UNIVERSIDADE ESTADUAL DO MARANHÃO - UEMA
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO E SISTEMAS - PECS

GESTÃO DO PROJETO URCA (USO RACIONAL DE CARROS) PARA MOBILIDADE URBANA NAS CIDADES INTELIGENTES

AURELIANNY ALMEIDA DA CUNHA

SÃO LUÍS - MA
2022

UNIVERSIDADE ESTADUAL DO MARANHÃO - UEMA
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
MESTRADO PROFISSIONAL EM ENGENHARIA DE COMPUTAÇÃO E SISTEMAS - PECS

AURELIANNY ALMEIDA DA CUNHA

GESTÃO DO PROJETO URCA (USO RACIONAL DE CARROS) PARA MOBILIDADE URBANA NAS CIDADES INTELIGENTES

Trabalho apresentado ao curso de Mestrado Profissional em Engenharia da Computação e Sistemas na Universidade Estadual do Maranhão como pré-requisito para obtenção de título de Mestre sob orientação do Prof. Dr. Carlos Henrique Rodrigues de Oliveira.

SÃO LUÍS - MA
2022

Cunha, Aurelianny Almeida da.
Gestão do Projeto URCA (Uso Racional de Carros) para mobilidade urbana nas cidades inteligentes / Aurelianny Almeida da Cunha. – São Luís, 2022.
114 f

Dissertação (Mestrado) - Programa de Pós-Graduação em Engenharia de Computação e Sistemas, Universidade Estadual do Maranhão, 2022.

Orientador: Prof. Dr. Carlos Henrique Rodrigues de Oliveira.

1.Comunicação de dados. 2.Mobilidade urbana. 3.Cidades inteligentes. I.Título.

CDU: 004.89:656

Elaborado por Giselle Frazão Tavares - CRB 13/665

AURELIANNY ALMEIDA DA CUNHA

GESTÃO DO PROJETO URCA (USO RACIONAL DE CARROS) PARA MOBILIDADE URBANA NAS CIDADES INTELIGENTES

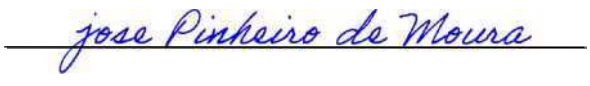
Dissertação apresentada ao Programa de Pós-graduação em Engenharia de Computação e Sistemas da Universidade Estadual do Maranhão como pré-requisito para obtenção do título de Mestra.

Aprovado em: 31/10/2022

BANCA EXAMINADORA

Prof. Dr. Carlos Henrique R. de Oliveira
(Orientador) Universidade Estadual do Maranhão

Prof. Dr. Mauro Sergio Silva Pinto
Universidade Estadual do Maranhão



Prof. Dr. José Pinheiro de Moura
Universidade Estadual do Maranhão

AGRADECIMENTOS

A Deus, por me dar as capacidades para chegar até aqui.

Ao professor Carlos Henrique pelas orientações prestadas e pela confiança depositada em mim para a continuação deste projeto, aos meus companheiros de equipe URCA Ana Paula e Luiz Carlos. Também ao David por todo o auxílio sempre que necessário e a todos que vêm participando do projeto direta ou indiretamente, pelo suporte e apoio oferecido.

Também, de forma especial, à minha mãe, à minha irmã e meu pai, por todo suporte e amor que me oferecem e me fortalece e por todo apoio para seguir em meus projetos.

RESUMO

O projeto URCA (Uso Racional de Carros nas Cidades Inteligentes) nasceu com o objetivo de melhorar a mobilidade urbana e ser uma alternativa às grandes obras de infraestrutura com relação ao excesso de automóveis nas vias. Para tal, ele propõe uma solução de baixo custo cuja aplicação seja o uso racional das vias públicas por automóveis por meio do compartilhamento de caronas solidárias. Para isso a solução visa fornecer um sistema de monitoramento capaz de identificar a quantidade de passageiros dentro dos veículos que trafegam as principais avenidas e as suas respectivas placas, para que tais dados, a partir daí, possam integrar uma base de informações que servirá como estatística para os órgãos regulamentadores de trânsito, a fim de que eles possam propor incentivos fiscais ou até mesmo sanções. Este trabalho trata da Gestão do Projeto URCA que integrou as funcionalidades que foram desenvolvidas desde o início do Projeto e complementar a solução de requisitos do ambiente de implantação, por meio de: a) um protótipo em hardware para testes de campo composto por um mini computador, uma câmera com duas câmeras (RGB e IR), um sistema de comunicação e um sistema de ventilação e encapsulamento mecânico contra pó e umidade e b) integração do software de carona (URCARONA) com o software de análise de dados (*Analytics*), para que seja permitida a efetiva aplicação do Projeto URCA nas principais ruas das cidades, de acordo com o interesse dos órgãos reguladores. A montagem do protótipo e os testes de campo foram realizados com sucesso.

Palavras-chave: URCA; Comunicação de Dados; Mobilidade Urbana, Cidades Inteligentes.

ABSTRACT

The URCA (Rational Use of Cars in Smart Cities) project was born with the objective of improving urban mobility and being an alternative to large infrastructure works in relation to the excess of cars on the roads. To this end, he proposes a low-cost solution whose application is the rational use of public roads by cars through carpooling. For this, the solution aims to provide a monitoring system capable of identifying the number of passengers inside the vehicles that travel the main avenues and their respective plates, so that such data, from there, can integrate an information base that will serve as statistics. to traffic regulatory bodies, so that they can propose tax incentives or even sanctions. This work deals with the URCA Project Management in order to integrate the functionalities that have been developed since the beginning of the Project and complement the solution of requirements of the deployment environment, through: a) a prototype in hardware for composite field tests by a mini computer, a camera with two cameras (RGB and IR), a communication system and a ventilation system and mechanical encapsulation against dust and humidity and b) integration of the ride-sharing software (URCARONA) with the data analysis software (Analytics), in order to allow the effective application of the URCA Project in the main streets of the cities, according to the interest of Organs regulatory bodies. Prototype assembly and field tests were successfully carried out.

Keywords: URCA; Data communication; Urban Mobility, Smart Cities.

LISTA DE FIGURAS

Figura 1: *Smart City* 20

Figura 2: Modelo para um neurônio biológico 22

Figura 3: Representação de um neurônio Artificial 23

Figura 4: RNA *feed-forward* 25

Figura 5: Visão Computacional x Processamento de imagem 27

Figura 6: Proposta inicial do Projeto URCA 31

Figura 7: Câmera P-90 32

Figura 8: Projeto para a prova de conceito 34

Figura 9: *Realsense* modo Infravermelho – Frontal do Carro 35

Figura 10: *Realsense* modo Infravermelho – Lateral do Carro 35

Figura 11: Esquema de Teste com a *Realsense* para captura de pessoas e placas 36

Figura 12: Resultados do algoritmo de capturas para os pontos C (câmera de profundidade) e D (câmera RGB) 38

Figura 13: Tela de acompanhamento do URCA Analytics 39

Figura 14: Esquema de integração das aplicações até URCA *Analytics* 39

Figura 15: Gestão do Projeto URCA e os Trabalhos do Projeto integrados 42

Figura 16: *Intel RealSense D435.* 43

Figura 17: *Raspberry Pi 4 Model B* 45

Figura 18: Caixa de passagem hermética 47

Figura 19: mini PC 48

Figura 20: Especificações do Notebook i7 com Windows 49

Figura 21: Especificações do Macbook i5 com MacOS 49

Figura 22: Vistas das peças 3D do modelo utilizado 52

Figura 23: Caixa e Tampa do modelo para encapsulamento inicial da solução 52

Figura 24: Modelagem 3D do encapsulamento mecânico para o ambiente *outdoor* 53

Figura 25: Encapsulamento mecânico implementado 54

Figura 26: Filtro de ar integrante do encapsulamento 55

Figura 27: Tela de extensões do *VS Code* 56

Figura 28: classe *Observer* com utilitários para diferentes SOs..... 59

Figura 29: Implementação da otimização do processamento com *Observer* 60

Figura 30: Distâncias da câmera para testes com entrada de dados ao vivo pela *Realsense*.... 62

Figura 31: Posição da câmera com angulação a 90º..... 63

Figura 32: Posição da câmera a 83º para captura de placas 63

Figura 33: *Raspberry* e interface gráfica para acesso ao seu sistema..... 66

Figura 34: Teste de campo com entrada de filmagem ao vivo com *Raspberry* 67

Figura 35: Teste de campo com entrada de filmagem ao vivo com Mini PC 68

Figura 36: 1ª tentativa - carro branco a 60 km/h 69

Figura 37: Resultado do processamento na 1ª tentativa para o carro branco 69

Figura 38: 1ª tentativa - carro prata a 60 km/h 70

Figura 39: Resultado do processamento na 1ª tentativa para o carro prata 71

Figura 40: 2ª tentativa - carro branco a 60 km/h 72

Figura 41: 2ª tentativa - carro prata a 60 km/h 73

Figura 42: Resultado do processamento na 2ª tentativa para o carro prata 73

Figura 43: Testes de campo em tempo real com Notebook Windows 74

Figura 44: Imagem extraída para filmagem ao vivo a 20 km/h..... 75

Figura 45: Resultado do processamento com entrada de filmagem ao vivo com Notebook Windows..... 75

Figura 46: Testes de campo com entrada de filmagem ao vivo com *Macbook*..... 76

Figura 47: Processamento em tempo real com *Macbook*..... 77

Figura 48: Resultado do processamento para entrada de filmagem ao vivo com *Macbook* 77

Figura 49: Processamento do Notebook Windows para entrada de vídeo de 20 km/h 79

Figura 50: Processamento do Notebook Windows para entrada de vídeo de 40 km/h 79

Figura 51: Processamento do Notebook Windows para entrada de vídeo de 60 km/h 80

Figura 52: Processamento no *Macbook* para entrada de vídeo de 20 km/h 81

Figura 53: Processamento no *Macbook* para entrada de vídeo de 40km/h 81

Figura 54: Processamento no *Macbook* para entrada de vídeo de 60 km/h 82

Figura 55: Fixação do encapsulamento da solução no local de teste 83

Figura 56: Testando conectividade do equipamento encapsulado 84

Figura 57: Acesso remoto ao equipamento encapsulado em funcionamento 85

Figura 58: Teste de conectividade, acesso remoto e verificação de temperatura do equipamento 85

Figura 59: Teste de conectividade, acesso remoto e verificação de temperatura do equipamento em horário e dia diferentes 86

Figura 60: Tela inicial do aplicativo 87

Figura 61: Tela de Cadastro de usuário com o Modo Motorista 88

Figura 62: Tela Cadastro do carro para Modo motorista 89

Figura 63: Tela de Login com usuário cadastrado 90

Figura 64: Tela Home do aplicativo para Modo Motorista 91

Figura 65: Tela para o Motorista escolher para quem dar carona 92

Figura 66: Tela de Cadastro de usuário com o Modo Carona 93

Figura 67: Tela Cadastro preferências para Modo Carona 94

Figura 68: Tela de Login com usuário cadastrado no Modo Carona 95

Figura 69: Tela Home do aplicativo para Modo Carona 96

Figura 70: Tela para o Caroneiro escolher com quem pegar carona 97

Figura 71: Tela Esqueceu a senha 98

Figura 72: Acesso à Tela do *Analytics* 100

Figura 73: Tela com gráficos do *Analytics* 101

LISTA DE TABELAS

Tabela 1: Especificações da *Intel Realsense D435*. 44

Tabela 2: Especificações do Mini PC 48

Tabela 3: Orçamento dos materiais especificados 58

LISTA DE ABREVIATURAS E SIGLAS

ARM	*Audio Modem Riser*
CI	Cidade Inteligente
CPE	*Customer Premises Equipment*
CPU	*Central Process Unit*
CSI	*Camera Serial Interface*
CSV	*Comma-separated Values*
DSI	*Display Serial Interface*
ERB	Estação Rádio Base
FAQ	*Frequently Asked Questions*
FIRJAN	Federação das Indústrias do Rio de Janeiro
FPS	*Frames per second*
GB	*GigaByte*
HOV	*High Occupancy Vehicle*
IDE	*Integrated Development Environment*
INPI	Instituto Nacional de Propriedade Industrial
IoT	*Internet of Things*
IPVA	Imposto Sobre Propriedades de Veículos Automotores
IR	*Infrared*
LAN	*Local Area Network*
MICROSD	*Micro Secure Digital*
RGB	*Red Green Blue*
RNA	Redes Neurais Artificiais
RNC	Redes Neurais Convolucionais
SMP	*Symmetric Multi-Processing*
SMT	*Simultaneous Multithreading*
SO	Sistema Operacional
UEMA	Universidade Estadual do Maranhão
URCA	Uso Racional de Carros
USB	*Universal Serial Bus*

SUMÁRIO

1. INTRODUÇÃO 14
1.1. Objetivos 15
1.1.1.Objetivo Geral 15
1.1.2.Objetivos Específicos 15
1.2. Metodologia 16
1.3. Justificativa 17
1.4. Estrutura do documento 17
2. ESTADO DA ARTE 18
2.1. Trabalhos Correlatos 18
2.2. Cidades Inteligentes 19
2.3. Redes Neurais Artificiais 22
2.4. Visão Computacional e Processamento de imagens 25
2.5. Processamento Paralelo 28
2.6. O Projeto URCA 31
2.7. O Gerenciamento do Projeto URCA 40
3. MATERIAIS E MÉTODOS 43
3.1. Especificação técnica do material 43
3.1.1 Câmera Intel Realsense 43
3.1.2 Raspberry Pi 4 45
3.1.3 Caixa de passagem hermética 46
3.1.4 Mini PC 47
3.1.5 Notebooks do Projeto 48
3.2. Especificação técnica para a implantação no ambiente *outdoor* 49
3.3 Encapsulamento da solução 51
3.4 Tecnologias utilizadas 55
3.4.1Visual Studio Code 55
3.4.2Python56
3.5 Orçamento dos materiais 58
3.6 Otimização do processo de extração de dados 58
3.7 Parâmetros para a realização dos testes de campo 61
4. RESULTADOS E DISCUSSÃO 64
4.1 Testes de campo com entrada de fluxo de quadros contínuos da câmera *Realsense* 64
4.1.1.Testes com *Raspberry Pi* 4 65
4.1.2.Testes com Mini PC com Linux 67
4.1.3.Testes com Notebook com Windows 74
*4.1.4.*Testes com *Macbook* com *MacOS* 76
4.2 Testes com entrada de vídeo gravado 78
4.2.1.Testes com Notebook com Windows 78
*4.2.2.*Testes com *Macbook* com *MacOS* 80
4.3 Teste do encapsulamento mecânico 82
4.4 Aplicativo URCARONA v1.1 86
4.5 Integração do Aplicativo URCARONA ao *Analytics* do Projeto 99
5. CONCLUSÕES 102
5.1. Dificuldades encontradas 103
5.2. Aprendizados 103
5.3. Trabalhos futuros 103
REFERÊNCIAS 105
APÊNDICE A 108
APÊNDICE B 109
APÊNDICE C 110

1. INTRODUÇÃO

De acordo com o Relatório da Frota Circulante de 2022 elaborado pela SindiPeças [1], em 2021 ocorreu um aumento da frota de automóveis no Brasil de 0,7% ao se comparar com o ano de 2020, registrando 46,6 milhões de unidades circulantes, o que resulta em um automóvel para cada 4,6 habitantes. Como consequência, apesar do crescimento anual não ser tão expressivo quanto nos anos anteriores, ainda há um excesso de automóveis rodando nas ruas e avenidas devido à quantidade de novos automóveis que entram nas vias públicas ser muito maior que a quantidade de automóveis que são retirados de circulação [2].

Tal excesso resulta em vários inconvenientes para a população, atingindo motoristas, pedestres e moradores, que enfrentam: o alto índice de emissão de poluentes no ambiente devido à queima de combustível dos automóveis [3] e o atraso gerado nos deslocamentos – com o consequente aumento de estresse no dia a dia a da população –. Há ainda o impacto negativo na economia do país como um todo, o qual pode ser percebido através de um estudo do instituto FIRJAN com base em dados de 37 regiões metropolitanas, que informa que os custos com o tempo perdido nos deslocamentos casa-trabalho-casa dos cidadãos correspondem a um prejuízo de mais de R$ 111 bilhões [4]. Pode-se também tomar como exemplo a região da Grande São Luís no Maranhão, cujo custo com deslocamentos acima de 30 minutos aumentou 14,1%, comparando-se os anos de 2012 e 2011[5].

Nasceu então o projeto URCA [2] (Uso Racional de Carros nas Cidades Inteligentes), cujo objetivo é se apresentar como uma alternativa às grandes obras de infraestrutura com relação ao excesso de automóveis nas vias [6] ao propor uma solução de baixo custo que visa a melhoria da mobilidade urbana através do uso racional das vias públicas por automóveis [7].

A solução proposta pelo URCA pretende funcionar como um sistema de monitoramento capaz de capturar a quantidade de passageiros no interior do veículo que transita a via monitorada e também de identificar a placa deste automóvel. Desta forma, é possível obter dados que informam se o veículo está trafegando com apenas o motorista ou se há passageiros e, através da placa lida e identificada, vincular esta informação ao veículo em questão [8]. O URCA tem como foco as principais avenidas das cidades que, devido ao intenso tráfego de carros, apresentam maior incidência de congestionamentos e sobre as quais o funcionamento da solução terá um horário predefinido, visando obter, de forma eficiente, dados estatísticos que serão usados pelos órgãos reguladores de trânsito como parâmetros para tomada de decisões. [9]. Desta forma, o URCA apresenta como sugestões a concessão de benefícios pelo órgão

regulador, tais como, por exemplo, desconto no pagamento de impostos ou até mesmo punição com emissão de multa de trânsito para carros com apenas uma pessoa em seu interior [6].

O projeto URCA tem sido desenvolvido através do esforço de trabalho conjunto de sua equipe como visto em [1], [6], [8] e [9]. Através destes trabalhos foram feitas as etapas de reconhecimento e tratamento das imagens capturadas, comunicação de dados e a de armazenamento e tratamento destes dados no Banco de Dados. No entanto, no percurso da execução destas fases, foi identificada a necessidade de uma Gestão do Projeto para habilitar a sua implantação, para a transição de uma prova de conceito a uma solução a ser implantada em um ambiente real envolvendo, assim, as etapas anteriormente destacadas e adicionando soluções menores que foram identificadas como necessárias para que a solução URCA possa ser aplicada em um ambiente mais próximo do real. Esta Gestão faz-se necessária e consiste em integrar as funcionalidades do Projeto já existentes e tratar dos requisitos do ambiente de implantação que devem ser alcançados para que possa ser feita a implementação da solução proposta.

Desta forma, este trabalho apresenta o processo da Gestão do Projeto URCA para a implantação da sua solução proposta, de acordo com os objetivos adicionais que foram identificados como habilitadores para a implantação desejada, envolvendo também os procedimentos de integração das etapas anteriores, de forma que a execução do planejamento proposto nesta Gestão do Projeto permita a efetiva implantação do URCA nas ruas e avenidas.

1.1. Objetivos

1.1.1. Objetivo Geral

Viabilizar a implantação do Projeto URCA na cidade de São Luís como uma solução de baixo custo para melhoria da mobilidade urbana nas cidades inteligentes.

1.1.2. Objetivos Específicos

Para a contemplação do objetivo geral fazem-se necessárias as conclusões dos seguintes objetivos específicos:

- Definir a especificações técnicas adequadas para os equipamentos utilizados para a viabilização da solução nas avenidas;
- Produzir um protótipo em hardware para testes de campo composto por um mini computador, uma câmera com duas câmeras (RGB e IR), um sistema de

comunicação e um sistema de ventilação e encapsulamento mecânico contra pó, poluição e umidade;

- Realizar os testes de campo;
- Integrar o aplicativo URCARONA com o Banco de Dados (*Analytics)* do Projeto;
- Validar a solução junto aos órgãos reguladores de trânsito.

1.2. Metodologia

A metodologia será apresentada baseando-se nos objetivos específicos citados, que constarão em diferentes fases que delimitam subconjuntos de esforços a serem feitos para a obtenção do objetivo geral.

Inicialmente, foi feita a especificação técnica do material necessário para viabilizar a integração entre os equipamentos para que sejam compatíveis entre si e com o projeto e para que possa ser feita a aquisição do material adequado para a implementação. Também foram especificados os parâmetros técnicos habilitadores para que a solução possa ser implantada de modo efetivo nos locais de captura.

