas unidades de tráfego, seja para controle de acesso a locais privados, multa por excesso de velocidade ou até mesmo acompanhamento de rotas por parte das autoridades.

Em 31 de janeiro de 2020 o Conselho Nacional de Trânsito tornou obrigatória para todo o Brasil o padrão de placa Mercosul, desta forma adotou-se o modelo já utilizado nos países vizinhos, como Uruguai e Argentina. Esta placa possui 7 dígitos no total, sendo 4 letras e 3 números, coexistindo no país dois padrões de placas, o que dificulta a extração dos caracteres devido ao diferente padrão seguido. Verifica-se na Figura 1 a diferença no padrão de letra utilizada, sendo que nas placas antigas utilizava-se a fonte *Mandatory* e as placas Mercosul emprega-se a *Engschrift*, fonte anti-falsificação, deste modo o sistema OCR deve ser capaz de processar ambas fontes.

Figura 1 – Comparação entre os padrões de placas veiculares existentes no Brasil

Fonte: https://placafipe.com/converter-placa-para-mercosul

Neste momento em que ambos padrões coexistem foi criado pelo Conselho Nacional de Trânsito um padrão de conversão para a 5º posição dos caracteres, que no padrão Mercosul deve ser uma letra, devendo ser realizado conforme a Figura 2:

Figura 2 - Tabela de conversão entre números e letras

PLACA ANTIGA	PLACA NOVA
0	A
1	B
2	C
3	D
4	E
5	F
6	G
7	H
8	I
9	J

Fonte: https://www.detran.ce.gov.br/esclareca-suas-duvidas-sobre-a-placa-padrao-mercosul/

Considerando a grande frota de veículos, é salutar que se deseja ter um controle de quem pode acessar ou até mesmo apenas registrar o acesso a determinados locais: Condomínios fechados, *Shoppings Centers*, Repartições Públicas entre outros. Desta maneira pode-se utilizar o processamento de imagens e técnicas de aprendizagem de máquina para realizar a leitura de uma placa veicular, uma vez que se tem poder computacional para processar a imagem e ser capaz de extrair esses caracteres salvando-os em um banco de dados.

Sendo assim será possível um controle automatizado de quem e quando acessou determinado local, não sendo necessária a intervenção humana, sendo apenas disponibilizado um sistema supervisório para verificar o funcionamento do sistema e acesso ao banco de dados através de uma interface construída em PHP (*Hypertext Preprocessor*) para consulta e autorização dos veículos.

1.1 MOTIVAÇÃO

Neste trabalho, o problema a ser estudado é verificar a viabilidade de se criar um sistema de automatização para que o acesso de veículos a locais privado não necessite de intervenção humana, sendo esse processamento realizado remotamente. Para isso, ao final do trabalho será visto o desempenho do sistema e a capacidade de fazer essas tarefas de forma autônoma.

Em determinados locais necessita-se um maior controle dos veículos por questões de segurança, uma vez que o controle atualmente em alguns locais é inexistente ou falho, permitindo o acesso a pessoas/veículos desconhecidos. Sendo assim este trabalho visa a aumentar o controle de acesso uma vez que ficará os registros para cada entrada e saída do veículo.

1.2 JUSTIFICATIVA

Atualmente gastos com portaria são bem dispendiosos para as pessoas e organizações, pois dependem da intervenção humana para realizarem as tarefas de controle de acesso, este trabalho pode ser uma alternativa na economia e gerenciamento deste problema.

A proposta visa contribuir apresentando uma metodologia para implementação em locais que desejam um maior controle de acesso e sem a necessidade de uma pessoa para realizar essa tarefa, automatizando todo o processo.

1.3 OBJETIVOS

1.3.1 Objetivo geral

O objetivo geral deste trabalho consiste em desenvolver e verificar o desempenho de um sistema automatizado de controle de entrada de veículos em locais privados gerenciado e controlado remotamente.

1.3.2 Objetivos específicos

São objetivos específicos deste trabalho:

1) Escolha da câmera para captura das imagens dos veículos.
2) Escolha hardware para processamento das imagens.
3) Desenvolvimento de um *software* em Python para processamento das imagens e leitura das placas veiculares.
4) Configuração da comunicação entre os dispositivos.
5) Montagem e configuração do ESP8266 que fará o acionamento do portão.
6) Criação de um banco de dados para armazenar as imagens e os dados referentes a cada entrada de veículo.
7) Desenvolvimento de um sistema de supervisão para gerenciamento remoto.

8) Desenvolvimento interface em PHP para consulta dos veículos que acessaram o local.

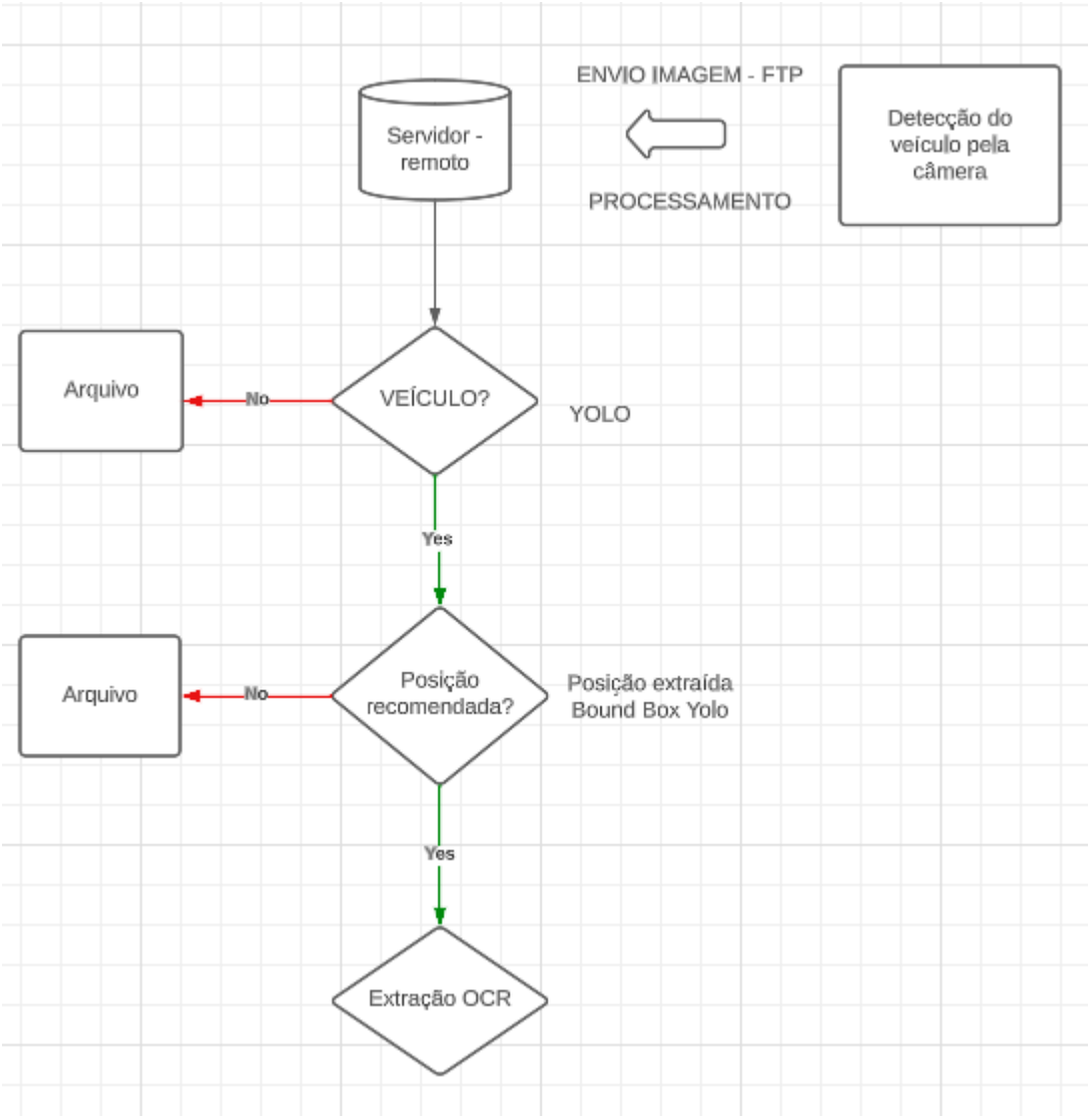
1.4 ESTRUTURA DO TRABALHO

Este trabalho está estruturado da seguinte forma: as referências teóricas são discutidas no Capitulo 2, juntamente com os trabalhos relacionados; no Capítulo 3 discute-se a metodologia e as ferramentas empregadas na resolução dos problemas propostos, o hardware e o *software* utilizado, juntamente com os protocolos de comunicação utilizados; no Capítulo 4 será visto o funcionamento descritivo de cada etapa do algoritmo e, no Capítulo 5, discutir-se-á os resultados obtidos e por fim, no Capítulo 6, as conclusões e as possíveis melhorias deste trabalho.

2. REFERENCIAL TEÓRICO

Neste capítulo será tratado sobre o referencial teórico utilizado na construção deste trabalho. Na Figura 3 mostra-se um organograma resumido do sistema proposto para melhor entendimento das etapas que compõe o desenvolvimento do projeto. A câmera realiza a detecção do veículo que envia via protocolo FTP para o servidor que fará o processamento da imagem. No servidor remoto utiliza-se o algoritmo YOLO para fazer a detecção do veículo, em seguida é extraído os caracteres das placas através de softwares (*Openalpr*, *Google Vision*). Na seção 3.8 será visto de maneira mais detalhada cada fase do trabalho proposto.

Figura 3 - Organograma resumido do sistema



Fonte: Autor (2023)

2.1 TRABALHOS RELACIONADOS

Segundo Prudente (2011) a automação predial e residencial pretende identificar todas aquelas tecnologias que permitem tornar automática uma série de operações no interior de um prédio ou habitação. Seguindo a linha IOT *(Internet of Things*), que consiste em uma revolução tecnológica que tem como objetivo conectar aparelhos utilizados no dia a dia a rede mundial de computadores, cada vez mais os aparelhos eletrônicos residências e comerciais tem adquiridos características de interação com usuário de forma remota, possibilitando o controle dos dispositivos através de computadores e celulares.

A aplicação da domótica, segundo Prudente (2011), pode ser em diversas áreas, entre elas, comando de veneziana, porta ou portão elétrico. Devido a melhorias dos hardwares ligados a comunicação, entre eles o Wi-Fi, a informação dos sensores e os comandos para os atuadores estão cada vez mais disponíveis e estáveis, desta maneira pode-se construir aplicações utilizando essas plataformas.

Conforme Albertin (2017), devido ao grande mercado consumidor e a melhorias de oferta de Internet em questões de custo e velocidade, tem feito com que haja diversas pesquisas e estudos na área da automação residencial. O trabalho de Merçon (2022) estudou a automatização da iluminação juntamente com o consumo energético de uma residência, para isso utilizou um Arduino Uno integrado ao ESP32 para conexão à Internet, dando ênfase a simplicidade e eficiência do projeto apresentado.

Pode-se encontrar trabalhos recentes na área de automação residencial, Santos e Lara Junior (2019) propõe um sistema de automação residencial utilizando ESP32 controlando via *smartphone*, onde foi criado dentro do ESP32 um servidor Web. Nesta linha Gonçalves (2019) propõe em seu trabalho as diversas possibilidades de aplicação da IoT, cujo propósito consiste em melhorar a qualidade de vida das pessoas.

O trabalho de Stopassole (2020) consistiu em criar um algoritmo capaz de detectar placas veiculares e classificá-las em padrão antigo ou novo (Mercosul). Para isso utilizou metodologias da visão computacional, como a detecção de contornos após realizar o tratamento da imagem. Em seu trabalho no capítulo de resultados, cita a dificuldade no *software* em trabalhar imagens com posicionamentos de veículos diferentes. Desta maneira o posicionamento correto do veículo é fundamental para a detecção e classificação da placa.

Um trabalho similar foi desenvolvido por Souza (2017) que consiste em localizar e segmentar os caracteres das placas veiculares utilizando visão computacional, e posteriormente

implementou uma rede neural artificial para verificar qual caractere estava contido na imagem. Para isso, utilizou técnicas de tratamento da imagem, como conversão em escala de cinza, posteriormente aplicou o algoritmo de *Canny* para detectar as bordas da imagem para poder descobrir o posicionamento da placa. Última etapa consistiu em treinar uma rede neural *Multi Layers Perceptron* com os caracteres extraídos, para que a mesma seja capaz de quando apresentado um caractere para a rede neural nos retornar o resultado conforme seu treinamento.

Amaral (2021) analisa em seu trabalho o desempenho de diferentes plataformas para detecção de placas veiculares bem como a extração dos seus caracteres, compara-se o desempenho do *software Microsoft Azure – Custom Vision AI* com a arquitetura YOLOV4 para detecção da placa. Para realizar o OCR comparou-se o desempenho do *Google Cloud Vision* com plataforma *Plate Recognizer*, ressaltando que em seu trabalho a plataforma YOLOV4 conseguiu melhores resultados em conjunto com o *Plate Recognizer*, que segundo o autor, esse melhor desempenho se deu devido ao treinamento específico para essa tarefa.

2.2 INTELIGÊNCIA ARTIFICIAL

Desde o surgimento dos primeiros computadores, por volta de 1946, o tema inteligência artificial (IA) vem crescendo exponencialmente, os primeiros testes realizados por Alan Turing, em 1950, tinham como propósito verificar se uma máquina possuía inteligência. Segundo Yoneyama (2004), conceituar IA é uma tarefa difícil, uma vez que existem diversos autores conceituando o referido tema, para esse trabalho optou-se pela seguinte definição: IA é o estudo de como fazer os computadores realizarem tarefas que, no momento, são feitas melhor por pessoas (RICH, 1983).

Diversas ferramentas são utilizadas em IA, neste projeto foram utilizadas as redes neurais artificiais (RNA), sendo essas utilizadas para aprendizado pelos *softwares* de extração de caracteres (OCR) utilizados neste trabalho.

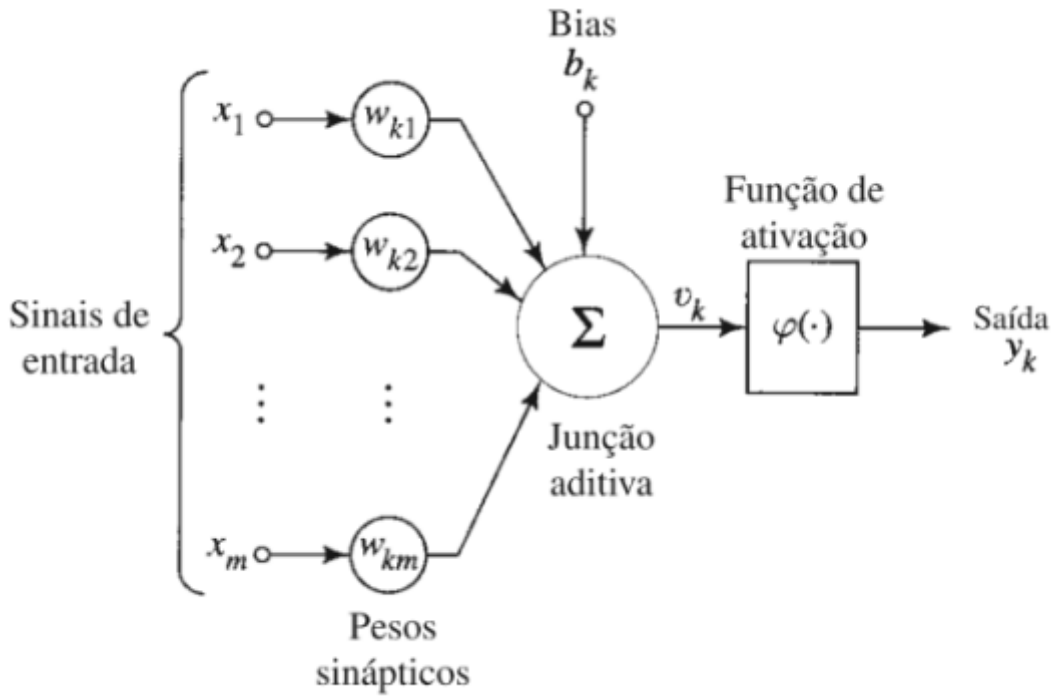
2.3 REDES NEURAIS ARTIFICIAIS

A espécie de homem *homo sapiens* domina atualmente o planeta Terra. Essa soberania advém de sua inteligência em se sobressair e se adaptar a novos ambientes, tendo sua inteligência o principal motivo de suas conquistas. Por isso, o aprendizado de máquina, baseado

em RNA, se espelham na estrutura humana do cérebro, esse constituído por cerca de 10^{10} a 10^{11} neurônios e por cerca de 5000 sinapses por neurônio, para Yoneyama (2004) o sistema de inteligência é dinâmico, realimentado, não linear, massivamente paralelo e assíncrono.

O RNA é composto por um neurônio artificial, que por sua vez é ligada a outro neurônio ou a entrada através de pesos, possuindo um bias a sua estrutura, essas duas variáveis (pesos e bias) são controladas para fazer com que nossa RNA seja capaz de realizar a tarefa com exatidão para o que foi projetada, ao final insere-se uma função de ativação, responsável pela comparação do valor com a saída desejada. A Figura 4 mostra a estrutura de um neurônio artificial, composto pelos pesos, bias, função de ativação e uma saída.

Figura *4* - Estrutura neurônio artificial



Fonte: Haykin (2007)

Desta maneira, uma RNA tem a função de processar conhecimento, armazenar e estar disponível para uso quando solicitado, para isso, deve-se treinar a RNA para a função que desempenhará, esse treinamento consiste na atualização dos pesos e bias a cada verificação da saída obtida com a saída desejada.

2.3.1 Funções de ativação

Como visto na seção 2.3, o aprendizado de máquina baseia-se na mudança de pesos e bias da rede neural, uma pequena mudança nesses valores causaria variação grande na saída da rede e consequentemente o aprendizado ficaria prejudicado, por isso, em nossa estrutura de RNA foi inserido a função de ativação, que sua função consiste basicamente em decidir se aquele neurônio específico deve ser ativado ou não. Desse modo a função de ativação é responsável pela não linearidade da rede, pode-se citar diversas funções de ativação usadas em RNA: Linear, Sigmoide, Tangente hiperbólica, ReLU, Softmax entre outras.

2.3.2 Descida do gradiente

Como estudado em cálculo, o gradiente de uma função, consistem em descobrir as derivadas parciais dessa função em relação a suas entradas, dessa maneira, consegue-se descobrir para quais valores na entrada minha função cresce mais rapidamente.

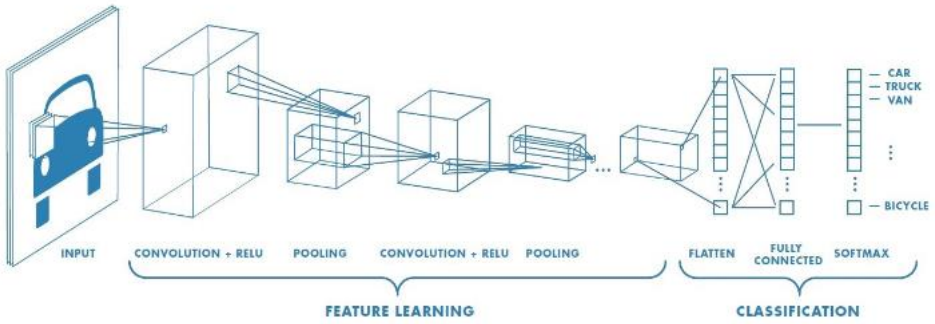
Uma das maneiras de otimizar o sistema de aprendizagem de nossa RNA, consiste em aplicar a descida do gradiente, o qual pode ser visto como uma “seta” apontando para a direção do maior decréscimo da função do erro da rede, por isso vai-se na direção oposta do gradiente calculado.

2.4 REDES NEURAIS CONVOLUCIONAIS (RNC)

É um tipo específico de redes para processamento de dados, possui um procedimento matemático o qual tem o nome de convolução. Essa rede se distingue das demais pelo meio de ligação dos neurônios, pois no método tradicional, estes se ligam a todas as camadas posteriores. Nas RNC cada neurônio se conecta a uma pequena quantidade de unidades da próxima camada. Deste modo pode-se classificar essas camadas em convolução, *pooling*, normalização, camadas totalmente conectadas, entre outras.

Uma RNC pode tratar de um volume grande de dados, uma vez que, consegue-se compactar os dados sem perder informação, como visto na Figura 5, aplicando diversas camadas os dados diminuem de dimensão, aumentado a profundidade e ao final transforma-se para um vetor coluna para comparação com a saída desejada.

Figura *5* - Estrutura Rede Neural Convolucional



Fonte: Saha (2018)

2.5 YOLO (YOU ONLY LOOK ONCE)

Consiste em um algoritmo de detecção de objetos que utiliza uma Rede Neural Convolucional para realizar a predição em imagens. Foi desenvolvido inicialmente em 2015, por Joseph Redmon e Ali Farhadi. Sua principal característica é que a imagem passa apenas uma vez (*Single Pass*) pela RNC, isso torna o método mais rápido que as predições comumente utilizadas, e de onde vem a derivação do nome: “Você só olha uma Vez”.

Conforme a Figura 6, a YOLO inicialmente divide a imagem em uma grade e atribui a cada célula da grade várias caixas delimitadoras (*Bounding Boxes*) e as probabilidades dessas caixas conterem o determinado objeto, o próximo passo é ajustar estas caixas para aumentar sua precisão. Para isso o algoritmo divide a imagem em *S x S* células, sendo esse tamanho adotado conforme a versão da YOLO. Em cada uma das células são previstas N (algarismo ilustrativo) possíveis caixas delimitadoras e a confiança atribuída a cada caixa. O sistema utiliza um filtro, o limiar de confiança, e as caixas abaixo desses valores são descartadas.

