CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE INDAIATUBA

CURSO DE TECNOLOGIA EM REDE DE COMPUTADORES

FABRICIO HENRIQUE CARDOSO ALVES

Sistema de controle de estacionamento com tecnologia OCR

INDAIATUBA
2017

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE INDAIATUBA

CURSO DE TECNOLOGIA EM REDES DE COMPUTADORES

FABRICIO HENRIQUE CARDOSO ALVES

Sistema de controle de estacionamento com tecnologia OCR

Projeto de Trabalho de Graduação apresentado por Fabricio Henrique Cardoso Alves como pré-requisito parcial para a conclusão do Curso Superior de Tecnologia em Redes de Computadores, da Faculdade de Tecnologia de Indaiatuba, elaborado sob a orientação do Prof. Dr. Jaime Cazuhiro Ossada.

INDAIATUBA
2017

CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

FACULDADE DE TECNOLOGIA DE INDAIATUBA

CURSO DE TECNOLOGIA EM REDES DE COMPUTADORES

FABRICIO HENRIQUE CARDOSO ALVES

Banca Avaliadora:

Jaime Cazuhiro Ossada	Orientador
André Luiz Silva	Professor Convidado
Carlos Alberto Bucheroni	Profissional da área

Data da Defesa: 28/06/2017

Dedico este trabalho à minha esposa, Barbara Alves, por sua paciência, incentivo e por estar ao meu lado me apoiando em todos os momentos. Dedico também à minha mãe por tudo o que sacrificou para me criar em bons caminhos e ensinar princípios valiosos, sendo um grande exemplo para mim.

Agradecimentos

Agradeço primeiramente a Deus por colocar as oportunidades em meu caminho.

Agradeço a minha esposa e meus filhos por me apoiar e me ajudar a chegar onde cheguei, alcançando mais uma etapa importante em minha vida. Ao meu irmão Danilo que compartilhou de sua experiência acadêmica.

Agradeço ao Prof. Dr. Jaime Cazuhiro Ossada, como professor e orientador, por tudo que contribuiu para a realização deste trabalho.

Agradeço a todos os professores da Faculdade de Tecnologia de Indaiatuba por compartilharem seus conhecimentos de modo a tornar possível o desenvolvimento deste trabalho.

RESUMO

Essa pesquisa mostra etapas básicas de um sistema óptico de reconhecimento de caracteres, Desde a captura da imagem, o tratamento e processamento da imagem e a exibição do resultado. Para a demonstração do funcionamento desse sistema foi desenvolvido um protótipo de controle de acesso de veículo, de modo que quando um veículo se aproxima da cancela, o sensor de presença aciona uma webcam, uma imagem da placa do veículo é gerada e através de um sistema desenvolvido em Python se o veículo tem autorização para a entrada. O hardware utilizado no protótipo é um arduino e o sistema OCR utilizado é a biblioteca Tesseract.

Palavras chave: Python, OCR (Optical Character Recognition), Tesseract, Arduino

LISTA DE FIGURAS

SUMÁRIO

INTRODUÇÃO

No século XVIII, na Inglaterra, logo que começou a Revolução Industrial, aos poucos as máquinas assumiram alguns postos de trabalhos das pessoas. Isso causou revolta nos trabalhadores, e então provocou violência, por acreditarem que esta substituição era a causa de tantos desempregos. Por conta disso apareceram movimentos como as "*tradeunions*", sindicatos ingleses, ou o pior deles, chamado de "ludista", que ocorreu na França e ficou conhecido como os "quebradores de máquinas", onde os integrantes desse "grupo" invadiam as fábricas e estragavam os equipamentos(BRAICK, P. R. E MOTA, 2005)

Com o passar dos anos, a "história tem mostrado" que os aparelhos eletroeletrônicos, eletromecânicos e os computadores trazem soluções e vantagens, diminuindo erros humanos.

Graças a esses avanços tecnológicos podemos ter a possibilidade de ter controle do "vai e vem" de veículos em diversos meios urbanos em nosso cotidiano.

Atualmente é muito comum nos depararmos com câmeras instaladas em casas, condomínios, nos comércios e nas ruas com o objetivo de monitorar e trazer mais segurança para as pessoas.

O controle de automóveis que circulam pelas ruas, avenidas e etc., tem sido necessário e cheio de desafios, devido ao fato, de ter aumentado a quantidade de automóveis. Hoje no Brasil, de acordo com o DENATRAN (Departamento Nacional de Trânsito), 91 milhões de carros circulam hoje em todo o território nacional[1]. Nos grandes centros onde a aglomeração dos veículos é maior, é normal encontrarmos câmeras de monitoramento que são utilizadas como radares.

A câmera vem sendo aperfeiçoadas a cada dia e ficando cada vez mais eficientes, devido o avanço da tecnologia, que proporciona computadores cada vez mais rápidos, facilitando assim o processamento de imagens das câmeras que capturam imagens dos veículos nas vias em tempo real. Junto a elas criou-se os sistemas que podem identificar as placas dos veículos e através delas saber a situação do veículo, se pode transitar ou não. Isso é possível devido ao sistema conhecido como OCR (Optical, Character Recognation) Reconhecimento Ótico de Caracteres, onde se pode reconhecer padrões pré-definidos na configuração do sistema e procurarem padrão nas imagens e transformar os caracteres na imagem em texto.

---

[1] http://www.denatran.gov.br/frota

[2] André luiz da câmara muniz UNICEUB

Diante disso surge a questão que norteia essa pesquisa: como desenvolver um sistema de identificação de caracteres alfanuméricos de placas de veículo, utilizando programas open source e hardware livre?

Assim, esse trabalho tem como objetivo, desenvolver um sistema de leitura de placas de veículos de baixo custo, utilizando plataformas open source que trabalham com sistema OCR para converter imagens de placas de carro no padrão brasileiro em texto, e enviar esse resultado um computador, que por sua vez ira interagir com sistema de hardware livre dando a permissão ou não do veículo ao estacionamento. No presente momento desse projeto não se abordará os métodos de segurança do sistema.

A metodologia a ser utilizada para desenvolver este estudo será a pesquisa experimental que, de acordo com (GIL, 2002), que consiste em um método de estudo no qual se define um objeto de estudo, selecionando as variáveis que têm a capacidade de influenciá-lo e então observar os efeitos causados no objeto. As variáveis podem, então, ser manipuladas uma a uma para se verificar o que ocorre com as demais, e como isso pode influenciar no resultado, definindo o pesquisador como um atuante, não sendo um espectador.

Para que esse objetivo seja alcançado, se criará o um protótipo simulando o ambiente do estacionamento privado: quando carro chegar ao portão uma câmera efetuara a captura da placa do veículo. Essas imagens serão enviadas ao computador que processará as imagens e comparará o resultado com o veículo autorizado a entrar, caso positivo acancela se abrirá automaticamente, caso de negativo um alerta será enviado a guarita do segurança.