Na fase 2, foi feito o encapsulamento mecânico da solução, integrando os equipamentos necessários para implantação nas avenidas, levando em consideração as variáveis de ambiente reais pelas quais deverá passar a solução encapsulada.

Na fase seguinte foram feitos testes de campo com a solução a fim de identificar se o funcionamento desta foi funcional e efetivo de acordo com os requisitos de alta velocidade dos carros e outros que foram especificados, para que, caso seja necessário que se realizem ajustes para corrigir possíveis problemas, as alterações e melhoramentos sejam executados.

Na fase 4, foi feita a continuação da implementação das funcionalidades do aplicativo URCARONA iniciado por [6], com a formulação deste na versão 1.1, o qual foi integrado ao A*nalytics* do projeto URCA, desenvolvido por [9], para que a solução URCA proposta esteja integrada e possa processar os dados e as respostas possam ser disponibilizadas através do aplicativo URCARONA.

Assim, após os passos anteriores, será feita a validação da solução e a procura pelos Órgãos reguladores de trânsito para que a solução possa ser testada nas avenidas como aplicação comercial.

1.3. Justificativa

Como frisado por [9] em seu trabalho, foi proposta uma solução que possa ser viável para resolver o problema de mobilidade urbana e ainda contrapor a alternativa de novas obras de infraestrutura, uma vez que estas soluções são de alta carga financeira. Além disso, o Projeto URCA se preocupa com a redução da emissão de poluentes, uma vez que o excesso de carros nas vias acarreta uma grande quantidade de propagação de gases como, monóxido de carbono, hidrocarbonetos, dióxido de carbono, dentre outros, que consequente afetam a qualidade de vida dos habitantes das regiões urbanas, acarretando problemas de saúde [3]. Ainda pode-se citar o impacto financeiro positivo da solução, tomando como base o fato de que os congestionamentos geram prejuízos financeiros na economia de país [4], uma vez que a frota de carros nas ruas será reduzida e incentivos fiscais (como, por exemplo, abatimento no valor do IPVA) deverão ser propostos aos motoristas que aderirem à ideia do Projeto.

1.4. Estrutura do documento

Este Trabalho está estruturado em 5 seções da seguinte forma: na seção 2 é apresentado o Estado da arte, onde serão introduzidos os conceitos e informações que servirão como base teórica para o prosseguimento do Relatório. Na seção 3 são apresentados os Materiais especificados e utilizados na realização do Trabalho, bem como os Métodos e processos que descrevem como estes materiais foram utilizados no decorrer da Gestão do Projeto URCA para alcançar os resultados. Na seção 4 são apresentados os Resultados e discussões. Por fim, na seção 5 serão feitas as Conclusões obtidas com todo o esforço deste Trabalho, bem como o que pôde ser aprendido e os Trabalhos futuros. Nos Apêndices estão os arquivos que mostram as produções científicas alcançadas durante no processo de desenvolvimento evolutivo do Projeto URCA, bem como o código produzido como fruto de parte deste Trabalho.

2. ESTADO DA ARTE

Nesta seção serão abordados os aspectos teóricos que oferecerão o embasamento necessário para nortear a compreensão de todo o processo da Gestão do Projeto URCA, tendo como fonte principal de referências Pesquisas cujo objetivo de trabalho envolve um ou mais aspectos teóricos utilizados para o desenvolvimento desta Pesquisa. As subseções tratam desde os principais conceitos teóricos utilizados nos algoritmos para captura e extração de placas e de quantidade de pessoas, até a explicação do próprio percurso tomado no Projeto URCA com o *roadmap* percorrido até aqui com os trabalhos anteriores [6] e [9], como forma de fundamentar e melhor situar como será feita a Gestão do Projeto nas seções seguintes.

2.1. Trabalhos Correlatos

Serão apresentados sucintamente alguns Trabalhos cujas propostas podem ser consideradas como relacionadas às deste presente Trabalho. Tal relacionamento foi feito a partir da identificação de um ou de alguns objetivos de cada um destes projetos, de forma a poder relacioná-los a partir de seus objetivos em comum com o do Projeto URCA.

As propostas comerciais mais conhecidas são os serviços oferecidos através de aplicativos desenvolvidos pela companhia *Uber* e também pela *Waze* (na modalidade *Waze Carpool*). As duas têm como fim a melhoria da mobilidade urbana, mas com diferentes propostas: a *Uber* visa "oferecer aos usuários uma melhor mobilidade"[1] em termos de facilitar o transporte de pessoas e de entregas mediante o pagamento por estes serviços. Já o *Waze Carpool* deseja "ajudar as cidades a lidar com problemas de mobilidade, como congestionamento, sustentabilidade e custo"[2], onde o usuário de seus serviços utiliza o aplicativo para pegar caronas compartilhadas, também pagando por isso, mas com o custo da viagem repartido entre motorista e demais passageiros que peguem a mesma carona através do aplicativo.

Também há Pesquisas como a apresentada em [10] por pesquisadores dos Estados Unidos, que propõe Testes de Sistemas de detecção nas vias para veículos de alta ocupação (HOV – *High Occupancy Vehicle*). As vias para *HOV* são faixas exclusivas para veículos com

---

[1] UBER TECHNOLOGY INC. **Quem somos**. Disponível em: <https://www.uber.com/br/pt-br/about/>. Acessado em 13 de Outubro de 2022.

[2] QUACH, K. ***Google says there's no Waze forward, carpool app axed***. Disponível em: <https://www.theregister.com/2022/08/25/google_waze_carpooling_closes/>. Acessado em 13 de Outubro de 2022.

mais de dois passageiros além do motorista, como vans, transporte coletivo ou mesmo carros partilhados, que por aí transitam, e a Pesquisa propõe uma forma de fiscalização para verificar se de fato, a ocupação dos veículos está conforme a quantidade de ocupantes habilitadora para o uso de tais vias. A fiscalização é proposta por meio da aplicação de tecnologias de detecção de ocupação de faixas das avenidas, para auxiliar o departamento de trânsito local no gerenciamento dos corredores de *HOV* do estado do Tennessee (EUA) e utilizou inicialmente uma tecnologia de infravermelho (IR) da *Xerox* como ferramenta de detecção. Assim, o sistema proposto por [10] foi um conjunto integrado de algoritmos de visão computacional e *hardwares* como câmeras, que faz a análise de vídeo para identificar o número de ocupantes em um veículo. Esta proposta tem como objetivo promover caronas e reduzir congestionamentos [10] ao fiscalizar os carros e incentivar que estes utilizem a faixa dedicada ao partilhar corridas.

No Brasil, uma outra iniciativa acadêmica voltada para mobilidade urbana foi desenvolvida por um grupo de alunos da Universidade Federal do Rio de Janeiro (UFRJ), chamada *Caronaê* [11]. O projeto tem como propósito conectar os membros da comunidade acadêmica por meio de um aplicativo e pontos de carona, possibilitando melhor ocupação dos veículos privados.

Por fim, há sistemas particulares de detecção de placas de carro que usam a tecnologia ALPR (do inglês – *Automated License Plate Recognition* – sistema de detecção automático de placas) que têm como objetivo o reconhecimento de placas, mas voltados para a comercialização de seus sistemas que podem ser aplicados, por exemplo, em estacionamentos ou outros locais como sistema que multe os carros através de suas placas.

Por fim, tendo em vista alguns dos principais fins das propostas acima e considerando a proposta de trabalho do Projeto URCA como um todo, o objetivo em comum que torna as propostas correlatas a este Trabalho é o objetivo da melhoria da mobilidade urbana. O Projeto URCA propõe uma solução que integra diferentes aplicações de *hardware* e *software* para ajudar a melhorar a mobilidade urbana através da promoção da diminuição da quantidade de carros circulantes nas principais vias de movimentação automóveis, para que a solução possa ser aplicada por órgãos governamentais.

2.2. Cidades Inteligentes

As Cidades Inteligentes (CI), também chamadas de *Smart Cities*, são consideradas como aquelas que utilizam a tecnologia de modo estratégico para melhorar sua infraestrutura, otimizar a mobilidade urbana, criar soluções sustentáveis e outras melhorias necessárias para a qualidade

de vida dos seus moradores [12]. Outra definição para *Smart Cities*, conforme feita pela União Europeia, é de um sistema de pessoas interagindo e usando energia, materiais, serviços e financiamento para catalisar o desenvolvimento econômico e a melhoria da qualidade de vida [13]. Na Figura 1, é ilustrado o conjunto de áreas e serviços de uma cidade, que, interagindo e sendo integradas através da tecnologia, fazem uma CI:

Figura 1: *Smart City*

Fonte: [12]

Explorando a sua terminologia, pode-se entender o conceito e o contexto das Cidades Inteligentes [14]: a este termo, são associadas algumas referências, entre as quais se destacam as de cidade sustentável e de cidade digital. Até a década de 1990, “cidades digitais” era o termo mais utilizado, devido ao termo “digital” ser então relacionado ao acesso a computadores e também à implantação da internet no espaço urbano. Já o termo “inteligente”, atualmente, se refere a processos computacionais imersivos em diferentes contextos, lidando com um grande volume de dados (*big data*), redes em nuvem e comunicação independente entre vários dispositivos conectados, através da Internet das Coisas (do inglês *Internet of Things - IoT*). Desta forma, em uma cidade inteligente há a produção, consumo e distribuição de um grande volume de informação em tempo real entre as estruturas que integram esta cidade.

Mas por que se deve pensar e planejar cidades inteligentes? De acordo com o estudo *The World Population Prospects: The 2017 Revision* [15], a população mundial chegará a 8,6 bilhões em 2030 e esse e outros índices apresentados neste estudo mostram que é essencial que cada vez mais as políticas e os planejamentos sejam orientados para o alcance dos novos objetivos que contemplem a melhoria da qualidade de vida, de serviços públicos e da

sustentabilidade dessa população. Desta forma, as cidades inteligentes apresentam-se como parte de uma solução atual ou integrante do planejamento estratégico para o alcance de tais objetivos, devido ao nível de qualidade de vida que elas se propõem a oferecer. Vale lembrar que, para isto, as CIs devem ser capazes de aliar a aplicação de tecnologias úteis ao planejamento urbano, habitação social, energia, mobilidade urbana, coleta de lixo, controle da poluição do ar, entre outros aspectos urbanos, para manter um bom nível qualitativo de vida e de suporte aos que moram e os que ainda virão a morar nelas.

Como exemplo de referência em Cidades Inteligentes, temos a cidade de *Songdo* na Coreia do Sul [12]. Ela foi planejada pensando totalmente na tecnologia e sustentabilidade: seus edifícios são conectados a sistemas que possibilitam o monitoramento da energia e alarmes de incêndio, reduzindo o custo com manutenção e otimizando o uso. Outro exemplo de inovação da *smart city* é o pneumático, um sistema localizado em todos os apartamentos, onde os resíduos jogados ali vão direto para a central de coleta de lixo.

No Brasil, há alguns exemplos de cidades inteligentes dentre algumas que vão evoluindo aos poucos, como: Curitiba, que inovou com a criação de uma frota de carros elétricos que prestam serviços públicos e que, desde a sua implantação, em 2014, a cidade poupou a emissão de 12264 quilogramas de gás carbônico na atmosfera [12]. Há também a cidade de Salvador [13], que investe em tecnologias com semáforos inteligentes, sensores de encostas, pluviômetros automáticos, estações de monitoramento de qualidade do ar e redes elétricas inteligentes (*smart grids),* tendo como retorno a redução de consumo de energia e rapidez nos serviços de atendimento relacionados a este setor.

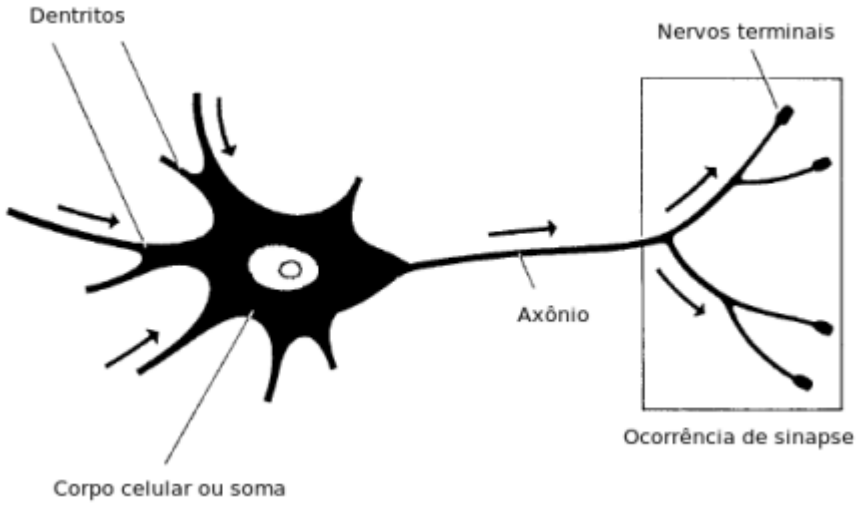
Já com relação à desejada mobilidade urbana, para a qual o Projeto URCA visa ser uma forma de solução, pode-se encontrá-la fazendo parte do conjunto dos indicadores de cidades inteligente utilizados pelo *IESE Cities in Motion Index*, um estudo publicado anualmente pela escola de negócios da Universidade de Navarra, que visa avaliar o desenvolvimento das cidades do mundo [13]. Este indicador urbano faz parte do conjunto de 9 índices utilizados para indicar o nível de inteligência de uma cidade, entre os quais estão: capital humano, coesão social, economia, governança, meio ambiente, mobilidade e transporte, planejamento urbano, alcance internacional e tecnologia.

2.3. Redes Neurais Artificiais

De acordo com [16] uma Rede Neural Artificial (RNA) é “um processador maciçamente paralelamente distribuído constituindo de unidades de processamento simples, que têm a propensão natural de armazenar conhecimento experimental e torná-lo disponível para o uso”. Os estudos e trabalhos em RNAs foram motivados pela identificação de que a “simulação” do funcionamento do cérebro humano pudesse habilitar a realização de novas tarefas computacionais as quais não podem ser feitas por um computador digital convencional como aproximação de funções complexas e reconhecimento de padrões. De forma geral, uma RNA se assemelha ao cérebro em dois aspectos: o conhecimento, que é adquirido pela rede a partir de seu ambiente através do processo de aprendizagem e os pesos sinápticos – ou forças de conexão entre neurônios –, que são utilizados para armazenar o conhecimento adquirido. Tais processos serão explicados a seguir.

Com relação ao cérebro humano, o neurônio é a sua unidade básica [17] Ele é uma célula especializada na transmissão de informações, pois nelas estão introduzidas propriedades de excitabilidade e condução de mensagens nervosas. Ele é constituído por três partes principais: a soma ou corpo celular, do qual emanam algumas ramificações denominadas de dendritos, e por fim outra ramificação descendente da soma, porém mais extensa chamada de axônio. Nas extremidades dos axônios estão os nervos terminais, que pelos quais é realizada a transmissão das informações para outros neurônios e tal transmissão é conhecida como sinapse. Uma ilustração de um neurônio biológico pode ser vista na Figura 2, onde suas principais partes citadas são apontadas:

Figura 2: Modelo para um neurônio biológico

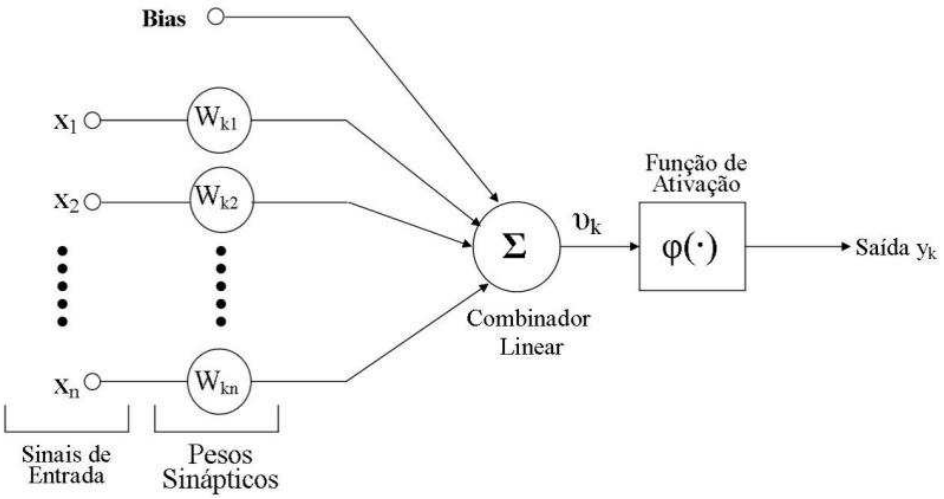


Fonte: [17]

O funcionamento de um neurônio biológico ocorre da seguinte forma [18]: o estímulo é recebido por canais localizados nas sinapses, permitindo que os íons fluam para dentro e para fora do neurônio. Um potencial de membrana aparece como resultado da integração das entradas neurais que, então, determinará se um dado neurônio produzirá um pico (ação potencial) ou não. Esse pico faz com que os neurotransmissores sejam liberados no final do axônio, formando sinapses com os dendritos de outros neurônios. O potencial de ação só ocorre quando o potencial de membrana está acima de um nível de limiar crítico. Entradas diferentes podem fornecer diferentes quantidades de ativação, dependendo de quanto neurotransmissor é liberado pelo emissor e quantos canais no neurônio pós-sináptico são abertos. Desta forma, a eficiência de uma sinapse, representada por um peso de conexão ou força associada, corresponde à informação armazenada no neurônio, e por consequência na rede.

Assim também vale para as Redes Neurais, as quais, baseadas no cérebro humano, têm seu conhecimento adquirido do ambiente por um processo conhecido como aprendizado. Este processo pode então ser entendido, a partir da explicação anterior para o funcionamento do cérebro humano, como o responsável por adaptar as forças de conexão entre os neurônios. Assim, o aprendizado se resume em encontrar as forças de conexões apropriadas para produzir padrões de ativação satisfatórios sob certas circunstâncias [18]. A Figura 3 mostra o modelo proposto para um neurônio artificial:

Figura 3: Representação de um neurônio Artificial

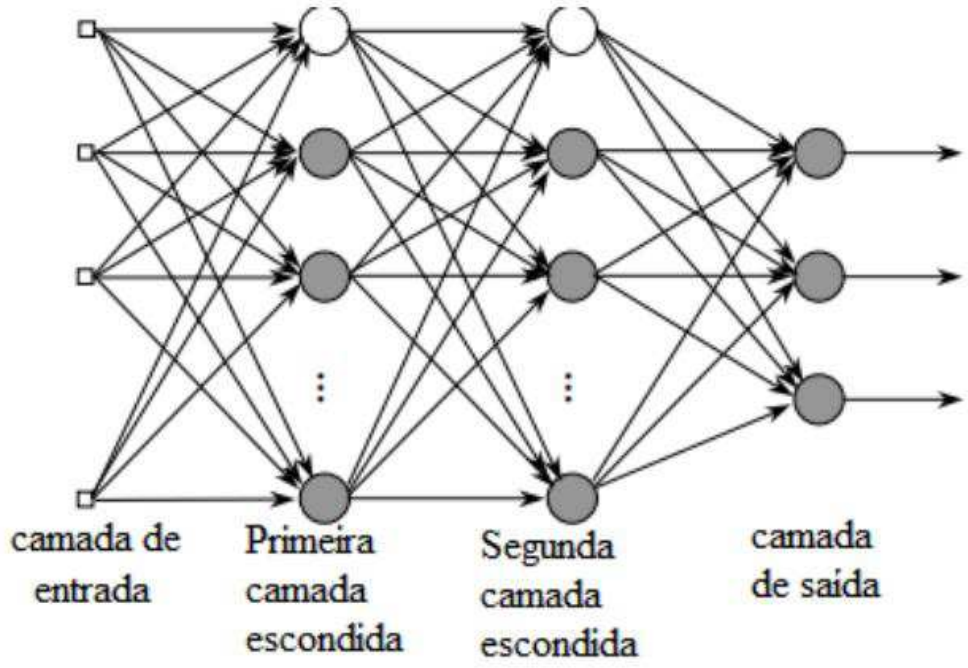


Fonte: [17]

Neste modelo, conforme [17], os impulsos elétricos provenientes de outros neurônios são representados pelos chamados sinais de entrada, (x_1, x_2, ..., x_n), dentre os vários estímulos recebidos onde alguns excitarão mais e outros menos cada neurônio receptor. Essa medida de quão excitatório é o estímulo recebido é representada através dos pesos sinápticos os quais tiverem maior valor do peso, mais excitatório será o estímulo. Os pesos sinápticos são representados por w_{kn}, onde *k* representa o índice do neurônio em questão e *n* se refere ao terminal de entrada da sinapse ao qual o peso sináptico se refere. A soma é representada por uma composição de dois módulos onde o primeiro é uma junção aditiva, somatório dos estímulos (sinais de entrada) multiplicado pelo seu fator excitatório (pesos sinápticos), e posteriormente uma função de ativação, que definirá com base nas entradas e pesos sinápticos, qual será a saída do neurônio. O axônio é aqui representado pela saída (y_k) obtida pela aplicação da função de ativação.