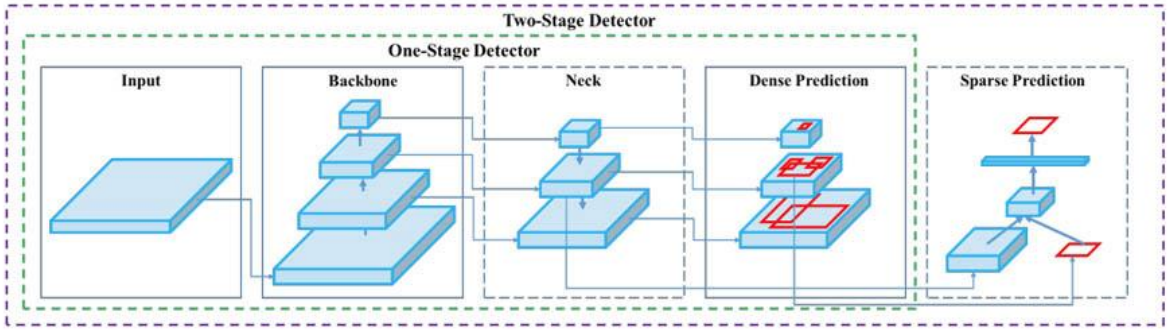
Para que em uma mesma célula não haja várias detecções o sistema utiliza um mapa de probabilidades de classes, desse modo, apenas uma classe pode ser atribuída por célula, independentemente do número de caixas delimitadoras. Cada caixa delimitadora possui alguns atributos, como intersecção pela união, a precisão da caixa, e as posições na imagem (*x,y,w,h*), onde *(x,y)* representa o posicionamento central e (*w,h*) representa sua altura e largura respectivamente.

Figura 6 – Regressão, o método YOLO

Fonte: IA EXPERT ACADEMY (2020)

Representa-se a arquitetura YOLO por 3 estágios: *Backbone, Neck e Head,* conforme a Figura 7. Sendo o *backbone* uma rede neural profunda composta por camadas de convolução, tendo por objetivo extrair os atributos essenciais da imagem. O *neck* sua função é coletar mapas de características de diferentes estágios, por fim o *Head* que tem finalidade realizar a previsão densa, no qual é composta por um vetor contendo as coordenadas da caixa delimitadora detectada e a confiança da previsão de cada rótulo.

Figura 7 – Arquitetura geral detector de objetos de um estágio



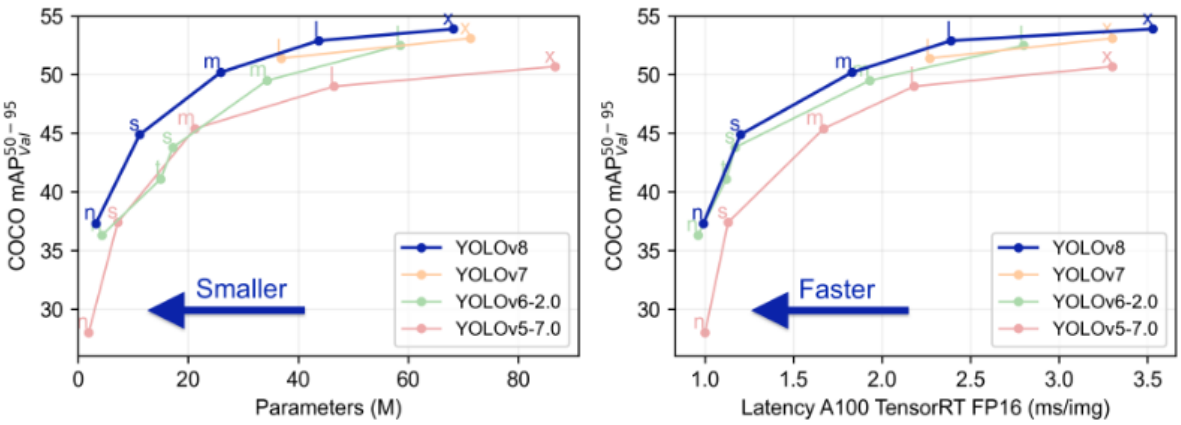
Fonte: https://aws.amazon.com/pt/blogs/aws-brasil/obtenha-desempenho-185-vezes-maior-para-deteccao-de-objetos-baseada-em-aprendizado-profundo-com-um-modelo-de-yolov4-compilado-pelo-aws-neuron-na-aws-inferentia/

Desde sua criação já foram lançadas diversas versões do *software*, todas elas de código aberto, sendo a última versão (V8) do desenvolvedor Ultralytics, a qual foi utilizada neste trabalho. Foi escolhido a versão 8 devido a mesma se apresentar estável e de fácil implementação, como verifica-se na implementação do algoritmo, permite treinar seus próprios

modelos e utilizar ou não uma GPU para processamento das imagens, sendo considerado um estado da arte devido ser um modelo de última geração, pois apresenta resultados ligeiramente melhores que sua última versão.

Na Figura 8, compara-se a rapidez e acurácia com as demais versões da YOLO, sendo o eixo X representado respectivamente pelos parâmetros de entrada e latência no processamento. No eixo Y tem-se a precisão média das predições *(mean average precision)*, onde constata-se que a última versão da YOLO tem um melhor desempenho comparado às anteriores. Ambas confrontações foram feitas no *dataset* COCO, o qual foi publicado pela Microsoft , e possui um variado conjunto de imagens para treinamento de predições usando aprendizado de máquina supervisionado.

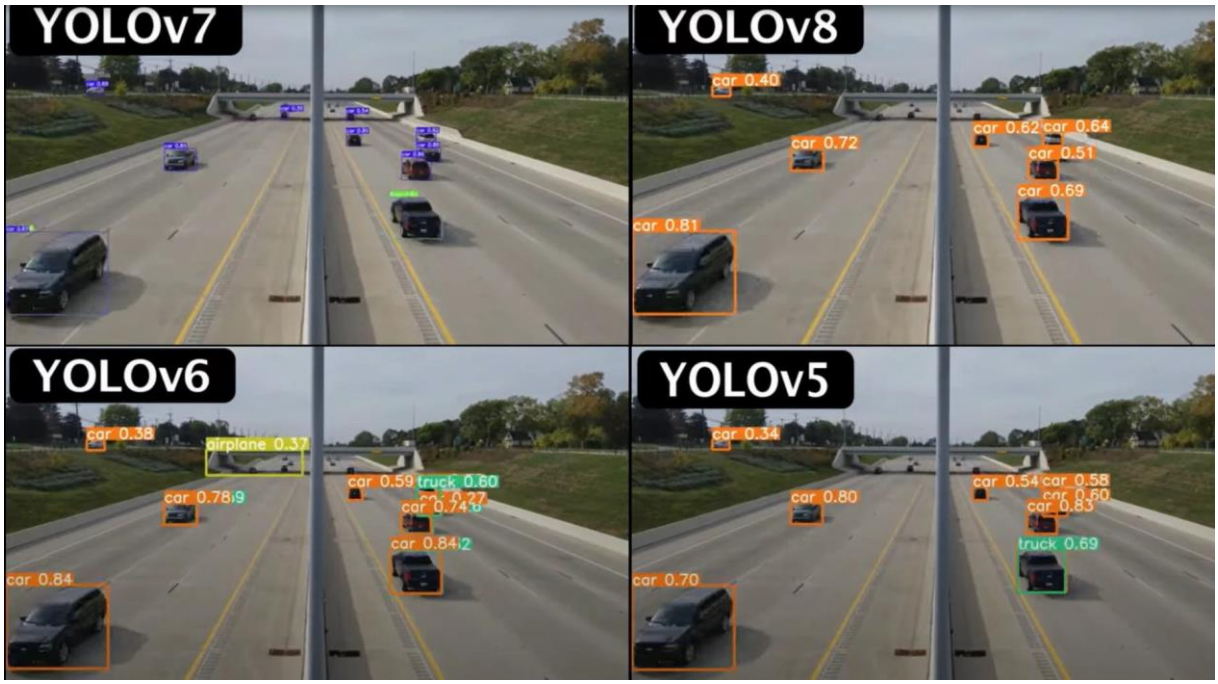
Figura 8 – Comparação resultados com outras versões da YOLO



Fonte: Github Ultralytics (2023)

Segundo JINDAL (2023), esta última versão da YOLO introduziu um novo Backbone Network, o qual é mais rápido e com melhor acurácia do que o utilizado na versão 7. Ainda possui uma rede neural Convolucional com 53 camadas profundas e pode classificar imagens em até 1000 categorias. Uma inovação foi o uso das *Features Pyramid Network*, as quais auxiliam na predição de objetos com diferentes tamanhos, pois utilizam diferentes escalas nos *features maps*. Na Figura 9 compara-se o as predições em diferentes versões da YOLO, sendo os resultados obtidos, apesar de próximos, apresentam algumas diferenças entre as imagens.

Figura 9 – Comparação predições entre diferentes versões da YOLO



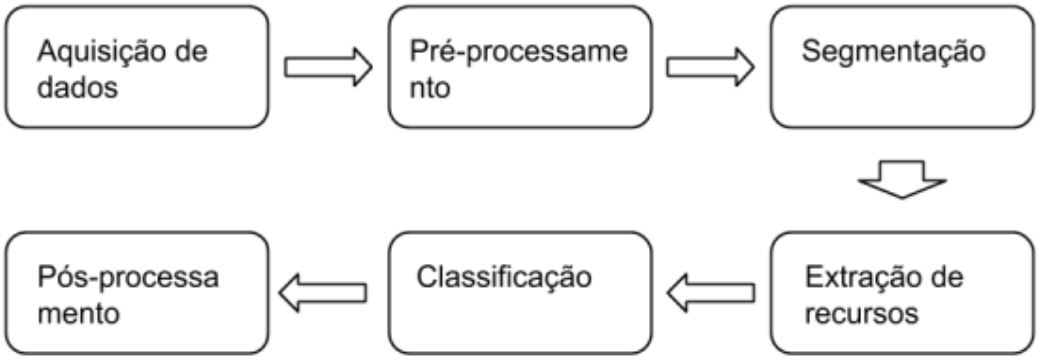
Fonte: https://encord.com/blog/yolo-object-detection-guide/

2.6 SISTEMA DE EXTRAÇÃO DE TEXTO EM IMAGENS (OCR)

Um sistema de reconhecimento ótico de caracteres (OCR) tem por finalidade reconhecer caracteres a partir de um arquivo de imagem ou de um mapa de bits, desta maneira pode-se de uma imagem ter um arquivo de texto que estará disponível para a finalidade que se deseja na aplicação proposta.

Um sistema OCR contém diversos componentes, conforme Feijó (2017), possui a seguinte sequência de etapas, conforme a Figura 10: aquisição de dados, pré-processamento, segmentação, extração de recursos, classificação e pós processamento. Na primeira etapa, adquire-se os dados através de uma câmera ou uma imagem de um arquivo, em seguida se aplica um pré-processamento com a finalidade de melhorar a imagem para extração dos caracteres, já na segmentação os caracteres são separados e extraídos para serem identificados, e na última etapa os caracteres são reconstruídos em palavras novamente.

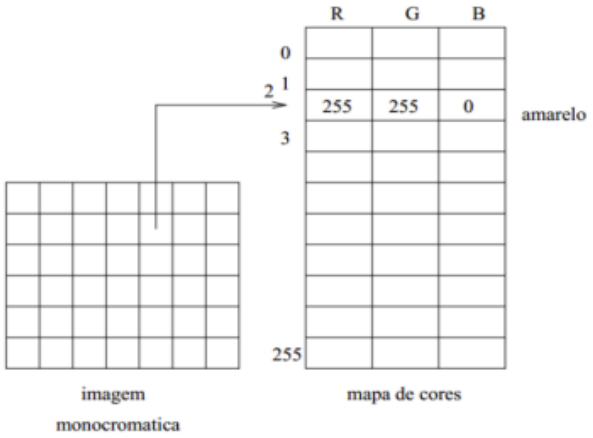
Figura 10 – Etapas em um sistema OCR



Fonte: Feijó (2017)

Para uma imagem colorida, existe a combinação de três cores primárias: Vermelho, verde e azul, sendo que cada imagem é composta pela junção de vários pixels, e que cada pixel tem profundidade de 1 *byte* por pixel. Os *softwares* de extração de caracteres de imagens utilizados neste trabalho (*Openalpr e Google Vision*), utilizam procedimentos para tratamento dessas imagens, sendo que um dos processos consiste em reduzir os valores das entradas RGB, alterando para apenas um canal, transformando a imagem colorida para preto e branco. A Figura 11 mostra o mapeamento de uma imagem colorida no formato RGB:

Figura *11* - Uso de Mapa de cores para imagens colorida

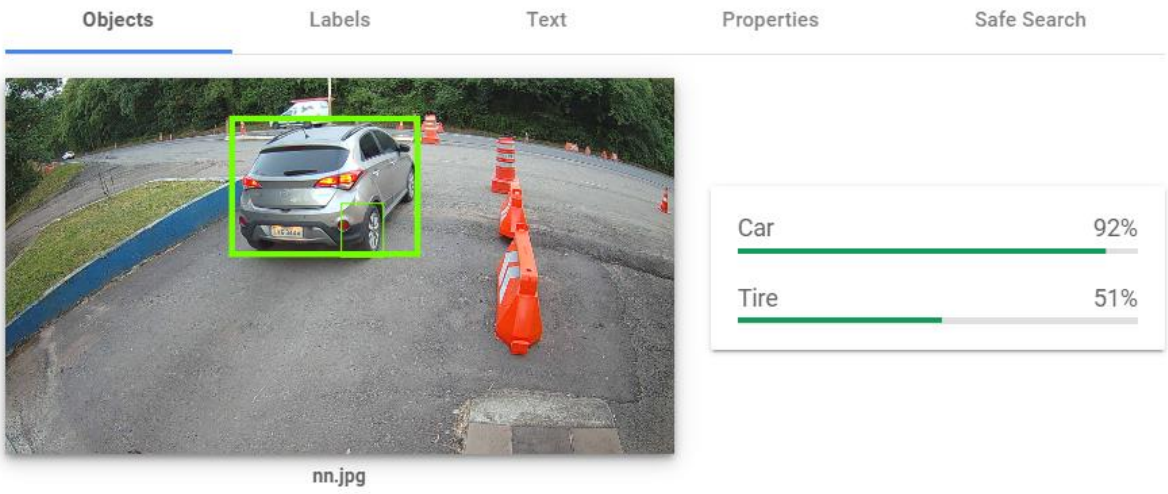


Fonte: Martins (2016)

2.7 GOOGLE VISION AI

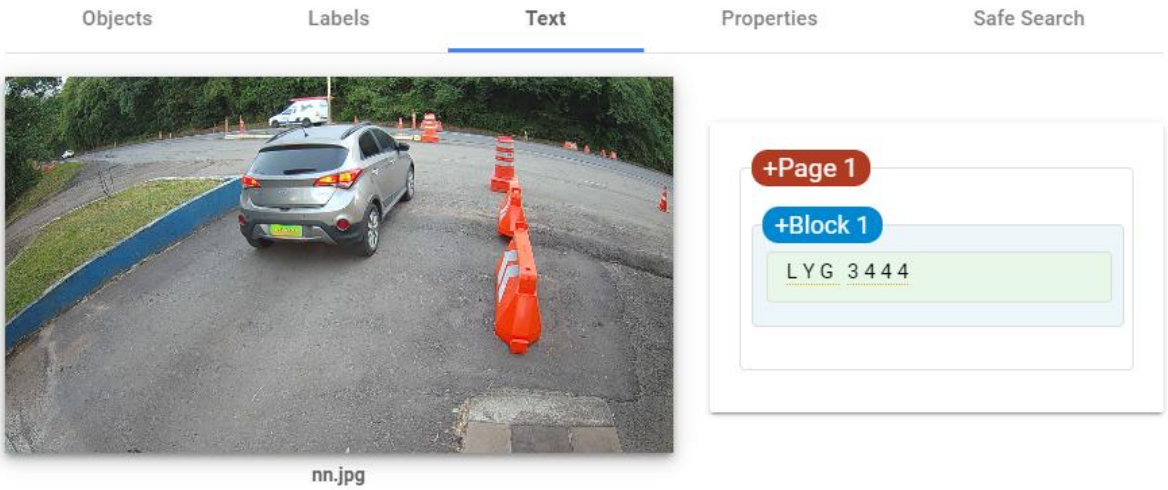
Consiste no processamento de imagens pelo Google através de uma API, sendo capaz de detectar objetos em uma imagem e sua posição, extrair textos, e as propriedades da imagem apresentada. Utiliza modelos de aprendizagem profunda, ou seja, redes neurais convolucionais, para realizar o processamento dos conteúdos das imagens, sendo permitido, caso seja necessário, treinar seu próprio modelo de detecção. É permitido realizar até 1000 consultas por mês de forma gratuita, após esse valor, uma tarifa é adicionada conforme o uso. Na Figura 12, analisa-se o resultado da predição de uma imagem referente aos objetos da imagem, na Figura 13 os textos detectados, e por último, na Figura 14 as propriedades da imagem, onde constam as suas cores dominantes.

Figura 12 – Detecção de objetos realizada pelo Google Vision



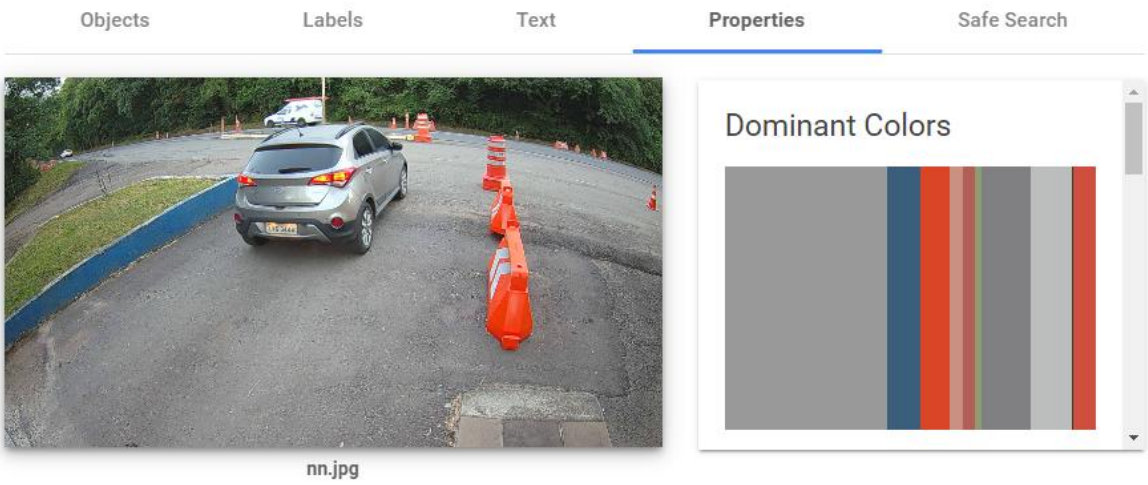
Fonte: Autor (2023)

Figura 13 - Detecção de textos realizada pelo Google Vision



Fonte: Autor (2023)

Figura 14 - Detecção de das propriedades da imagem realizada pelo Google Vision



Fonte: Autor (2023)

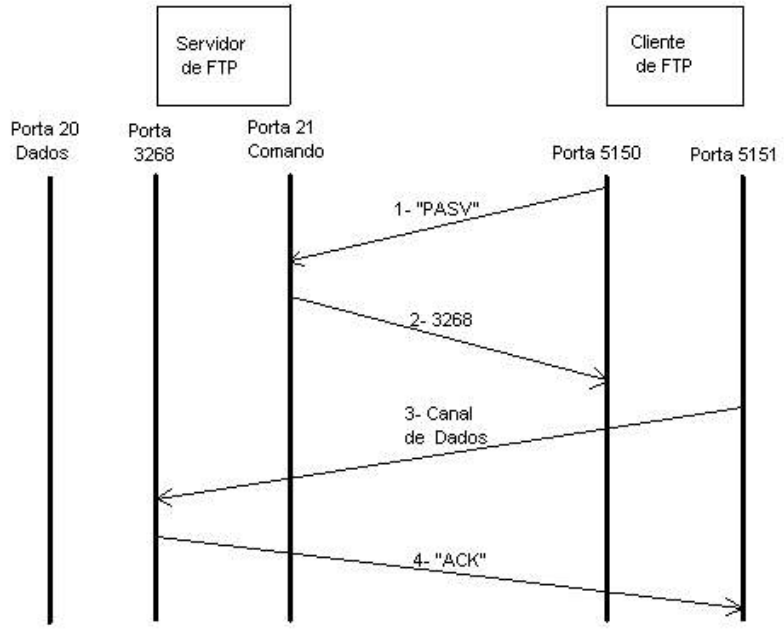
2.8 PROTOCOLO FTP

Este protocolo por estar disponível em nossa câmera de captura e atender os requisitos do trabalho, foi utilizado para envio das imagens para o servidor. Consiste em um protocolo da camada de aplicação do modelo ISO/OSI, especificado na RFC 959, utilizado para *download* e

upload de arquivos em conexões do tipo cliente/servidor. Em nossa aplicação a câmera será o cliente e o computador rodará um servidor FTP.

Podem ser estabelecidas dois tipos de conexões FTP: Ativo e o passivo. No modo ativo o cliente estabelece o canal de comandos (porta de conexão) e quando recebe a resposta do servidor, o cliente escolhe uma segunda porta para estabelecer a troca dos dados, geralmente acima de 1024. Para o modo passivo o cliente estabelecerá ambos canais de comunicação, ou seja, o de comandos e dados, quando o cliente se conecta no servidor através da porta de comando, o servidor já envia qual faixa de portas está disponível para troca de dados, assim o cliente somente pode escolher a porta dentro da faixa disponível. No trabalho utilizou-se o FTP passivo devido a evitar problemas de *firewall*. Na Figura 15 segue um exemplo da comunicação FTP no modo passivo, onde o cliente estabelece as portas de comunicação que serão utilizadas pelo servidor.