Para o desenvolvimento dessa pesquisa será utilizado a linguagem de programação Python, que de acordo com (BORGES, 2014) é uma linguagem de altíssimo nível e orientado a objeto, de tipagem dinâmica e forte. Devido ao Python ser um software livre, hoje existem diversos módulos e bibliotecas que facilitam sua utilização.

Esse trabalho é apresentado em três capítulos: o primeiro é a fundamentação teórica que aborda a visão computacional, seu avanço e utilidade nos dias de hoje;

O segundo capítulo traz a metodologia do projeto, que é experimental e efetua-se pela a elaboração do protótipo mostrando os detalhes de seu desenvolvimento;

No terceiro capitulo mostra a Apresentação, Documentação e Análise dos Dados;

E por fim, as considerações finais, as referências bibliográficas, os documentos utilizados para a pesquisa e os anexos.

CAPÍTULO I

1. Fundamentação Teórica

1.1. Visão Computacional

Os humanos podem perceber imagens com estruturas tridimensionais facilmente, (SZELISKI, 2010), cita como exemplo, quando se olha a um vaso de flor em cima de uma mesa, se pode descrever com clareza a forma do vaso os detalhes das pétalas, da luz, a paisagem atrás ou quando se vê uma fotografia é fácil identificar quantas pessoas está na foto, quem são ou onde estão. Há anos pesquisadores tem se dedicado a conhecer o funcionamento do olho humano.

O humano primeiramente vê para aprender algo. Já a visão computacional é o processo inverso, a máquina precisa primeiro receber a informação, para depois reconhecer uma imagem.

Segundo o artigo de (MUNIZ, 2009), a visão computacional também pode se assemelhar à forma como os humanos enxergam, seguindo a mesma ordem, em que primeiro tem a captura da imagem. Logo após em segundo a "cena" é processada e depois vem a etapa em que se identifica a imagem. Apesar do processo ser semelhante, a forma da máquina traduzir uma imagem difere de como a visão do homem faz. Ele diz que as máquinas não conseguem compreender a conexão lógica ou familiar de uma imagem, mas, a reconhece por meio de aspectos particulares, constituindo um reconhecimento impreterivelmente comum, e elas não tem capacidade para separar aspectos que não contribuem para a execução do acontecimento em destaque. Em compensação, a visão dos seres humanos, trabalham de um método mais complicado, em que é provável desconsiderar todos os fatos que não tem ligação presente com a situação a ser detectada, filtrando e guardando somente o que é importante para a compreensão da ocorrência.

De acordo com(BALLARD; BROWN, 1982), visão computacional é a ciência que estuda a visão de uma máquina, fazendo com que se extraia informações através de dispositivos

como câmera de vídeo, sensores, scanners e outros dispositivos que consigam captar alguma informação significativa, com essas informações capturada o computador pode reconhecer, manipular e pensar sobre os objetos que compõem os dados adquiridos.

O entendimento da imagem é muito diferente do processamento da imagem, as instruções são um pré-requisito para o reconhecimento, manipulação dos objetos da imagem. Sobre essa questão, (LEVIN, 2006)explica que um computador sem uma programação adicional é incapaz de responder as perguntas mais simples sobre uma imagem, por exemplo, se contém uma pessoa ou um objeto, se a imagem foi feita de dia ou de noite, etc. Então o campo da visão computacional foi criado resolver esses problemas.

Desde os anos 50 já se falava de visão computacional, mas os primeiros experimentos significativos para a evolução da visão computacional ocorreram na década de 70, onde aliada com a matemática tridimensional e a inteligência artificial, acreditava-se que em pouco tempo seria possível trazer o sentido da visão para os computadores, mas com o passar dos anos descobriu-se que o processo era mais complexo do que se imaginava, pois a maior dificuldade era fazer com que o computador funcionasse como o cérebro humano, ao ver uma imagem identificasse de forma rápida e processar sua informação identificando o objeto.(GREENEMEIER, 2008)

1.2.Aplicações da Visão Computacional

Hoje em dia, com o avanço da tecnologia, a visão computacional ganhou uma maior abrangência em seus usos, nas mais variadas áreas, como na agronomia: utilizada para identificação de pragas e doenças no campo, robotização da colheita; na medicina utilizada para reconhecimento de câncer e outros tumores; na indústria de produção para fazer controles de qualidade, braços mecânicos que fazem tarefas de alta precisão através dos sensores instalados neles; na área militar utilizada para reconhecimento de áreas inimigas e alvos utilizando imagens de satélites; na segurança pública utiliza o reconhecimento de faces para encontrar pessoas procuradas em áreas com grande concentração de pessoas como aeroportos e grandes centros da cidade; os veículos autônomos que são carros podem se guiar sozinhos com a capacidade de reconhecer obstáculos a sua volta; no transito utilizado para fazer a análise de tráfico, além dos sistemas de radares inteligentes instalados principalmente nas grandes cidades como por exemplo na cidade de São Paulo, onde diversos radares fixos com câmeras são instalados em diversos pontos da cidade, esses radares tem por função além,

de verificar a velocidade do veículo que está trafegando, as câmeras tem uma tecnologia conhecida como OCR, Reconhecimento Ótico de Caracteres, que por sua vez consegue reconhecer as placas dos veículos e assim saber se aquele veículo está em dia com a documentação, se o veículo é roubado ou até verificar se o carro está infligindo o rodízio de carros.

Assim esses sistemas em uma fração de segundos conseguem localizar e detectar os carros infratores e aplicar as multas ao mesmo tempo, eliminando assim as Blitz policiais que bloqueiam as faixas causando lentidão no trânsito. Nas cidades onde os sistemas de câmeras são instalados para fazer a segurança pública, alguns sistemas são capazes de reconhecer sozinhos um acidente, seja uma colisão de veículos ou um atropelamento, ele aciona um alarme avisando as autoridades e mostra o local, para que eles tomem as medidas cabíveis o mais breve possível (MARQUES, 2011).

(SZELISKI, 2010) cita exemplos de David Lowe's[2] que nos mostram várias aplicações interessantes da visão computacional, que pessoas comuns usam no seu dia a dia, por exemplo ao utilizar a câmera fotográfica que dispara através do reconhecimento de um sorriso, ao fazer uma foto panorâmica onde a câmera "mescla" várias fotos formando uma única foto (figura 1a), utilizar filtros nas fotos através de aplicativos nos smartphone alterando a exposição da luz criando uma nova imagem. (Figura 1 b), mesclar a foto de um amigo com outro utilizando o processo da morfologia (figura 1c), A detectação de faces é um recurso comum nas redes sociais hoje, diversos laptop smartphones e até caixas eletrônicos utilizam da biometria para fazer o reconhecimento das pessoas.