As funções de ativação são representadas pela letra grega *phi* φ(v) e podem ser divididas em três tipos: linear (1), usualmente utilizada como uma saída binária; linear por partes (2), utilizada como amplificador de saída porque uma de suas saídas é a própria entrada e Sigmoide (3), amplamente utilizada na construção de redes neurais [16]. A saída do neurônio é a resposta da rede neural, porém uma rede neural geralmente é constituída de muitos neurônios, dispostos em diferentes camadas, conforme Figura 4, indicando que a saída de um neurônio pode ser utilizada como entrada para um neurônio numa camada posterior.

Figura 4: RNA *feed-forward*



Fonte: [16]

A Figura 4 identifica uma topologia comum de RNAs [17]: as redes *feed forwards* (redes alimentadas adiantes). Algumas características importantes desta topologia são: os neurônios são arranjados em camadas – conjuntos de neurônios distintos e ordenados sequencialmente –; a camada inicial recebe os sinais de entrada e a camada final obtém as saídas. As camadas intermediárias são chamadas de camadas ocultas; cada neurônio de uma camada é conectado com todos os neurônios da camada seguinte; não pode haver conexões entre neurônios de uma mesma camada. A informação (alimento) provém da camada de entrada ou neurônios (nós) fonte e posteriormente é transmitida para as camadas seguintes até a camada de saída.

2.4. Visão Computacional e Processamento de imagens

A Visão Computacional procura modelar e replicar a visão humana usando software e hardware, tentado superar as limitações do processamento de imagens comum quando comparado com o complexo processamento da visão humana. Tentando simular a visão natural, a Visão Computacional tem seu embasamento científico decorrente de estudos de algoritmos que buscam analisar e compreender conceitos da visão humana nas áreas de biologia, óptica e matemática (álgebra linear, geometria e estatística) [19].

Algumas etapas da Visão Computacional podem ser consideradas, sendo, de modo geral as seguintes [18]:

- Aquisição de imagem, que compreende a obtenção de um sinal do mundo real com técnicas para gerar uma representação computacional do sinal de entrada;

- Pré-processamento, que pode ser necessário para deixar a representação obtida anteriormente em condições adequadas à etapa de processamento propriamente dita, para ajustá-las de modo a evitar erros posteriores;

- Extração de características que pode ser feita a partir de expressas matemáticas, em níveis de complexidade elevados;

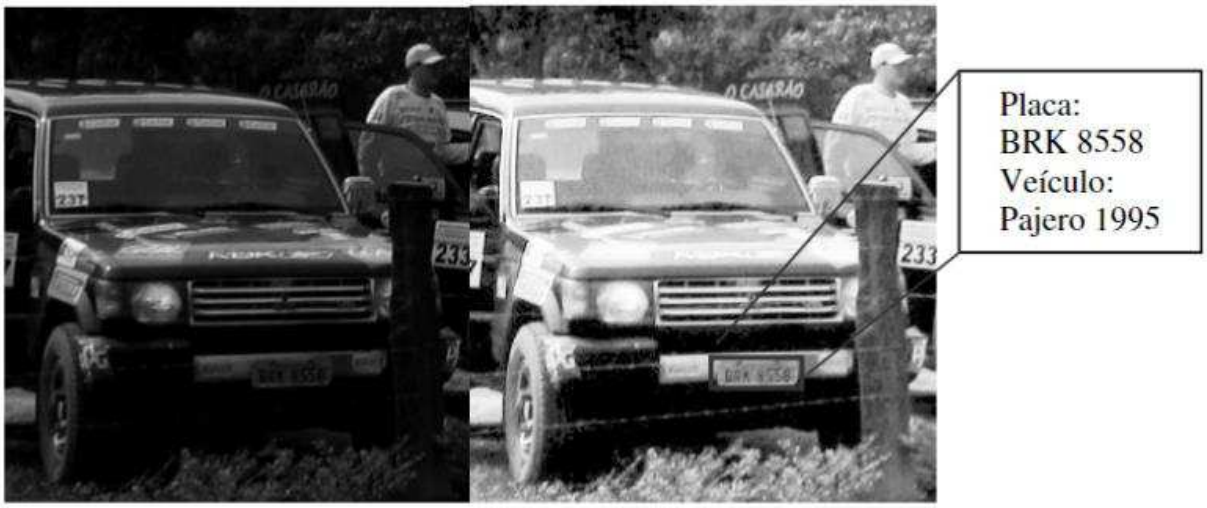
- Detecção e segmentação: consiste em discriminar regiões de interesse e separá-las para posterior análise;

-Processamento de alto nível: consiste em verificar a qualidade dos dados, fazer estimativas sobre a imagem, bem como classificar objetos detectados em diferentes categorias.

As aplicações da área de visão computacional são diversas e as mais conhecidas compreendem: controle de processos em robôs industriais e veículos autônomos, modelagem de objetos ou de ambientes, entre outras.

Vale ressaltar que pode ser feita a distinção entre Visão Computacional e Processamento de imagens. De acordo com [20], embora não seja clara a fronteira entre o processamento de imagens e visão computacional, pode-se dizer que o processamento de imagens é um processo em que a entrada do sistema é uma imagem e a saída é um conjunto de valores numéricos como saída (como objetos complexos, vetores e matrizes), que podem ou não compor uma outra imagem. Já a Visão Computacional procura emular a visão humana, tendo também como entrada uma imagem, mas apresentando como saída uma interpretação da imagem como um todo, ou parcialmente. De modo resumido, os processos de Visão Computacional geralmente iniciam com o processamento de imagens. A Figura 5 identifica as diferenças entre Processamento de imagens e Visão Computacional:

Figura 5: Visão Computacional x Processamento de imagem



Fonte: [20]

A comparação entre os processos vistos na Figura 6 pode ser entendida da seguinte forma, também conforme [20]: a imagem da esquerda mostra o veículo, porém, não é possível ler a placa do veículo, pois ela está muito escura. Na imagem da direita foi feita uma operação de equalização para ajuste dos valores de intensidade da imagem de forma a melhorar o seu contraste, o que é tipicamente uma operação de processamento de imagem, resultando em uma imagem mais clara e que permite a leitura da placa do veículo. Já uma operação de visão computacional é a aplicação de um operador que extrai a placa do veículo e identifica as letras e números da placa, possibilitando que os dados do veículo sejam encontrados em um banco de dados.

No Projeto URCA, é utilizado um exemplo de aplicação de Visão Computacional que permite o reconhecimento de caracteres de placas de carro, que é a técnica de Reconhecimento Óptico de Caracteres (do inglês OCR, *Optical Character Recognition*). O OCR é um método de reconhecimento de padrões em imagens que permite a extração de caracteres alfanuméricos, além de caracteres especiais como pontos de exclamação e vírgulas [21]. De maneira geral, os passos para reconhecimento de placas são: análise e leitura da foto, imagem ou documento escaneado; comparação dos caracteres daquela imagem com as fontes já existentes e categorizadas em seu banco de dados; reconhecimento das especificações de cada caractere incluído na imagem processada; conversão de todos os caracteres em um texto, aplicando em suas finalidades.

Considerando o funcionamento generalizado do OCR apresentado anteriormente, o processo de reconhecimento de caracteres em imagens utiliza de modo geral três etapas da Visão Computacional: o pré-processamento, a detecção e segmentação e o processamento de

alto nível com a determinação dos caracteres que têm as maiores chances de ser correspondente ao caractere extraído [22].

No Projeto, utilizou-se a biblioteca OpenALPR que faz uso da técnica de OCR, na qual a entrada deve ser a imagem que contenha a placa do carro trafegando sob a região monitorada pela solução e a saída é são os caracteres da placa correspondente para ser utilizada nas demais aplicações dentro do URCA. OpenALPR (*Automatic License Plate Recognition*) é uma biblioteca de Reconhecimento Automático de Placas de Carro de código aberto escrita em C++ com *bindings* (vínculo pelo qual diferentes linguagens acessam a tecnologia com suas próprias sintaxes) nas linguagens de programação C#, Java, Node.js, Go e Python. A biblioteca é utilizada para identificar placas a partir da análise de imagens e fluxos de vídeo e apresenta como saída a representação de texto de quaisquer caracteres da placa de licença [23].

De acordo com [24], o OpenALPR é um serviço em nuvem com uma taxa de acerto muito alta, sendo inclusive atualmente empregado para sistemas de aplicação de multas não só no Brasil como também Canadá, EUA, Europa e Oceania. Embora ele também seja baseado no *Tesseract* e OpenCV, há muitos refinamentos para garantir a precisão do serviço.

2.5. Processamento Paralelo

Para reduzir o tempo de processamento geral de uma aplicação, é possível chamar diversos serviços simultaneamente, em vez de processar cada serviço em série. As Arquiteturas Paralelas foram apresentadas como formas de obter uma maior capacidade de processamento para execução de tarefas com altas cargas de processamento. Como o aumento da velocidade dos processadores esbarrava no custo e no limite da capacidade tecnológica para obtenção de circuitos rápidos, a solução tendeu para o emprego de vários processadores trabalhando em conjunto na obtenção de uma maior capacidade de processamento. É quando surge o termo Processamento Paralelo, para designar diferentes técnicas que atendem a essas necessidades. [25]

As configurações para processamento paralelo permitem a execução das tarefas em menor tempo, através da execução de diversas tarefas paralelamente em vez de executar uma a uma de forma sequencial. Existem diferentes configurações destas arquiteturas, sendo elaboradas para aplicações específicas. Desta forma, diante dessa diversidade de conceitos que envolvem e utilizam processamento paralelo, pode-se considerar diferentes níveis de paralelismo, podendo ser enquadrados por:

- Nível de Instrução, que ocorre quando as instruções podem ser reordenadas e combinadas em grupos que são executadas em paralelo, sem mudar o resultado final do programa. Este nível de paralelismo ocorre em arquiteturas pipeline, superescalares e o paralelismo entre instruções;

- Nível de Thread ou de Tarefas, que é a característica de um programa paralelo em que diferentes cálculos são executados no mesmo ou em diferentes conjuntos de dados. O paralelismo em tarefas ocorre em arquiteturas SMT (*Simultaneous MultiThreading*);

- Nível de Processo, no qual as arquiteturas de computador deste nível introduzem o paralelismo com soluções de processamento em múltiplas CPUs (*Central Process Unit*), o que pode ocorrer na forma de multicomputadores – com múltiplos computadores interligados por uma rede – ou na forma de multiprocessadores, podendo ser multiprocessadores simétricos (SMP - *Symmetric Multi-Processing*) – em uma única máquina – ou com vários processadores agregados em diferentes máquinas – na forma de clusters.

Os 3 níveis identificam termos que podem ser assim simplificados: Instrução refere-se a um estágio básico cuja soma destes estágios/instruções integra os comandos para realizar uma funcionalidade, enquanto o termo Processo refere-se a uma função cuja chamada é feita pelo sistema operacional para execução de um programa. Já uma Thread compõe uma parte autônoma de código, criada dentro de um mesmo Processo. [25]

Destacando-se a forma de processamento paralelo mais comum que temos utilizado em nossos computadores nos últimos anos, temos os multiprocessadores simétricos com memória compartilhada, com a arquitetura do tipo *multi-core* (vários núcleos), na qual várias unidades de processamento (processadores), chamadas de núcleo, são integradas em uma única pastilha (chip), com apenas uma interface para gerenciar todo o sistema. Todos os núcleos acessam fisicamente uma única memória, que é comum a todos, e tal fato permite que seus processadores tenham menor custo de sincronização, facilitando o trabalho de divisão de tarefas de forma paralela. Assim, ao colocar vários núcleos físicos de processamento dentro de um único chip, o processador não tem sua velocidade de processamento elevada, mas permite executar mais tarefas simultâneas em um determinado período, o que quer dizer que ao invés de executar uma única tarefa com velocidades cada vez maiores, os processadores de arquitetura *multi-core* permitem que múltiplas tarefas sejam executadas em diferentes núcleos do mesmo chip. [26]

Para exemplificar este modelo, existem os processadores de fabricantes como Intel e NVidia, dentre os quais pode-se exemplificar com o processador presente no MiniPC utilizado pelo Projeto URCA: sua descrição é de um processador Core i9 de 10ª geração com 8 núcleos e 16 threads. De forma simplificada, ele é um processador com um único chip que encapsula 8

processadores (ou cores). Como vantagem disto, tem-se que quanto mais núcleos, mais tarefas paralelas poderão ser executadas.

Dentro do processo de Gestão do Projeto URCA, foi feita a exploração da potencialidade de processamento paralelo oferecida pelo processador do MiniPC para otimizar a solução especialmente através de Threads e da quantidade de diferentes cores. Isto deveu-se pelo fato de o processamento dos dados de entrada na forma de quadros de vídeos ser bastante exigente e ao mesmo tempo necessário ser processado com eficiência, porque assim poderia permitir que fossem feitas as extrações de informações de forma a cumprir os requisitos propostos para garantir o gerenciamento da solução do Projeto.

Assim, de forma a melhor compreender como foi feita a otimização aplicada no código integrante da solução, também serão abordados alguns pontos conceituais sobre Threads: conforme citado anteriormente, um processo pode ser definido como um programa que é chamado pelo sistema operacional e está em seu momento de execução. Quando temos um programa que passa a ser executado, ele se torna um processo e, assim, possui o seu fluxo de execução. Por sua vez, relacionando tais conceitos, uma *Thread* pode ser considerada como um fluxo de execução. [27]

Dentre as categorias de sistemas que executam programas de forma paralela, está a de *Multi-Thread*, e seu objetivo como sistema é gerenciar múltiplas tarefas independentes, porque geralmente ele contém mais processos para executarem suas *Threads* do que processadores/núcleos disponíveis. [27]

Também há a tecnologia chamada *Hyper-Threading*, que foi desenvolvida pela Intel e permite que aplicações que fazem uso de *Multi-Threads* executem suas *Threads* de forma paralela, pois tal tecnologia faz com que um único processador físico seja visto por uma aplicação como dois processadores lógicos, sendo que estes processadores compartilham um único conjunto de recursos físicos destinados à execução dos mesmos. Do ponto de vista de um software, isto significa que sistemas operacionais e programas de usuários podem atribuir processos ou *Threads* para os processadores lógicos como se estes fossem processadores físicos convencionais. [27]

Nos processadores Intel com *Hyper-Threading*, normalmente há *n* núcleos e *2n threads*, como pode ser visto no processador do MiniPC: com 8 núcleos e 16 *threads*. Isso significa que o chip possui 8 processadores físicos independentes na matriz e, em conjunto, eles possibilitam que até 16 processos ativos (*Threads*) sejam executados, sem causar filas de execução no sistema operacional. [26]. A ressalva para o aproveitamento de todos esses recursos de forma

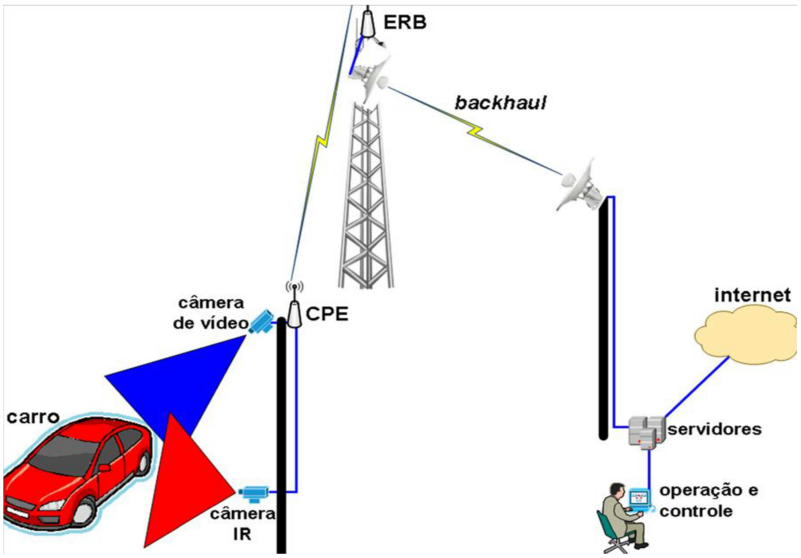
paralela é que não pode haver dependência entre as *Threads*, isto é, um trecho em execução de um processo não pode depender do resultado de cálculo de outro.

Por fim, o objetivo geral do uso de *Threads* é tornar conveniente escrever programas que executem várias tarefas ao mesmo tempo e fazer isto com eficiência. O efeito prático da implementação de mais núcleos e do sistema M*ulti-Threading* nos processadores foi a maior estabilidade dos processos computacionais que, juntamente com o recurso de *Hyper-Threading*, permitiram que vários aplicativos possam ser utilizados simultaneamente, mesmo quando outra aplicação já esteja em execução no sistema, aumentando a produtividade e eficiência na execução de diferentes programas ao mesmo tempo. [26]

2.6. O Projeto URCA

Conforme [6], as pesquisas para desenvolvimento do projeto URCA tiveram início no segundo semestre de 2015. Após as verificações da disponibilidade técnica dos equipamentos necessários para as simulações que seriam realizadas no primeiro ano, idealizou-se a seguinte estratégia para este primeiro momento: uma câmera de vídeo filmaria as avenidas e seria a responsável por capturar a placa do carro e uma segunda câmera de infravermelho seria a responsável pela captura da quantidade de passageiros dentro do veículo. As câmeras ficariam ligadas a um CPE (*Customer Premises Equipment*) – equipamento instalado do lado do cliente – e este ficaria responsável por enviar as informações coletadas a uma ERB – Estação Rádio Base – que enviaria as informações, por meio de um *backhaul*, aos equipamentos servidores do centro de operações e controle, onde as informações seriam tratadas e posteriormente enviadas pela Internet aos órgãos reguladores. A disposição dos elementos do esquema proposto é mostrada na Figura 6.

Figura 6: Proposta inicial do Projeto URCA



Fonte: [6]

A partir daí, foram executadas as etapas necessárias para se realizar a prova de conceito para a proposta inicial, conforme [6]. Assim, foram realizados estudos da viabilidade técnica e verificação da disponibilidade comercial do sensor PIR (*Passive Infrared Sensor*) ou câmera infravermelho e os projetos da rede piloto de radiocomunicação e da rede de dados, além dos algoritmos da contagem de passageiros e de identificação das placas dos carros.

Para a prova de conceito, ainda foram realizadas as etapas do projeto de como se daria a comunicação de dados, o envio das informações extraídas e da sua hospedagem em um banco de dados para tratamento destas informações no centro de gerenciamento da rede.