Figura 15 – Exemplo comunicação para o FTP Passivo



Fonte: https://www.pop-rs.rnp.br/~berthold/etcom/redes2-2000/trabalhos/FTP_EltonMarques.htm

Uma das vulnerabilidades desse protocolo, apesar de a conexão ser autenticada com nome de usuário e senha, consiste que seus dados não são criptografados ao ser enviado pela rede,

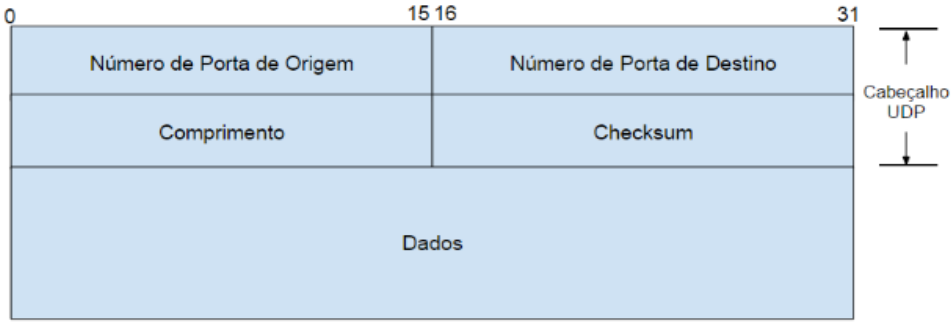
deste modo, ao ser interceptado pode-se ter acesso ao seu conteúdo, quando este requisito é fundamental para a aplicação, pode-se utilizar o protocolo SFTP que possui a criptografia dos dados e autenticação segura.

2.9 PROTOCOLO UDP

Protocolo da camada de transporte do modelo ISO/OSI, que tem como principal característica não haver estabelecimento de conexão para envio dos dados, o que torna o protocolo mais rápido, se comparado ao TCP, por exemplo. É utilizado em aplicações mais simples ou que não demandam a certeza da confirmação das trocas de mensagens. Por isso o protocolo se torna mais rápido nas trocas dos datagramas. Pode-se citar o uso deste protocolo em sistemas VoIP - telefonia, pois nessa comunicação prefere-se que o atraso na comunicação seja mínimo a ter certeza da entrega dos pacotes.

O datagrama UDP é encapsulado em um pacote IP e possui estrutura bem simples. Seu cabeçalho possui tamanho de 8 bytes e contém apenas o número da porta de comunicação de origem e destino, comprimento da mensagem e o *checksum*, este último tem a finalidade de verificar erros do cabeçalho e dos dados transmitidos, conforme Figura 16.

Figura 16 – Estrutura do cabeçalho pacote UDP



Fonte: http://www.bosontreinamentos.com.br/redes-computadores/curso-de-redes-protocolo-udp-user-datagram-protocol/

3. MATERIAIS E MÉTODOS

Neste capítulo será visto a metodologia e os materiais necessários para construção deste trabalho.

3.1 CÂMERA DE SEGURANÇA

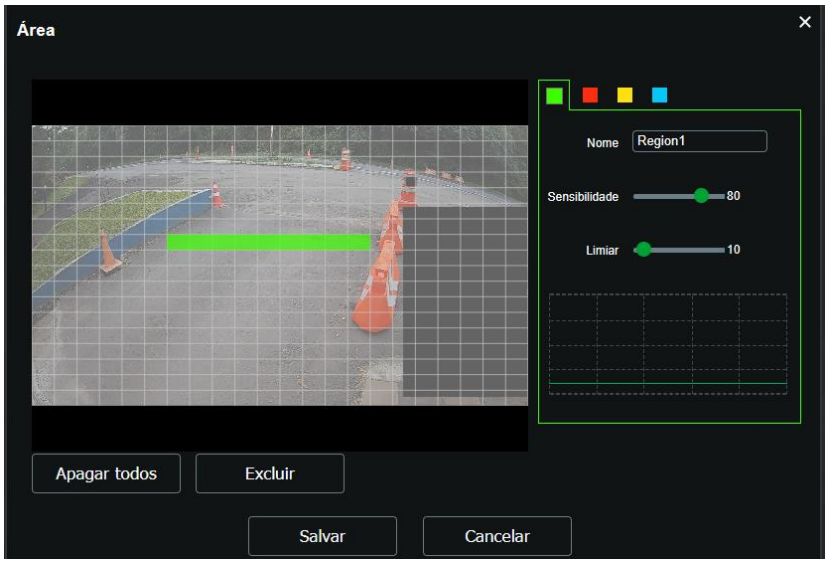
Para realizar a função da captura das imagens, foi escolhido conforme a Figura 17, a câmera IP do fabricante Intelbras, modelo VIP-3260-Z-G2. Foi dada preferência para o modelo IP, deste modo é possível o gerenciamento remoto ao *hardware* para realizar os ajustes necessários para a melhor captura da imagem, esta versão possui grau de proteção IP67, ou seja, pode ser instalada diretamente em ambientes externos, pois é resistente a umidade e poeira.

Figura 17 – Câmera Intelbras VIP-3260-Z-G2

Fonte: Intelbras (2023)

Uma importante característica que a câmera deve possuir e que foi utilizado nesse projeto, consiste em ter a capacidade de detectar movimento em uma área específica e enviar a imagem para o o servidor via protocolo FTP. Como observa-se na Figura 18, é possível configurar a área de interesse da imagem, que quando a câmera detectar alguma mudança na imagem irá enviá-la para processamento, sendo escolhido uma sensibilidade alta, ou seja, ocorrendo uma mínima mudança na imagem, essa já será processada e enviada ao servidor.

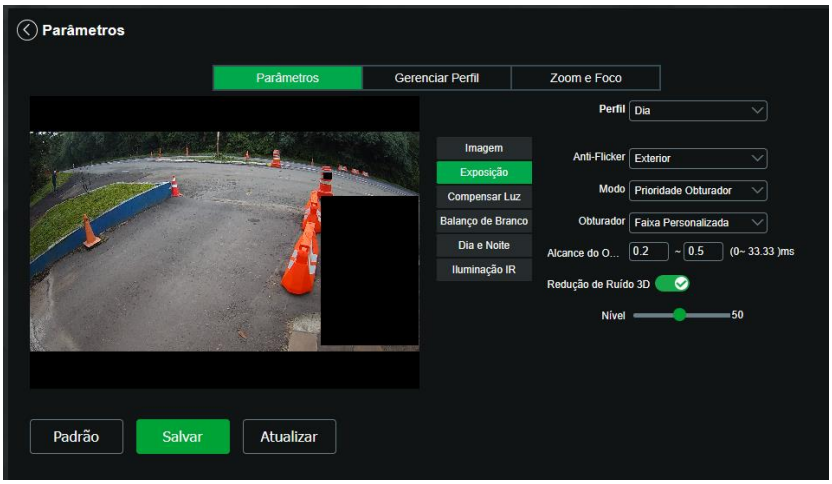
Figura 18 – Configuração da área de detecção na câmera Intelbras VIP 3260



Fonte: Autor (2023)

Um dos objetivos deste trabalho, consiste em verificar se a velocidade de processamento da imagem e abertura do portão são razoáveis para que usuário não fique demasiadamente esperando o processamento. Uma característica importante que a câmera deve possuir, é que permita ajuste no tempo do obturador, o qual é responsável por conseguir tirar fotografias de objetos em movimento, desta maneira não há necessidade de esperar que o carro para em uma posição determinada, pois ao aproximar-se do portão a câmera já é capaz de realizar a captura da imagem de forma que seja possível extrair os caracteres da placa. Desse modo, conforme a Figura 19, configura-se a câmera para ter prioridade para o obturador com abertura em uma faixa entre 0,2 e 0,5 ms.

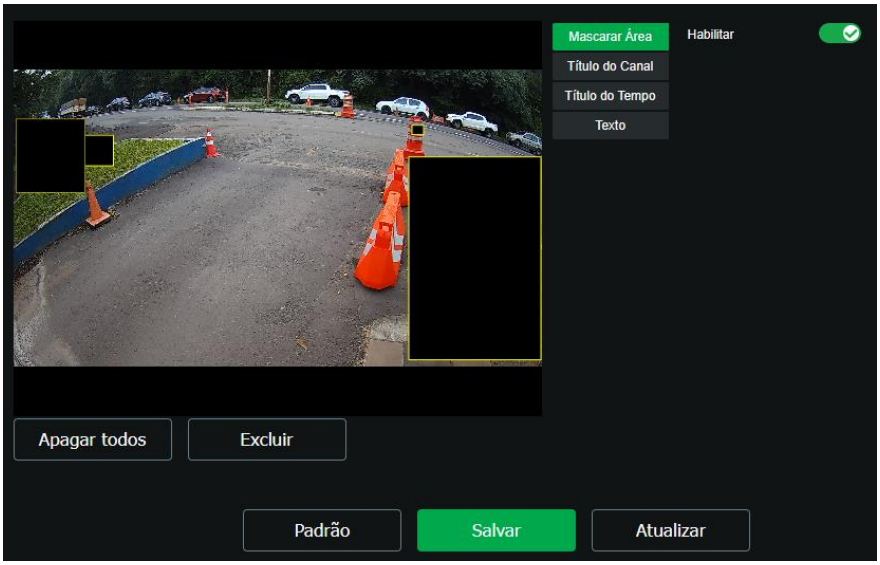
Figura 19 – Configuração exposição da imagem



Fonte: Autor (2023)

Para otimizar o funcionamento do *software*, a câmera possui a possiblidade de mascarar uma determinada área, ou seja, pode-se desconsiderar algumas partes das imagens, as quais não interferem na aplicação. Desta maneira pode-se concentrar em uma região de interesse e otimizar o algoritmo de processamento, seja na detecção do veículo na posição determinada ou na extração do OCR. Essa ferramenta foi utilizada como demonstrado na Figura 20.

Figura 20 – Mascaramento de áreas que não são interesse da aplicação.



Fonte: Autor (2023)

Para a noite, quando a luminosidade do local fica prejudicada, utiliza-se a configuração automática da câmera quanto aos parâmetros de velocidade do obturador, compensação de luz entre outras. Desta forma quando o veículo está se movimentando a imagem não fica nítida o suficiente conforme a Figura 21, sendo necessário aguardar até a parada na posição correta para uma nova imagem, esta sim com capacidade de leitura dos caracteres.

Figura 21 – Veículo em movimento em baixa luminosidade, incapacidade da leitura OCR

Fonte: Autor (2023)

3.2 COMPUTADOR PARA PROCESSAMENTO DAS IMAGENS

O *hardware* é composto por um computador com processador da Intel I3, tendo 8 GB de RAM, disco rígido SSD tipo NVMe com capacidade de 500 GB, com sistema operacional Windows 10 instalado, conectado à Internet através de uma conexão de 400 Mbps através de uma placa de rede Gigabyte.

Para realizar o processamento da imagem, foi incorporado ao hardware uma placa de vídeo da fabricante NVIDIA, modelo GTX 970, com isso pode-se usar uma GPU para acelerar o processamento da imagem, o que acarretou um melhor desempenho no processamento pela YOLO. Com a GPU ativada reduziu-se cerca de 10 vezes o tempo de resposta, sendo que em torno de 70 a 80ms o algoritmo desenvolvido em Python retorna os objetos da imagem, conforme a Figura 22.

Figura 22 – Tempo de processamento de uma imagem com uso da GPU

```
image 1/1 C:\Users\Andre Amaral\Desktop\cam_gate\AKOK2102037T9\m.jpg: 384x640 1 person, 2 cars, 73.5ms
Speed: 0.0ms preprocess, 73.5ms inference, 1.0ms postprocess per image at shape (1, 3, 640, 640)
yc2:  [608, 241]
```

Fonte: Autor (2023)

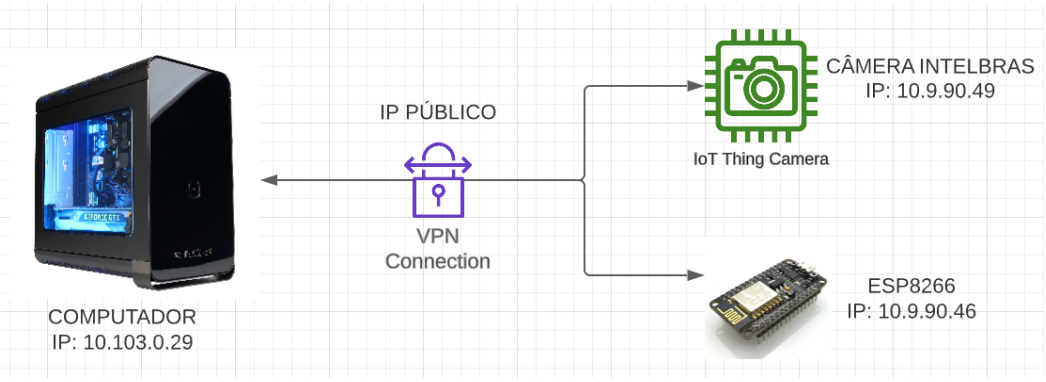
3.3 COMUNICAÇÃO DE DADOS ENTRE OS EQUIPAMENTOS

3.3.1 VPN (Virtual Private Network)

Por questões de segurança, optou-se em utilizar uma rede virtual privada entre os equipamentos que controlam o portão e o computador que processa os dados e envia novamente o comando para abertura do mesmo. Utilizando uma VPN os dados estão criptografados e impede que alguém com um *sniffer* de rede consiga descobrir a comunicação trocada entre os dispositivos da rede.

Para estabelecimento desta conexão foi utilizado o *software FortClient* VPN, e a configuração dos IPs da rede seguem conforme Figura 23.

Figura 23 – Diagrama da rede



Fonte: Autor (2023)

3.3.2 *Software* Filezzila Server (FTP)

Para recebimento das imagens da Câmera IP, foi instalado um servidor FTP, sendo escolhido o *software Filezzila Server* na versão 1.6.1 por ser de uso gratuito. Demonstrou-se ser estável e permitir taxas de transmissões condizentes com a aplicação. Desse modo configurou-se o servidor na câmera passando o IP, login e senha do servidor FTP. Optou-se por

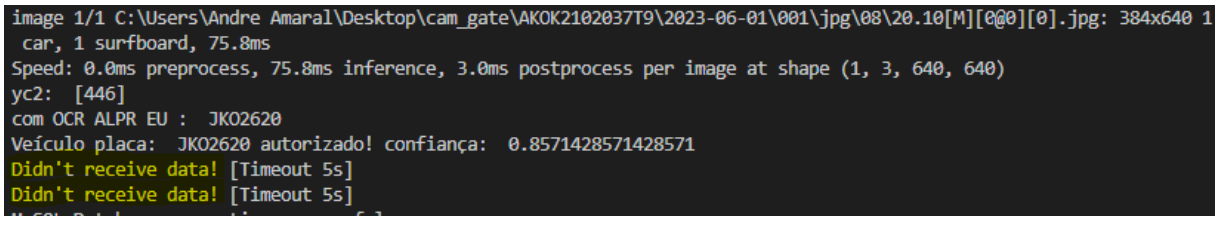
passar o IP Público do computador para a configuração do FTP. Desse modo para o envio das imagens não é necessário o uso da VPN, apenas restringiu-se tal uso para envio do comando de abertura/fechamento via UDP.

3.3.3 UDP

A comunicação através o protocolo UDP é a responsável pelos comandos de abertura e fechamento do portão, juntamente com a verificação se o mesmo está aberto ou fechado. Informação obtida através do sensor de fim de curso, sendo após a leitura da placa veicular autorizada a entrar no local ou através do sistema supervisório com a autorização do operador. Para isso foi configurado no ESP8266 o servidor UDP que faz as leituras dentro do *loop*, e recebendo o código correto envia o comando para o controle do portão através de um transmissor de rádio frequência como apresentado na seção 3.6.2.

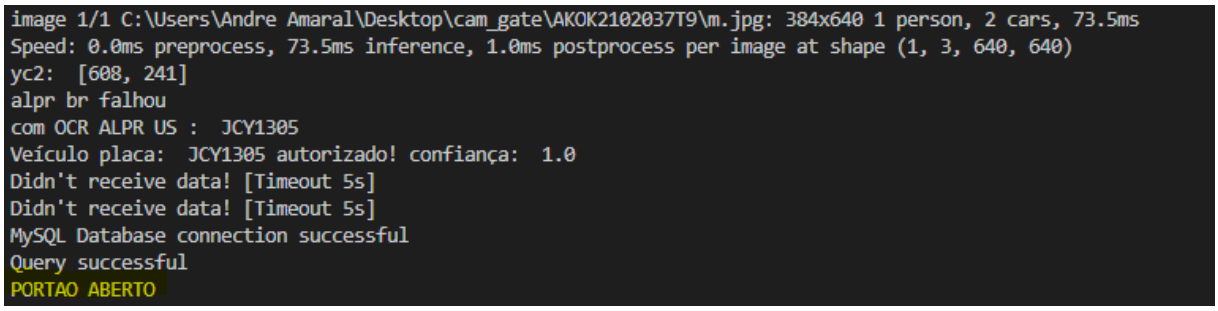
Devido o protocolo UDP não ser orientado a conexão, ou seja, apenas a aplicação envia na rede o comando sem se preocupar se o servidor está escutando e consequentemente não existe a confirmação ou não do recebimento dos pacotes enviados. Configura-se no servidor para que o mesmo envie ao cliente a resposta ao pacote recebido. Desta maneira a aplicação consegue verificar se o pacote foi recebido com sucesso. Verifica-se na Figura 24 a falta de resposta do servidor o que significa que o pacote UDP não foi recebido, resultando o não processamento pelo ESP8266, uma vez que a informação não chegou até o servidor UDP. Sendo na Figura 25 a resposta de que o portão está aberto e não é necessário abri-lo, desta forma a aplicação sabe que o pacote UDP foi processado.

Figura 24 – Problemas no recebimento do pacote UDP



Fonte: Autor (2023)

Figura 25 – Confirmação do pacote UDP



Fonte: Autor (2023)

3.*4* *SOFTWARES* OCR

Para extração dos caracteres das placas, foram incorporados ao programa principal dois *softwares*, ambos adaptados para aplicação na linguagem Python: *Openalpr e Google API Vision.*

3.4.1 Openalpr

Utilizado na versão 2.3.0, *software Open Source*, o qual é capaz de detectar a placa do veículo e extrair seus caracteres, foi desenvolvido na linguagem C++. Pode ser compilado em Python, tendo como base de dados de treinamento veículos de diversos países. Sendo assim para melhor detecção utiliza-se a base de dados do Brasil, EUA, e Europa, sendo priorizado o padrão Brasileiro, e caso este não detecte nenhuma placa, passa-se para as outras possibilidades.

Como ferramenta de extração de caracteres o *software* utiliza a biblioteca Tesseract, que tem sua arquitetura baseada em um tipo de rede neural recorrente: *Long Short Term Memory (LSTM),* consiste ser uma rede em *loop*, permitindo que as informações persistam dentro do laço de aprendizado. Para o tratamento da imagem o *Openalpr* utiliza a biblioteca de processamento de imagem de código aberto: *OpenCv* na versão 2.4.8.

3.4.2 API Google Vision – OCR

Como redundância para a extração dos caracteres das placas, quando o *software Openalpr* não for capaz de detectar uma região de interesse, ou seja, a placa veicular, utiliza-se

a API do *Google Vision*, que consiste em passar uma imagem e o algoritmo é capaz de retornar os objetos e os textos desta imagem.

Tendo em vista que o algoritmo devolve toda a leitura de caracteres existente na fotografia, na maioria das vezes deve-se fazer o tratamento deste resultado para buscar somente a placa veicular. Para efetivar esse filtro, utiliza-se uma *regex*, a qual consiste em uma expressão regular, um padrão que é buscado no texto. Sendo implementada em *Python*, busca-se o padrão das placas existentes no país atualmente utilizando a seguinte *regex*: *re.search("[a-zA-Z]{3}[0-9][A-Za-z0-9][0-9]{2}"*. Com isso o algoritmo busca dentro da lista devolvida pela API uma sequência de *strings* contendo 3 letras, 1 número, 1 número ou 1 letra e ao final 2 números. Na Figura 26 demonstra-se um exemplo do filtro aplicado.

Figura 26 – Criação de uma *regex* para aplicar nos resultados API *Google Vision*

```
image 1/1 C:\Users\Andre Amaral\Desktop\cam_gate\AKOK2102037T9\2023-06-02\001\jpg\11\31.31[M][0@0][0].jpg: 384x640 3
 cars, 2 trucks, 74.8ms
Speed: 1.0ms preprocess, 74.8ms inference, 2.0ms postprocess per image at shape (1, 3, 640, 640)
yc2:  [254, 121, 517]
alpr br falhou
ALPR US2 falhou
['TuntorEFFS4B89', 'Tuntor', 'EFFS4B89']
com OCR GOOGLE :  FFS4B89
```

Fonte: Autor (2023)

3.5 BANCO DE DADOS

Com a finalidade de manter os registros de acesso e a melhoria do sistema, é salvo em um banco de dados utilizando *Mysql*, onde cria-se as tabelas de veículos autorizados a acessar o local, dos acessos onde consta a placa do veículo, data e hora e qual a confiabilidade da leitura e o OCR utilizado. Portanto verifica-se o momento exato em que o veículo acessou o local e o índice de acerto do programa e na seção 4.5.2 mostra-se como recuperar a imagem do acesso.

3.6 HARDWARES

Neste tópico será detalhado os *hardwares* utilizados e suas características, que são responsáveis pela abertura e controle do portão: ESP8266, transmissor 433 MHz e a chave fim de curso.

3.6.1 ESP8266

Para este projeto, foi escolhido o microcontrolador ESP8266, Figura 27, devido o mesmo possuir interface de comunicação *Wi-Fi* para acesso à Internet, portas digitais que foram utilizados duas, uma para o sensor fim de curso, outra para acionamento do relé. Para aumentar a confiabilidade do sistema, foi ativado a função *watchdog timer*, a qual verifica constantemente se o *software* está travado dentro do loop e caso isso aconteça, reinicia automaticamente o microcontrolador.