[2](http://www.cs.ubc.ca/spider/lowe/)

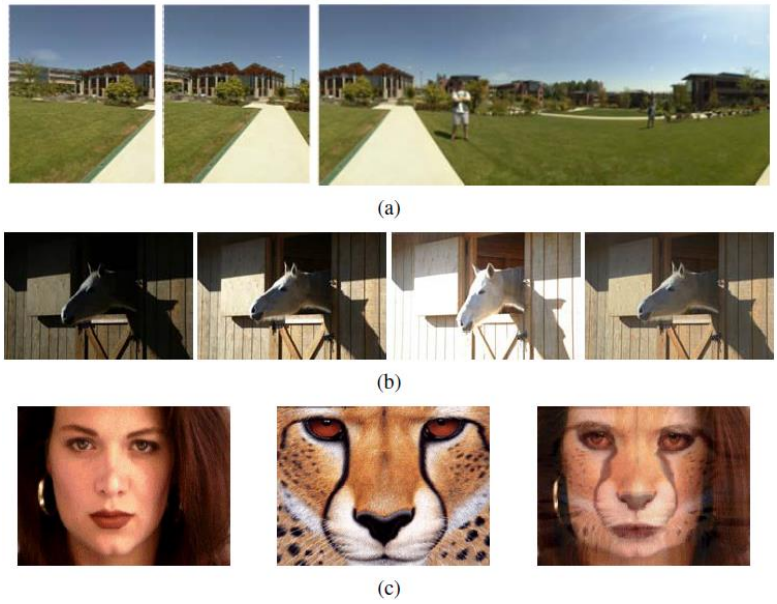

Figura 1: Algumas aplicações da visão computacional.

1.3. Soluções para Visão Computacional

Uma das soluções conhecidas para visão computacional é o OpenCV (Open Source Computer Vision Library) que é uma biblioteca em código aberto com mais de 2500 algoritmos otimizados atendendo os mais diversos campos da área da visão computacional como reconhecimento de rostos, identificação de objetos, detectação de movimentos, reconhecimento de placas e caracteres, comparação de imagens etc. É estimado que mais de 47 milhões de pessoas utilizem o OpenCV, dentre elas grandes empresas como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota além de grupo de pesquisas e órgãos governamentais. Essa plataforma é utilizada nos mais diferentes tipos de projetos como por exemplo detecção de intrusos em vídeos de vigilância em Israel, monitoramento de equipamentos para minas na China, ajudando robôs navegar e pegar objetos no Willow Garage que desenvolve hardware e software open source para aplicações de robótica pessoais, detecção de acidentes de afogamento em piscinas na Europa, verificando pistas para detritos na Turquia, inspecionando os rótulos dos produtos em fábricas ao redor do mundo para a detecção de rosto rápido no Japão.[3]

O Tesseract é um desses algoritmos utilizado no Opencv, é um API de código aberto utilizado para reconhecimento ótico de caracteres, inicialmente desenvolvida e mantida pela HP entre os anos de 1985 a 1985. Em 2005 a Google deu continuidade ao projeto[4]. Além do

[3]Conforme disponível em: <http://opencv.org/about.html> Acessado em Abr/2017
[4]https://github.com/tesseract-ocr/tesseract/blob/master/doc/tesseract.1.asc

seu grande desempenho na busca de caracteres nas imagens, se destaca também pela facilidade de sua integração com outras linguagens de programação.

1.4. Componentes de um sistema de Processamento de Imagem

(GONZALEZ; WOODS, 2002) nos apresentam cada um dos principais componentes básicos de um sistema de processamento de imagem como ilustrado na figura 2

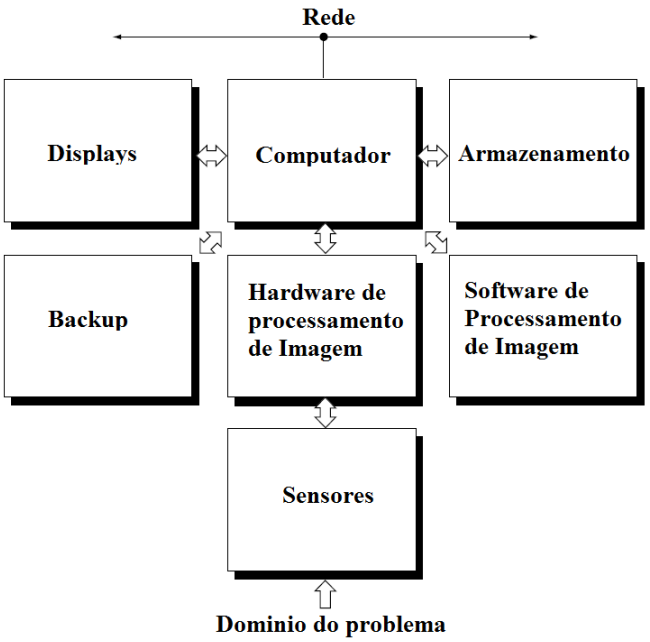

Figura 2: Principais componentes de um sistema de processamento de imagem.

O processamento da imagem inicia-se nos sensores, que podem ser definidos como dispositivos físicos que são sensíveis à energia irradiada pelo objeto que desejamos capturar, então após a captura do sinal de entrada, os sensores convertem os dados em sinal digital, e enviando-os para a próxima etapa.

- **Hardware de Processamento de Imagem:** Esse dispositivo recebe o sinal de entrada e faz um pré-processamento das imagens, e envia para a próxima etapa.
- **Computador:** Como se pode ver na figura 2 o computador é o ponto central do sistema de processamento de imagem, onde se faz o recebimento e a distribuição das tarefas. O computador indicado para essa tarefa dependerá do nível do processamento de imagens podendo variar de um computador de uso geral até um super-computador.
- **Software de Processamento de imagem:** é um sistema com vários algoritmos pré-definido, onde buscam informações na imagem e retorna dados importantes ao computador.

- **Armazenamento:** pode-se classificar o armazenamento em 3 categorias (1) de curto prazo para o uso durante o processamento, (2) o de recuperação rápida como por exemplo a memória ram, e (3) o de arquivamento utilizado para buscas posteriores, como exemplo os HD.
- **Displays:** é o dispositivo que retorna a saída em modo gráfico para o usuário, exemplos: TVs, Monitores
- **Backup:** Dispositivos que fazem uma cópia do resultado do processamento de imagem apresentado, exemplos Dvds, Pen drives, impressoras laser e jato de tinta.
- **Rede:** é quase uma função essencial em qualquer sistema de computação hoje. Devido à grande quantidade de dados em aplicações de processamento de imagem, um fator fundamental na transmissão de imagem é a largura de banda. Em redes dedicadas, isso normalmente não é um problema, mas em redes que utilizam a internet como o meio de transferência dos dados nem sempre terá um bom desempenho. Mas com o crescimento da fibra óptica nas redes privadas e na internet, esse problema tende a melhorar.