Houve também a procura pela existência de alguma opção comercial já disponível no mercado que se assemelhasse à proposta do URCA [6] e encontrou-se a câmera "*P-90 for Highway applications*" desenvolvida pela *VehicleOccupancy – Detection Corporation*, mas exigindo um alto custo de investimento de US$ 100 000.00 por câmera. Sendo assim, de acordo com [6], buscou-se alguma maneira de realizar o Projeto URCA para oferecer uma solução com o diferencial de ser de baixo custo – comparada com as opções comerciais como a P-90 –. A Figura 7 mostra a câmera P-90:

Figura 7: Câmera P-90

Fonte: [6]

A primeira etapa a ser alcançada durante o desenvolvimento do projeto foi a de contagem de passageiros dentro do carro. Para isto, buscou-se uma câmera que atendesse às

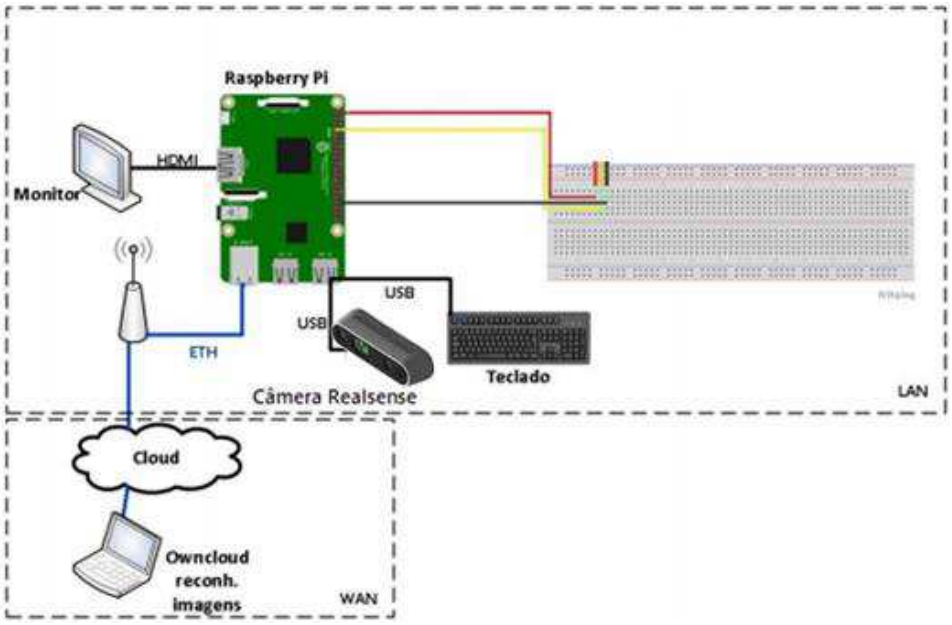
necessidades técnicas de poder permitir o reconhecimento facial e também de poder ultrapassar a barreira do vidro nos carros, e que estivesse também dentro do requisito de ser uma câmera de baixo custo. Chegou-se então a optar inicialmente pela câmera Kinect, da Microsoft [6].

A Kinect possibilitou provar que era viável a contagem de passageiros e detecção da placa, conforme planejado para o projeto URCA [6], graças aos seguintes recursos: a câmera RGB (*Red, Green, Blue*) – o que serviria para a detecção das placas –, o sensor de profundidade – que permite o uso de imagens de profundidade, que ultrapassaria a película do vidro e permitiria o reconhecimento de pessoas – e a capacidade de detecção de 48 pontos de articulações do corpo humano – que permitiria o reconhecimento de movimentos –. Apesar disto, a Kinect possibilitou apenas testes em ambiente *indoor*, pois no momento que era exposta ao sol, suas câmeras retornavam apenas clarões. Por isso, buscou-se um novo dispositivo que fosse semelhante a Kinect, também de baixo custo, e que pudesse ser utilizado em ambientes *outdoor*.

Por isso foi adquirida uma câmera desenvolvida pela Intel, a *Realsense* modelo D435. Ela traz um conjunto de tecnologias de profundidade e rastreamento, que tem como objetivo proporcionar recursos de percepção de profundidade. Ela também possibilita a visualização de imagens coloridas RGB e em profundidade (IR), contando com dois sensores de imagem (direito e esquerdo), um projetor infravermelho e uma câmera RGB, além de possuir um maior alcance e maior resolução, tanto da câmera RGB como de profundidade [6]. O seu diferencial foi permitir a aplicação do algoritmo de reconhecimento de pessoas em ambiente *outdoor* e hoje é a câmera que está sendo utilizada pelo Projeto. Mais outras especificações técnicas sobre a *Realsense* serão tratadas na seção 3.

Ainda com relação à especificação do material necessário, utilizou-se também neste primeiro momento a placa *Raspberry*, pois tanto a câmera Kinect quanto a *Realsense*, por si só, não são capazes de se conectar à rede, portanto, a placa possibilitaria a conexão e comunicação de dados graças às suas interfaces de rede. Também, para que as informações pudessem ser visualizadas e manipuladas foram adicionados à *Raspberry* os periféricos como monitor e teclado [6]. As especificações técnicas da placa *Raspberry* e dos demais equipamentos serão feitas na seção 3.1. A Figura 8 mostra como ficou o Projeto para a Prova de conceito:

Figura 8: Projeto para a prova de conceito



Fonte: adaptado de [6]

Até então, a prova de conceito da solução do Projeto foi realizada em duas fases: uma em ambiente *indoor* (no Nutenge – Núcleo Tecnológico de Engenharia da UEMA), utilizando a câmera Kinect, e outra em ambiente *outdoor* (no Pórtico de entrada da Universidade Estadual do Maranhão), através da *Realsense*. Para o caso do ambiente externo, mais próximo do real de aplicação do Projeto URCA, alguns parâmetros de posicionamento da *Realsense* foram testados para estabelecer qual a melhor forma de utilizar a câmera a fim de verificar a possibilidade de que somente uma já seria suficiente para realizar as duas detecções – de placas e de pessoas – com um mesmo ângulo. Em um dos testes *outdoor*, a câmera foi posicionada em frente ao vidro frontal do carro, conforme Figura 9, de forma a pegar passageiros e placas, mas verificou-se que a incidência solar nesta posição dificulta a penetração do feixe de infravermelho dentro do carro com a consequente dificuldade de identificação de pessoas através da câmera de profundidade da *Realsense*. Portanto foi preciso utilizar outro posicionamento para a câmera.

Figura 9: *Realsense* modo Infravermelho – Frontal do Carro

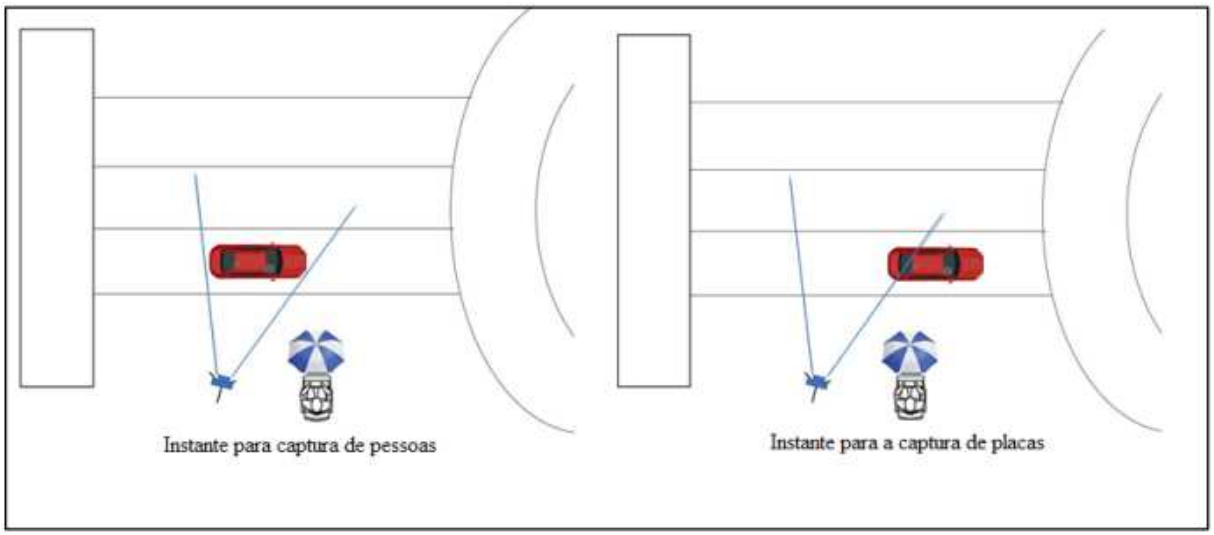
Fonte: [6]

Desta forma, na tentativa de diminuir a interferência da irradiação para possibilitar a detecção de pessoas, optou-se pela captura do vidro lateral do carro, conforme mostra a Figura 10. O esquema de posicionamento da *Realsense* para a prova de conceito neste momento ficou assim definido (conforme Figura 11) [6]:

Figura 10: *Realsense* modo Infravermelho – Lateral do Carro

Fonte: [6]

Figura 11: Esquema de Teste com a *Realsense* para captura de pessoas e placas



Fonte: [6]

De acordo com o esquema da Figura 11 tem-se: a passagem do carro após o pórtico no sentido da esquerda para a direita, que entra no campo de visão da câmera, com ela posicionada a 3,5 m da lateral do carro e com ângulo de abertura entre 70º e 90º. Desta forma, as capturas seriam feitas por uma mesma câmera, onde a captura de quantidade de pessoas aconteceria quando o carro estivesse no centro do campo de visão (conforme o lado esquerdo da Figura 11) e a captura de placas quando o carro estivesse perto do fim do campo de visão da câmera (conforme lado direito da Figura 11).

Até aqui os resultados alcançados por meio das etapas anteriores foram extrações de quantidade de pessoas e de placas para carros trafegando a velocidade de até 60 km/h [6], com os recortes das imagens de entrada para os algoritmos sendo feitos de forma manual.

A partir daí, identificou-se a necessidade de dar continuidade ao Projeto através do desenvolvimento das seguintes fases: criação de um banco de dados para guardar as informações das capturas com a automatização do envio destas ao banco; implementação da aplicação Urca *Analytics*; implementação de um protótipo da solução URCA com as aplicações desenvolvidas até então de forma integrada. Estas etapas foram trabalhadas por [9] e serão sucintamente explicadas a seguir.

O banco de dados foi feito através da modelagem dos dados e posterior criação das tabelas por meio da ferramenta MySQL *Workbench*, a fim de ser utilizado para guardar as informações coletadas pelo algoritmo de capturas, bem como fornecer estas informações à aplicação URCA *Analytics*. Para isto, foi preciso estabelecer a comunicação entre o banco e a

aplicação de capturas, para que fosse possível testar se a conexão é capaz de povoar as tabelas corretamente com informações que posteriormente possam ser manipuladas por meio do URCA *Analytics*.

Seguiu-se então com a tarefa de automatização do envio dos dados capturados. Conforme [9], houve a necessidade de se automatizar tal processo para tornar o funcionamento do URCA mais prático e eficiente, que permitisse o preenchimento automatizado das tabelas do banco de dados com informações vindas diretamente da aplicação de capturas, uma vez que as entradas (quadros de vídeo) para extração de informação, até então, eram feitas manualmente.

A solução encontrada por [9] foi desenvolver um gatilho (*trigger*) via *software* para capturar os quadros do vídeo em instantes pré-determinados e, dessa maneira, obter automaticamente as entradas necessárias para cada tipo de extração de informações. Após um período de estudos, chegou-se à implementação de uma aplicação que combinou o algoritmo de capturas com técnicas de detecção de objetos e visão computacional (apresentadas em [9]), chegando-se ao passo final para a implementação do gatilho, onde foram criados quatro pontos (A, B, C e D) com alguns valores estimados, chamados de "trigger points" – com cada *trigger point* correspondente a uma posição dentro do quadro –.

Para determinar esses pontos, foram feitos diferentes testes e, após algumas tentativas em um ambiente controlado – utilizando dois carros próprios pertencentes a membros do projeto – e associando com as saídas geradas pela automatização, conseguiu-se ajustar os pontos estimados para os *trigger points*, de modo que alguns deles salvassem imagens que mais se aproximavam das características de entradas desejadas para extração de quantidade de passageiros e extração de placas. Foram então eleitas como entradas: uma imagem engatilhada no ponto C do quadro da câmera de profundidade para a extração de passageiros e uma imagem engatilhada no ponto D do quadro da câmera RGB como as que mais se aproximavam das entradas desejadas.

Desse modo, como pode ser visto na Figura 12, as imagens passadas como entrada para o algoritmo de capturas utilizando os pontos C e D estimados, retornaram informações corretas sobre a quantidade de pessoas e os caracteres da placa. Assim, de acordo com esse processo, pôde-se constatar que a coleta das informações do algoritmo pode ocorrer de maneira automática e estas informações poderiam então ser enviadas ao banco de dados [9].

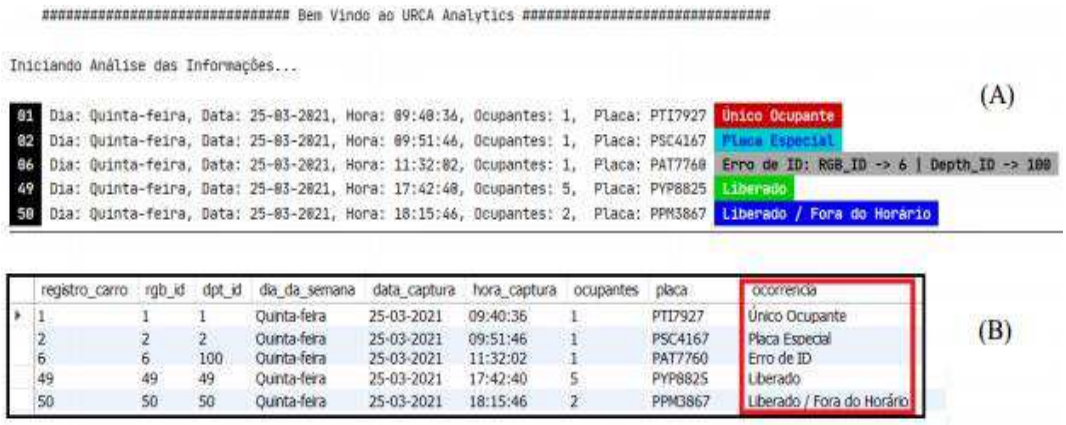
Figura 12: Resultados do algoritmo de capturas para os pontos C (câmera de profundidade) e D (câmera RGB)

Fonte: [9]

O passo seguinte do trabalho foi o desenvolvimento do URCA *Analytics*, que foi desenvolvido por [9] como uma interface para o banco de dados criado, integrando-se a este com o intuito de manipular as informações geradas e salvas no banco de dados pelo algoritmo de capturas da solução, fazendo o tratamento das mesmas e classificando-as de acordo com regras previamente definidas.

As funcionalidades do URCA *Analytics* deveriam permitir acesso aos dados armazenados apresentando-os de forma sistemática, gerando informações sobre tipos de ocorrência de acordo com a quantidade de passageiros e do horário do dia, acompanhamento das placas e associação com a quantidade de passageiros nos carros através de uma tela de acompanhamento [9]. Outras funcionalidades desenvolvidas foram: geração de gráficos, buscas seletivas de dados para que o operador possa retornar e mostrar na tela apenas os dados que lhe são de interesse. A Figura 13 mostra algumas destas funcionalidades.

Figura 13: Tela de acompanhamento do URCA Analytics

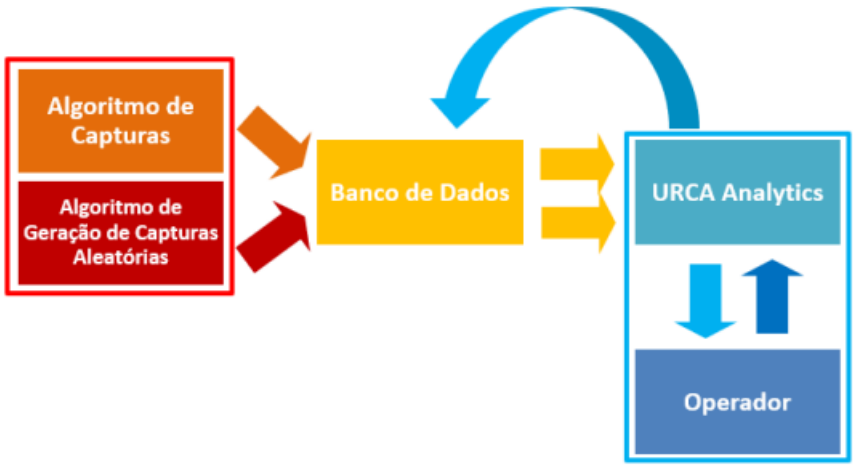


Fonte: [9]

A etapa final do trabalho de [9] foi a implantação de um protótipo da solução URCA realizando a integração das etapas concluídas anteriormente, que foi feita em uma rua com baixa movimentação de carros do Campus Paulo VI da UEMA.

A Figura 14 mostra o esquema da integração das aplicações que foi proposta para o momento do trabalho de [9], onde tem-se que o algoritmo de capturas recebe as entradas das imagens “trigadas” de forma automática e inicialmente envia essas informações extraídas à tabela no banco de dados para que, em seguida, quando a aplicação URCA *Analytics* for ativada, o seu operador tenha acesso a essas informações, por meio das funcionalidades desenvolvidas para o *Analytics* que, para cada entrada, irá interagir com o banco de dados.

Figura 14: Esquema de integração das aplicações até URCA *Analytics*



Fonte: [9]

2.7. O Gerenciamento do Projeto URCA

Nesta seção será explicado como foi feita a Gestão do Projeto URCA. Conforme visto na seção 2.6, já foi percorrido um longo caminho, fruto de muita pesquisa e dedicação pelos membros do projeto URCA através dos trabalhos [6], [8] e [9], o que possibilitou mostrar a viabilidade do Projeto como uma solução de baixo custo com possibilidade de ser implantado nas avenidas. De forma resumida, até então, o *roadmap* do URCA passa por: um protótipo formado pelos materiais e equipamentos especificados (encapsulamento mecânico), pela aplicação de capturas e extração de placas e passageiros, por um banco de dados implementado e uma aplicação para interface gráfica para acesso às informações armazenadas nesta base de dados (*Analytics*) e um aplicativo para conectar os futuros usuários da solução (URCARONA).

Ter esta ideia geral sobre processo percorrido pelo Projeto URCA é importante para situar os objetivos já alcançados até aqui e entender a necessidade de se gerenciar os próximos passos, tendo como fim viabilizar o fornecimento da solução URCA como uma solução pronta para implantação nas ruas, através do alcance de novas etapas e, por meio disto, da entrega de uma solução com as aplicações integradas que possa ser testada em um ambiente aberto, mais próximo do ambiente das avenidas da cidade.

Para isto, pensando no ambiente das avenidas movimentadas e situando-se com o que o Projeto já tinha em mãos até então, foram levantados novos requisitos para nortear o esforço deste trabalho de Gestão das aplicações URCA. Eles estão relacionados ao alcance de objetivos que tornem a solução URCA capaz de lidar e ser eficaz com relação a diferentes variáveis do ambiente *outdoor*, como a segurança e acomodação da solução nas avenidas e também às diferentes velocidades com que os carros trafegam nestas vias, de forma que a captura de placas e quantidade de passageiros não seja comprometida.

É preciso ressaltar também que, conforme os testes foram sendo realizados com os materiais disponíveis até então, e visando satisfazer os novos requisitos, foi identificada a necessidade de melhoria de resposta em alguns pontos, que poderiam ser melhor atendidos com a adaptação dos equipamentos disponíveis ou mesmo com o *upgrade* destes. Por isso foram necessárias novas especificações para alguns equipamentos e também tecnologias. Também se identificou como requisito importante a integração das aplicações já desenvolvidas, especialmente com relação ao aplicativo URCARONA, levando em conta também o fornecimento de uma vertente da aplicação para os motoristas.

Desta forma, a Gestão do Projeto URCA se dá no processo de planejamento e execução das seguintes etapas:

- Testes das aplicações já desenvolvidas dentro do Projeto, de modo que elas forneçam saídas em forma de dados de placa e quantidade de passageiros, tendo como dados de entrada o fluxo de quadros da câmera *Realsense*;
- Nova especificação técnica para escolher equipamentos que permitam um melhor desempenho da aplicação de capturas, agora para velocidades mais altas, próximo do ambiente das avenidas, tendo sido estabelecidas como metas três velocidades de carros trafegando, para as quais a solução deverá conseguir extrair informações: a 20 km/h, a 40 km/h e a 60 km/h;
- Otimização da estratégia de extração de informações referente à parte do programa de detecção e de extração de placas de carros e de passageiros visando o alcance da extração para velocidades mais altas e o melhor aproveitamento do potencial de processamento do *hardware* utilizado;
- Produção de encapsulamento mecânico dos equipamentos em um só conjunto para permitir fixação segura contra furtos e ao mesmo tempo eficiente contra as intempéries como fortes chuvas, vento ou calor excessivo comuns ao local de implantação;
- Integração destas aplicações ao URCA *Analytics* e ao aplicativo URCARONA.