Figura 27 – Microcontrolador ESP8266

Fonte: https://www.casadarobotica.com/internet-das-coisas/placas/esp/placa-esp-nodemcu-v3-wifi-802-11-b-g-n

3.6.2 Transmissor 433 MHz

Esse componente, conforme Figura 28, consiste em um controle remoto na frequência de 433 MHz deverá ser configurado na central do portão eletrônico para que o mesmo passe a aceitar os comandos provenientes do controle. Quando a saída normalmente aberta do relé é ativada por cerca de 2 segundos, o controle está no modo ligado e envia o sinal para o portão, o qual encontra-se aberto e realiza o fechamento, e caso fechado, a abertura, sendo essa lógica configurada no algoritmo responsável pelo monitoramento e controle de todo o sistema.

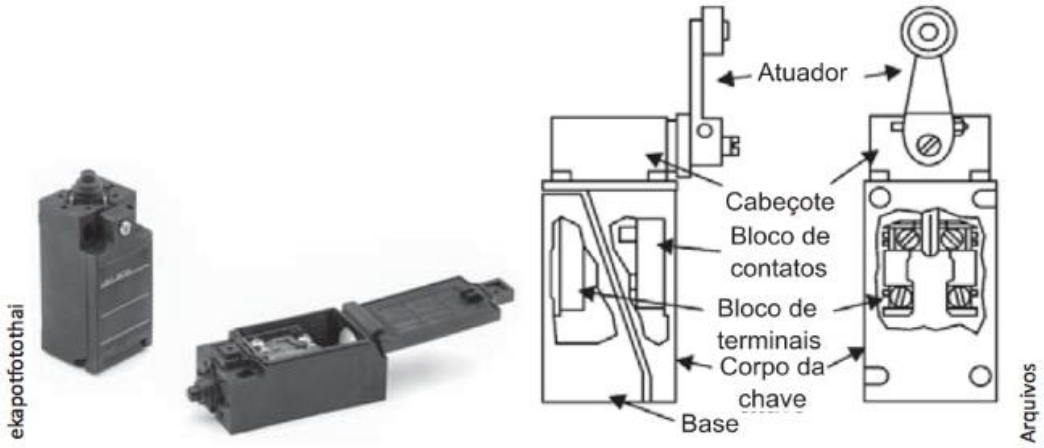
Figura 28 – Transmissor frequência 433 MHz

Fonte: https://www.pontofrio.com.br/controle-de-portao-tx-car-a-prova-de-agua-farol-luz-alta-carro-moto-43392-mhz-code-learn/p/1508487949

3.6.3 Chave Fim de Curso

Conforme Franchi (2015), as chaves fim de curso são dispositivos auxiliares de comando e de acionamento que atuam em um circuito com funções diversificadas, entre elas, determinar a posição de um elemento móvel. Utiliza-se uma chave fim de curso para controle, para detectar se o portão está aberto ou fechado. Desta forma ao chegar um veículo e o sistema realizar a leitura da placa, ele irá enviar o comando para o microcontrolador. Contudo esse não acionará o relé pois a chave fim de curso está aberta. Na Figura 29 observa-se a chave fim de curso e suas principais partes.

Figura 29 – Chave fim de curso

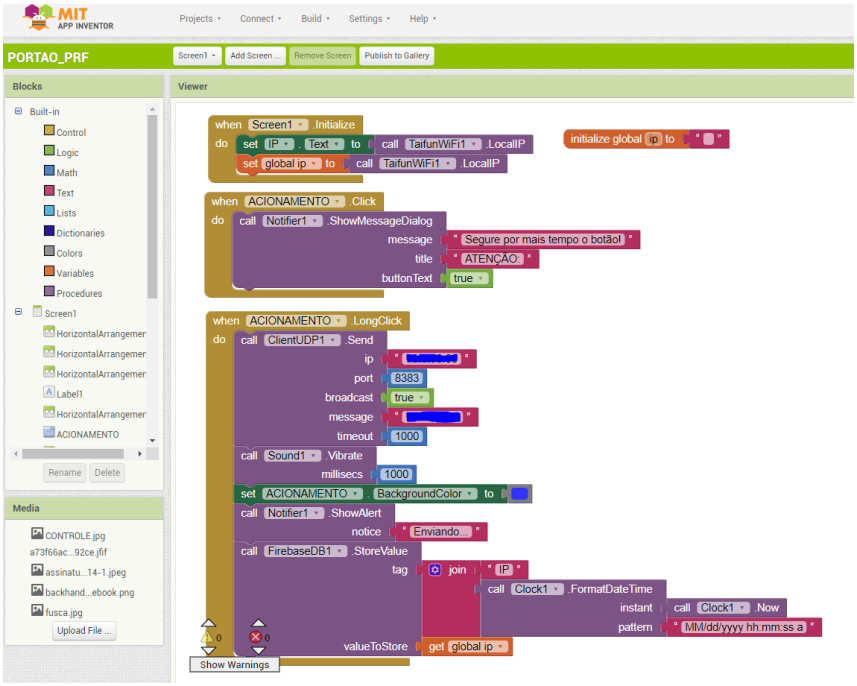


Fonte: Franchi (2015)

3.7 *SOFTWARE* DE DESENVOLVIMENTO MIT APP INVENTOR

Consiste em um ambiente de programação de fácil utilização desenvolvido para iniciantes na área de programação para celular. Foi desenvolvido pelo *Massachusetts Institute of Technology* (MIT) e sua utilização é on-line, onde é programado primeiramente a parte visual do aplicativo e posteriormente seu código através de blocos, utilizando apenas a lógica de programação, conforme ilustrado na Figura 30.

Figura 30 - Blocos de programação MIT APP Inventor



Fonte: Autor (2023)

3.8 ORGANIZAÇÃO DO SISTEMA PROPOSTO

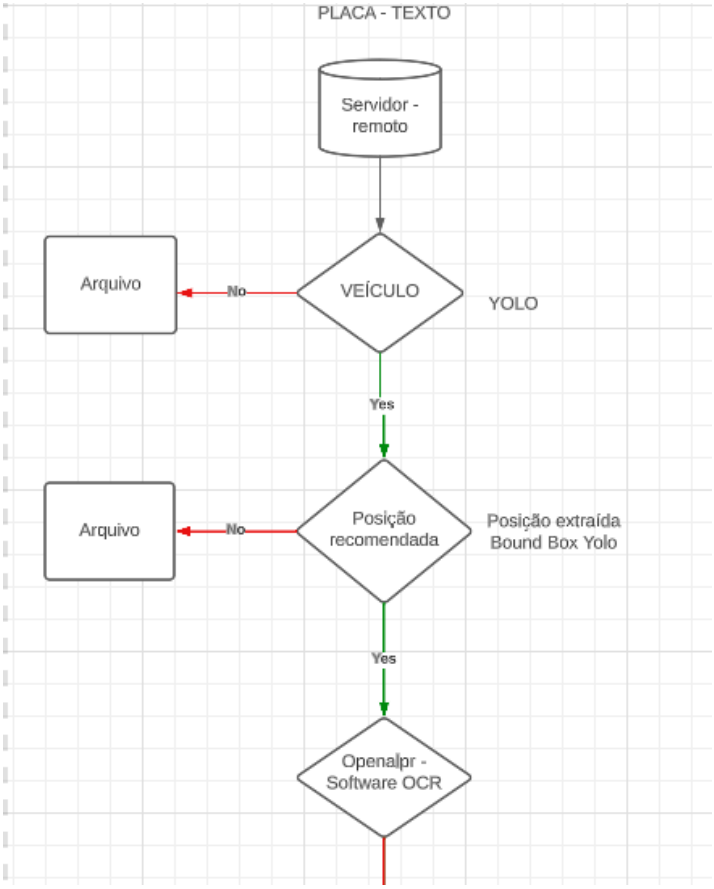
Neste tópico aborda-se sobre o organograma do sistema em forma de diagrama, desta maneira consegue-se verificar as decisões tomadas pelo algoritmo. Conforme a Figura 31 a câmera detecta o movimento conforme configurada e envia via FTP para o servidor remoto que fará o processamento.

Figura 31 – Ilustração da detecção de movimento e envio das imagens

Fonte: Autor (2023)

Após o servidor receber a imagem, o primeiro processamento é verificar se existe um veículo e extrair sua posição, utiliza-se a *YOLO* para essa tarefa, caso a resposta seja negativa a imagem não é processada. Se o veículo estiver na posição correta inicia-se a tentativa da extração dos caracteres através do *software openalpr*, conforme nos mostra a Figura 32.

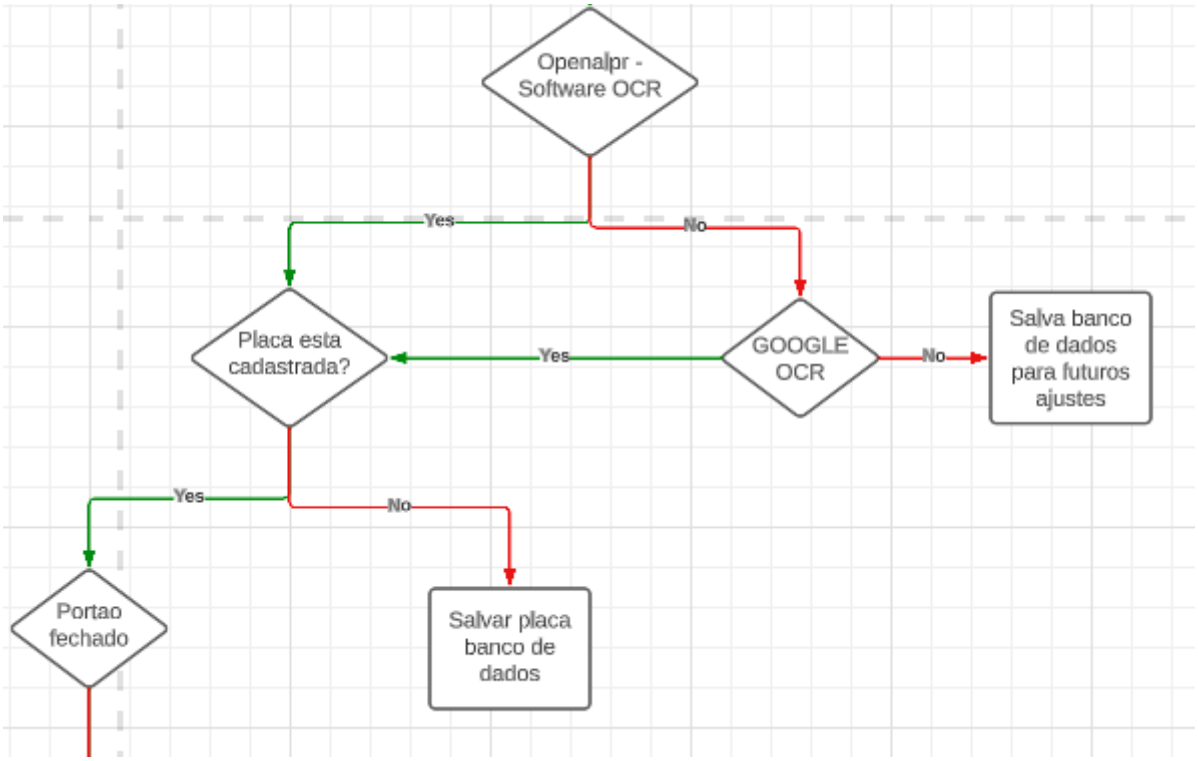
Figura 32 – Etapa processamento *YOLO-Openalpr*



Fonte: Autor (2023)

Caso a detecção de placa não ocorra através do *Openalpr*, utiliza-se a *API Google Vision* para detectar e extrair a placa veicular. Caso este não consiga, mesmo assim, a imagem e a tentativa de acesso são salvas no banco de dados para ajustes e melhorias no sistema. Detectada a placa e extraído os caracteres, verifica-se no banco de dados se o veículo está autorizado a entrar no estacionamento, conforme nos mostra a Figura 33.

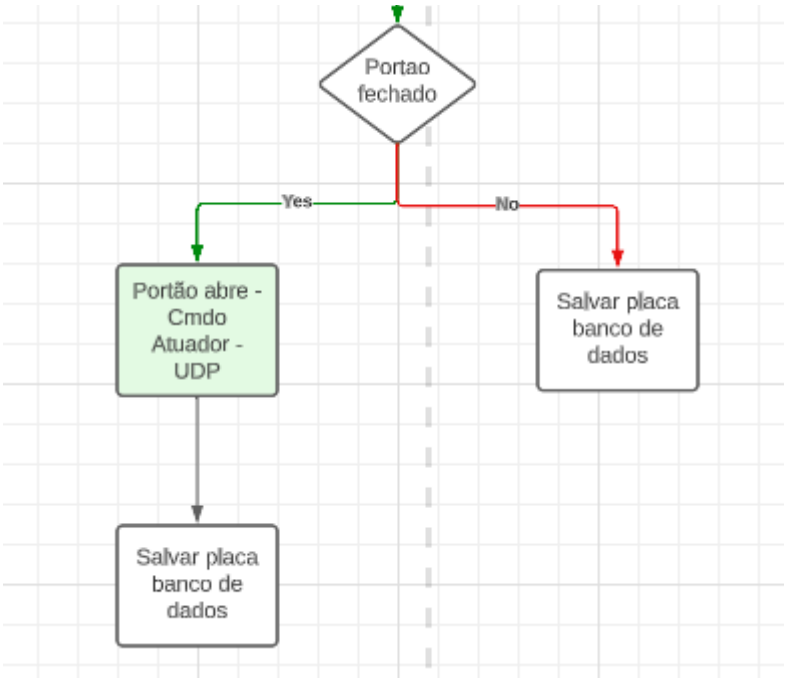
Figura 33 – Etapa de verificação e extração dos caracteres do veículo



Fonte: Autor (2023)

Última etapa do processamento consiste, caso o veículo esteja autorizado a entrar no local, verificar se portão está aberto ou fechado. Caso esteja aberto a imagem e os dados de entrada são salvos no servidor, caso negativo o microcontrolador acionará o relé e consequentemente enviará o sinal em 433 MHz para abertura do portão. Este processo está conforme a Figura 34.

Figura 34 – Última etapa do processamento



Fonte: Autor (2023)

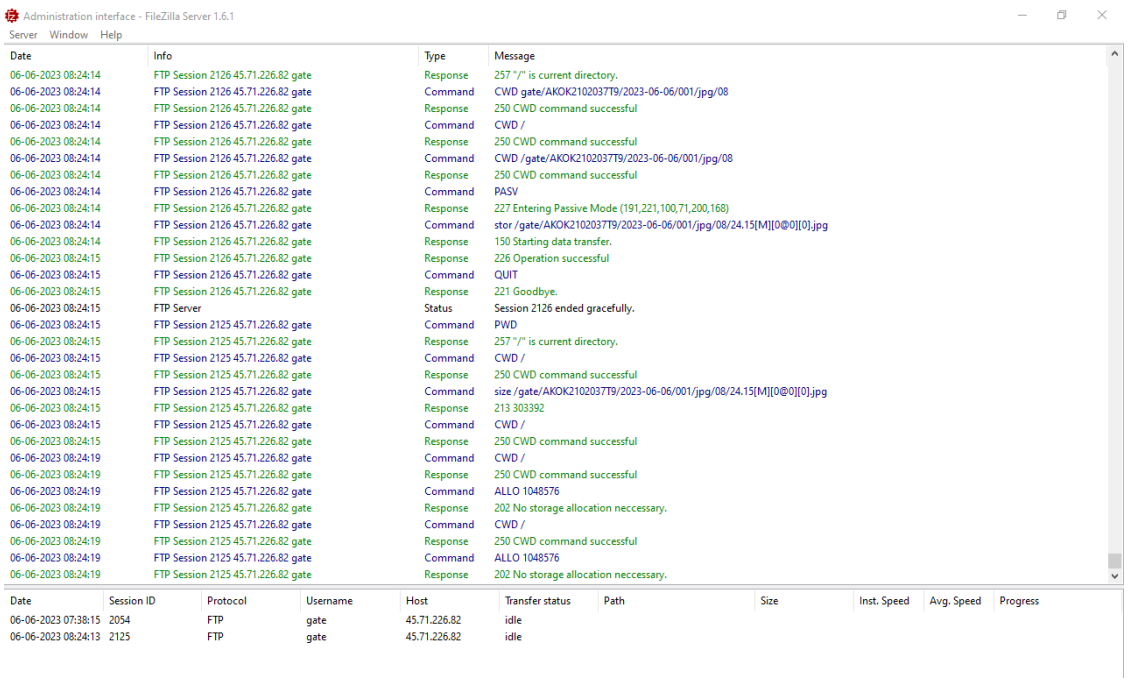
4. RESULTADOS

Neste capítulo apresenta-se os resultados obtidos referente ao funcionamento do sistema, verificando o tempo de processamento e as precisões das leituras efetivadas das placas veiculares, e ao final os custos envolvidos no projeto são apresentados.

4.1 OBTENÇÃO DAS IMAGENS

As imagens provenientes da câmera ficam armazenadas no servidor organizadas por dia do mês, e hora do recebimento. Em média cada imagem possui cerca de 200 KB, está no formato JPG, com resolução de 1920 x 1080 pixels, levando em torno de 40ms para transferência entre câmera/servidor. Pode-se acompanhar através do programa utilizado *FileZilla Server*, conforme a Figura 35, o status da câmera. Desta maneira caso ocorra um problema na conexão o gerenciador de rede poderá ser alertado.

Figura 35 – *Software Filezzila Server* em execução

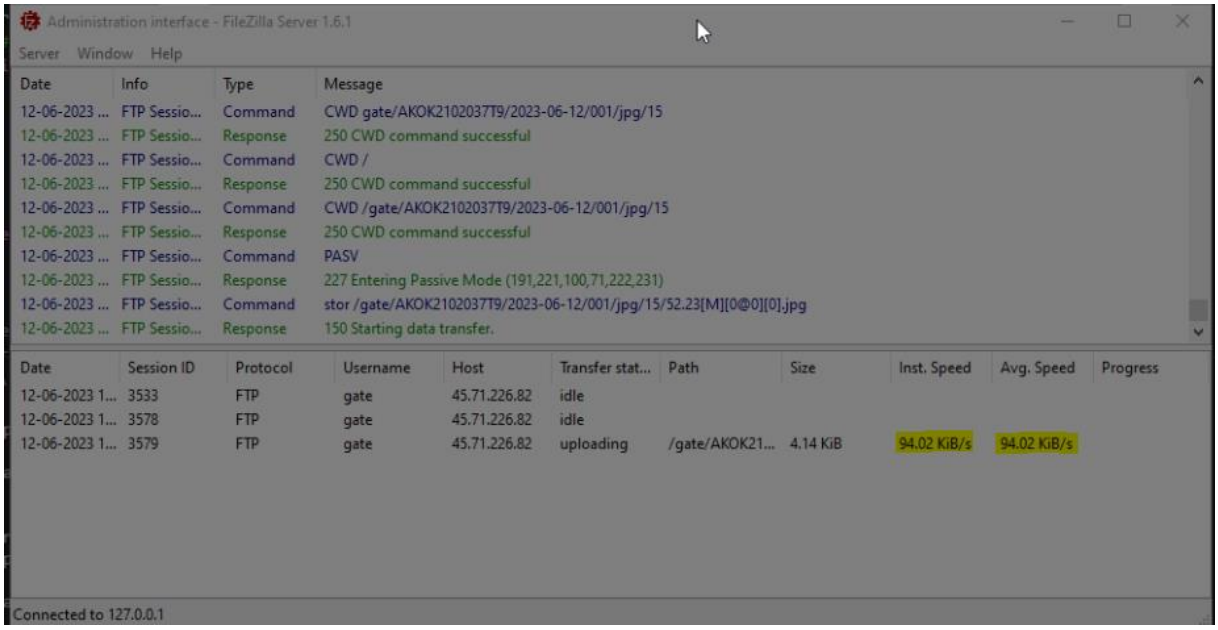


Fonte: Autor (2023)

Conforme a Figura 36, as taxas de transferências via FTP entre a câmera e o servidor ficam em torno de 94 KiB/s (kibibyte por segundo), ou 94,02 KiB/s, levando menos de 1

segundo para transferir a imagem, pois neste exemplo, a mesma possui em torno de 423 KB. Esse tempo deverá ser somado ao tempo total para o processamento da imagem, apesar de ser um valor pequeno. Assim é obtido o tempo total de operação do sistema, ou seja, quanto tempo de espera do veículo em frente ao portão até que se inicie a abertura do mesmo.

Figura 36 – Taxa de transferência via FTP



Fonte: Autor (2023)

4.2 PROCESSAMENTO DAS IMAGENS

4.2.1 Resultados com a YOLO

Utilizou-se a YOLO para detectar a presença de veículos na imagem e se este se encontra na posição que seja possível a leitura pelo algoritmo OCR. Não é um dos objetivos deste trabalho realizar o treinamento da YOLO, por isso utiliza-se um modelo treinado de predição de objetos disponibilizado no site da Ultralitics: "**yolov8l.pt**", sendo esse modelo capaz de detectar, entre outros, os seguintes instrumentos*: 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'stop sign'*.

Quando a YOLO detecta um carro na imagem, ela retorna à posição encontrada no eixo coordenado "Y" e verifica-se o valor numérico do contorno da imagem em formato retangular

(*bounding box)* do veículo está compreendido entre 360 e 700, os quais se mostraram com uma maior assertividade dos caracteres lidos. Na Figura 37 é possível ver o resultado da detecção de 4 carros, mas nenhum na posição correta para processamento.