1.5. Revisão da Literatura

Nesta seção são apresentados estudos e pesquisas relacionados ao tema deste trabalho. O levantamento realizado foi orientado pela busca de pesquisas científicas e/ou tecnológicas que têm em seus objetivos o desenvolvimento/implementação e/ou análise de ferramentas, tecnologias OCR para leitura de placas de veículos.

A ferramenta que serviu de referência para isso foi o *Google* Acadêmico por meio do qual buscamos mapear as pesquisas dessa natureza circunscritas nos últimos anos.

(MUNIZ, 2009) em seu trabalho para o Centro Universitário de Brasília (UniCEUB), apresentou um protótipo de um controle de acesso de veículos automotores em ambientes privados.

O protótipo é constituído de um sensor de presença na cancela de entrada esse sensor reconhece o veículo e faz uma foto de sua placa, ao ser consultado em um banco de dados verifica se o veículo tem autorização para entrada. O sistema que gerência todo o sistema foi desenvolvido em Delph, e é interligado ao hardware através da porta paralela do computador.

Após os testes os resultados obtidos do funcionamento do mecanismo do protótipo foram satisfatórios, as restrições citadas pelo autor foram com respeito a luz incidente na webcam e sua distância da captura, os testes foram realizado capturando a tela de um monitor com a webcam, a fonte utilizada durante todo o processo foi arial tamanho entre 20 a 40, com

a webcam posicionada a 20cm e 30cm do monitor. Caso esses requisitos não se cumprir, a leitura OCR do sistema pode ser comprometida.

No ano de 2011, (PASSELLA; SOUZA, 2011), em seu trabalho para o UNISEB fizeram um sistema que pudesse encontrar e reconhecer placas de veículos de maneira automática.

Como metodologia foram utilizada as ferramentas Java e API Image J para linguagem JAVA e em seu trabalho eles informaram que um resultado satisfatório mesmo com 100 imagens foi alcançado em partes, e que é necessário melhorar o reconhecimento de caracteres.

Tiveram dificuldades também de encontrar formas que pudessem solucionar os problemas de reconhecimento de placas vermelhas e brancas e que fizessem a diferenciação de símbolos e faróis.

Acreditam que num trabalho posterior, poderia haver um aprimoramento de algoritmos e técnicas, especialmente no fracionamento dos caracteres, achar uma forma de fazer com que a máscara que faz a varredura da placa na foto seja diferente de maneira automática dependendo do tamanho que tiver a placa e depois levar o sistema para o Android que usa a linguagem Java.

(NASCIMENTO, 2012) desenvolveu no Centro Universitário De Brasília - UniCEUB, um software para identificar placas veiculares.

Para o procedimento de captura, foram utilizados dois equipamentos para filmar: uma webcam que faz filmagem em qualidade VGA e a câmera de um Iphone 4S de oito Mega pixel. Houve a realização de três testes diferentes: para verificar a exatidão em detectar a placa do veículo; do desempenho do algoritmo sendo utilizado as bibliotecas OpenCV e Tesseract OCR, e da precisão na mudança dos caracteres da gravura para o texto.

Ele concluiu dizendo que a meta do trabalho foi concluída para grande parte dos testes, porém teve dificuldades por conta da baixa qualidade da imagem feita pela câmera e a baixa iluminação, foi necessário fazer uma edição de imagem, aplicando filtros para reduzir os ruídos. Entre outras dificuldades.

(WEBER, 2013) desenvolveu na Universidade Federal do Rio Grande do Sul, um protótipo para detecção automática de placas de veiculares brasileiras.

A metodologia da pesquisa consiste na montagem de um protótipo utilizando um raspberryPi B equipada com uma câmera, utilizando um sistema desenvolvido em C++ além de um compilador MinGW/GCC e a biblioteca do OpenCV, tudo aplicado sobre o sistema operacional Raspbian. O autor faz diversos teste para verificar a melhor forma de se efetuar o tratamento da imagem e em seguida apresentas os resultados desse experimento. A captura

das placas para análise foi feita através de um vídeo com veículos, simulando o protótipo instalado em uso real.

Com a pesquisa, ficou concluído que é possível construir um algoritmo eficiente de reconhecimento placa utilizando baixo processamento. Foi constatado que a captura das placas traseiras dos veículos gera menos processamento e mais eficiência no resultado, porém adesivos nos para-choque dos carros podem resultar em um falso positivo.

No ano de 2014, (CABRAL; MACHADO, 2014) em seu trabalho para a Universidade Tecnológica Federal do Paraná desenvolveu um Identificador de Placa Veicular, para melhorar a segurança de alunos em escolas particulares ou públicas.

Para desenvolver o aplicativo ele fez uma pesquisa com a equipe que faz a segurança de uma instituição de ensino para averiguar qual o melhor jeito de utilizar o aplicativo, quais dados que o aplicativo precisa mostrar como informações cadastrais dos alunos e que foram definidos pela gestora e assessora do colégio. Houve a utilização de um *WebService* para o módulo de sincronização do aparelho que foi ajustado pela equipe de Tecnologia do colégio e que foi feito para um aparelho *smartphone* com plataforma *Android,* que tinha que possuir câmera e acesso a uma rede de dados. Então para identificar os veículos houve a utilização de uma biblioteca OCR (*Optical Character Recognition),* que interpreta a foto da placa do veículo feita pela câmera do aparelho. Depois do processamento tem-se uma resposta do OCR em forma de um texto, e então ele pesquisa na base de dados do aparelho para encontrar o registro do veículo.

Realizaram vários testes e pelo OCR foi obtido 58% de sucesso e concluiu que caso a foto não fosse reconhecida, a pessoa responsável pelo uso do aplicativo precisaria informar as informações da placa manualmente.

No ano de 2015, (NARDI; POLAK; VALIATI, 2015), em seu trabalho para o Universidade Tecnológica Federal Do Paraná, desenvolveram um sistema de controle de acesso veicular baseado em reconhecimento digital de caracteres (tecnologia OCR) em imagens.

Para realizar o trabalho foi utilizado um computador simples, o *Raspberry-Pi*modelo B que tem uma tecnologia de custo baixo e poderoso computacionalmente, um servo motor para abrir a cancela e dois sensores ultrassônicos para verificar a presença dos automóveis. O programa utilizado foi: a biblioteca OpenCV, a biblioteca Tesseract-OCR, a biblioteca ALPR com pouquíssimas alterações específicas para que atendesse melhor à proposta do projeto além das soluções criadas para acionar o motor e os sensores.