A fim de ilustrar melhor onde se situa o Gerenciamento do Projeto URCA, pode-se verificar pela Figura 15 as etapas já percorridas e alcançadas pelos Trabalhos em [6], [8] e [9], bem como é destacada a integração realizada entre estas etapas, como requisito do Gerenciamento do Projeto URCA para produzir uma solução como um todo.

Figura 15: Gestão do Projeto URCA e os Trabalhos do Projeto integrados

Fonte: do Autor

Nas seções seguintes, será explicado todo o processo realizado neste trabalho de Gestão do Projeto URCA e os resultados alcançados.

3. MATERIAIS E MÉTODOS

Nesta seção serão detalhados os processos e técnicas utilizados para o alcance dos requisitos previamente definidos como forma de gerenciar o Projeto URCA, desde a especificação para os equipamentos necessários e para o encapsulamento mecânico, até a montagem da solução encapsulada com seu respectivo orçamento e à definição dos parâmetros para a realização dos testes de campo da solução URCA.

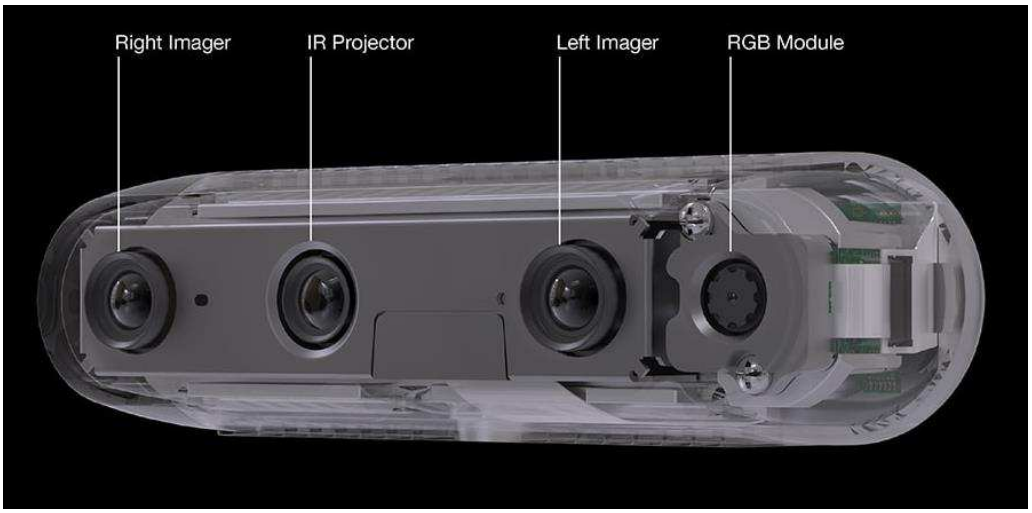
3.1. Especificação técnica do material

3.1.1 Câmera Intel Realsense

De acordo com a *Intel* [28], as tecnologias *Intel RealSense* estão basicamente reformulando o futuro ao equiparem dispositivos com a capacidade de ver, entender, interagir e aprender com o ambiente delas. Com anos de experiência na criação de todos os tipos de visão computacional e soluções de profundidade, os dispositivos *Intel RealSense* são a opção perfeita, quaisquer que sejam suas necessidades de sensibilidade de profundidade.

Dentre as câmeras oferecidas pela *Intel*, a câmera de profundidade *Intel RealSense D435*, mostrada na Figura 16, é uma solução de rastreamento estéreo, oferecendo profundidade de qualidade para uma variedade de aplicações e chamou a atenção do projeto URCA. Seu amplo campo de visão é perfeito para aplicações como robótica ou realidade aumentada e virtual, onde ver o máximo possível da cena é de vital importância [29]. Com alcance de até 10m, esta câmera de formato pequeno pode ser integrada a qualquer solução com facilidade e vem completa com o kit de desenvolvimento de Software *Intel RealSense SDK 2.0* e suporte a várias plataformas [29].

Figura 16: *Intel RealSense D435.*



Fonte: [29].

Como o *SDK 2.0* da *Intel Realsense D435* é de código aberto e possibilita a integração com linguagens de programação como *Python*, *C*, *C++* dentre outras, tal câmera se torna um dispositivo atrativo para as aplicações do projeto, então é válida uma especificação resumida sobre este dispositivo.

As especificações da *Intel Realsense 435* encontra-se na Tabela 1:

Tabela 1: Especificações da *Intel Realsense D435*.

Recursos	• Ambiente de uso: *Indoor/Outdoor* • Tecnologia de sensor de imagem: Obturador global, tamanho de pixel de 3µm x 3µm • Alcance: Aprox. 10 metros. Obs.: A precisão varia dependendo da calibração, cena e condição de iluminação.
Câmera de Profundidade	• Tecnologia de profundidade: *Active IR Stereo* • Campo de Visão de Profundidade: 87°±3° x 58°±1° x 95°±3° • Distância mínima de profundidade (Min-Z): 0,105 m • Resolução de saída de profundidade e taxa de quadros: resolução de profundidade estéreo ativa de até 1280 x 720. Até 90 fps.
Câmera *RGB*	• Resolução do sensor RGB e taxa de quadros: 1920 x 1080 • Taxa de quadros RGB: 30 fps • Campo de visão do sensor RGB (H x V x D): 69,4 ° x 42,5 ° x 77 ° (+/- 3 °)
Componentes Principais	• Módulo da câmera: Câmera *Intel RealSense Module D430 + RGB* • Placa do processador *Vision*: *Intel RealSense Vision Processor D4*
Características Físicas	• Fator de forma: Periférico da câmera • Comprimento x Profundidade x Altura: 90 mm x 25 mm x 25 mm • Conectores: *USB C 3.1 Gen 1*

	• Mecanismo de montagem: Um ponto de montagem de rosca 1 / 4-20 UNC. Dois pontos de montagem de rosca M3.

Fonte: [30].

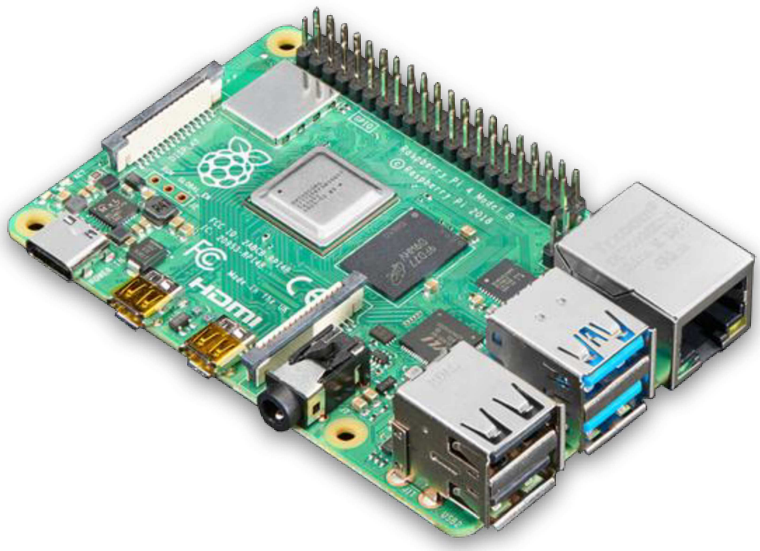
3.1.2 Raspberry Pi 4

De posse da câmera especificada, faz-se então necessário um outro dispositivo para realizar a comunicação dos dados a através da rede. Como a câmera *Intel Realsense D435* não possui interface de rede, foi especificado um dispositivo chamado *Raspberry Pi 4*.

O *Raspberry*, lançado em 2006, é um computador do tamanho de um cartão de crédito, desenvolvido no Reino Unido pela Fundação *Raspberry Pi*, que pode ser conectado a periféricos como um monitor de computador ou TV, bem como a um teclado e um mouse padrão através de suas interfaces. Todo o hardware é integrado em uma única placa [6].

De modo específico, o modelo utilizado é o *Raspberry Pi 4 Model B*. De acordo com [31], este dispositivo, mostrado na Figura 17, é o produto mais recente da linha de computadores *Raspberry Pi*, que oferece aumentos inovadores na velocidade do processador, desempenho de multimídia, memória e conectividade. Este minicomputador apresenta como principais recursos: suporte a dois monitores em resoluções de até 4K por meio de um par de portas micro-HDMI, decodificação de vídeo por hardware de até 4K, 8 GB de RAM, LAN sem fio nas bandas 2.4/5.0 GHz, Bluetooth 5.0, Gigabit Ethernet e portas USB 3.0.

Figura 17: *Raspberry Pi 4 Model B*



Fonte: [32].

O *Raspberry Pi 4 Model B* possui as seguintes especificações [26]:

- Processador: *Broadcom BCM2711*, quad-core Cortex-A72 (ARM v8);
- Memória RAM: 8 GB LPDDR4;
- Conectividade: 2,4 GHz e 5,0 GHz IEEE 802.11b/g/n/ac sem fio, LAN, Bluetooth 5.0, Gigabit Ethernet, 2 portas USB 3.0 e 2 portas USB 2.0;
- Vídeo e som: 2 portas micro HDMI (com suporte para até 4Kp60), porta para display MIPI DSI de 2 faixas, porta para câmera MIPI CSI de 2 faixas, áudio estéreo de 4 polos e porta de vídeo composto;
- Multimídia: decodificação H.265 e H.264, OpenGL ES, gráficos 3.0;
- Suporte para cartão SD: slot para cartão Micro SD para carregamento de sistema operacional e armazenamento de dados;
- Potência de entrada: 5 V DC via conector USB-C (mínimo 3A), 5 V DC via cabeçalho GPIO (mínimo 3A), Power over Ethernet (PoE) – habilitado;
- Ambiente: Temperatura de operação 0–50ºC.

Essas especificações foram decisivas para aquisição deste dispositivo, pelo fato de a câmera necessitar de entradas USB 3.0 e de uma certa capacidade de processamento para que possa funcionar corretamente.

3.1.3 Caixa de passagem hermética

Como parte da especificação do material integrante do Projeto URCA, pensou-se na necessidade de um artefato que pudesse acomodar os equipamentos eletrônicos – a princípio a placa *Raspberry* e a câmera *Realsense* – de forma a protegê-los das condições climáticas adversas de um local aberto, como incidência de chuvas ou de forte calor, que podem diminuir a vida útil destes equipamentos ou mesmo danificá-los caso eles não tenham o devido resguardo. Então, como parte da estratégia de encapsulamento mecânico da solução URCA, foi adquirida uma caixa de passagem com tampa transparente, modelo 300 da Kraus-Muller [33].

Ela foi escolhida porque suas especificações vão de encontro à necessidade de proteção dos equipamentos responsáveis pela captura e extração de informações dos carros, ao mesmo tempo que permite a filmagem da câmera através da tampa transparente. De acordo com [33] algumas dessas características são: as dimensões de 300x220x148 mm, que permitem acomodar com tranquilidade a câmera, a placa e seus cabos de conexões; indicada para montagem de equipamentos elétricos, passagem e comando; fabricação com dupla isolação permitindo um grau de proteção maior, IP 65 (contra poeira e contra jatos d'água), temperatura de Operação

de -5ºC à 70ºC e proteção contra raios U.V de até 5 anos. A Figura 18 mostra como é a caixa de passagem escolhida:

Figura 18: Caixa de passagem hermética

Fonte: [33]

3.1.4 Mini PC

O Mini PC foi escolhido como material integrante da solução URCA para substituir a *Raspberry* (processo que será explicado na seção 4), devido ao seu maior poder de processamento – fornecidos por seu processador i9 de 10ª geração em combinação com sua memória RAM DDR4 e Memória SSD Nvme – e às suas dimensões, que permitem que ele possa ser encapsulado na caixa de passagem hermética e possa ser levado às ruas para permitir o funcionamento da solução no ambiente *outdoor*. A Figura 19 ilustra o mini PC escolhido:

Figura 19: mini PC

Fonte: do Autor

A Tabela 2 indica as principais especificações do mini PC utilizado:

Tabela 2: Especificações do Mini PC

CPU	Intel Core i9 10980HK 8 núcleos, 16 *threads*, 2 GHz - 5,10 GHz
Placa de vídeo	Intel UHD *graphics* 350 MHz – 1,25 GHz
Memória	16 GB DDR4
Sistema Operacional	Windows 10/Linux Ubuntu 18.04
Armazenamento	512 SSD Nvme
Rede	2 portas RJ45 Gigabit Ethernet
Entradas	6 x USB 3.0, 1 x HDMI, 1 x Áudio, 1 x DC

Fonte: do Autor

3.1.5 Notebooks do Projeto

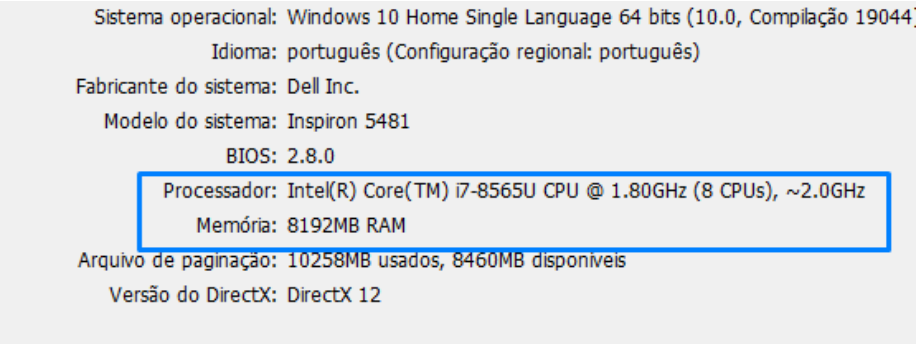
Também são considerados como equipamentos úteis e que fazem parte deste Trabalho, os notebooks já pertencentes ao Projeto URCA: o notebook com sistema operacional Windows 10 e o *Macbook Air* com sistema operacional *MacOS Monterey*.

Dentre estes, o notebook com Windows foi bastante utilizado nos testes iniciais para interface dos programas de captura com a câmera *Realsense* (conforme explicado na seção 2.6) bem como serviu de ferramenta de trabalho para o desenvolvimento de diversas finalidades intermediárias que puderam ser realizadas por meio dele.

Os dois notebooks foram utilizados nos testes realizados nesta Gestão do Projeto, tanto os testes de campo com entrada por vídeo da câmera com processamento sequencial como com os testes com entrada de vídeo das gravações feitas. Tal utilização se deu a fim de se fazer um comparativo envolvendo os dois notebooks e os computadores especificados acima – *Raspberry* e Mini PC – quanto aos seus respectivos desempenhos para a captura de imagens e extração de informações. Os resultados deste comparativo e os demais serão discutidos na seção 4.

As Figuras 20 e 21 destacam algumas das especificações principais do notebook com Windows e do *Macbook Air*, respectivamente.

Figura 20: Especificações do Notebook i7 com Windows



Fonte: do Autor

Figura 21: Especificações do Macbook i5 com MacOS

Fonte: do Autor

3.2. Especificação técnica para a implantação no ambiente *outdoor*

A seguir serão apresentados alguns requisitos técnicos indicados para permitir uma boa implantação de sistemas de monitoramento em ambientes *outdoor*, os quais nortearam o projeto do encapsulamento mecânico da solução URCA.

Uma solução de videomonitoramento bem-feita pode trazer inúmeros benefícios para que as capturas sejam bem-sucedidas e também para eficácia operacional. Para isto, além do estudo e seleção das tecnologias tratadas na seção 3.1, é preciso projetar o encapsulamento desta solução em um formato que envolva o conjunto dos equipamentos que formam a solução proposta para a captura e comunicação de dados, levando em conta as diferentes variáveis do ambiente real onde será implantada a solução. Assim, devem ser consideradas diferentes exigências deste ambiente real ao projetar o encapsulamento da solução de forma que tais problemas possam ser tratados ou contornados.

O foco do monitoramento proposto pelo URCA são os pontos por onde trafegarão os carros, de modo que seja possível capturar e identificar as suas placas e a quantidade de passageiros aí presentes. Por isto, é preciso determinar a melhor maneira de implantar a solução para garantir a sua eficiência mesmo diante das variáveis do ambiente de implantação que possam dificultar seu desempenho.

Para isto, algumas especificações de fatores que devem ser levados em conta para o projeto de encapsulamento da solução do Projeto URCA, considerando o tipo de ambiente *outdoor*, onde ela deverá ser implantada [34]:

- É preciso estudar quais são os fatores de riscos do local e seus pontos críticos;
- É preciso fazer a escolha correta da altura em que as câmeras serão posicionadas, pois é o primeiro fator que interfere diretamente na qualidade das filmagens e até na preservação dos equipamentos. Por exemplo, se o posicionamento for muito baixo e em uma área externa, o equipamento poderá ser exposto ao vandalismo e outros tipos de danificações. Por sua vez, câmeras colocadas em uma altura muito elevada irão exigir uma resolução superior e iluminação reforçada. A melhor altura vai depender da finalidade do monitoramento, do tipo de equipamento e do tamanho da área a ser monitora. De uma maneira geral, o recomendado é que as câmeras fiquem a pouco mais de 3 metros de altura do chão, para que não possam ser alcançadas facilmente e não limitem demais o seu campo de visão;
- O equipamento a ser instalado em ambiente *outdoor* deverá resistir a adversidades como chuva, calor forte, ventos de alta velocidade, dentre outros fatores naturais intrínsecos ao ambiente *outdoor*. Uma opção é optar por caixas protetoras à prova d'água e de temperatura, que são ideais para essas circunstâncias.

- Com relação ao posicionamento, o recomendável é posicionar a câmera na direção contrária à luz do sol, uma vez que os raios podem superaquecer o sensor das câmeras;
- É preciso atentar para a distância da câmera aos demais equipamentos, como os de rede, atendendo às distâncias máximas dos cabos para permitir comunicação de dados e alimentação;
- É preciso considerar a iluminação do local onde será posicionada a solução. Além do superaquecimento do equipamento já citados, a luz solar externa também pode interferir na visibilidade de algum ponto a ser monitorado, que ficará prejudicado por horas até que o sol mude de posição. A recomendação é de posicionar a câmera contra a luz natural ou, pelo menos, apontá-las em um ângulo indireto para trazer nitidez às imagens captadas.

Seguindo estas indicações, foi possível montar um plano de ações para projetar o encapsulamento mecânico da solução e gerir a sua melhor forma de implantação visando encapsular os equipamentos de forma a conseguir contornar tais possíveis problemas.

3.3 Encapsulamento da solução

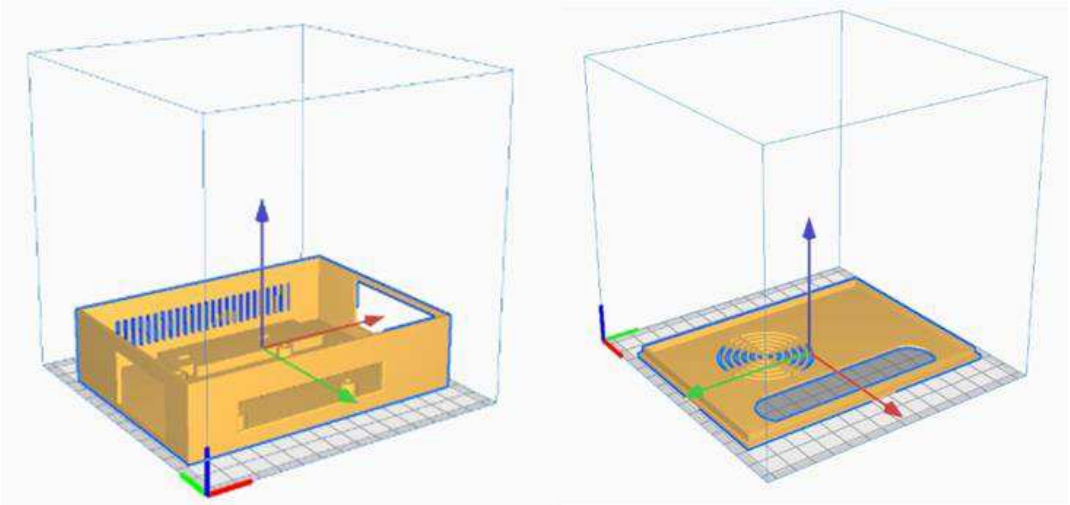
Levando em conta as especificações tratadas na seção 3.2 e como parte da Gestão do Projeto URCA para integrar os trabalhos feitos até aqui e para permitir sua efetiva implantação, fez-se necessário fazer o encapsulamento do conjunto de equipamentos utilizados para que possam funcionar como um todo, uma solução integrada, superando os parâmetros intrínsecos ao ambiente de implantação. Assim, considerando as diferentes exigências deste ambiente real, o encapsulamento foi projetado de forma tratá-los ou contorná-los.