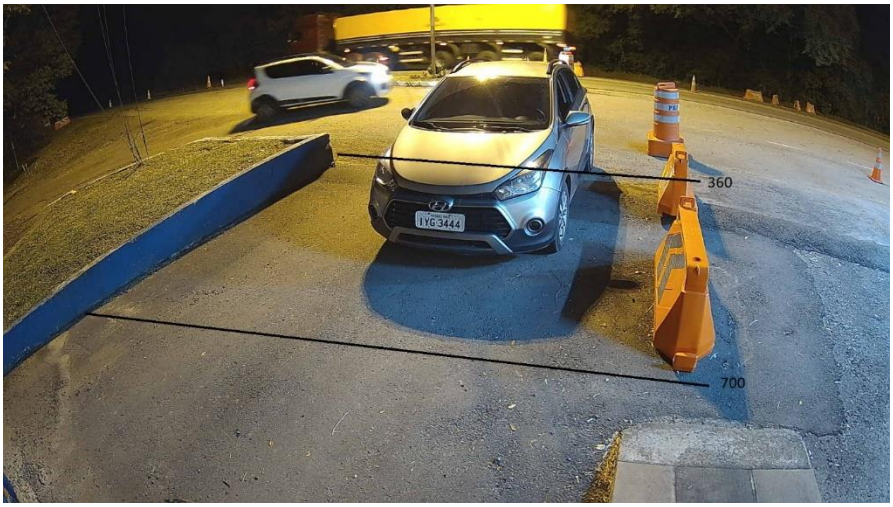
Figura 37 – Detecção de veículos pela YOLO fora da posição de processamento

```
image 1/1 C:\Users\Andre Amaral\Desktop\cam_gate\AKOK2102037T9\2023-06-01\001\jpg\16\29.53[M][0@0][0].jpg: 384x640 4
 cars, 1 fire hydrant, 1 potted plant, 64.8ms
Speed: 1.0ms preprocess, 64.8ms inference, 2.0ms postprocess per image at shape (1, 3, 640, 640)
yc2:  [130, 144, 275, 274]
Nenhum veículo na posição determinada
```

Fonte: Autor (2023)

Dentro da posição desejada (360 a 700) o algoritmo passa para a próxima fase, que seria o processamento da imagem para extração do OCR, sendo possível verificar na Figura 38 a detecção do veículo e o processamento da imagem na Figura 39.

Figura 38 – Posicionamento correto do veículo



Fonte: Autor (2023)

Figura 39 – Processamento da imagem para posição correta do veículo

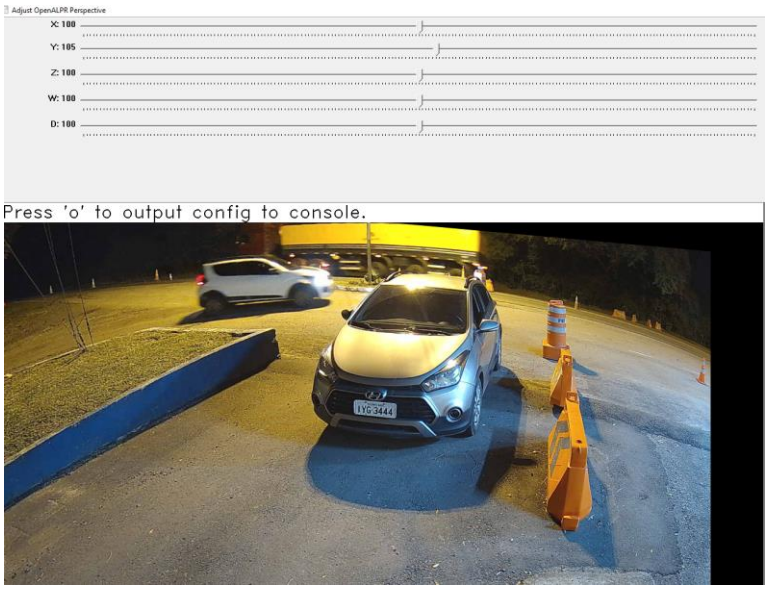
```
image 1/1 C:\Users\Andre Amaral\Desktop\cam_gate\AKOK2102037T9\k.jpg: 384x640 2 cars, 1 fire hydrant, 1 surfboard, 7
8.8ms
Speed: 0.0ms preprocess, 78.8ms inference, 2.0ms postprocess per image at shape (1, 3, 640, 640)
yc2:  [546, 242]
com OCR ALPR EU :  JYG3444
Veículo placa:  JYG3444 autorizado! confiança:  0.8571428571428571
PORTAO ABERTO
MySQL Database connection successful
Query successful
```

Fonte: Autor (2023)

4.2.2 Resultados *softwares* OCR

Para a correta detecção das placas é necessário calibrar o sistema *Openalpr* inicialmente. Consiste em realizar o ajuste da imagem da câmera para que essa fique com características de posicionamento similar ao treinamento de detecção do *software*. Com isso, consegue-se aproximar as imagens reais com as do treinamento do sistema. Para isso utiliza-se a ferramenta de calibração conforme a Figura 40, e adicionando a seguinte linha no código de configuração do *Openalpr: "prewarp = planar,1280.000000,720.000000,-0.000000,0.000350,0.100000,1.000000,1.000000,0.000000,0.000000"*

Figura 40 - Calibração do *Openalpr*



Fonte: Autor (2023)

Para verificar a eficiência do processamento das imagens, foi inserido um contador de tempo dentro do algoritmo, quando detectado do recebimento da imagem no servidor proveniente da câmera, inicia-se um cronômetro, quando finalizado o processamento da imagem encerra-se o *timer* e a informação é salva no banco de dados. Deste modo consegue-se mensurar quando tempo o sistema leva do recebimento da imagem até o portão iniciar sua abertura.

Utilizando biblioteca *Python Statiscs* para analisar os dados salvos no banco de dados, verifica-se que o tempo médio de processamento de uma imagem é de 3,41s. Considerando os dois *softwares* de extração de caracteres, das 676 extrações do OCR, 323 foram realizadas pelo

Google Vision, e que utilizando esse sistema de extração de caracteres o sistema demora em média 4,6s. Sendo o pior tempo de processamento do *Google Vision* em torno de 11s, como nos mostra a Tabela 1.

Verifica-se que a extração dos caracteres realizadas pelo *Openalpr* tem um tempo menor, em média 1,90s, e o pior processamento em torno de 4s. Sendo assim, é mais eficiente iniciar o processamento pelo *Openalpr* e após caso esse não seja capaz de realizar o OCR, utilizar o Google Vision, inclusive levando em conta que a API do Google precisa de conexão com os servidores remotos o que consequentemente leva um tempo maior.

Tabela 1 – Desempenho do sistema para diferentes *softwares* OCR

Sistema OCR	Nº Imagens	Média tempo (s)	Maior tempo (s)
Openalpr	353	1,9	4
Google Vision	323	4,6	11
TOTAL	676	-	-

Fonte: Autor (2023)

Principalmente em condições adversas de luminosidade ou qualquer acessório próximo a placa pode dificultar ou até mesmo fazer com que o *software* OCR não detecte o texto ou erre o caractere. Desta maneira compara-se o resultado obtido com as placas cadastradas no sistema e sua entrada é autorizada quando 5 caracteres na mesma posição derem o *match*, ou seja, se a placa original autorizada for IYG3444, e o sistema realizar a leitura LYG8444, o sistema autorizará a entrada. Mas se a leitura for GIY4344 o acesso será negado, porque compara-se a posição por posição dos caracteres com a placa autorizada.

Desta forma a precisão estará entre um número entre 0 e 1, sendo a partir de 0.71 autorizada a entrada do veículo. Na Figura 41 é possível ver na coluna “precisão” os acertos e erros do sistema quando comparado com a placa cadastrada no bando de dados, ou seja, individualmente. Deste modo tem se a opção de abrir a imagem e verificar o caractere com problema para melhorar o desempenho do programa.

Figura 41 - Precisão da leitura OCR

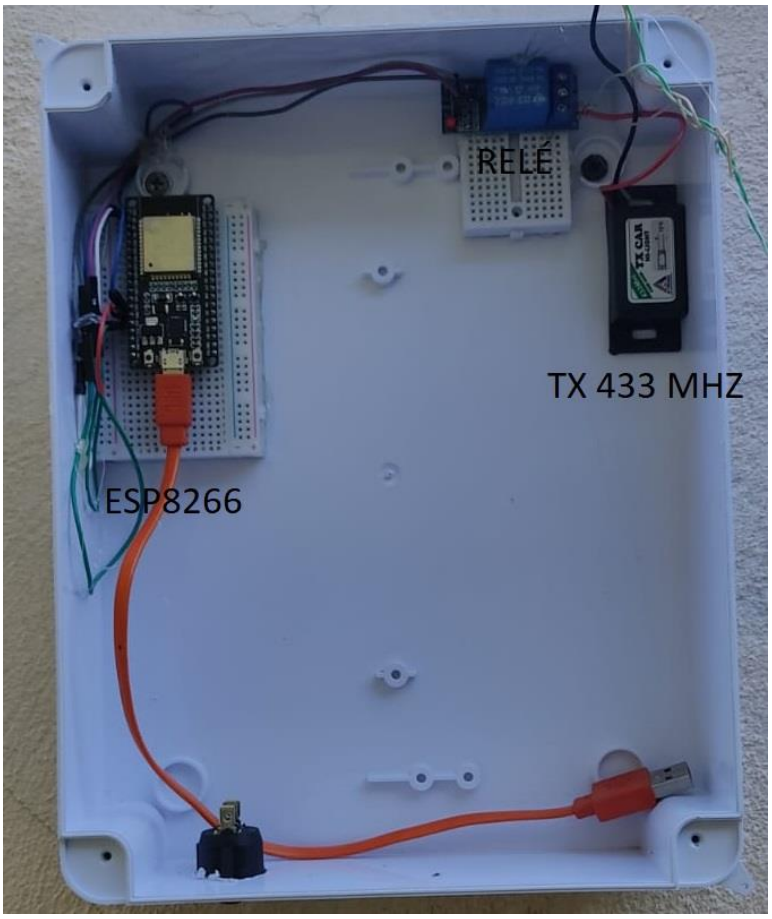
OCR	Precisão
ALPR(EU)	0.428571
ALPR(BR)	1
ALPR(EU)	1
GOOGLE OCR	0.428571
ALPR(US)	1
GOOGLE OCR	1
ALPR(EU)	1

Fonte: Autor (2023)

4.3 CIRCUITO DE COMANDO

Instalado no local de acesso, conforme a Figura 42, em uma caixa com grau de proteção IP 65, encontra-se o *hardware* necessário para acionamento do portão. Ele é composto por um ESP8266 responsável pelo recebimento do datagrama UDP, e acionamento do relé fechando o circuito para que o transmissor 433 MHz acione o motor elétrico instalado no local.

Figura 42 – *Hardware* instalado no local



Fonte: Autor (2023)

Devido o sistema ser responsável pela entrada de pessoas no local e a indisponibilidade de realizar cópias para todos os envolvidos, foi adicionado externamente um botão que também realizar o acionamento, apenas fechando o contato e acionando o transmissor. Com isso caso ocorra alguma falha no controle o sistema pode ser operado manualmente. Após realizado os testes necessários esse botão será desativado. Caso o *hardware* precise ser reiniciado pode-se acionar o botão vermelho, o qual desliga todo o sistema. Na Figura 43 é visto o resultado do protótipo final.

Figura 43 – Protótipo final do controlador

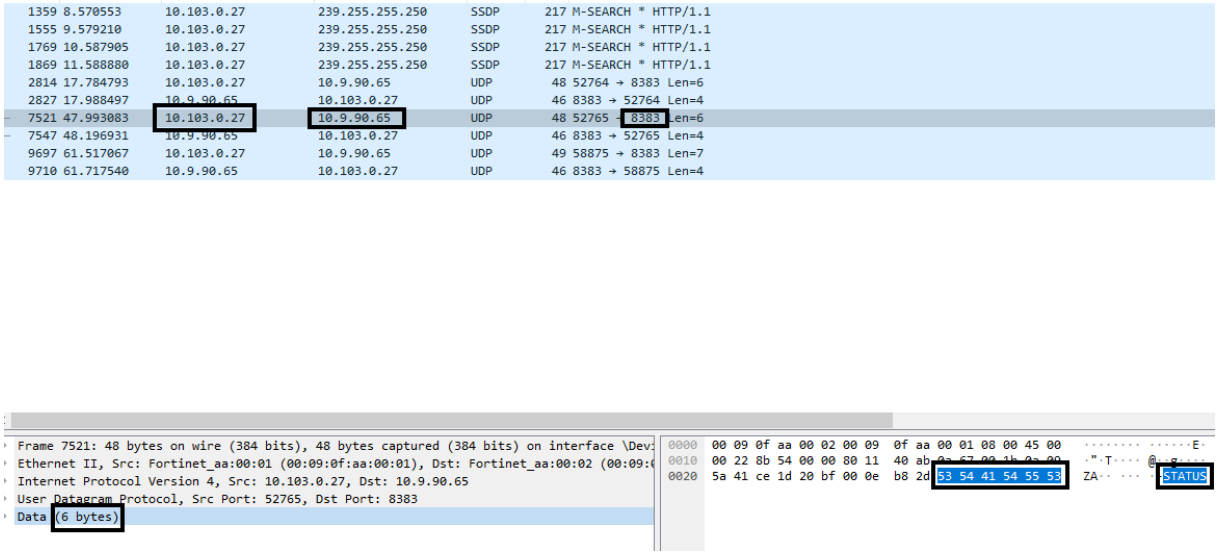
Fonte: Autor (2023)

4.4 COMUNICAÇÃO COM MICROCONTROLADOR

Utilizando o *software Wireshark* captura-se os quadros Ethernet enviados e recebidos pelo servidor UDP (ESP8266) oriundos da comunicação com o cliente (aplicação em *Python*), o qual mostra o conteúdo em bytes no formato hexadecimal toda a comunicação incluindo as portas utilizadas e os dados trocados.

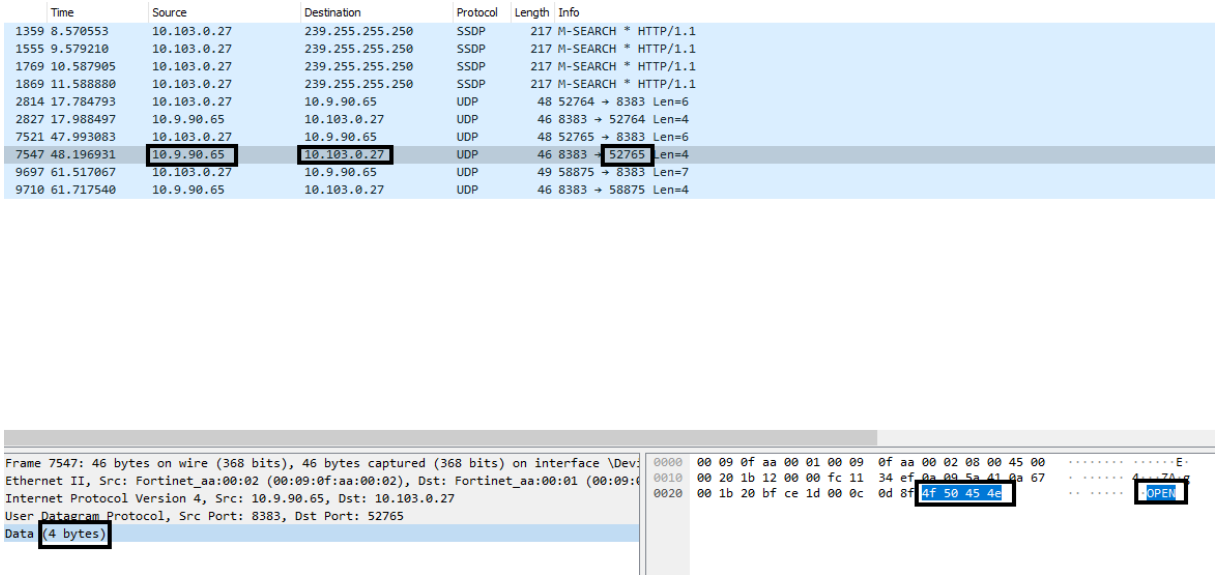
A Figura 44 mostra a requisição do cliente (IP 10.103.0.27) para o servidor (IP 10.9.90.65), na porta 8383, contendo 6 *bytes* de dados, HEX: 53 54 41 54 55 53, o que representa em decimal “STATUS”. Na Figura 45, o servidor responde ao cliente conforme solicitado na porta de destino 52765, contendo 4 *bytes* de dados, HEX: 4F 50 45 4E, em decimal “OPEN”, o que significa para a aplicação que o portão está aberto, sendo essa informação adquirida através do sensor fim de curso.

Figura 44 – Comunicação UDP Cliente x Servidor



Fonte: Autor (2023)

Figura 45 – Comunicação UDP Servidor - Cliente



Fonte: Autor (2023)

4.5 INTERFACE USUÁRIO

Para consulta dos dados e operação do sistema foram criadas duas interfaces para comunicação homem-máquina. Em Python foi criado o sistema de supervisão remota, o qual se comunica com o controlador através da Internet. Noutra interface, em PHP pode-se realizar as consultas e autorizações de veículos para acessar o local.

4.5.1 Interface Usuário – Python

Criada com a biblioteca para Python *Tkinter*, este sistema permite a supervisão da entrada dos veículos e o funcionamento do sistema em tempo real, pois caso se deseje, pode-se colocar um operador que tem a capacidade de liberar a entrada para um veículo não autorizado ou realizar o fechamento do portão.

Utilizando a biblioteca *OpenCV*, é possível ter acesso a câmera em tempo real, deste modo, além do sensor fim de curso que nos traz a informação do estado do portão (aberto ou fechado) a imagem permite a visualização da entrada e gerenciamento de todo o acesso.

As seguintes informações estão disponíveis conforme a Figura 46:

- **Número de placas** autorizadas: Permite verificar quantos veículos estão autorizados a entrar no local

- **Monitoramento:** Verifica se o *software* está monitorando as imagens provenientes da câmera

- **Placa/data**: Retorna a última placa liga e a data e hora que ingressou no local

- **Autorizado**: Verifica se o veículo está na lista de autorizados

- **Status do portão**: Monitora se o mesmo está aberto ou fechado, utiliza comunicação UDP para essa tarefa.

- ***Start:*** Inicia o monitoramento

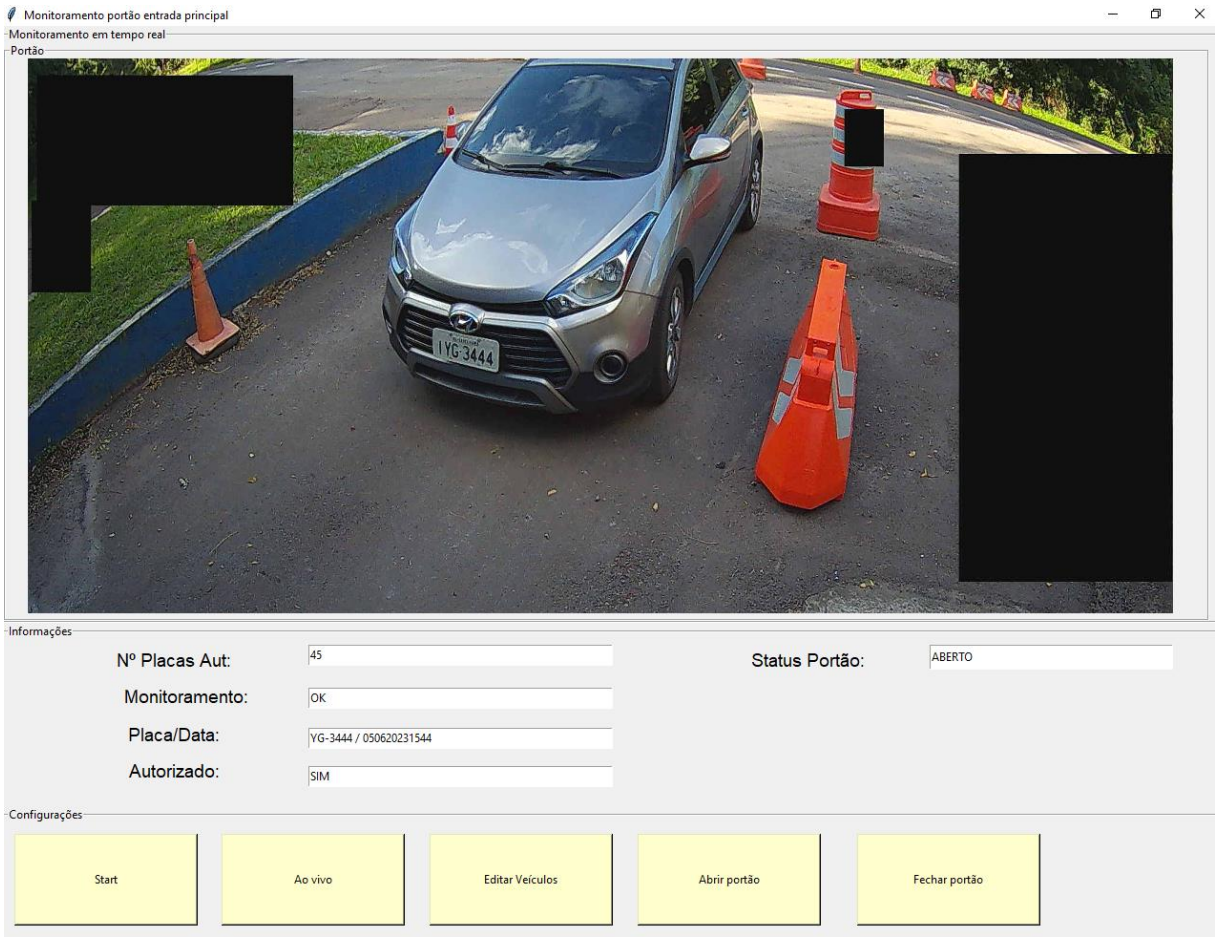
- **Ao vivo**: Exibe imagens da câmera de segurança presente no local

- **Editar veículos**: Abre página na Internet para adicionar ou retirar veículos da lista dos autorizados

- **Abrir portão**: Aciona o portão para abertura caso o mesmo esteja fechado

- **Fechar portão**: Aciona o portão para fechamento caso o mesmo esteja aberto.