Foram realizados variados testes e puderam observar que não há 100% de reconhecimento dos caracteres das placas e algumas vezes não reconheceu. A biblioteca APLR possui alguns defeitos para reconhecer alguns caracteres, por exemplo o número 8 que foi identificado como letra B. Porém perceberam que nas fotos recebidas pela Raspicam, quando aparecia o carro na foto, houve um acerto de mais ou menos uns 70% na identificação dos caracteres.

CAPÍTULO II

2. Metodologia

2.1.Natureza da Pesquisa

Segundo (GIL, 2002), uma pesquisa é definida como um método sistemático e racional de se obter respostas para os problemas apresentados. Torna-se necessário o uso desse recurso quando não se possui informações suficientes para a resolução do problema ou então quando se possui as informações, porém essas estão desorganizadas, impossibilitando a sua ligação com o problema proposto.

No decorrer do desenvolvimento de uma pesquisa, (GIL, 2002)afirma que diversas etapas estão envolvidas, iniciando pela apresentação do problema até atingir resultados de modo satisfatório, nesse processo devem ser utilizados técnicas, métodos e outros procedimentos científicos.

As pesquisas podem ser realizadas por diversos motivos, porém dois deles devem ser considerados como referência: *razões de ordem intelectual*, quando há o desejo de conhecer para satisfazer o próprio conhecimento e *razões de ordem prática*, que é quando se deseja obter o conhecimento para realizar algo com mais eficiência e eficácia. (GIL, 2002).

Tendo em vista o problema proposto, a natureza da pesquisa a ser utilizada para o desenvolvimento deste estudo será a pesquisa experimental, entendida aqui como um método no qual é definido um objeto de estudo, selecionada as variáveis que podem influenciá-lo e então se observa os efeitos causados nesse objeto. As variáveis então podem ser manipuladas uma a uma para se verificar os impactos dessas e como influenciam no resultado, assim, o pesquisador é um atuante, e não um espectador.(GIL, 2002).

Para ser considerada uma pesquisa experimental, (GIL, 2002)afirma que é necessário que haja manipulação de ao menos uma das características dos elementos do estudo; sejam inseridos controles no contexto do experimento; e que haja aleatoriedade ao designar os elementos que participarão dos grupos de pesquisa. Por mais que contenha muitas limitações, a pesquisa experimental possui diversas possibilidades de controle, fazendo dela uma excelente maneira de se testar hipóteses e relacionar a causa e o efeito das variáveis.

2.2.Experimento da Pesquisa

Para a demonstração do experimento foi elaborado um protótipo utilizando um hardware livre, que para essa pesquisa estaremos utilizando o Arduino Uno que é um microprocessador baseado no ATmega328. Segundo (BANZI; SHILOH, 2015): "O Arduino é uma plataforma de computação física de fonte aberta. Pode ser utilizado para desenvolver objetos interativos independentes, ou conectado a softwares do seu computador [...]". Mas o grande diferencial do arduino é sua facilidade e a praticidade de se comunicar com diversos hardwares e softwares.

Como visto no capítulo anterior, na revisão de literaturas, as pesquisas apresentadas foram feitas utilizando a linguagem JAVA, Delph, C++. Para esta pesquisa se tem o objetivo de utilizar a linguagem de programação Python, que é uma linguagem desenvolvida por Guido Van Rossum em 1991, que foi desenvolvida para manter um código limpo e fácil e de maneira rápida. As bibliotecas e plug-ins disponíveis são um grande diferencial da linguagem.

A proposta desta pesquisa é fazer um protótipo de um estacionamento privado, onde a liberação será efetuada através da tecnologia OCR onde a leitura da placa do veículo feita por uma foto da placa. Com esse protótipo em funcionamento será avaliado a viabilidade do uso da linguagem de programação Python e suas bibliotecas para esses fins.

Os Caracteres impressos para simular as placas dos veículos será impressa na fonte Mandatory que desde 2008 é a fonte utilizada conforme a Resolução 231 do Contran[5]. Para verificar a eficiência da captura colocaremos algumas imagens de placas de transito atuais para verificar o resultado.

Nessa pesquisa não será abordado questões referentes a segurança do sistema e banco de dados.

[5]http://www.denatran.gov.br/download/Resolucoes/RESOLUCAO_231.pdf

2.3.Materiais e métodos

Para a criação do protótipo foi utilizado um arduino Uno, um sensor ultrassônico HC-SR04, um Micro Servo 9g SG90, um protoboard, Leds, resistência 1k2 ohm, fios para a conexão do circuito e uma webcam Microsoft Lifecam VX-3000. A figura 3 mostra a conexão da montagem do arduino.

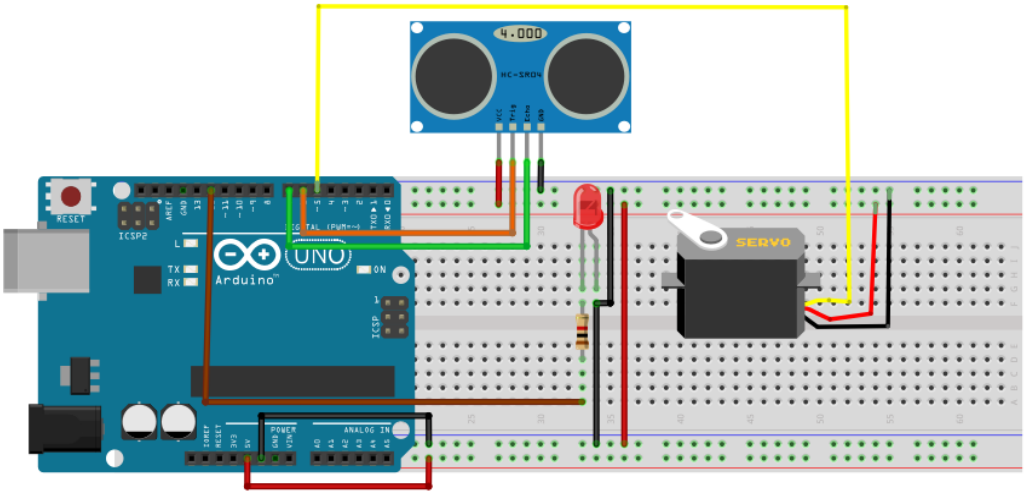

Figura 3 - Montagem do circuito

O diagrama da figura 4 apresenta o funcionamento geral do protótipo. O sensor ultrassônico HC-SR04 fica em constate comunicação com o arduino verificando se chegou algum veículo, por sua vez o arduino verificando alteração no resultado do sensor envia um sinal para o servidor. O servidor então tira uma foto através da webcam, essa foto é processada e transformada em texto, assim se torna possível que o servido possa consultar o veículo em um banco de dados. Caso o veículo esteja autorizado a entrar o servidor envia um sinal ao arduino permitindo o acionamento da cancela, caso contrário um led vermelho se acenderá sinalizando a que o carro não tem permissão.