A primeira questão a ser levada em conta foi a proteção dos equipamentos, que é parte essencial de um sistema de monitoramento efetivo ao passo que ela permite aumentar a longevidade do sistema de monitoramento, manter a efetividade operacional a longo prazo e evitar furtos [35]. O encapsulamento também foi projetado para fornecer cobertura para garantir proteção da câmera contra poeira, água da chuva e ferrugem, considerando que esta proteção não prejudique os movimentos nem o campo de visão do aparelho, nem permita a fragilização dele, para diminuir custos de manutenção futuros e garantir que a câmera funcione mesmo em climas desfavoráveis.

Vale registrar que, antes da adoção do Mini PC em substituição da placa *Raspberry*, foi feito um protótipo inicial com uma caixa protetora por meio de impressão 3D, projetada para abrigar o conjunto placa *Raspberry Pi 4* com a câmera *Realsense*. O modelo foi disponibilizado

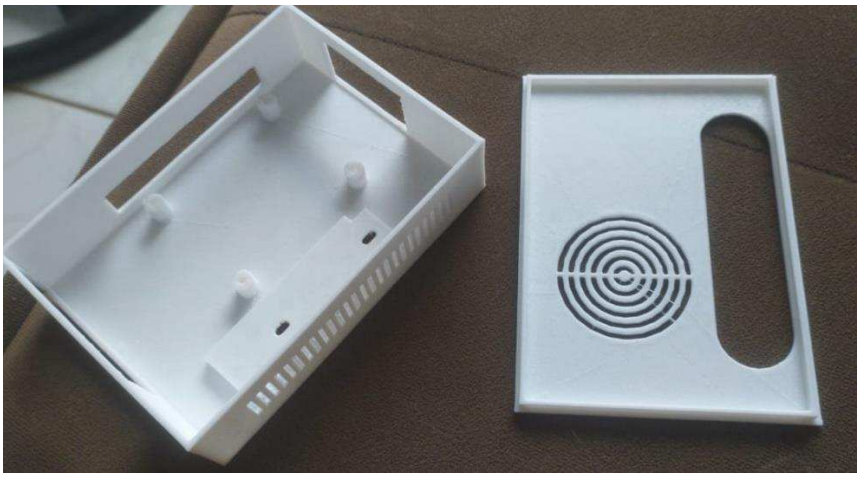
por [36], que compartilha o arquivo no formato próprio para impressão 3D (*.stl*) das duas peças que formam a proteção do protótipo inicial sendo elas caixa e tampa. A Figura 22 mostra o *case* protetor do conjunto *Raspberry* e câmera em formato 3D, enquanto a 22 mostra o modelo impresso. Este protótipo inicial, apesar de acomodar bem a câmera e a *Raspberry*, não foi continuado, pois verificou-se que a proteção fornecida não era suficiente diante do que foi especificado para o ambiente *outdoor*.

Figura 22: Vistas das peças 3D do modelo utilizado



Fonte: Do Autor.

Figura 23: Caixa e Tampa do modelo para encapsulamento inicial da solução



Fonte: Do Autor.

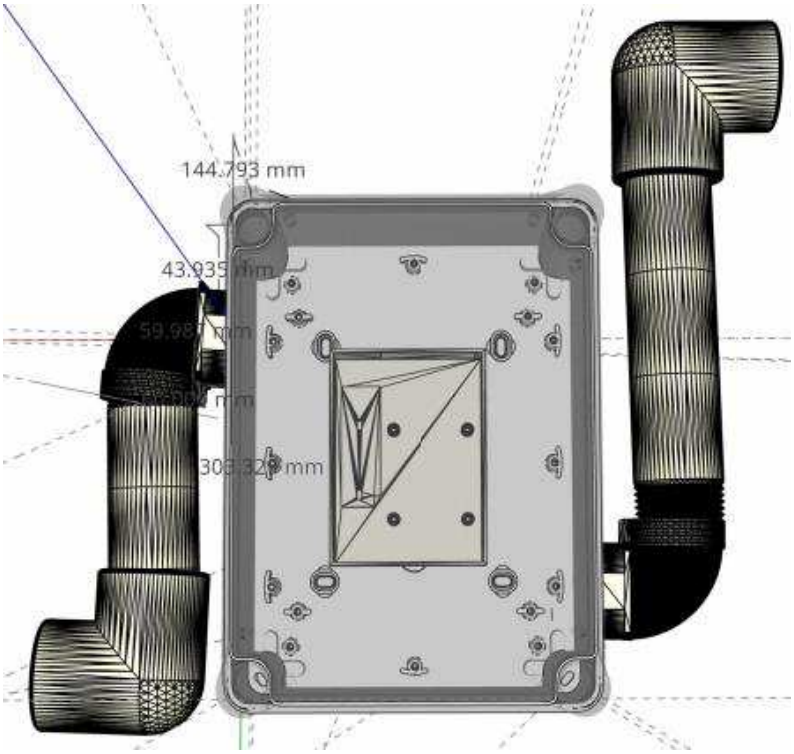
Continuou-se a pensar e a projetar então um encapsulamento que atendesse a mais exigências. Outro fator levado em conta nesta engenharia foi o superaquecimento dos aparelhos, que, ao serem implantados no local estarão sob a forte ação do sol, com temperaturas de sensação térmica que durante o dia chegam a aproximadamente 40º C – que se somam ao

aquecimento próprio destes aparelhos, que por si só já é alto durante a execução de seus processos contínuos.

Assim, para contornar as questões de segurança e temperatura, o projeto do encapsulamento lançou mão de uma caixa hermética de passagem (especificada na seção 3.1) combinada com duas ventoinhas USB cujo funcionamento deverá ter uma destas com a função de retirar o ar quente circulando entre os aparelhos enquanto a outra servirá para impulsionar ar de um lado com temperatura menor para dentro da caixa.

Foi feita uma modelagem 3D com as dimensões aproximadas das reais da caixa hermética, de modo a facilitar o posicionamento das demais peças e depois das furações para a instalação das duas ventoinhas. Através da Figura 24 é possível identificar a estratégia adotada para o encapsulamento: além da caixa com as ventoinhas, também são ilustradas peças adicionais como joelhos – cada um acoplado a uma ventoinha – e extensões de canos nestes joelhos, formando um conjunto de proteção aos aparelhos numa disposição entre eles tal que forneça ao máximo uma barreira para a entrada de respingos de água das chuvas e permita a ventilação artificial para resfriamento dos equipamentos.

Figura 24: Modelagem 3D do encapsulamento mecânico para o ambiente *outdoor*



Fonte: Do Autor.

Já a Figura 25 identifica o encapsulamento mecânico implementado, fruto das estratégias anteriormente faladas e que será utilizado futuramente no ambiente real com a solução rodando encapsulada e com as devidas proteções projetadas. Podem ser observados o

Mini PC, a câmera *Realsense* dispostos dentro da caixa de passagem e os joelhos acoplados às ventoinhas, prontos para serem adicionados extensões de canos de diâmetro de 50 mm.

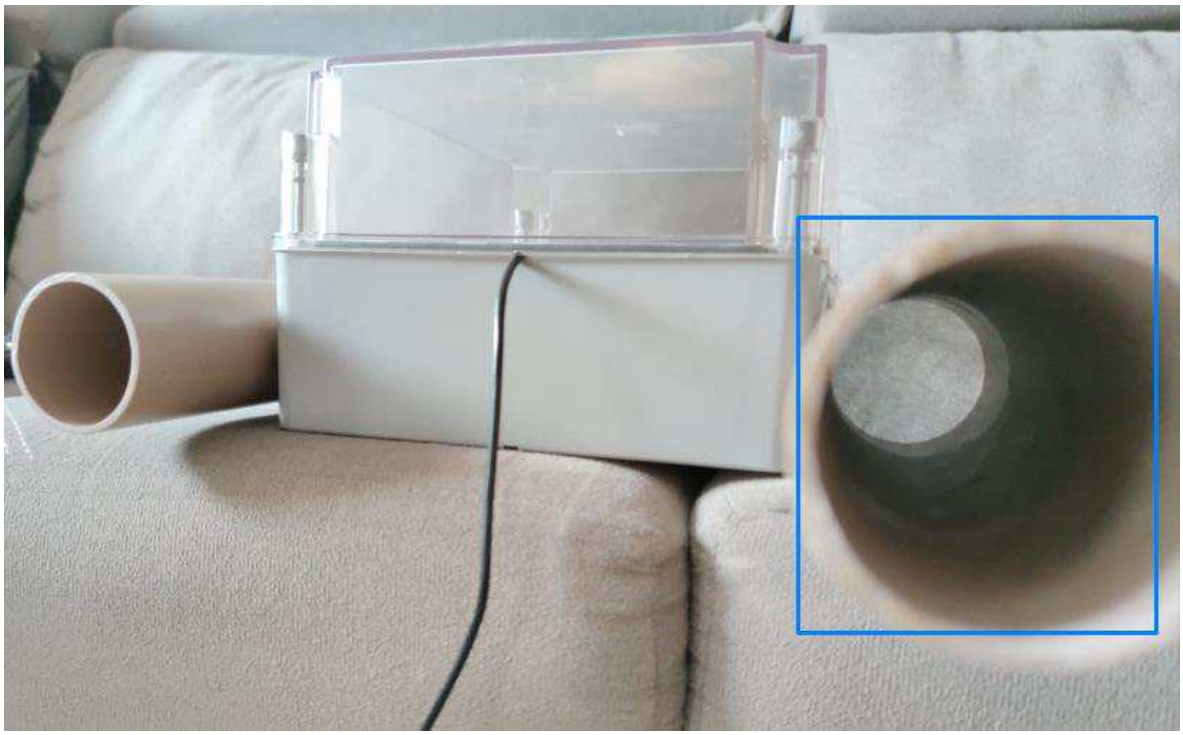
Figura 25: Encapsulamento mecânico implementado

Fonte: Do Autor.

Em vista de futuros testes com a solução encapsulada, também foi adicionada mais uma forma de proteção contra fatores ambientais que podem prejudicar o funcionamento dos equipamentos protegidos, através do uso de um filtro em um dos lados dos dutos para impedir que o ar puxado pela ventoinha circule com partículas de poluição. Tal fato foi considerado e implementado pois a presença de partículas como as de fuligem e também de outras formas, podem ser prejudiciais aos microcomponentes eletrônicos presentes no MiniPC e também na *Raspberry*.

Na Figura 26 pode-se identificar a caixa hermética com os dutos de ventilação já acoplados, bem como a presença de um filtro de tecido de cor clara do lado direito (na ventoinha que puxa o ar para dentro), para barrar a entrada direta de micropartículas que estejam presentes no ambiente.

Figura 26: Filtro de ar integrante do encapsulamento



Fonte: do Autor

3.4 Tecnologias utilizadas

3.4.1 Visual Studio Code

O *Visual Studio Code* (VS *Code*) é uma ferramenta de código aberto para desenvolvimento de programas desenvolvida pela Microsoft. De acordo com a própria Microsoft ele é “um editor de código-fonte autônomo que é executado no Windows, macOS e Linux” [37] e possui as funcionalidades mais simples como: edição de código com suporte a várias linguagens de programação, terminal de comandos integrado além de controle de versão de código [38].

Apesar de ser somente um editor de código e ter somente a versão gratuita, o *VS Code* tem sido amplamente adotado pelos desenvolvedores pelo oferecimento de funcionalidades que podem ser reunidas em uma só ferramenta e que, por vezes, se mostra tão eficiente quanto outras IDEs (do inglês *Integrated Develpment Enviroment* – Ambiente de Desenvolvimento Integrado) pagas. Entre essas funcionalidades, as seguintes são destacadas [37]:

- *Intellisense* que fornece preenchimentos inteligentes com base em tipos de variáveis, definições de funções e módulos importados permitindo ir além do realce de sintaxe e do preenchimento automático;

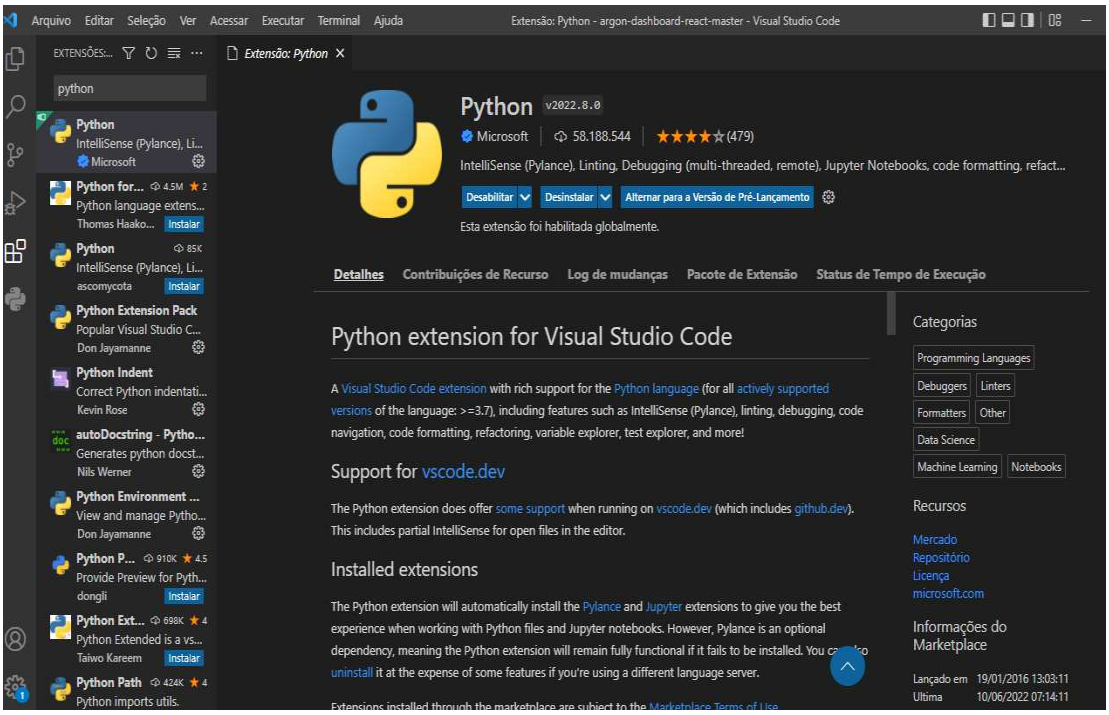
- Depuração de código direto do editor, que permite se inicie ou anexe diferentes aplicativos em execução e se depure estes aplicativos com pontos de interrupção, pilhas de chamadas e um console interativo;

- Personalização e Loja de Extensões rica e cheia de utilidades, que se torna um outro ponto forte do *VS Code*, onde é possível instalar extensões para adicionar novos idiomas, temas, depuradores e conectar-se a serviços adicionais e com cada uma dessas extensões instaladas sendo executada em processos separados, garantindo que não deixem o editor mais lento.

O VS Code é uma ferramenta leve e altamente customizável onde, por exemplo, ao configurarmos um projeto de desenvolvimento de código, podemos fazer login no VS Code para manter nossas configurações salvas na nuvem. Dessa forma podemos manter nossas preferências e extensões sincronizadas em computadores diferentes [38].

A Figura 27 mostra uma tela do *Visual Studio Code* onde podem ser encontradas várias extensões que permitem ao desenvolvedor trabalhar com a linguagem de programação Python:

Figura 27: Tela de extensões do *VS Code*



Fonte: Do Autor

3.4.2 Python

Python é uma linguagem de programação interpretada simples, porém poderosa, que preenche a lacuna entre a programação C e *shell* (intérprete de comandos para acessar recursos do Sistema Operacional). Sua sintaxe é composta de construções emprestadas de uma variedade de outras linguagens enquanto o interpretador Python pode ser facilmente estendido com novas funções e tipos de dados implementados em C. Esta poderosa linguagem está disponível para

vários sistemas operacionais, entre os quais vários tipos baseados em UNIX (incluindo distribuições Linux), Apple Macintosh O.S., MS-DOS e Windows [39].

Python é uma linguagem de alto nível e de propósito geral. Seu design a torna muito legível, o que é mais importante do que parece. Por exemplo: todo programa de computador é escrito apenas uma vez, mas lido e revisado muitas vezes, e muitas vezes por diferentes pessoas, então, a facilidade de legibilidade oferecida pela linguagem a torna mais fácil de ser revisada, de ser aprendida e lembrada, e, portanto, mais gravável. Pode-se também afirmar que Python tem uma curva de aprendizado mais suave, comparada com a de outras linguagens de programação [40].

Python foi criada em 1991 e nos últimos anos encontra-se consistentemente entre as 10 linguagens de programação mais populares. Utilizar Python para escrever programas e funcionalidades já tem sido utilizado como prática habitual por empresas como Google, YouTube, Dropbox, Netflix e Hulu. Python é amplamente utilizado por ser uma linguagem flexível e de código aberto e tem uma reputação de produtividade que atrai organizações que prezam por este requisito, por terem rápido crescimento e necessitarem de tal velocidade no desenvolvimento de suas aplicações. Podemos encontrar Python em muitos ambientes de computação, incluindo os seguintes [40]:

- Em linha de comando (através do terminal ou console) ou em interface gráfica, através de uma IDE ou Editor de código;
- Interfaces gráficas de usuário, incluindo a Web;
- A Web, nos lados do cliente e do servidor;
- Servidores de *backend* que suportam grandes sites populares;
- A nuvem (servidores gerenciados por terceiros);
- Dispositivos móveis;
- Dispositivos embarcados.

Ainda como exemplo de aplicação da linguagem, temos [41] bancos e instituições financeiras que usam Python para processar dados; instituições acadêmicas e centros de pesquisa, que usam a linguagem para visualização e processamento de informações; empresas de previsão do tempo, de construção de modelos financeiros e corretoras de seguros que também a usam. Dentro do vasto campo da Ciência de dados, a linguagem Python torna-se uma das preferidas pois fornece um banco de dados massivo de bibliotecas para inteligência artificial e aprendizado de máquina. Algumas das bibliotecas mais populares incluem *Numpy, Scipy, TensorFlow, Matplotlib, Pandas*, dentre outras.

No projeto URCA, foi utilizada a linguagem Python na versão 3 e todo o aparato que ela oferece para o desenvolvimento do algoritmo de *Centroid Tracker* apresentado por [9], bem como dos demais módulos e bibliotecas complementares que foram necessários para que o programa fosse capaz de detectar e rastrear os carros em trânsito, tanto com entrada de quadros de vídeo ao vivo pela *Realsense* como em vídeo gravado, e ter suas capturas de imagem automatizadas.

3.5 Orçamento dos materiais

Como parte integrante de Materiais e Métodos, a Tabela 3 informa o custo médio da solução URCA integrada, com os valores dos materiais eletrônicos – cuja necessidade foi sendo feita através das especificações técnicas para o desenvolvimento da solução – e também do material do encapsulamento mecânico projetado e de sua mão-de-obra, cuja escolha e planejamento foram apresentados nas seções 3.1, 3.2 e 3.3.