Figura 46 – Sistema de supervisão criado em Python



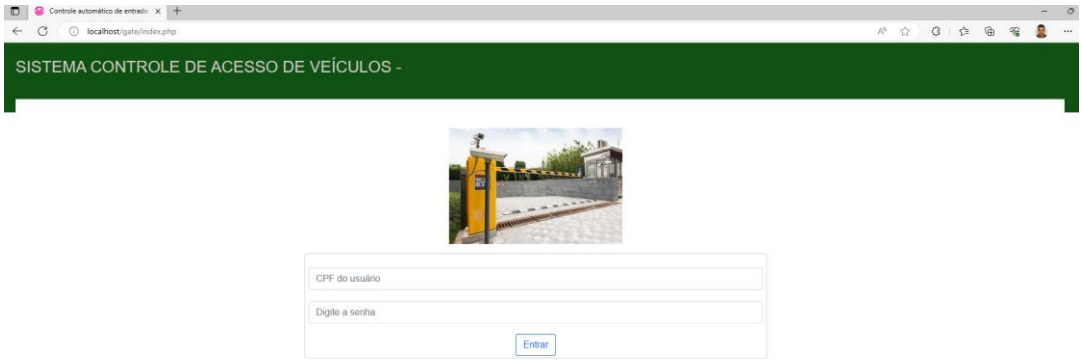
Fonte: Autor (2023)

4.5.2 Interface Usuário – PHP

Para verificar os veículos que acessaram o local e a data/hora, foi criado um servidor *Apache* na máquina local, inserindo páginas criadas em PHP, HTML e NodeJS. Sendo o PHP responsável por fazer as consultas e inserções no banco de dados.

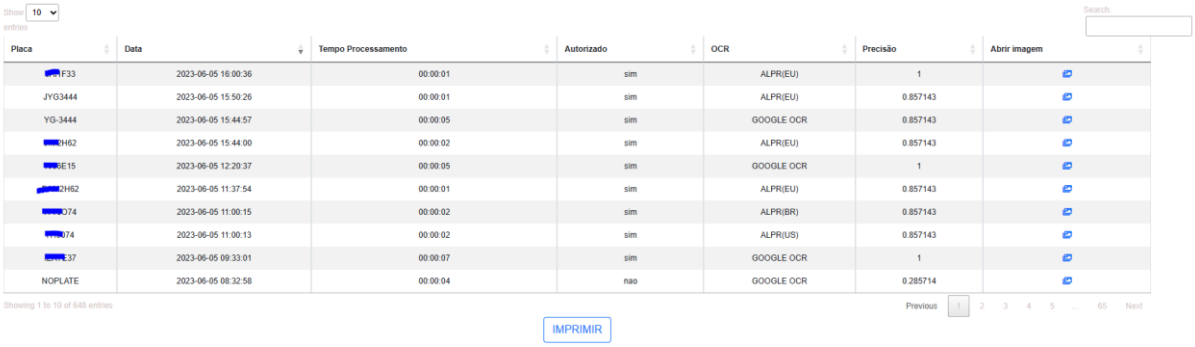
Conforme a Figura 47 o acesso ao sistema foi protegido através de login e senha de usuário. Na Figura 48 nos mostra os últimos 10 acessos realizados, sendo possível conforme a Figura 49 verificar a fotografia utilizada para processamento e, por último, na Figura 50 realiza-se a inserção ou retirada de autorização de veículos.

Figura 47 – Página de acesso ao site armazenado na máquina local



Fonte: Autor (2023)

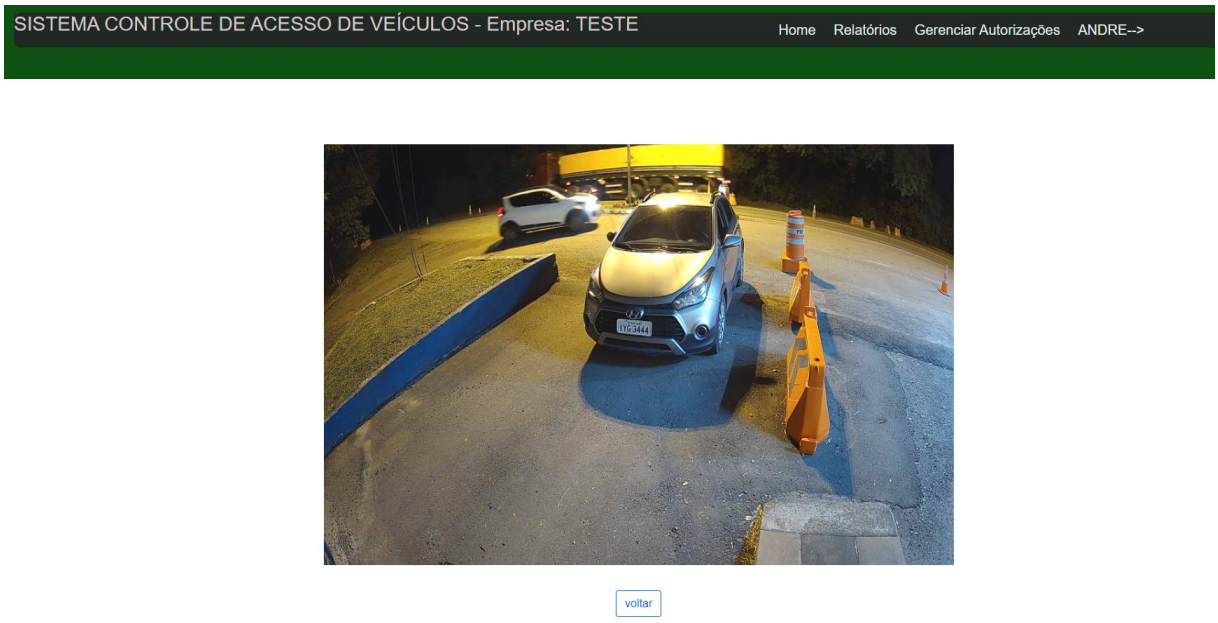
Figura 48 – Verificação dos veículos que acessaram o local



Placa	Data	Tempo Processamento	Autorizado	OCR	Precisão	Abrir imagem
[redacted]F33	2023-06-05 16:00:36	00:00:01	sim	ALPR(EU)	1	
JYG3444	2023-06-05 15:50:26	00:00:01	sim	ALPR(EU)	0.857143	
YG-3444	2023-06-05 15:44:57	00:00:05	sim	GOOGLE OCR	0.857143	
[redacted]H62	2023-06-05 15:44:00	00:00:02	sim	ALPR(EU)	0.857143	
[redacted]6E15	2023-06-05 12:20:37	00:00:05	sim	GOOGLE OCR	1	
[redacted]2H52	2023-06-05 11:37:54	00:00:01	sim	ALPR(EU)	0.857143	
[redacted]074	2023-06-05 11:00:15	00:00:02	sim	ALPR(BR)	0.857143	
[redacted]074	2023-06-05 11:00:13	00:00:02	sim	ALPR(US)	0.857143	
[redacted]37	2023-06-05 09:33:01	00:00:07	sim	GOOGLE OCR	1	
NOPLATE	2023-06-05 08:32:58	00:00:04	nao	GOOGLE OCR	0.285714	

Fonte: Autor (2023)

Figura 49 – Visualização do veículo



Fonte: Autor (2023)

Figura 50 – Editar os veículos cadastrados no sistema

Fonte: Autor (2023)

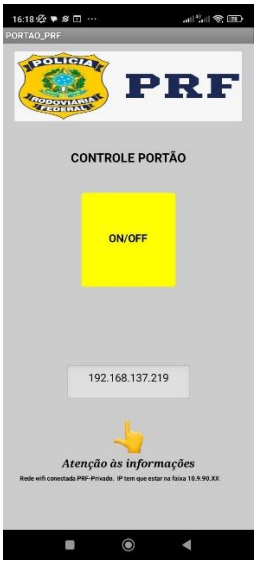
4.6 APLICATIVO PARA CONTROLE DO PORTÃO

Para realizar a automação completa do sistema, foi desenvolvido utilizando a ferramenta *APP Inventor* do *MIT*, um aplicativo para celulares *Android*, que é capaz de abrir e fechar o portão. Por questões de segurança o mesmo somente funciona na rede local, com o endereço IP na mesma faixa da rede privada local.

Quando acionado o botão, conforme a Figura 51, ocorre a comunicação através do protocolo UDP, o aplicativo envia o comando para o IP do microcontrolador na porta

configurada. Tendo em vista que o mesmo somente funciona no local, não foi necessário configurar a resposta ao comando recebido, uma vez que o usuário estará na frente do portão.

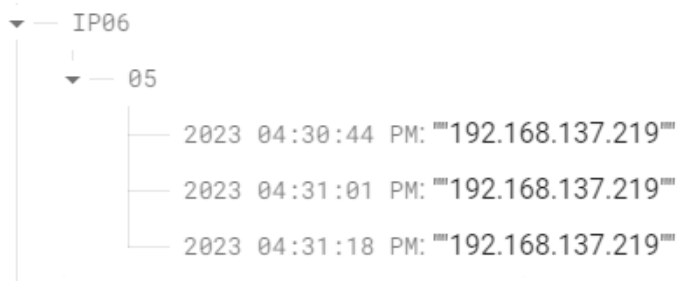
Figura 51 – Aplicativo para abertura/fechamento portão



Fonte: Autor (2023)

Tendo em vista um melhor gerenciamento dos acessos, a cada vez que usuário realiza a abertura e/ou fechamento do portão grava-se no *firebase* os *logs* desse acesso, contendo o IP e data/hora do acionamento, conforme a Figura 53.

Figura 52 – Gravação do IP - data/hora do acionamento no bando de dados *Firebase*



Fonte: Autor (2023)

Para gerenciamento remoto, foi criado um *bot* do *telegram*, o qual é capaz de receber avisos do ESP8266. Sendo assim o gerente do local pode receber um aviso diretamente no

celular de que o portão foi acessado naquele momento e quando o sistema por algum motivo reiniciou, conforme Figura 53.

Figura 53 - *Bot Telegram* informando do acionamento do portão.

Fonte: Autor (2023)

4.7 CUSTOS DE IMPLEMENTAÇÃO DO SISTEMA

Conforme a Tabela 2 o custo de implantação total do sistema foi de R$ 4.448,00, comparado ao valor de mercado de apenas uma câmera LPR, a qual possui um *software* embutido que realiza o OCR, a qual está custando cerca de R$ 6.000, o sistema se torna competitivo. Pode-se com o mesmo computador monitorar diversas entradas, portanto o custo por unidade é reduzido. Sendo que apenas a câmera LPR não possui as funcionalidades desta aplicação desenvolvida de forma intuitiva e fácil de ser executada pelo cliente, dentro dessas funcionalidades destaca-se a abertura e fechamento remoto, verificação de data/hora da entrada do veículo e autorizações ou exclusões da lista de forma rápida e eficiente.

Tabela 2 – Custos da implementação do sistema

Descrição	Quantidade	Valor Unitário	Total (R$)
Câmera IP 2 MP	01	1260,00	1260,00
Caixa de proteção IP 65	01	90,00	90,00
Computador I3, 8 Gb RAM, placa Vídeo GTX 970	01	3000,00	3000,00
ESP8266	01	40,00	40,00
Relé 1 canal	01	5,00	5,00
Transmissor 433 Mhz	01	15,00	15,00
Protoboard	01	5,00	5,00
Sensor fim de curso	01	30,00	30,00
Botão liga/desliga	01	3,00	3,00
TOTAL			**4448,00**

Fonte: Autor (2023)

5. CONCLUSÕES

Neste trabalho realizou-se a implantação de um sistema para automatizar e gerenciar de forma remota a entrada em locais fechados. Para isso foi utilizado uma câmera de captura de imagens do tipo IP, com a funcionalidade de detecção de movimento e envio das imagens via protocolo FTP para processamento do servidor. Sendo possível a usuários locais a utilização de um aplicativo para A*ndroid* que é capaz de abrir e fechar o portão, sendo que o sistema registra no banco de dados, a cada abertura e fechamento do local, os dados (placa, data, hora) para posteriores consultas.

Foram apresentados a metodologia empregada e os recursos utilizados, onde utiliza-se a linguagem de programação *Python*, *PHP* e os *softwares open source Openalpr e Google Vision*. Importante salientar o desempenho da detecção de veículos através da YOLO V8, a qual foi capaz de trazer resultados em torno de 70ms utilizando a GPU da placa de vídeo GTX 970. O desempenho do protocolo de comunicação FTP para envio das imagens ao servidor demonstrou-se adequado e eficiente, devido a não apresentar falhas e conexão estável. O protocolo UDP que foi utilizado para troca de mensagens entre a aplicação e o ESP8266, teve um bom desempenho devido apresentar estabilidade no funcionamento.

Após analisa-se o desempenho da automatização criada, onde verificou-se que o tempo médio para abertura do portão desde a chegada do veículo está variando de 3 a 10 segundos, o que são valores razoáveis de espera dependendo do local, por isso, com esse desempenho em locais com muitas entradas e saídas de veículos pode-se usar como alternativa o uso do APP. Referente ao índice de acertos do sistema, para que não houvesse prejuízo para os veículos cadastrados no que tange ao erro de leitura da placa, foi ajustado para um índice de acerto de 5 dos 7 caracteres, sendo esse valor alterável conforme desejado.

Portanto, para um trabalho futuro, pode-se ajustar o sistema para melhoria do desempenho do tempo de processamento e consequentemente abertura do portão. Soma-se como melhoria o ajuste da precisão dos caracteres lidos, implementando e melhorando o desempenho dos sistemas OCR embarcados no projeto.

REFERÊNCIAS

ALBERTIN, Alberto Luiz; ALBERTIN, Rosa Maria de Moura. **A Internet das Coisas irá muito além das Coisas**. 2017.

ALVES, Gabriel. **Detecção de objetos com YOLO – uma abordagem moderna. 2020**. Acesso em 10 maio 2023. Disponível em: <https://iaexpert.academy/2020/10/13/deteccao-de-objetos-com-yolo-uma-abordagem-moderna/>.

AMARAL, Carlos Ivan Santos do. **Avaliação de plataformas para o reconhecimento de placas veiculares brasileiras**. 2021. Trabalho de Conclusão de Curso. Brasil.

ARAÚJO, Aline Moura et al. **Detecção e destaque em vídeo de objetos utilizando YOLO**. 2022.

BARRETO, Saulo Cardoso. **Reconhecimento de placas veiculares utilizando deep learning: análise da influência de dados sintéticos no processo de reconhecimento**. 2018.

FEIJÓ, José Victor Feijó de Araujo et al**. Análise e Classificação de imagens para aplicação de OCR em cupons fiscais. 2017**. Dissertação de Mestrado. Florianópolis, SC.

FRANCHI, Claiton Moro. **Instrumentação de Processos Industriais Princípios e aplicações**. Saraiva Educação SA, 2015.

GONÇALVES, Moura; LEONARDO, Rangel. **Automatização residencial: um estudo de caso da aplicação da internet das coisas**. Sistemas de Informação-Florianópolis, 2019.

GRANATYR, Jones. **Introdução ao YOLOV8**. 2023. Acesso em 11 maio 2023. Disponível em: < https://iaexpert.academy/2023/03/09/introducao-ao-yolov8/>.

HAYKIN, Simon. **Redes neurais: princípios e prática**. Bookman Editora, 2001.

JINDAL, Punnet. **YOLOv8 is here, and it gets better**. Acesso em 28 junho 2023. Disponível em: < https://pub.towardsai.net/yolov8-is-here-and-it-gets-better-54b12b87e3b9/>.

MARQUES, Bruno Henrique Pereira. **Avaliação de algoritmos baseados em deep learning para localizar placas veiculares brasileiras em ambientes complexos**. 2019. Trabalho de Conclusão de Curso. Brasil.

MARQUES, Elton. **Um byte de cada vez: Seu FTP Ativo e Passivo**. Acessado em 29 junho 2023. Disponível em: https://www.pop-rs.rnp.br/~berthold/etcom/redes2-2000/trabalhos/FTP_EltonMarques.htm#:~:text=No%20FTP%20ativo%20(normal)%2C,de%20porta%20provida%20pelo%20servidor.

MARTINS, Samuel. **Introdução ao Processamento Digital de Imagens**. Universidade Estadual de Campinas(UNICAMP), 2016.

MERÇON, Victor de Abreu. **Sistema de controle de iluminação e monitoramento de consumo elétrico residencial via aplicativo.** 2022.

NIELSEN, Michael A. **Neural networks and deep learning**. San Francisco, CA, USA: Determination press, 2015.

PRUDENTE, Francesco. **Automação Predial e Residencial - Uma Introdução**. 2011**.** Acesso em: 11 jun. 2023. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/978-85-216-2024-2/.

RICH, Elaine. **Knowledge representation. Artificial intelligence**, p. 135-172, 1983.

SAHA, Sumit. **A comprehensive guide to convolutional neural networks—the ELI5 way**. Towards data science, v. 15, 2018.

SANTOS, Jean Willian; LARA JUNIOR, Renato Capelin de. **Sistema de automatização residencial de baixo custo controlado pelo microcontrolador esp32 e monitorado via smartphone**. 2019. Trabalho de Conclusão de Curso. Universidade Tecnológica Federal do Paraná.

SILVA, Ronildo Oliveira da. **Análise de desempenho da Google Cloud Vision API em leitura de textos provenientes de imagens naturais**. 2019.

STOPASSOLE, Alisson. **Reconhecimento automático de placas veiculares para o controle de estacionamentos baseado em aprendizado de máquina e visão computacional**. 2020. Trabalho de Conclusão de Curso. Universidade Tecnológica Federal do Paraná.

APÊNDICE A – PROCESSAMENTO YOLOV8

```
from ultralytics import YOLO
import torch
import cv2

def detecta_veiculo(img):
   model = YOLO('yolov8l.pt')
   #model.fuse()
   result = model.predict(img,conf=0.25)
   cls = result[0].boxes.cls # seleciona as classes, tensor
   cls_list=cls.tolist() #trocar para lista
   vertices = (result[0].boxes.xyxy) #cordenadas normalizadas, utilizar whn
   vertices_list=vertices.tolist() #trocar para lista


   names={0: 'person', 1: 'bicycle', 2: 'car', 3: 'motorcycle', 4: 'airplane',
5: 'bus', 6: 'train', 7: 'truck', 8: 'boat', 9: 'traffic light', 10: 'fire
hydrant', 11: 'stop sign',  12: 'parking meter', 13: 'bench', 14: 'bird', 15:
'cat', 16: 'dog', 17: 'horse', 18: 'sheep', 19: 'cow', 20: 'elephant', 21:
'bear', 22: 'zebra', 23: 'giraffe', 24: 'backpack', 25: 'umbrella', 26:
'handbag', 27: 'tie', 28: 'suitcase', 29: 'frisbee', 30: 'skis',     31:
'snowboard', 32: 'sports ball', 33: 'kite', 34: 'baseball bat', 35: 'baseball
glove',  36: 'skateboard', 37: 'surfboard', 38: 'tennis racket', 39: 'bottle',
40: 'wine glass',41: 'cup', 42: 'fork', 43: 'knife', 44: 'spoon', 45: 'bowl',
46: 'banana', 47: 'apple', 48: 'sandwich', 49: 'orange', 50: 'broccoli', 51:
'carrot', 52: 'hot dog', 53: 'pizza', 54: 'donut', 55: 'cake', 56: 'chair', 57:
'couch', 58: 'potted plant', 59: 'bed', 60: 'dining table', 61: 'toilet', 62:
'tv', 63: 'laptop', 64: 'mouse', 65: 'remote', 66: 'keyboard', 67: 'cell phone',
68: 'microwave', 69: 'oven', 70: 'toaster',71: 'sink', 72: 'refrigerator', 73:
'book', 74: 'clock', 75: 'vase', 76: 'scissors', 77: 'teddy bear', 78: 'hair
drier', 79: 'toothbrush'}

   lista_encontrados=[]
   posicao_encontrados=[]

   for i in range(len(cls_list)):
      lista_encontrados.append(names[cls_list[i]])
      posicao_encontrados.append(vertices_list[i])
   #print(posicao_encontrados)
    #posicao_encontrados = vetor posicao na imagem
    # pos_person=posicao no array
    #transforma ponto flutuante em int
   posicao_encontrados_int=[]
   for i in range(len(posicao_encontrados)):
```

```
44          posicao_encontrados_int.append(list(map(int, posicao_encontrados[i])))
45      #print(posicao_encontrados_int)
46      pos_car = []
47      for i in range(len(lista_encontrados)):
48         if 'car' in lista_encontrados[i]:
49              pos_car.append(i)
50
51      xc =  []
52      xc1=  []
53      yc1 = []
54      xc2 = []
55      yc2 = []
56
57      for i in range(len(pos_car)):
58        xc.append(posicao_encontrados_int[pos_car[i]])
59      for i in range(len(xc)): #separo lista x das imagens dos carros
60              xc1.append((xc[i][0]))
61              xc2.append((xc[i][2]))
62              yc1.append((xc[i][1]))
63              yc2.append((xc[i][3]))
64
65      #print("xc1: ", xc1)
66      #print("xc2: ", xc2)
67      #print("yc1: ", yc1)
68      print("yc2: ", yc2)
69
70      found=[]
71      for i in range (len(pos_car)):
72        if 385 < yc2[i] <  700:  #faixa da imagem da captura
73            found.append(yc2[i])
74      #print(len(found))
75      return found
```