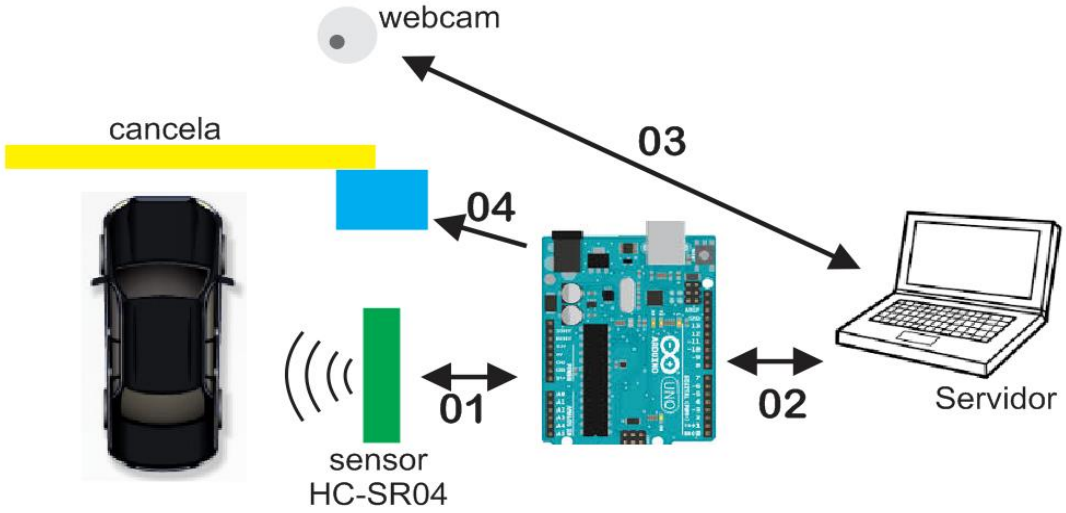

Figura 4 - Fluxo do sistema

CAPÍTULOIII

3. Apresentação, Documentação e Análise de Dados

3.1.Desenvolvimento da Aplicação

O primeiro experimento feito foi verificar a confiabilidade do Pytesser. O Pytesser o modulo do Tesserract adaptada para o python desenvolvido por Michael J.T. O'Kelly,
onde se tem o objetivo de tomar como entrada um arquivo de imagem e fazer a leitura procurando padrões conhecidos, nesse caso os caracteres, na imagem e convertendo em uma string em sua saída.

Foi utilizado algumas imagens de placas de veículos e imagens geradas com a fonte Mandatory para a criação das placas de veículos, a tabela 01 nos mostra o resultado.

A Figura 05 mostra o código fonte do teste que foi feito com o Pytesser.

```
1  from pytesser import *      #Importa o modulo  pytesser
2  im = Image.open('p4.jpg')  # Define qual a imagem será utilizada
3  text = image_to_string(im) # coloca na vareavel o resultado da converção OCR
4  print (text)               # Imprime na tela o resultado.
```

Figura 5 - Código fonte Python Pytesser
Fonte: Autor

ENTRADA	SAIDA	% ACERTOS
	XYD-9587	100%
	ABC-l234	100%
	HOW-5678	87%
	CSC-2DI3	75%
	BCD-345B	87%

Tabela 1 - Resultados da Leitura do Pytesser

Pode-se perceber que quando algum fundo é colocado na placa, a porcentagem de acerto é menor devido o fundo criar algum tipo de ruído no resultado da imagem. Então para tentar resolver o problema foi colocado uma nova biblioteca no código. The Python Imaging Library (PIL) é uma biblioteca de manipulação de imagem com essa biblioteca usamos um filtro que faz a detectação de bordas na imagem e cria um contraste. A figura 06 mostra a nova implementação.

```
1  from pytesser import *  # Importa o modulo  pytesser
2  from PIL import image   # The Python Imaging Library (PIL)
3
4  image = Image.open('p5.jpg') # Define qual a imagem será utilizada
5  mask=image.convert("L") # convertendo a imagem em escala de cinza
6  th=90 # Esse valor ajusta a qualidade do contraste da imagem
7  mask = mask.point(lambda i: i < th and 255) #verificando a bordas da imagem e
8                                              #criando o contraste
9  mask.save('edge.jpg')   # Salva a imagem da primeira saida.
10
11 im = Image.open('edge.jpg')
12 text = image_to_string(im)  # coloca na vareavel o resultado da converção OCR
13 print (text)    # Imprime na tela o resultado.
```

Figura 6 - Codigo Fonte Python Contraste da Imagem
Fonte: Autor

A tabela 02 mostra o resultado desse teste.

ENTRADA	Saída 01	Saida 02	% Acerto
		XYD-9587	100%
		ABC-l234	100%
		HOW-5678	87%
		CSC-2013	100%
		BCD-3456	100%

Tabela 2 - Resultados da Leitura do Pytesser Utilizando Contraste.

Após verificar os resultados satisfatório com o modulo do Pytesser, iniciou-se um experimento de como o Python iria fazer a verificação da saída do Pytesser com uma comparação de strings. Então adicionou-se no código uma variável que nessa pesquisa faz o papel do retorno de uma consulta no banco de dados, ou seja, o conteúdo dessa variável é a placa do veículo autorizado. E no final do código fazemos uma comparação logica entre as variáveis.

A figura 7 demonstra essa alteração do código.

```
1  from pytesser import *
2  from PIL import Image, ImageEnhance, ImageFilter
3
4
5  #SIMULANDO O BANCO DE DADOS
6  placa = "XYD-9587"
7
8  im = Image.open('1.jpg')
9  mask=im.convert("L")
10 th=90
11 mask = mask.point(lambda i: i < th and 255)
12 mask.save('edge.jpg')
13
14 im = Image.open('edge.jpg')
15 text = image_to_string(im) # CONVERTE A IMAGEM EM STRING
16
17 print "PLACA DA LEITURA: ", text
18 print "PLACA DO BANCO: ", placa
19
20 #FAZ A COMPARAÇÃO DAS VARIAVEIS
21 if placa == text:
22     print "Entrada Permitida"
23 else:
24     print "Entrada Proibida"
25
```

Figura 7 – Teste de comparação de igualdades.
Fonte: Autor.