Tabela 3: Orçamento dos materiais especificados

Material	**Valor (R$)**
Câmera Intel *Realsense* D435	1899,00
Mini PC	3981,70
Encapsulamento – mão de obra	50,00
Caixa de passagem hermética	138,90
Total	5979,60

Fonte: do Autor

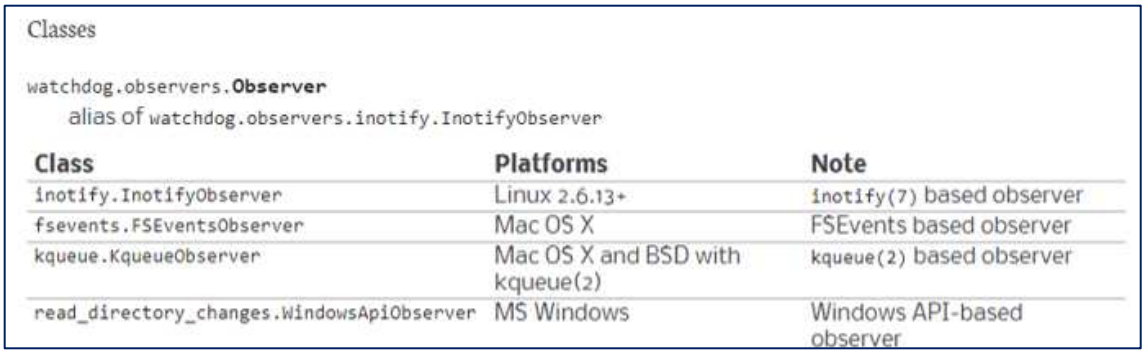
3.6 Otimização do processo de extração de dados

Para implementar o processamento e extração de placas e de quantidade de pessoas de forma paralela à captura dos quadros, foi utilizada uma biblioteca do Python chamada *Watchdog*. Ela é uma Biblioteca de API para Python que fornece utilitários de *shell* para diferentes plataformas (como Windows, Linux e Mac), com o objetivo de monitorar eventos do sistema de arquivos. O seu sistema de monitoramento e de notificações é baseado no padrão *Observer*, que é um padrão de projeto comportamental que permite que um objeto notifique outros objetos sobre alterações em seu estado. [42]. De acordo com a sua documentação oficial para Python, a implementação deste padrão é feita por meio de *Threads* de *Observers*, que monitoram diretórios e despacham chamadas para manipuladores de eventos conforme o evento identificado sobre o diretório alvo. [43]

A Figura 28 mostra como a biblioteca possui interface para utilitários *shel script* (interpretadores de comando de um sistema operacional) em diferentes plataformas, que podem

ser implementados a partir do uso da Classe *Observers* da *Watchdog*, permitindo que diferentes sistemas de arquivos sejam monitorados.

Figura 28: classe *Observer* com utilitários para diferentes SOs



Classes

watchdog.observers.**Observer**
alias of watchdog.observers.inotify.InotifyObserver

Class	Platforms	Note
inotify.InotifyObserver	Linux 2.6.13+	inotify(7) based observer
fsevents.FSEventsObserver	Mac OS X	FSEvents based observer
kqueue.KqueueObserver	Mac OS X and BSD with kqueue(2)	kqueue(2) based observer
read_directory_changes.WindowsApiObserver	MS Windows	Windows API-based observer

[Fonte: adaptado de 43]

Para demonstrar os conceitos acima descritos, será mostrada a implementação deles e como sua aplicação no programa pôde trazer resultados mais otimizados, conforme esperado.

Com já explorado, o uso de *Threads* pode ajudar a melhorar o desempenho do sistema operacional na execução paralela de diferentes processos. Em sistemas cliente-servidor, como o utilizado para a requisição da resposta da identificação da placa pelo servidor do ALPR, é feita uma solicitação a um serviço remoto. Se tal aplicação fosse feita em um sistema *monothread*, de forma sequencial, o programa deveria esperar o resultado requisitado antes de poder prosseguir sua execução [44]. Mas, aproveitando-se dos recursos disponíveis do miniPC, a requisição que espera a resposta do servidor passou a ser executada em uma *thread* de forma assíncrona, liberando o sistema para outras tarefas, enquanto a resposta não era devolvida ao programa em execução que a solicitou. Desta forma, enquanto os quadros do vídeo de entrada são processados um a um, numa exigência de alto processamento, a tarefa de solicitar e trazer o resultado da identificação das placas ficava delegada a uma outra *Thread,* acontecendo assim, de forma paralela, sem se tornar um gargalo para a sequência de quadros que deveriam esperar a resposta do servidor.

A estratégia então, consistiu em delegar o processamento das imagens capturadas para uma *Thread* dedicada, que seria acionada no momento da criação destes arquivos das imagens, de acordo com eventos que ocorressem no diretório específico onde estas imagens fossem salvas. O processamento inicial é feito em cima da entrada de dados, que é todo o fluxo de quadros de vídeo contínuo, onde o Trigger tenta identificar, quadro a quadro, a presença do carro e, em pontos específicos enquanto o carro é identificado, registra imagens como uma "foto" do carro passando nestes pontos específicos – conforme desenvolvido em [9]. O resultado dessa captura é gerado em forma de imagem .png em um caminho específico, na pasta

“crop_imagens”. Então, enquanto o processamento quadro a quadro acontece, a responsabilidade de extrair informações – placa e quantidade de passageiros – é delegada para outra linha de execução, feita em paralelo, de forma que não seja comprometido o processamento dos quadros para identificação dos pontos do *Trigger*.

Assim, seguindo a estratégia de otimização, foi implementado um *Observer*, executado de forma paralela ao processamento dos quadros, com o objetivo de “vigiar” a pasta onde são salvas as imagens capturadas pelo *Trigger*. Seguindo os conceitos do funcionamento do *Observer*, [43], a pasta “crop_imagens” seria monitorada continuamente e, ao ser identificado o evento de criação de um novo arquivo nesta pasta – imagem capturada pelo *Trigger* –, o *Observer* seria notificado e associaria a este evento a execução de um manipulador de eventos. Então, a implementação da requisição para identificação das placas ou do processamento para identificação de quantidade de passageiros, seriam executados por este manipulador de arquivos, disparado conforme os arquivos de imagem são criados na pasta alvo vigiada.

Na Figura 29 é destacada a parte do código utilizado na estratégia de otimização, na qual o *Observer* é instanciado e configurado, atrelado ao caminho a ser monitorado, bem como ao manipulador de eventos que será disparado quando o evento de criação de arquivo acontecer.

Figura 29: Implementação da otimização do processamento com *Observer*

processamento_extracao_placas_pessoas.py

```
242    #cada um dos métodos acima definidos serão realizados pelo manipulador de eventos:
243    manipulador_eventos.on_created = ao_criar #será chamado quando um arquivo ou diretório for criado
244    manipulador_eventos.on_deleted = ao_deletar #será chamado quando um arquivo ou diretório for deletado
245
246    #pasta onde estão todas as imagens de placas e pessoas
247    #cujos crops (recortes) foram feitos pelo Trigger:
248    caminho = "./crop_imagens"
249    pesquisar_subpastas=True
250    observer = Observer()
251    observer.schedule(manipulador_eventos, caminho , recursive=pesquisar_subpastas)
252
253    #iniciando o observer dos arquivos:
254    observer.start()
255    try:
256        while True:
257            time.sleep(1)
258    except KeyboardInterrupt: #para parar/interromper o script, apertar 'ctrl+c'
259        observer.stop()
260    observer.join()
```

Fonte: do Autor

O código com a implementação da estratégia proposta para otimização do processamento pode ser verificado no Apêndice C.

3.7 Parâmetros para a realização dos testes de campo

Reunidos os equipamentos necessários que foram especificados anteriormente, foi possível realizar os testes de campo. Foi realizada uma boa quantidade de testes, com o objetivo final de permitir as extrações das informações dos carros de forma a atender aos novos requisitos propostos: que tais extrações sejam executadas com entrada pela câmera *Realsense* que forneçam dados de saída confiáveis para carros trafegando na região monitorada com velocidades de até 60km/h.

Para a realização dos testes, inicialmente seguiu-se os parâmetros utilizados por [6] e [9] e, assim, foram aplicados: a câmera foi posicionada a uma distância horizontal de aproximadamente 3,5 m do ponto onde os carros transitavam no sentido da esquerda para a direita; a uma distância vertical de 1,3 m do chão; com ângulo perpendicular (90º) com o objetivo de permitirem à câmera ter campo de visão para detectar a placa para identificação de seus caracteres e de identificar a quantidade de pessoas dentro dos carros.

Conforme a necessidade de alcançar detecções para um carro trafegando com velocidades mais altas – até conseguir detecções a 60 km/h –, houve a escolha de um local específico para os testes. O ambiente escolhido foi a avenida Milton Santos da UEMA, com o posicionamento dos equipamentos em frente a esta avenida numa localidade próxima ao prédio da UEMANET. A escolha foi feita por este local ser propício às necessidades operacionais para a realização dos testes, como possibilidade de passagem de carros transitando a velocidades mais altas, segurança para o uso dos equipamentos eletrônicos e também à disponibilidade de tomada para alimentação do Mini PC que permitisse que este e os demais materiais pudessem ficar posicionados em um local estratégico para o monitoramento dos carros.

Após a realização de consecutivos testes neste local e na tentativa de obtenção de resultados para as faixas de velocidades de 20, 40 e 60 km/h, identificou-se uma série de parâmetros base para a realização de testes futuros. Conforme os resultados iam sendo progressivamente alcançados durante os testes, os valores dos parâmetros aplicados foram medidos e anotados a fim de se identificar quais os valores de referência para a realização de testes futuros e permitir a avaliação de eficiência ou necessidade de mais melhoras na utilização destes. Assim, utilizando como base as dimensões de um Ford Ka 2015, que foi repetidamente utilizado nos testes, as medidas referenciais e seus respectivos parâmetros foram de:

- 4,60 m: distância horizontal da câmera até a posição na pista que foi utilizada como referência de ponto de passagem do carro posicionando de modo que o seu meio ficasse a esta distância;

- 4 m: distância horizontal da câmera até o carro – mais especificamente da posição do dos pneus do lado direito do carro passando da esquerda para a direita – com uma tolerância para mais ou para menos de 0,1 m;
- 0,62 m: distância horizontal do pneu do carro ao meio da pista (da avenida do local referido);
- 1,30 m: distância do chão à câmera com a tolerância de 0.03m para mais ou para menos;
- 83 graus: o ângulo de visão horizontal da câmera.

As Figuras 30, 31 e 32 representam os parâmetros de referência utilizados nos testes em campo na Universidade Estadual do Maranhão:

Figura 30: Distâncias da câmera para testes com entrada de dados ao vivo pela *Realsense*

Fonte: do Autor

Figura 31: Posição da câmera com angulação a 90º

Fonte: do Autor

Figura 32: Posição da câmera a 83º para captura de placas

Fonte: do Autor

4. RESULTADOS E DISCUSSÃO

Nesta seção serão relatadas as atividades realizadas e apresentados os resultados alcançados por meio delas, embasadas no processo da Gestão do Projeto URCA explicado por meio das etapas tratadas nas seções anteriores, todas fundamentais para a aplicação dos testes descritos a seguir, que têm como finalidade obter os resultados desejados que viabilizem a implantação da solução URCA nas avenidas.

Nos testes realizados para obtenção dos resultados discutidos a seguir, foram utilizados os algoritmos apresentados em [6] e [9] e os materiais e métodos especificados na seção 3. Os resultados serão apresentados em dois momentos: no primeiro, com aqueles alcançados através da aplicação de testes de campo cuja entrada do programa para extração das informações foi o fluxo de quadros contínuo da câmera *Realsense*, com a obtenção das saídas logo após o processamento, feito assim que os carros passavam por esta câmera; já no segundo, com o mesmo programa, só que desta vez fazendo o processamento da entrada de dados não mais em ao vivo, mas com entrada de vídeos gravados, registrados em outro momento através da *Realsense* com os mesmos parâmetros da câmera e do ambiente de testes para carros transitando com as três faixas de velocidade. Tal divisão foi feita a fim de avaliar quais formas de aplicação da solução foram capazes de capturar e extrair os dados de placa e quantidade de passageiros dos carros às velocidades especificadas de 20, 40 e 60 km/h.

Vale ressaltar que os testes a seguir foram feitos de forma a simular o ambiente de implantação, utilizando algumas adaptações que são destacadas: o foco da captura é feito para uma faixa (um sentido da via pública) com um carro transitando em somente um sentido; para que o mini PC pudesse ser utilizado para o processamento da solução, foi utilizada uma extensão elétrica para a alimentação adequada do computador; já o acesso à internet para a consulta ao ALPR foi feito através de um *hotspot* proveniente de internet móvel.

4.1 Testes de campo com entrada de fluxo de quadros contínuos da câmera *Realsense*

A seguir serão apresentados os resultados obtidos através das realizações dos testes de campo com entrada da aplicação de captura e extrações sendo o fluxo de quadros contínuos da câmera *Realsense*, no qual enquanto a filmagem dos carros era feita, o programa realizava o processamento para fornecer as respostas.

É importante ressaltar a dificuldade de realização dos testes de campo com uma maior frequência conforme fora desejada e até necessária, pois para a realização de cada um dos testes era preciso que houvesse a disponibilidade de pelo menos um carro e uma outra pessoa para

dirigi-lo (a fim verificar as respostas para as velocidades de referência), além da autora deste trabalho – que já estaria com a tarefa de manusear o equipamento –.

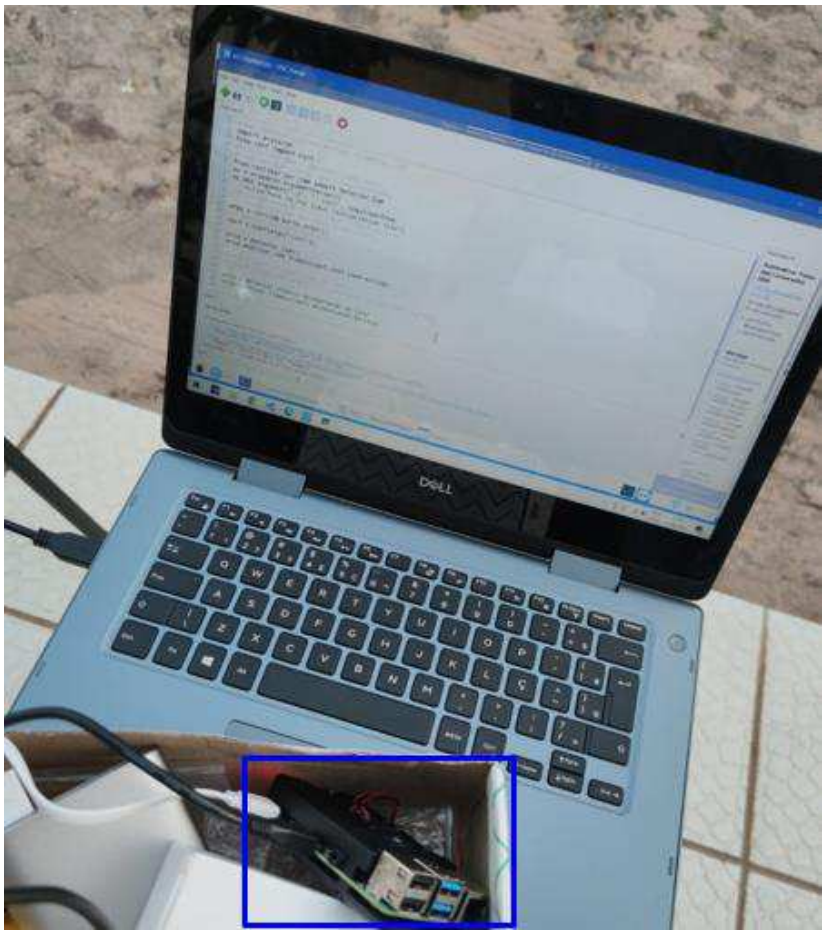
Além do fator disponibilidade, outro que também dificultou a realização dos testes de campo foi o período com forte ocorrência de chuvas, comuns nos primeiros meses do ano na cidade de São Luís, o que impedia ou forçava a suspensão da realização dos testes devido ao fato do local de teste estar molhado ou mesmo da chuva poder danificar os equipamentos eletrônicos necessários utilizados. Portanto, o período de tempo para a realização dos testes até o alcance dos resultados esperados, tornou-se dessa forma, mais extenso do que o esperado.

Os testes realizados até aqui foram, em sua maioria, em ambiente com as variáveis de velocidade controladas, através de um carro passando com velocidades conhecidas a fim de poder se avaliar a eficiência das extrações realizadas com entrada ao vivo do fluxo de quadros da câmera. Por vezes, foi necessário que o carro, ou mais de um quando havia a disponibilidade, fossem manobrados diversas vezes até que as capturas e extrações de informações fornecessem saídas que correspondessem aos dados dos carros vistos a olho nu ou até se confirmar o limite de velocidade com que cada configuração ou computador conseguiria fornecer respostas eficientes.

4.1.1. Testes com *Raspberry Pi* 4

Inicialmente, conforme a especificação técnica inicial feita em [6] o processamento da solução foi feito com a placa *Raspberry*, cujos dados técnicos foram mostrados na seção 3.1. Como ela só apresenta as interfaces para conexão com periféricos, para os testes de campo, optou-se por utilizar uma ferramenta de acesso remoto com interface gráfica chamada *Real VNC*, pela qual pode-se acessar a *Raspberry* por meio de uma versão *Viewer* instalada no Notebook do Projeto e de uma versão *Server* instalado na *Raspberry*. À *Raspberry*, foram conectadas apenas a câmera *Realsense* e uma bateria externa de um *power bank*, para alimentação da placa. As Figuras 33 e 34 mostram como foi feita a utilização da *Raspberry* nos testes de campo:

Figura 33: *Raspberry* e interface gráfica para acesso ao seu sistema



Fonte: do Autor

Figura 34: Teste de campo com entrada de filmagem ao vivo com *Raspberry*

Fonte: do Autor

- ✓ Resultados: os resultados não foram satisfatórios, visto que a velocidade máxima observada após a realização de diferentes testes com a *Raspberry* foi de que ela foi capaz de realizar detecções com entrada da *Realsense* ao vivo para até 12 km/h.

Desta forma, houve a necessidade de procurar uma nova opção com especificação técnica mais robusta para realizar a função de processamento que até então estava sendo feita com a *Raspberry*, para que pudesse permitir uma melhor resposta para extrações para velocidades mais altas.

4.1.2. Testes com Mini PC com Linux

A opção encontrada foi um Mini PC, cujas dimensões e poder de processamento atendiam ao que se necessitava até então: um equipamento com tamanho que permitisse ser encapsulado e levado à rua e com poder de processamento tal que pudesse suportar o fluxo de quadros contínuo da câmera, processando cada quadro e realizando detecções sobre estes de forma contínua e melhorando a resposta das extrações, que necessitam que este processamento sobre o fluxo de vídeo seja mais efetivo.

A Figura 35 registra um dos testes de campo realizados com o Mini PC. A configuração e posicionamento da câmera *Realsense* foram feitos de forma a seguir os parâmetros referenciados anteriormente e o acesso ao Mini PC feito através de um monitor, teclado USB, mouse USB, além da conexão com a *Realsense* e o uso das ferramentas para executar o programa com os algoritmos de detecção.

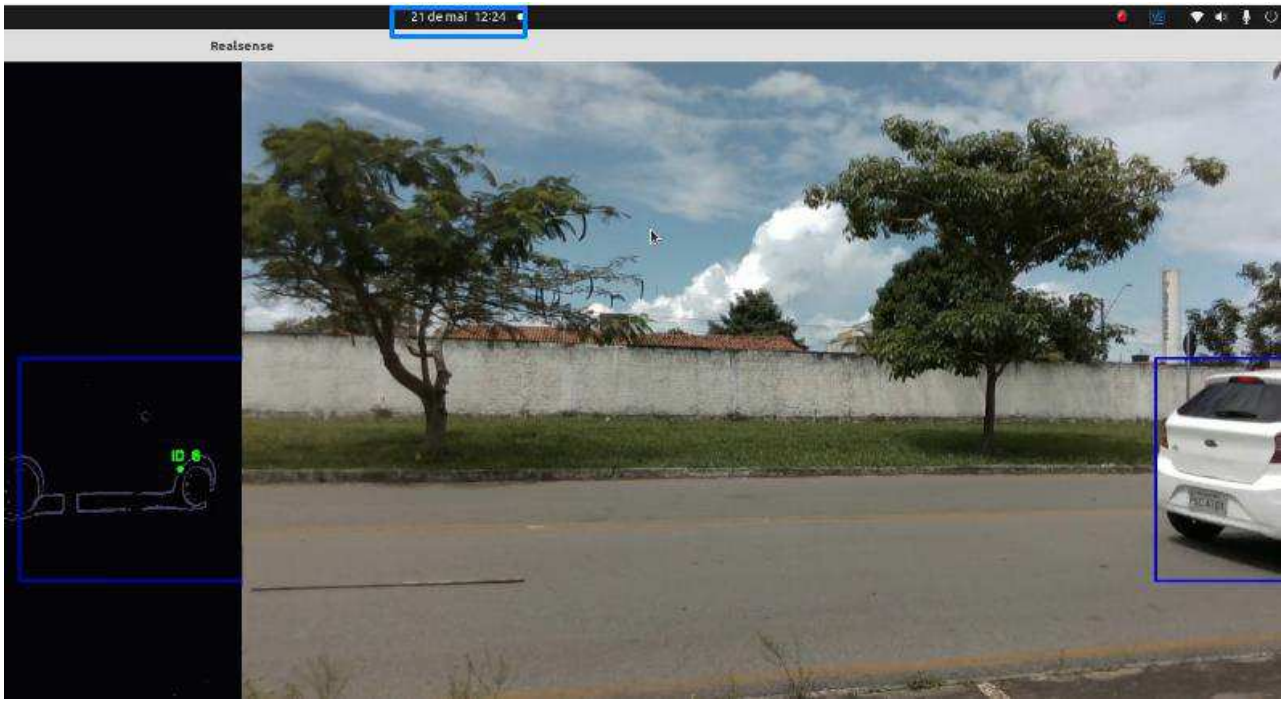
Figura 35: Teste de campo com entrada de filmagem ao vivo com Mini PC

Fonte: do Autor

Os testes de campo com o Mini PC foram realizados de forma a analisar as saídas dos algoritmos de detecção de placa e de quantidade de pessoas para dois carros seguidos passando a 60 km/h num intervalo de poucos segundos um após o outro.