APÊNDICE B – PROCESSAMENTO GOOGLE VISION

```
from PIL import Image
from google.cloud import vision
import urllib3
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
import os
os.environ["GOOGLE_APPLICATION_CREDENTIALS"]="/Users/Andre
Amaral/Desktop/andre.json"
import re
import cv2


def detect_text(path):
    """Detects text in the file."""
    from google.cloud import vision
    import io
    client = vision.ImageAnnotatorClient()

    with io.open(path, 'rb') as image_file:
        content = image_file.read()

    image = vision.Image(content=content)

    response = client.text_detection(image=image)
    texts = response.text_annotations

    candidatos = []
    for text in texts:
        candidatos.append(text.description)

        #print('\n"{}"'.format(text.description))

        #vertices = (['({},{})'.format(vertex.x, vertex.y)
                    # for vertex in text.bounding_poly.vertices])

        #print('bounds: {}'.format(','.join(vertices)))
    #print(teste)
    #print(len(teste))
    placas=[]
    for i in candidatos:
        res = re.sub('[-]', '', i)
        res = re.sub('[\n]', '', i)
        res=res.replace(" ","")
        res = re.sub(r"\s+$", "", res)
```

```
44          placas.append(res)
45      print(placas)
46      if len(placas)!=0:
47        for i in range(len(placas)):
48          if len(placas[i])==7:
49             look_number = any(char.isdigit() for char in placas[i]) #verifica
50  se contem números
51             look_letras = any(char.isalpha() for char in placas[i]) #verifica
52  se contem letras
53             if look_number and look_letras == True:
54              placa=placas[i]
55              break
56          if len(placas[i])==8:
57             placa = (placas[i])[1:]
58             look_number = any(char.isdigit() for char in placa) #verifica se
59  contem números
60             look_letras = any(char.isalpha() for char in placa) #verifica se
61  contem letras
62             if look_number and look_letras == True:
63                  placa=placa
64                  break
65          if len(placas[i])>8:
66               x = re.search("[a-zA-Z]{3}[0-9][A-Za-z0-9][0-9]{2}", placas[i])
67               #print(x.group())
68               if x != None:
69                   placa=x.group()
70                   break
71               else:
72                   placa="NOPLATE"
73          else:
74             placa="NOPLATE"
75      if len(placas)==0:
76          placa="NOPLATE"
77      return placa
```

APÊNDICE C – PROCESSAMENTO BANCO DE DADOS

```
import mysql.connector
from mysql.connector import Error
import pandas as pd
import datetime
from statistics import mean

def create_db_connection(host_name, user_name, user_password, db_name):
    connection = None
    try:
        connection = mysql.connector.connect(
            host=host_name,
            user=user_name,
            passwd=user_password,
            database=db_name
        )
        print("MySQL Database connection successful")
    except Error as err:
        print(f"Error: '{err}'")

    return connection

def execute_query(connection, query):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        connection.commit()
        print("Query successful")
    except Error as err:
        print(f"Error: '{err}'")

def read_query(connection, query):
    cursor = connection.cursor()
    result = None
    try:
        cursor.execute(query)
        result = cursor.fetchall()
        return result
    except Error as err:
        print(f"Error: '{err}'")

'''
#exemplos
```

```
44
45  connection = create_db_connection("localhost", "root", "xxxxxxx","sys")
46  q1 = """
47  SELECT *
48  FROM autorizados;
49  """
50
51  results = read_query(connection, q1)
52
53  for result in results:
54    print(result)
55
56  placa_autorizada = """
57  INSERT INTO autorizados (PLACA) VALUES
58  ('IYS5122')
59  """
60
61  execute_query(connection, placa_autorizada)
62

63  ############ CALCULAR TEMPO PROCESSAMENTO ########################
64  def tempo_processamento():
65     tempo = []
66     connection = create_db_connection("localhost", "root", "xxxxxxx","sys")
67     q1 = """
68     SELECT TIME
69     FROM gate_control WHERE OCR='ALPR(EU)'
70     ;
71     """
72     results = read_query(connection, q1)
73     print(type(results))
74     for result in results:
75       tempo.append(result)
76
77     return tempo
78
79  tempo = tempo_processamento()
80  print(type(tempo[0][0]))
81
82  segundos_list = []
83  for i in range(len(tempo)):
84      segundos_list.append(tempo[i][0].total_seconds())
85
86  #3td_in_sec =
87  tempo[6][0].total_seconds()                                    # getting
88  the total amount of time in seconds
89  print(segundos_list[0])
90  print(type(segundos_list[0]))
```

```
avg = mean(segundos_list)
print(avg)  # média do tempo abertura do portão considerando todos os dados
(3.41) // Considerando google ocr (4.60) // Considerando alpr (1.90)



def contagem():
   contagem = []
   connection = create_db_connection("localhost", "root", "xxxxxxxx","sys")
   q1 = """
   SELECT OCR
   FROM gate_control WHERE OCR='GOOGLE OCR'
   ;
   """
   results = read_query(connection, q1)
   print(type(results))
   for result in results:
     contagem.append(result)

   return contagem

total = contagem()
print(len(total))

'''
```

APÊNDICE D – PROCESSAMENTO PLACAS

```
from time import sleep
from os import path
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
import os
import shutil
import cv2
import json
import sys
from openalpr import Alpr
import easyocr
import time
from datetime import datetime
import processa_yolo_car
import server
from difflib import SequenceMatcher
import ocr_google
import envia_udp_abre

os.add_dll_directory("C:\openalpr_64")
lastNome = ""
lastPlaca = "" #utilizado para não repetir salvamento da mesma placa

#função ajuste para todos os OCR apos lida a placa
def processa_placa(texto):
      look_number = any(char.isdigit() for char in texto) #verifica se contem
números
      look_letras = any(char.isalpha() for char in texto) #verifica se contem
letras

      if len(texto)==8:
         texto = texto[1:]
      if look_number and look_letras == True:
        maiusculas = texto.upper()
        my_list = list(maiusculas)

        if my_list[0]== "1" :
           my_list[0] = "I"
        if my_list[0]== "W" : #somente para o RS
           my_list[0] = "I"
        if my_list[0]== "U" : #somente para o RS
           my_list[0] = "I"
```

```
44          if my_list[1]== "8" :
45             my_list[1] = "B"
46          if my_list[1]== "6" :
47             my_list[1] = "G"
48          if my_list[2]== "6" :
49             my_list[2] = "G"
50          if my_list[2]== "0" :
51             my_list[2] = "O"
52          if my_list[3]== "O" :
53             my_list[3] = "0"
54          if my_list[3]== "B" :
55             my_list[3] = "8"
56          if my_list[3]== "Q" :
57             my_list[3] = "9"
58          if my_list[3]== "Z" :
59             my_list[3] = "7"
60          if my_list[5]== "S" :
61             my_list[5] = "5"
62          if my_list[5]== "I" :
63             my_list[5] = "1"
64          if my_list[5]== "B" :
65             my_list[5] = "8"
66          if my_list[5]== "Q" :
67             my_list[5] = "O"
68          if my_list[5]== "D" :
69             my_list[5] = "0"
70          if my_list[6]== "L" :
71             my_list[6] = "4"
72          if my_list[6]== "D" :
73             my_list[6] = "0"
74          if my_list[6]== "Z" :
75             my_list[6] = "2"
76          if my_list[6]== "j" :
77             my_list[6] = "1"
78          if my_list[6]== "S" :
79             my_list[6] = "5"
80          if my_list[6]== "I" :
81             my_list[6] = "1"
82          placa = ''.join(map(str, my_list))
83          return placa
84
85  def alpr_eu(imagem):
86      msg1 = "Null"
87      msg2 = "Null"
88      img = imagem
89      alpr = Alpr("eu",r"\Users\Andre
90  Amaral\Desktop\openalpr.conf",r"\Users\Andre Amaral\Desktop\runtime_data")
```

```
91      if not alpr.is_loaded():
92          print("Error loading OpenALPR")
93          sys.exit(1)
94      alpr.set_top_n(20)
95      alpr.set_default_region("br")
96      results = alpr.recognize_file(img)
97
98      if len(results['results']) != 0:
99
100         for (plate) in results['results']:
101            placa = (plate['plate'])
102          #  placa_processada = processa_placa(placa)
103            x = plate['coordinates'][0]['x']
104            y = plate['coordinates'][0]['y']
105            w = plate['coordinates'][2]['x']-plate['coordinates'][0]['x']
106            h = plate['coordinates'][2]['y']-plate['coordinates'][0]['y']
107
108            #crop = imagem_2[y:y+h, x:x+w]
109         return placa
110     else:
111        # shutil.move(imagem, "/Users/Andre
112 Amaral/Desktop/cam_sm_01/processadas/perau/no_plates" + imagem)
113         return msg1,msg2
114
115 def alpr_br(imagem):
116     msg1 = "Null"
117     msg2 = "Null"
118     img = imagem
119     alpr = Alpr("br",r"\Users\Andre
120 Amaral\Desktop\openalpr.conf",r"\Users\Andre Amaral\Desktop\runtime_data")
121     if not alpr.is_loaded():
122         print("Error loading OpenALPR")
123         sys.exit(1)
124     alpr.set_top_n(20)
125     alpr.set_default_region("br")
126     results = alpr.recognize_file(img)
127
128     if len(results['results']) != 0:
129
130         for (plate) in results['results']:
131            placa = (plate['plate'])
132          #  placa_processada = processa_placa(placa)
133
134            x = plate['coordinates'][0]['x']
135            y = plate['coordinates'][0]['y']
136            w = plate['coordinates'][2]['x']-plate['coordinates'][0]['x']
137            h = plate['coordinates'][2]['y']-plate['coordinates'][0]['y']
```

```
138
139             #crop = imagem_2[y:y+h, x:x+w]
140         return placa
141     else:
142        # shutil.move(imagem, "/Users/Andre
143 Amaral/Desktop/cam_sm_01/processadas/perau/no_plates" + imagem)
144         return msg1,msg2
145
146 def alpr_us(imagem):
147     msg1 = "Null"
148     msg2 = "Null"
149     img = imagem
150     alpr = Alpr("us",r"\Users\Andre
151 Amaral\Desktop\openalpr.conf",r"\Users\Andre Amaral\Desktop\runtime_data")
152     if not alpr.is_loaded():
153         print("Error loading OpenALPR")
154         sys.exit(1)
155     alpr.set_top_n(20)
156     alpr.set_default_region("gu")
157     results = alpr.recognize_file(img)
158
159     if len(results['results']) != 0:
160
161         for (plate) in results['results']:
162            placa = (plate['plate'])
163          #  placa_processada = processa_placa(placa)
164
165            x = plate['coordinates'][0]['x']
166            y = plate['coordinates'][0]['y']
167            w = plate['coordinates'][2]['x']-plate['coordinates'][0]['x']
168            h = plate['coordinates'][2]['y']-plate['coordinates'][0]['y']
169
170            #crop = imagem_2[y:y+h, x:x+w]
171         return placa
172     else:
173        # shutil.move(imagem, "/Users/Andre
174 Amaral/Desktop/cam_sm_01/processadas/perau/no_plates" + imagem)
175         return msg1,msg2
176
177 def detecta_veiculo(img):
178    veiculo=processa_yolo_car.detecta_veiculo(img)
179    return veiculo
180
181 def placas_autorizadas():
182    placas_autorizadas = []
183    connection = server.create_db_connection("localhost", "root",
184 "xxxxxxxx","sys")
```

```
185     q1 = """
186     SELECT *
187     FROM authorized
188     ;
189     """
190     results = server.read_query(connection, q1)
191 
192     for result in results:
193       placas_autorizadas.append(result[1])
194     return placas_autorizadas
195 
196 def save_plate(placa,hora_entrada,autorizacao,ocr,time,resultado,nome_foto):
197     global lastPlaca
198 
199     if placa != lastPlaca:
200        lastPlaca=placa
201        if autorizacao == 'sim':
202           contador = 0
203           result=envia_udp_abre.enviaudp()
204           if result=='FALHA' and contador < 2: #reenvia o sinal caso não recebe
205 confirmação pct UDP
206              result=envia_udp_abre.enviaudp()
207              contador+=1
208 
209        salva_entrada = f"""
210        INSERT INTO gate_control (PLACA,DATA,AUTORIZADO,OCR,TIME,RESULTADO,FOTO)
211 VALUES
212        ('{placa}','{hora_entrada}','{autorizacao}','{ocr}','{time}','{resultado
213 }','{nome_foto}')
214        """
215        connection = server.create_db_connection("localhost", "root",
216 "xxxxxxxx","sys")
217        server.execute_query(connection, salva_entrada)
218 
219 def comparaPlaca(placa,placa_aut): #liberar acesso com 6 dígitos
220     return SequenceMatcher(None, placa, placa_aut).ratio()
```