Ao executar o código obtive o seguinte resultado:

```
PLACA DA LEITURA:  XYD-9587


PLACA DO BANCO:  XYD-9587
Entrada Proibida
>>>
```

Figura 8 - Console do Python - resultado da comparação das strings.
Fonte: Autor.

Mesmo que as variáveis aparentemente apresentem o mesmo conteúdo o Python não reconhecia a igualdade. Então adicionamos mais um parâmetro para melhorar o tratamento da saída do Pytesser. Então um novo teste foi realizado, para se ter certeza que nenhuma "sujeira" estava vindo na conversão da imagem para string adicionamos um novo parâmetro no momento do armazenamento na variável: [0:8]. Esse parâmetro faz com que somente os 8 primeiros caracteres sejam armazenados na string. Após essa alteração o resultado da comparação da string foi positivo.

Para efetuar a captura da imagem pela câmera utilizou-se o modulo PyGame utilizada para aplicações multimídia no python. A figura 9 mostra o código fonte utilizado no experimento para se capturar a imagem pela webcam.

```
1    import pygame.camera, sys
2
3    #Define o diretorio onde será salvo a imagem
4    WEBCAM_DIR = "C:\img"
5
6    #parametros de funcionamento da webcam
7    pygame.init()
8    pygame.camera.init()
9    cam = pygame.camera.Camera(0,(640,480),"RGB")
10
11   #iniciando o processo de capitura da imagem
12   cam.start()
13   image = cam.get_image()
14   cam.stop
15
16   #Salvando a imagem
17   filename = "%s/%s.jpg" % (WEBCAM_DIR, 'foto')
18   pygame.image.save(image, filename)
```

Figura 9 - Codigo Fonte Pygame - Capturando imagem pela webcam.

Fonte: Autor

A biblioteca Serial no python, é utilizada para fazer a comunicação entre o python e o arduino. O anexo1 mostra o código fonte utilizado no arduino, a figura 10 mostra o código fonte no python dos comandos utilizada da biblioteca serial.

```
1   import serial
2
3   #DEFINIÇÃO DA PORTA COM UTILIZADA E VELOCIDADE
4   ser = serial.Serial("COM9" , 9600)
5
6   #FAZ A LEITURA DA PORTA SERIAL
7   verifica = ser.readline()
8
9   #ESCREVE NA PORTA SERIAL
10  ser.write (camando)
```

Figura 10 - Biblioteca Serial Comandos para comunicação com o arduino.

Fonte: Autor

O arduino utiliza três parâmetros para se comunicar com a linguagem python. Se o arduino escrever “3” na porta serial o python ira saber que um veículo chegou. Por sua vez o python irá responder ao arduino pela porta serial: “1” para abrir a cancela ou “2” para acender o Led.

3.2. Montagem do Protótipo

Após os testes com as bibliotecas iniciou-se a montagem do protótipo unindo todos os elementos como já apresentado nos tópicos anteriores dessa pesquisa. O computador utilizado para rodar e controlar o protótipo é um HP430 com um processador Intel I3-2310M, 4 GB de memória, 500GB de armazenamento. O ambiente para interpretar o Python instalado foi a versão 2.7.12 do python, baixado diretamente pelo site https://www.python.org/downloads. A bibliotecas utilizada foram: PIL (*Python Imaging Library*) versão 1.1.7, Pygame Versão 1.9.2b1, Pyserial versão2.7, Pytesseract versão 0.1.6.

Com o ambiente do computado instalado o próximo passo foi a montagem do diagrama como apresentado na figura 3, onde se conectou os o sensor e o servo motor ao arduino. O programa utilizado para efetuar o controle do arduino se encontra no Anexo 1 dessa pesquisa.

A comunicação do arduino com o computador é feita através da comunicação serial utilizando uma porta USB, para este experimento foi utilizado a porta COM 9.

O Anexo 2 contém a aplicação python que fica aguardando o comando do arduino para acionar a webcam e fazer o processamento da imagem para verificar se o veículo tem permissão ou não para entrar no estacionamento.

3.3.Teste Prático do Protótipo

Para testar o funcionamento do sistema de reconhecimento de caracteres, foram impressos 8 cartões para que a webcam pudesse capturar a imagem desses cartões simulando uma placa de veículo com combinações utilizando todas as letras do alfabeto e os números de 0 a 9 aleatoriamente.

Mediu-se também o tempo de execução da aplicação, desde o momento que o arduino envia o comando para o computador, captura a imagem, faz o tratamento da imagem, faz a consulta e comparação e retorna a resposta para o arduino.

A tabela 3 mostra a média dos resultados desse teste. O teste foi efetuado 8 vezes.

Entrada	Saída	%Acerto	Tempo (s)
ISV-1246	ISV-1246	87%	1,8
AFH-1597	AFH-L597	87%	1,5
WNK-1091	WNK-\09\	75%	1,7
PZJ-7531	PZJ-753!	87%	1,6
CBE-6033	CBE-6033	98%	1,5

DMU-5788	DMU-5788	98%	1,5
OXL-4268	OXL-4298	97%	1,4
QRT-2934	QRT-2934	96%	1,5

Tabela 3 - Resultado dos testes de leitura OCR pela Webcam.

Todos os erros apresentados nesse teste estão relacionados ao "1", onde o ao converter a imagem para caracteres interpretava o número "1" com a letra I maiúsculo, o L minúsculo, barra ou sinal de exclamação. A figura 11 mostra um exemplo dos cartões utilizado onde se pode ter a perceber a semelhança.

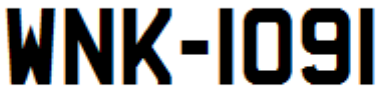

Figura 11 - Fonte Mandatory
Fonte: Autor.

CONSIDERAÇÕES FINAIS

Este trabalho que teve como problema de pesquisa, como desenvolver um sistema de identificação de caracteres alfanuméricos de placas de veículo, utilizando programas opensource e hardware livre, chega ao seu final com as seguintes considerações:

Para o desenvolvimento de um protótipo, buscou-se inicialmente estudos e pesquisas científicas e/ou tecnológicas que têm em seus objetivos o desenvolvimento/implementação e/ou análise de ferramentas tecnologias para a identificação de caracteres alfanuméricos de placas de veículo. A partir dos padrões encontrados nesse processo, foi possível então desenvolver e implementar um protótipo utilizando o arduino e python. Ao utilizar a linguagem Python, sua eficiência e simplicidade foi surpreendente. A documentação das bibliotecas do python e o vasto conteúdo disponível sobre o assunto, ajudaram muito para as escolhas das bibliotecas a usar para alcançar o objetivo dessa pesquisa.

Nessa pesquisa foram abordadas as principais etapas do processamento de imagem envolvido no processo de OCR, desde a captação da imagem a exibição.