APENDICE E – INTERFACE PRINCIPAL

```
#----- IMPORTS  ------#
from tkinter import *
from tkinter import messagebox
import _thread
import cv2
import sys
import MySQLdb
from PIL import Image, ImageTk
from tkVideoPlayer import TkinterVideo
import tkinter as tk
from time import sleep
from os import path
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
import os
import shutil
import json
import sys
import time
from datetime import datetime
import processa_gate
import webbrowser
import socket
import re

placas_ok = []
lastNome = ""
lastPlaca = ""

#----- PRE CONFIG  ------#

class WatchdogHandler(FileSystemEventHandler):
   def __init__(self, watch_path):
        self.watch_path = watch_path


   def on_created(self, event):
        """Triggered when a file or folder is created."""
        global lastNome
        global placas_ok
        resultado=0
        """Triggered when a file or folder is created."""
        print(f"Imagem recebida: {event.src_path}")
```

```
45          inicio = time.time()
46          full_path = event.src_path
47          #time.sleep(0.1)
48          arquivo, extensao = os.path.splitext(full_path)
49          if extensao == '.jpg' and full_path != lastNome: #testa nome do
50  arquivo é diferente, para não processar mesmo arquivo
51           lastNome=full_path
52           veiculo = processa_gate.detecta_veiculo(full_path)
53           if len(veiculo) != 0:
54              (placa) = processa_gate.alpr_eu(full_path)
55              #print (len(placa))
56              if len(placa) > 6:
57                 #print(placa)
58                 placa=processa_gate.processa_placa(placa)
59                 if placa != None: # caso detecte placa direfente do padrao
60  brasileiro
61                    print("com OCR ALPR EU : ", placa)
62                    show_img_1(full_path)
63                    clearTextInput()
64                    #imagem_2 = cv2.imread(full_path)
65                    #redimensiona = cv2.resize(imagem_2, dsize=(960, 540),
66  interpolation=cv2.INTER_CUBIC)#reduz pela metade
67                    #cv2.imwrite(full_path, redimensiona) #salva no disco
68                    data_e_hora_atuais = datetime.now()
69                    time_foto = data_e_hora_atuais.strftime('%d%m%Y%H%M')
70                    dados=placa+" / "+time_foto
71                    logPlaca.insert(0,dados)
72                    for i in range(len(placas_ok)):
73                       res = processa_gate.comparaPlaca(placa, placas_ok[i])
74                       if res>resultado:
75                          resultado=res
76                    if resultado > 0.7:
77                       print('Veículo placa: ',placa,'autorizado!',"confiança:
78  ",resultado)
79                       logAut.insert(0,"SIM")
80                       fim = time.time()
81                       time_proc=fim - inicio
82                       local=placa+time_foto+".jpg"
83                       processa_gate.save_plate(placa,data_e_hora_atuais,'sim','
84  ALPR(EU)',time_proc,resultado,local)
85                       shutil.move(full_path, "/wamp64/www/gate/fotos/" + placa
86  + time_foto + ".jpg")
87
88                    else:
89                       logAut.insert(0,"NÃO")
90                       print('Veículo placa: ',placa,'não autorizado!')
91                       fim = time.time()
```

```
                    time_proc=fim - inicio
                    local=placa+time_foto+".jpg"
                    processa_gate.save_plate(placa,data_e_hora_atuais,'nao','
ALPR(EU)',time_proc,resultado,local)
                    shutil.move(full_path, "/wamp64/www/gate/fotos/" + placa
+ time_foto + ".jpg")

                else:
                 print("alpr EU falhou")
                 shutil.move(full_path, "/Users/Andre
Amaral/Desktop/cam_gate/no_plates")

            elif len(placa) > 1:
               (placa) = processa_gate.alpr_br(full_path)
               if len(placa) > 6:
                  #print(placa)
                  placa=processa_gate.processa_placa(placa)
                  if placa != None: # caso detecte placa direfente do padrao
brasileiro
                     print("com OCR ALPR BR : ", placa)
                     show_img_1(full_path)
                     clearTextInput()
                     #imagem_2 = cv2.imread(full_path)
                     #redimensiona = cv2.resize(imagem_2, dsize=(960
                     # , 540), interpolation=cv2.INTER_CUBIC)#reduz pela
metade
                     #cv2.imwrite(full_path, redimensiona) #salva no disco
                     data_e_hora_atuais = datetime.now()
                     time_foto = data_e_hora_atuais.strftime('%d%m%Y%H%M')
                     dados=placa+" / "+time_foto
                     logPlaca.insert(0,dados)
                     for i in range(len(placas_ok)):
                        res = processa_gate.comparaPlaca(placa, placas_ok[i])
                        if res>resultado:
                           resultado=res
                     if resultado > 0.7:
                         print('Veículo placa:
',placa,'autorizado!',"confiança: ",resultado)
                         logAut.insert(0,"SIM")
                         fim = time.time()
                         time_proc=fim - inicio
                         local=placa+time_foto+".jpg"
                         processa_gate.save_plate(placa,data_e_hora_atuais,'si
m','ALPR(BR)',time_proc,resultado,local)
                         time.sleep(0.1)
                         shutil.move(full_path, "/wamp64/www/gate/fotos/" +
placa + time_foto + ".jpg")
```

```
                    else:
                        print('Veículo placa: ',placa,'não autorizado!')
                        logAut.insert(0,"NÂO")
                        fim = time.time()
                        time_proc=fim - inicio
                        local=placa+time_foto+".jpg"
                        processa_gate.save_plate(placa,data_e_hora_atuais,'nao
','ALPR(BR)',time_proc,resultado,local)
                        time.sleep(0.1)
                        shutil.move(full_path, "/wamp64/www/gate/fotos/" +
placa + time_foto + ".jpg")

                else:
                    print("alpr br falhou")

                    (placa) = processa_gate.alpr_us(full_path)
                    #print(len(placa))

                    if len(placa) > 6:
                        #print(placa)
                        placa=processa_gate.processa_placa(placa)
                        if placa != None: # caso detecte placa direfente do
padrao brasileiro
                            print("com OCR ALPR US : ", placa)
                            show_img_1(full_path)
                            clearTextInput()
                           # imagem_2 = cv2.imread(full_path)
                           # redimensiona = cv2.resize(imagem_2, dsize=(960, 540),
interpolation=cv2.INTER_CUBIC)#reduz pela metade
                            # cv2.imwrite(full_path, redimensiona) #salva no disco
                             data_e_hora_atuais = datetime.now()
                             time_foto = data_e_hora_atuais.strftime('%d%m%Y%H%M')
                             dados=placa+" / "+time_foto
                             logPlaca.insert(0,dados)
                             for i in range(len(placas_ok)):
                               res = processa_gate.comparaPlaca(placa,
placas_ok[i])
                               if res>resultado:
                                   resultado=res
                             if resultado > 0.7:
                                logAut.insert(0,"SIM")
                                print('Veículo placa:
',placa,'autorizado!',"confiança: ",resultado)
                                fim = time.time()
                                time_proc=fim - inicio
                                local=placa+time_foto+".jpg"
```

```
                            processa_gate.save_plate(placa,data_e_hora_atuais,'
sim','ALPR(US)',time_proc,resultado,local)
                            time.sleep(0.1)
                            shutil.move(full_path, "/wamp64/www/gate/fotos/" +
placa + time_foto + ".jpg")
                        else:
                            print('Veículo placa: ',placa,'não autorizado!')
                            logAut.insert(0,"NÂO")
                            fim = time.time()
                            time_proc=fim - inicio
                            local=placa+time_foto+".jpg"
                            processa_gate.save_plate(placa,data_e_hora_atuais,'
nao','ALPR(US)',time_proc,resultado,local)
                            time.sleep(0.1)
                            shutil.move(full_path, "/wamp64/www/gate/fotos/" +
placa + time_foto + ".jpg")
                       else:
                            print("ALPR US1 falhou")
                            placa=processa_gate.ocr_google.detect_text(full_path)
                            if placa != "NOPLATE":
                                placa=processa_gate.processa_placa(placa)
                            print("com OCR GOOGLE : ", placa)
                            show_img_1(full_path)
                            clearTextInput()
                            data_e_hora_atuais = datetime.now()
                            time_foto = data_e_hora_atuais.strftime('%d%m%Y%H%M')
                            dados=placa+" / "+time_foto
                            logPlaca.insert(0,dados)
                            for i in range(len(placas_ok)):
                              res = processa_gate.comparaPlaca(placa,
placas_ok[i])
                              if res>resultado:
                                  resultado=res
                            if resultado > 0.7:
                               #processa_gate.envia_udp_abre.enviaudp()

                               print('Veículo placa:
',placa,'autorizado!',"confiança: ",resultado)
                               logAut.insert(0,"SIM")
                               fim = time.time()
                               time_proc=fim - inicio
                               local=placa+time_foto+".jpg"
                               processa_gate.save_plate(placa,data_e_hora_atuais,'
sim','GOOGLE OCR',time_proc,resultado,local)
                               time.sleep(0.1)
                               shutil.move(full_path, "/wamp64/www/gate/fotos/" +
placa + time_foto + ".jpg")
```

```
232                         else:
233                            print('Veículo placa: ',placa,'não autorizado!')
234                            logAut.insert(0,"NÃO")
235                            fim = time.time()
236                            time_proc=fim - inicio
237                            local=placa+time_foto+".jpg"
238                            processa_gate.save_plate(placa,data_e_hora_atuais,'
239   nao','GOOGLE OCR',time_proc,resultado,local)
240                            time.sleep(0.1)
241                            shutil.move(full_path, "/wamp64/www/gate/fotos/" +
242   placa + time_foto + ".jpg")
243                     else:
244                        print("ALPR US2 falhou")
245                        placa=processa_gate.ocr_google.detect_text(full_path)
246                        if placa != "NOPLATE":
247                           placa=processa_gate.processa_placa(placa)
248                        print("com OCR GOOGLE : ", placa)
249                        show_img_1(full_path)
250                        clearTextInput()
251                        data_e_hora_atuais = datetime.now()
252                        time_foto = data_e_hora_atuais.strftime('%d%m%Y%H%M')
253                        dados=placa+" / "+time_foto
254                        logPlaca.insert(0,dados)
255                        for i in range(len(placas_ok)):
256                           res = processa_gate.comparaPlaca(placa, placas_ok[i])
257                           if res>resultado:
258                                 resultado=res
259                        if resultado > 0.7:
260                             #processa_gate.envia_udp_abre.enviaudp()
261                              logAut.insert(0,"SIM")
262                              print('Veículo placa:
263   ',placa,'autorizado!',"confiança: ",resultado)
264                              fim = time.time()
265                              time_proc=fim - inicio
266                              local=placa+time_foto+".jpg"
267                              processa_gate.save_plate(placa,data_e_hora_atuais,'
268   sim','GOOGLE OCR',time_proc,resultado,local)
269                              time.sleep(0.1)
270                              shutil.move(full_path, "/wamp64/www/gate/fotos/" +
271   placa + time_foto + ".jpg")
272                        else:
273                              print('Veículo placa: ',placa,'não autorizado!')
274                              logAut.insert(0,"NÃO")
275                              fim = time.time()
276                              time_proc=fim - inicio
277                              local=placa+time_foto+".jpg"
```

```
278                             processa_gate.save_plate(placa,data_e_hora_atuais,'
279 nao','GOOGLE OCR',time_proc,resultado,local)
280                             time.sleep(0.1)
281                             shutil.move(full_path, "/wamp64/www/gate/fotos/" +
282 placa + time_foto + ".jpg")
283          else:
284            data_e_hora_atuais = datetime.now()
285            time_foto = data_e_hora_atuais.strftime('%d%m%Y%H%M%S')
286            print("Nenhum veículo na posição determinada")
287            #shutil.move(full_path, "/Users/Andre
288 Amaral/Desktop/cam_gate/no_veiculos/" + time_foto + ".jpg")
289
290
291 class FolderWatchDog:
292     def __init__(self, handler):
293         """
294         Constructor
295         :param handler: Who will handle what happens in our watch_path.
296         """
297         self.handler = handler
298         self.watch_path = handler.watch_path
299         self.observer = Observer()  # This is the class that does the actual
300 watching.
301
302     def start(self):
303
304
305         self.observer.schedule(self.handler,
306                                path=self.handler.watch_path,
307                                recursive=True)  # If set to true, will watch
308 every folder recursively.
309
310         try:
311             print(f"Starting to watch folder: {self.handler.watch_path}")
312             self.observer.start()
313             while True:
314                 sleep(5)
315
316         except KeyboardInterrupt:
317             self.stop()
318         except Exception as e:
319             print(f"Something went wrong...")
320             print(e)
321
322         finally:
323             print("All done!")
324
```

```
325         def stop(self):
326             self.observer.stop()
327             self.observer.join()
328 
329     #----- FUNCTIONS  ------#
330 
331     def start_vigia():
332         th = _thread.start_new(vigiar,()) #necessário para não ficar preso loop
333 
334     def vigiar():
335         th = _thread.start_new(checa_udp,()) #necessário para não ficar preso loop
336         global placas_ok
337         placas_ok=processa_gate.placas_autorizadas()
338         print(placas_ok)
339         logFrame.insert(0,'OK')
340         qtdfplacasEntry.insert(0,len(placas_ok))
341         to_watch = "/Users/Andre Amaral/Desktop/cam_gate/AKOK2102037T9"
342         w_dog = FolderWatchDog(handler= WatchdogHandler(watch_path=to_watch))
343         w_dog.start()
344 
345     def checa_udp():
346 
347         while True:
348            print("checagem UDP")
349            enviaudp_teste()
350            time.sleep(30) # Sleep for seconds
351 
352     def start_video():
353         th = _thread.start_new(show_frames,()) #necessário para não ficar preso
354     loop
355 
356     def show_frames():
357      while True:
358 
359         #This is to check whether to break the first loop
360         isclosed=0
361         cap =
362     cv2.VideoCapture('rtsp://xxxxxx:xxxxxxx@10.9.90.66:554/cam/realmonitor?channel
363     =3&subtype=1')
364 
365         while (True):
366 
367             ret, frame = cap.read()
368             # It should only show the frame when the ret is true
```

```
369         if ret == True:
370 
371             cv2.imshow('frame',frame)
372             if cv2.waitKey(1) == 27:
373                 # When esc is pressed isclosed is 1
374                 isclosed=1
375                 break
376         else:
377             break
378     # To break the loop if it is closed manually
379 
380  cap.release()
381  cv2.destroyAllWindows()
382 
383 def show_img_1(img):
384     imagem = cv2.imread(img)
385     redimensiona = cv2.resize(imagem, dsize=(1200,800),
386 interpolation=cv2.INTER_CUBIC)#reduz pela metade
387     cv2image = cv2.cvtColor(redimensiona, cv2.COLOR_BGR2RGBA)
388     img2 = Image.fromarray(cv2image)
389     imgtk = ImageTk.PhotoImage(image=img2)
390     mostra_car_1.imgtk = imgtk
391     mostra_car_1.configure(image=imgtk)
392 
393 def clearTextInput():
394     logPlaca.delete("0","end")
395     logAut.delete("0","end")
396 
397 def enviaudp_2():
398 
399     msgFromClient       = "xxxxxxx"
400     bytesToSend         = str.encode(msgFromClient)
401     serverAddressPort   = ("10.9.90.65", xx83)
402     bufferSize          = 50
403 
404     # Create a UDP socket at client side
405     UDPClientSocket = socket.socket(family=socket.AF_INET,
406 type=socket.SOCK_DGRAM)
407     UDPClientSocket.settimeout(5)
408 
409     # Send to server using created UDP socket
410     UDPClientSocket.sendto(bytesToSend, serverAddressPort)
411 
412     while True:
413         try:
414             msg = UDPClientSocket.recvfrom(bufferSize)
415             msg = format(msg[0])
```

```
416             msg = re.sub('[\'b]', '', msg)
417             if msg == 'OK':
418                print("Recebi UDP - PORTAO FECHADO")
419                return msg
420             if msg == 'OPEN':
421                print("PORTAO ABERTO")
422                return msg
423         except socket.timeout:
424           print("Didn't receive data! [Timeout 5s]")
425           return "FALHA"
426
427 def enviaudp_3():
428
429     msgFromClient       = "xxxxxx"
430     bytesToSend         = str.encode(msgFromClient)
431     serverAddressPort   = ("10.9.90.65", xx83)
432     bufferSize          = 50
433
434     # Create a UDP socket at client side
435     UDPClientSocket = socket.socket(family=socket.AF_INET,
436 type=socket.SOCK_DGRAM)
437     UDPClientSocket.settimeout(5)
438
439     # Send to server using created UDP socket
440     UDPClientSocket.sendto(bytesToSend, serverAddressPort)
441
442     while True:
443         try:
444             msg = UDPClientSocket.recvfrom(bufferSize)
445             msg = format(msg[0])
446             msg = re.sub('[\'b]', '', msg)
447             if msg == 'OK':
448                print("Recebi")
449                return msg
450             if msg == 'OPEN':
451                print("PORTAO")
452                return msg
453         except socket.timeout:
454           print("Didn't receive data! [Timeout 5s]")
455           return "FALHA"
456
457 def enviaudp_teste():
458
459     msgFromClient       = "xxxxxxx"
460     bytesToSend         = str.encode(msgFromClient)
461     serverAddressPort   = ("10.9.90.65", xx83)
462     bufferSize          = 50
```

```
463
464         # Create a UDP socket at client side
465         UDPClientSocket = socket.socket(family=socket.AF_INET,
466     type=socket.SOCK_DGRAM)
467         UDPClientSocket.settimeout(5)
468
469         # Send to server using created UDP socket
470         UDPClientSocket.sendto(bytesToSend, serverAddressPort)
471
472         while True:
473             try:
474                 msg = UDPClientSocket.recvfrom(bufferSize)
475                 msg = format(msg[0])
476                 msg = re.sub('[\'b]', '', msg)
477                 if msg == 'CLOSE':
478                    logStatus.delete("0","end")
479                    logStatus.insert(0,'FECHADO')
480                    print("Recebi UDP - PORTAO FECHADO")
481                    return msg
482                 if msg == 'OPEN':
483                    logStatus.delete("0","end")
484                    logStatus.insert(0,'ABERTO')
485                    print("PORTAO ABERTO")
486                    return msg
487             except socket.timeout:
488               logStatus.delete("0","end")
489               logStatus.insert(0,'FALHA COMUNICAÇÃO')
490               print("Didn't receive data! [Timeout 5s]")
491               return "FALHA"
492
493     #---- MAIN FRAME ------#
494     root = Tk()
495
496     root.geometry('700x650+200+200')
497     root.wm_title("Monitoramento portão entrada principal")
498
499     saveplate = IntVar()
500
501     #---------- DETECTS FRAME ----------
502     detectsFrame = LabelFrame(root, text="Monitoramento em tempo real")
503     detectsFrame.place(relwidth=1, relheight=0.65)
504
505     car_ImgFrame = LabelFrame(detectsFrame, text="Portão") #relx=começa a partir
506     de 30, relwidht: tamanho largura, relheight: tamanho altura
507     car_ImgFrame.place(relx=0.00, relwidth=0.99, relheight=1)
508
509     mostra_car_1 = Label(car_ImgFrame, text="Sem veículos")
```

```
510  mostra_car_1.place(relwidth=0.99, relheight=0.99)
511
512  #----- ALARMES FRAME  ------#
513
514  alarmeFrame = LabelFrame(root, text="Informações")
515  alarmeFrame.place( rely=0.65,relwidth=1, relheight=0.25)
516
517  placasFrame = Label(alarmeFrame, text="Nº Placas Aut:", font=("Helvetica",
518  14))
519  placasFrame.place(relwidth=0.28, relheight=0.20)
520
521  qtdfplacasEntry = Entry(alarmeFrame)
522  qtdfplacasEntry.place(relx=0.25, relwidth=0.25, relheight=0.1, rely=0.025)
523  qtdfplacasEntry.insert(0,'')
524
525  MostraPlacas = Label(alarmeFrame, text="Monitoramento:", font=("Helvetica",
526  14))
527  MostraPlacas.place(rely=0.2, relwidth=0.3, relheight=0.15)
528
529  logFrame = Entry(alarmeFrame)
530  logFrame.place(relx=0.25, relwidth=0.25, relheight=0.1, rely=0.230)
531  logFrame.insert(0,'')
532
533  placa_Frame = Label(alarmeFrame, text="Placa/Data:", font=("Helvetica", 14))
534  placa_Frame.place(rely=0.38, relwidth=0.28, relheight=0.15)
535
536  logPlaca = Entry(alarmeFrame)
537  logPlaca.place(relx=0.25, relwidth=0.25, relheight=0.1, rely=0.420)
538  logPlaca.insert(0,'')
539
540  aut_Frame = Label(alarmeFrame, text="Autorizado:", font=("Helvetica", 14))
541  aut_Frame.place(rely=0.55, relwidth=0.28, relheight=0.15)
542
543  logAut = Entry(alarmeFrame)
544  logAut.place(relx=0.25, relwidth=0.25, relheight=0.1, rely=0.600)
545  logAut.insert(0,'')
546
547  status_Frame = Label(alarmeFrame, text="Status Portão:", font=("Helvetica",
548  14))
549  status_Frame.place(relx=0.52, rely=0.025, relwidth=0.28, relheight=0.15)
550
551  logStatus = Entry(alarmeFrame)
552  logStatus.place(relx=0.76, rely=0.026, relwidth=0.20, relheight=0.12)
553  logStatus.insert(0,'')
554

555  #----- CONF FRAME  ------#
556
```

```
557   confFrame = LabelFrame(root, text="Configurações")
558   confFrame.place( rely=0.85,relwidth=1, relheight=0.25)
559
560   streamStartBnt = Button(root, bg='#FFFFCC',text="Start", command=start_vigia)
561   streamStartBnt.place(relwidth=0.15, relheight=0.10, relx=0.01, rely=0.88)
562
563   streamStartBnt = Button(root, bg='#FFFFCC',text="Ao vivo",
564   command=start_video)
565   streamStartBnt.place(relwidth=0.15, relheight=0.10, relx=0.18, rely=0.88)
566
567   streamStartBnt = Button(root,bg='#FFFFCC', text="Editar Veículos",
568   command=lambda: webbrowser.open('http://localhost/gate/index.php'))
569   streamStartBnt.place(relwidth=0.15, relheight=0.10, relx=0.35, rely=0.88)
570
571   streamStartBnt = Button(root, bg='#FFFFCC',text="Abrir portão",
572   command=enviaudp_2)
573   streamStartBnt.place(relwidth=0.15, relheight=0.10, relx=0.52, rely=0.88)
574
575   streamStartBnt = Button(root, bg='#FFFFCC',text="Fechar portão",
576   command=enviaudp_3)
577   streamStartBnt.place(relwidth=0.15, relheight=0.10, relx=0.70, rely=0.88)
578   #----- START APP  ------#
579
580
581   root.mainloop()
582
```

APENDICE F – CÓDIGO ESP8266

```
//Controle portao pela porta UDP
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <WiFiUdp.h>//Biblioteca do UDP.
#include <ArduinoJson.h>
#include "CTBot.h"
#include <esp_task_wdt.h> //Biblioteca do watchdog

CTBot myBot;
WiFiUDP udp;//Cria um objeto da classe UDP.
String req;//String que armazena os dados recebidos pela rede.
const char* ssid = "xxxxxxxx";
//const char* ssid = "xxxxxxxxxx";
//const char* ssid = "xxxxxxxxxxx";
const char* password = "xxxxxxxxxxx";
//const char* password = "xxxxxxxxxxx";
// Informações de acesso para rede de internet / IP Fixo
IPAddress local_IP(10, 9, 90, 65);
//IPAddress local_IP(192, 168, 137,119);
IPAddress gateway(10, 9, 90, 254);
//IPAddress gateway(192, 168, 137, 1);
IPAddress subnet(255, 255, 255, 0);
//IPAddress subnet(255, 255, 255, 0);
//IPAddress primaryDNS(192, 168, 137, 1);
IPAddress primaryDNS(8, 8, 8, 8);
IPAddress secondaryDNS(1, 1, 1, 1);
int rele = 13;
int sfc = 12;
int status_sensor;
//Variaveis bot telegran
String token = "xxxxxxxxxxxxxxxxxxx"; // your Bot Token (Get from Botfather)
unsigned long currentTime=0;
unsigned long previousTime=0;

WiFiClientSecure client;

void setup() {

esp_task_wdt_init(9, true); //tempo nodemcu pode ficar travado
esp_task_wdt_add(NULL); //
myBot.setTelegramToken(token);
pinMode(LED_BUILTIN, OUTPUT);     // Initialize the LED_BUILTIN pin as an
output
```

```
45  pinMode(rele, OUTPUT); //saida rele
46  pinMode(sfc, INPUT_PULLUP);
47  digitalWrite(rele,HIGH); //iniciar desligado
48  Serial.begin(115200);
49  Serial.println();
50   // Configures static IP address
51    if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
52      Serial.println("STA Failed to configure");
53    }
54  Serial.print("Conectando-se a ");
55  Serial.println(ssid);
56  WiFi.begin(ssid, password);
57  while (WiFi.status() != WL_CONNECTED) {
58   digitalWrite(LED_BUILTIN, HIGH);   // Liga led azul quando estiver conectando
59  ao WIFI
60   delay(500);
61   Serial.print(".");
62  }
63  Serial.println("");
64  Serial.println("WiFi conectada.");
65  Serial.println("Endereço de IP: ");
66  Serial.println(WiFi.localIP());
67  digitalWrite(LED_BUILTIN, LOW);   // Desliga  o led se conseguiu conectar
68  udp.begin(83xx);//Inicializa a recepçao de dados UDP na porta 555
69  myBot.sendMessage(xxxxxxxxx, "portao PRF - RESET - ");  // notify the sender
70  //client.setTrustAnchors(&cert); // Add root certificate for api.telegram.org
71
72  //bot.sendMessage(CHAT_ID, "portao PRF - RESET -", "");
73
74  }
75
76  // the loop function runs over and over again forever
77  void loop() {
78    if(WiFi.status() != WL_CONNECTED)
79    {
80    delay(1000);
81    conecta();
82    delay(1000);
83    }
84  if((currentTime-previousTime)>10800000){    //a cada 3 hora reseta udp
85      previousTime=currentTime;
86      reset_udp();
87  }
88
89  status_sensor=digitalRead(sfc);
90  listen();//Sub-rotina para verificar a existencia de pacotes UDP.
91  //Reseta o temporizador do watchdog
```

```
92  esp_task_wdt_reset();
93  }
94
95  void listen()//Sub-rotina que verifica se há pacotes UDP's para serem lidos.
96  {
97     if (udp.parsePacket() > 0)//Se houver pacotes para serem lidos
98     {
99         req = "";//Reseta a string para receber uma nova informaçao
100        while (udp.available() > 0)//Enquanto houver dados para serem lidos
101        {
102            char z = udp.read();//Adiciona o byte lido em uma char
103            req += z;//Adiciona o char à string
104        }
105        //Após todos os dados serem lidos, a String estara pronta.
106        Serial.println(req);//Printa a string recebida no Serial monitor.
107     //  digitalWrite(LED_BUILTIN, HIGH);  // Turn the LED off by making the
108 voltage HIGH
109       if ((req == "xxxxxxxxx")&&(status_sensor == LOW)) {
110
111        digitalWrite(rele,LOW); //iniciar desligado
112        delay(2000);
113        digitalWrite(rele,HIGH); //iniciar desligado
114        digitalWrite(LED_BUILTIN, HIGH);   // Turn the LED on (Note that LOW is
115 the voltage level
116        delay(50);
117        digitalWrite(LED_BUILTIN, LOW);   // Turn the LED on (Note that LOW is
118 the voltage level
119        delay(50);
120
121       IPAddress remoteIp = udp.remoteIP();
122       Serial.print(remoteIp);
123       Serial.print(", port ");
124       Serial.println(udp.remotePort());
125       int udpPort = udp.remotePort();
126       uint8_t buffer[50] = "OK";
127       udp.beginPacket(remoteIp, udpPort);
128       udp.write(buffer, 2);
129       udp.endPacket();
130       memset(buffer, 0, 50);
131    //processing incoming packet, must be called before reading the buffer
132       udp.parsePacket();
133       delay(50);
134
135       }
136
137    if ((req == "xxxxxx")&&(status_sensor == HIGH)) {
138
```

```
139         IPAddress remoteIp = udp.remoteIP();
140         Serial.print(remoteIp);
141         Serial.print(", port ");
142         Serial.println(udp.remotePort());
143         int udpPort = udp.remotePort();
144         uint8_t buffer[50] = "OPEN";
145         udp.beginPacket(remoteIp, udpPort);
146         udp.write(buffer, 4);
147         udp.endPacket();
148         memset(buffer, 0, 50);
149 
150       //processing incoming packet, must be called before reading the buffer
151         udp.parsePacket();
152         delay(50);
153         }
154      if (req == "xxxxxxxxxx"){
155 
156           digitalWrite(rele,LOW); //iniciar desligado
157           delay(2000);
158           digitalWrite(rele,HIGH); //iniciar desligado
159           digitalWrite(LED_BUILTIN, HIGH);   // Turn the LED on (Note that LOW is
160    the voltage level
161           delay(50);
162           digitalWrite(LED_BUILTIN, LOW);   // Turn the LED on (Note that LOW is
163    the voltage level
164           delay(50);
165         IPAddress remoteIp = udp.remoteIP();
166         Serial.print(remoteIp);
167         Serial.print(", port ");
168         Serial.println(udp.remotePort());
169         int udpPort = udp.remotePort();
170         uint8_t buffer[50] = "OK";
171         udp.beginPacket(remoteIp, udpPort);
172         udp.write(buffer, 2);
173         udp.endPacket();
174         memset(buffer, 0, 50);
175       //processing incoming packet, must be called before reading the buffer
176         udp.parsePacket();
177         delay(50);
178         myBot.sendMessage(xxxxxxxx, "portao PRF - ACIONADO - ");  // notify the
179    sender
180         }
181    if ((req == "xxxxxxxx")&&(status_sensor == HIGH)) {
182 
183         IPAddress remoteIp = udp.remoteIP();
184         Serial.print(remoteIp);
185         Serial.print(", port ");
```

```
186         Serial.println(udp.remotePort());
187         int udpPort = udp.remotePort();
188         uint8_t buffer[50] = "OPEN";
189         udp.beginPacket(remoteIp, udpPort);
190         udp.write(buffer, 4);
191         udp.endPacket();
192         memset(buffer, 0, 50);
193       //processing incoming packet, must be called before reading the buffer
194         udp.parsePacket();
195         delay(50);
196         }
197
198   if ((req == "xxxxxxxxxx")&&(status_sensor == LOW)) {
199
200         IPAddress remoteIp = udp.remoteIP();
201         Serial.print(remoteIp);
202         Serial.print(", port ");
203         Serial.println(udp.remotePort());
204         int udpPort = udp.remotePort();
205         uint8_t buffer[50] = "CLOSE";
206         udp.beginPacket(remoteIp, udpPort);
207         udp.write(buffer, 5);
208         udp.endPacket();
209         memset(buffer, 0, 50);
210       //processing incoming packet, must be called before reading the buffer
211         udp.parsePacket();
212         delay(50);
213         }
214       }
215   }
216   void conecta() {
217   ESP.restart();
218   }
219
220   void reset_udp(){
221
222         udp.stop();
223         udp.begin(83xx);//Inicializa a recepçao de dados UDP na porta 555
224         myBot.sendMessage(xxxxxxxxx, "portao PRF - UDP reset - ");  // notify
225   the sender
226
227   }
```