Com base nos testes efetuados na aplicação, apresentou uma pequena porcentagem de erro, esses erros aconteceram devido a fonte utilizada e não por ineficiência do sistema OCR, pois esses erros podem ser tratados na aplicação.

Um trabalho futuro e de grande contribuição para essa pesquisa, seria incluir a biblioteca ALPR (*Automatic License Plate Recognition*), incluir uma interface web ao programa além da integração com um banco de dados.

REFERÊNCIAS

BALLARD, D. H.; BROWN, C. M. Computer Vision. 1982.

BANZI, M.; SHILOH, M. Primeiros passos com o Arduino. **São Paulo: Novatec**, p. 17, 2015.

BORGES, L. E. **Python para Desenvolvedores: Aborda Python 3.3.** [s.l.] Novatec Editora, 2014.

BRAICK, P. R. E MOTA, M. B. **História: das cavernas ao terceiro milênio.** São Paulo: Moderna, 2005.

CABRAL, F. A.; MACHADO, V. H. P. Identificador de placa veicular: alternativa para segurança em escolas. **Universidade Tecnológica Federal Do Paraná**, 2014.

GIL, A. C. **Como elaborar projetos de pesquisa.** São Paulo: Atlas, 2002.

GONZALEZ, R.; WOODS, R. Digital image processing. **Prentice Hall**, p. 190, 2002.

GREENEMEIER, L. **Visionary Research: Teaching Computers to See Like a Human, 2008.** Disponível em: <https://www.scientificamerican.com/article/visionary-research/>. Acesso em: 4 abr. 2017.

LEVIN, G. No TitlComputer Vision for Artists and Designers: Pedagogic Tools and Techniques for Novice Programmerse. **Journal of Artificial Intelligence and Society**, v. Vol. 20.4., 2006.

MARQUES, N. **Como funcionam os radares de trânsito no dia do rodízio**. Disponível em: <https://olhardigital.uol.com.br/video/como_funcionam_os_radares_de_transito_no_dia_do_rodizio/18005>. Acesso em: 20 mar. 2017.

MUNIZ, A. L. D. C. Controle de acesso de veículos automotores. 2009.

NARDI, E.; POLAK, L.; VALIATI, L. Sistema De Controle De Acesso Utilizando Reconhecimento De Caracteres Em Placa De Automóvel. 2015.

NASCIMENTO, J. D. DO. Detecção e reconhecimento de placa automotiva com baixo custo. 2012.

PASSELLA, P. H.; SOUZA, G. S. S. DE. Reconhecimento Automático De Placas De Veículos Utilizando Processamento Digital De Imagens E Inteligência Artificial. 2011.

SZELISKI, R. Computer Vision : Algorithms and Applications. **Computer**, v. 5, p. 832, 2010.

WEBER, H. Um protótipo móvel para detecção automática de placas veiculares brasileiras. 2013.

ANEXOS

ANEXO 01

Código Fonte do controle do Arduino

```
1  //Incluindo a bibliotecas
2  #include <Ultrasonic.h>
3  #include <Servo.h>
4
5  //Chamando as classes das bibliotecas
6  Ultrasonic u(6,7);
7  Servo s;
8
9  //Definindo as variaveis
10int can;
11 intcancela;
12 long microsec = 0;
13 constintledVermelho = 12; //Definindo porta 12 para o led
14
15 voidsetup() {
16   //Iniciando a comunicação serial do arduino
17 Serial.begin(9600);
18   s.attach(5);// Definindo a porta 5 para o servo
19   s.write(0); //Colocando o servo na posição 0
20
21   // Inicializando o Led
22   pinMode(ledVermelho,OUTPUT);
23   digitalWrite(ledVermelho,LOW);
24
25 }
26
27 void loop() {
28   microsec = u.timing();
29
30   if (microsec< 200){
31       Serial.print("3");
32       Serial.print("\n");
33       delay(1000);
34
35        while(microsec< 200){
36
37       if (Serial.available() > 0){
38          cancela = Serial.read();
39
40           if (cancela == '1'){
41              for(can = 0; can < 90; can++){
42               s.write(can);
43               delay(15);
44               }
45               delay(2000);
```

```
46              for(can = 90; can >= 0; can--){
47              s.write(can);
48              delay(15);
49              microsec = 300;
50              }
51           }
52           if (cancela == '2'){
53             digitalWrite(ledVermelho,HIGH);
54             delay(2000);
55             digitalWrite(ledVermelho,LOW);
56             microsec = 300;
57           }
58         }
59
60       }
61   }
62     delay(1000);
63 }
```

ANEXO 02

Código Fonte da aplicação Python

```
1  from pytesser import *
2  from PIL import Image, ImageEnhance, ImageFilter
3  import pygame.camera, sys
4  import serial
5
6
7  #PARAMETROS DE CONFIGURACAO DO SISTEMA
8
9  #Define o diretorio onde sera salvo a imagem
10 WEBCAM_DIR = "C:\img"
11
12 #parametros de funcionamento da webcam
13 pygame.init()
14 pygame.camera.init()
15 cam = pygame.camera.Camera(0,(640,480),"RGB")
16
17 #parametros de funcionamento da serial
18 ser = serial.Serial("COM9" , 9600)
19
20
21 whileTrue:
22     #verificando serial do arduino
23     verifica = int(ser.readline())
24
25     if (verifica == 3):
26         print "Chegou Carro:"
27
28         #iniciando o processo de capitura da imagem
29         cam.start()
30         image = cam.get_image()
31         cam.stop
32
33         #Salvando a imagem
34         filename = "%s/%s.jpg" % (WEBCAM_DIR, 'foto')
35         pygame.image.save(image, filename)
36
37         #PEGANDO A IMAGEM SALVA PELA WEBCAM
38         im = Image.open('foto.jpg')
39         #CONVERTENDO A IMAGEM EM BORDAS E CONTRASTE
40         mask=im.convert("L")
41         th=90
42         mask = mask.point(lambda i: i<th and 255)
43         mask.save('edge.jpg')
```

```
44
45      #CONVERTENDO A IMAGEM EM STRING
46      im = Image.open('edge.jpg')
47      text = image_to_string(im)[0:8]
48
49      #SIMULANDO RESULTADO DA CONSULTA NO BANCO DE DADOS
50      placa = "XYD-9587"
51
52      #VERIFICACAO DAS IGUALDADES
53      print "PLACA DA LEITURA: ", text
54      print "PLACA DO BANCO: ", placa
55
56      if placa == text:
57        comando= "1"
58        print "Entrada Permitida"
59        ser.write (comando)
60        print "---------------------------- \n"
61      else:
62         comando = "2"
63         print "PROIBIDA ENTRADA"
64         ser.write (comando)
65         print "---------------------------- \n"
```