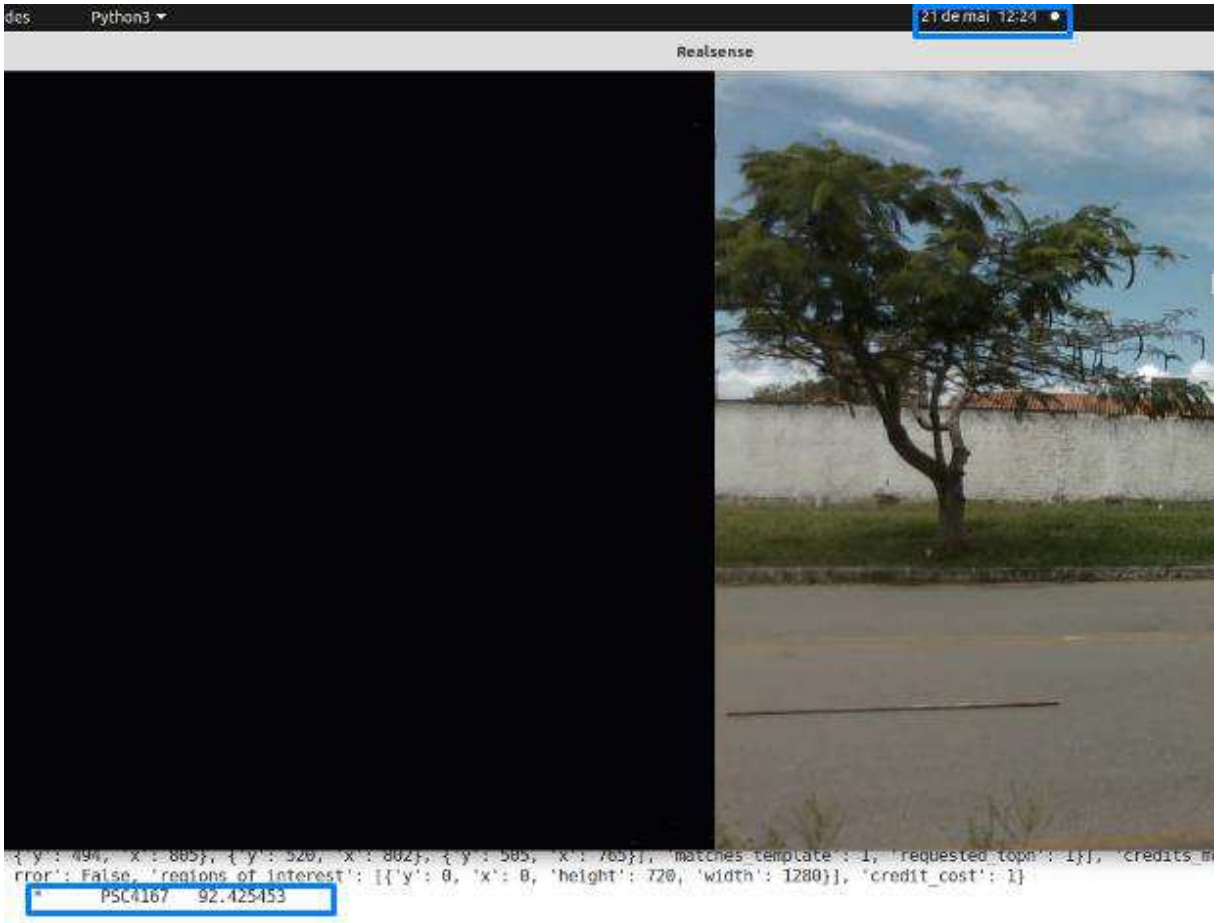
- Processamento para o 1º carro: a Figura 36 identifica um dos quadros da câmera *Realsense* em que o 1º carro (Ford Ka branco), que passa a 60 km/h, é detectado, enquanto a Figura 37 mostra o processamento da solução URCA. Observando-se a data e hora destacadas com um retângulo azul nas duas Figuras, pode-se perceber que a saída do processamento foi fornecida logo em seguida:

Figura 36: 1ª tentativa - carro branco a 60 km/h



Fonte: do Autor

Figura 37: Resultado do processamento na 1ª tentativa para o carro branco

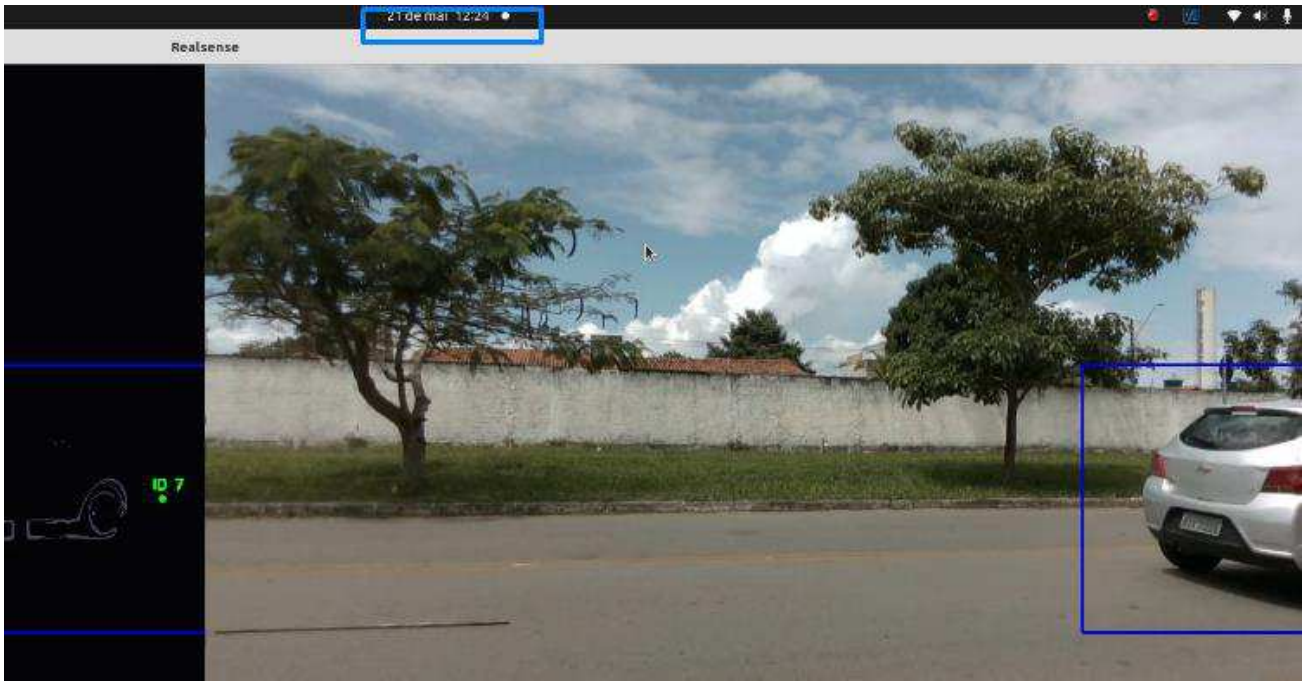


Fonte: do Autor

- ✓ Resultado: conforme mostra a Figura 37, a placa do 1º carro a 60 km/h foi lida corretamente.

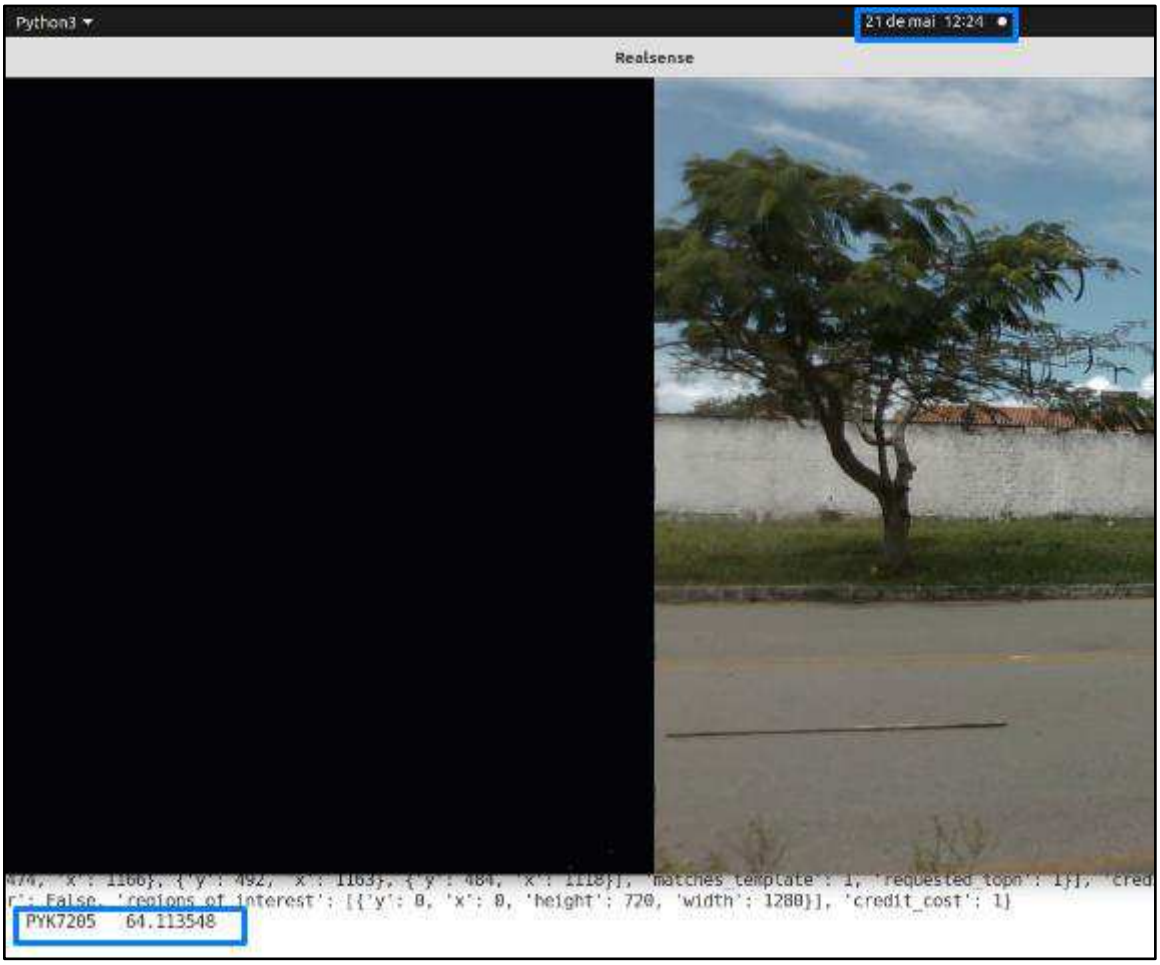
- ➢ Processamento para o 2º carro: a Figura 38 identifica um dos quadros da câmera *Realsense* em que o 2º carro (Chevrolet Onix prata), que passa a 60 km/h, é detectado, enquanto a Figura 3 mostra o processamento da solução URCA. Observando-se a data e hora destacadas com um retângulo azul nas duas Figuras, pode-se perceber que a saída do processamento foi fornecida logo em seguida:

Figura 38: 1ª tentativa - carro prata a 60 km/h



Fonte: do Autor

Figura 39: Resultado do processamento na 1ª tentativa para o carro prata



Fonte: do Autor

- ✓ Resultado: conforme mostra a Figura 39 a placa do 2º carro que passou logo após o 1º, também a 60 km/h, não foi lida de forma totalmente correta, pois houve a troca do caractere ‘T’ da placa real pelo caractere ‘Y’ feito pelo processamento do algoritmo.

Houve ainda uma 2ª tentativa nos mesmos moldes e com os mesmos parâmetros da 1ª, e os resultados observados foram:

- ➢ Processamento na 2ª tentativa para o 1º carro: a Figura 40 identifica um dos quadros da câmera *Realsense* em que o 1º carro (Ford Ka branco), que passa a 60 km/h, é detectado:

Figura 40: 2ª tentativa - carro branco a 60 km/h

Fonte: do Autor

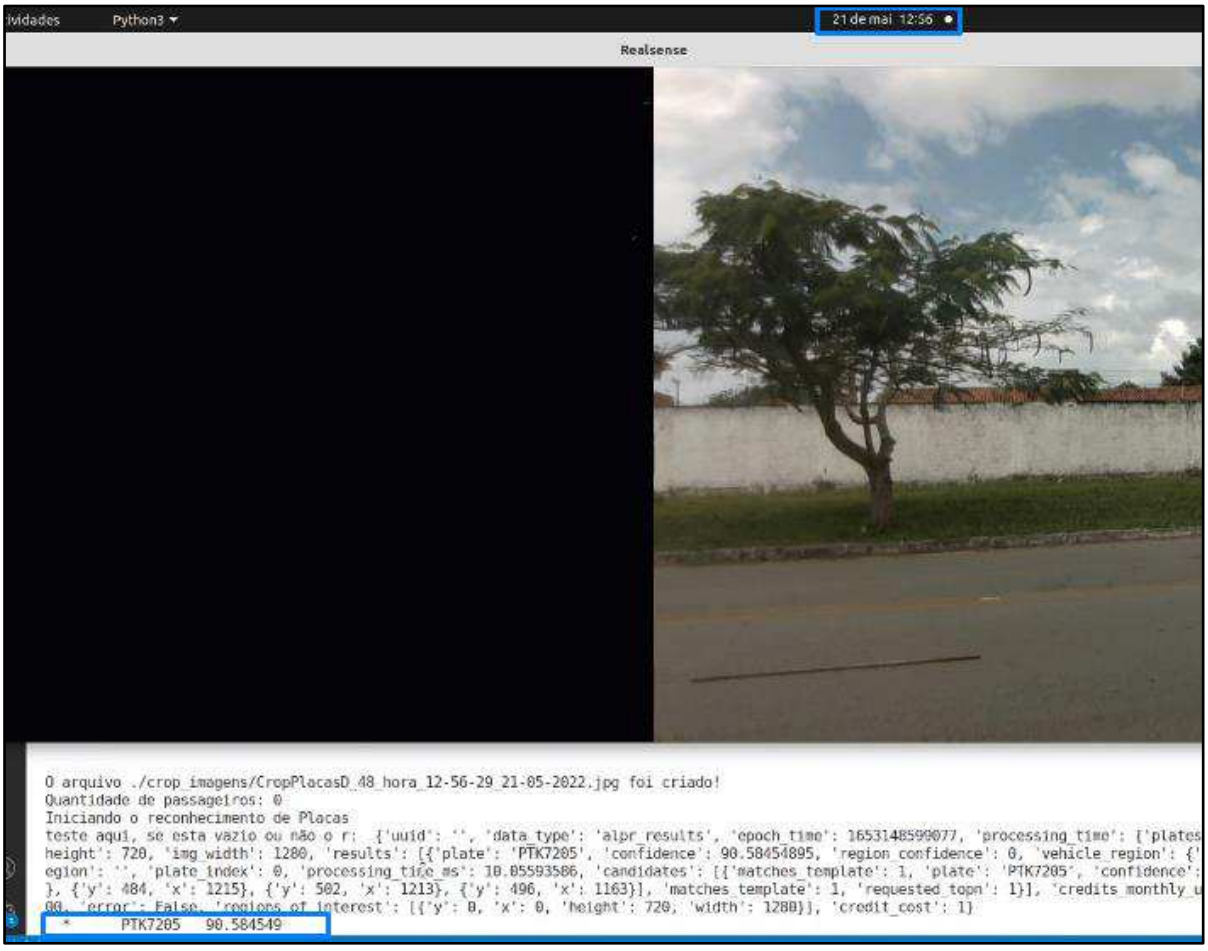
- ✓ Resultado: apesar da detecção do carro, o processamento desta vez não retornou nenhuma saída, ou seja, não houve reconhecimento de placa do carro branco a 60 km/h na 2ª tentativa.

- ➢ Processamento na 2ª tentativa para o 2º carro: a Figura 41 identifica um dos quadros da câmera *Realsense* em que o 2º carro (Chevrolet Onix prata), que passa a 60 km/h, é detectado, enquanto a Figura 42 mostra o processamento da solução URCA. Observando-se a data e hora destacadas com um retângulo azul nas duas Figuras, pode-se perceber que a saída do processamento foi fornecida em logo em seguida:

Figura 41: 2ª tentativa - carro prata a 60 km/h

Fonte: do Autor

Figura 42: Resultado do processamento na 2ª tentativa para o carro prata



Fonte: do Autor

- ✓ Resultado: conforme mostra a Figura 42, a placa do 2º carro que passou logo após o 1º, também a 60 km/h, foi lida corretamente.

4.1.3. Testes com Notebook com Windows

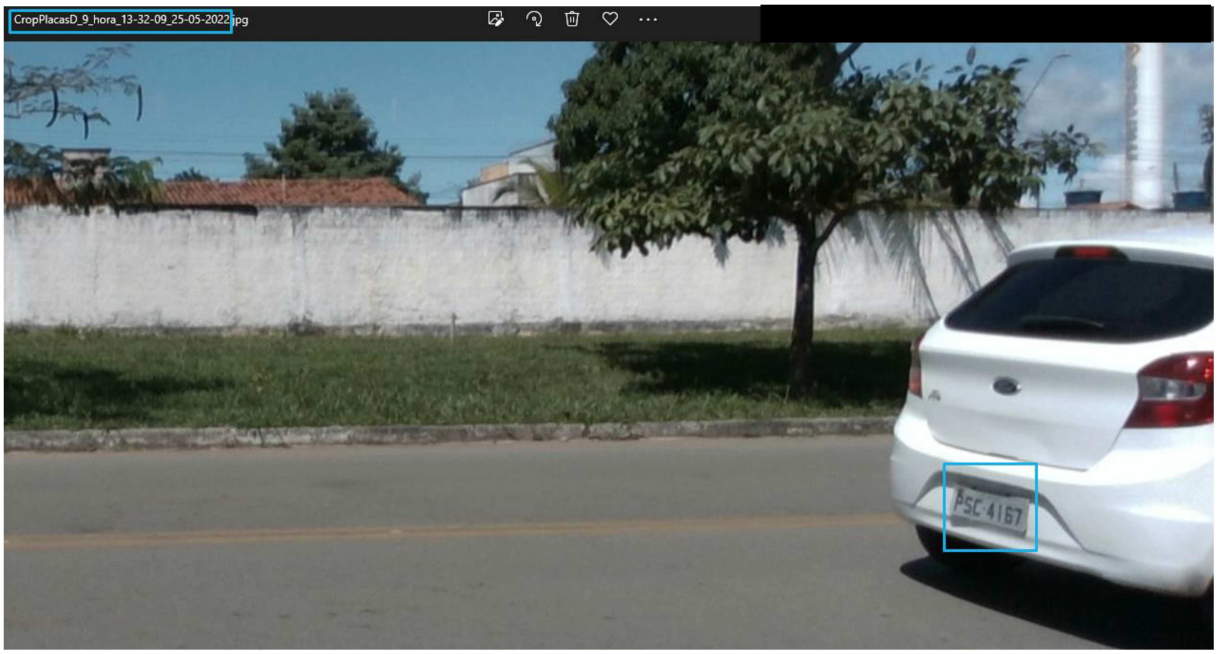
Para fins de comparação de desempenho e de nivelamento com relação ao desempenho oferecido pelo Mini PC, também foi realizado um teste de campo com processamento com entrada da *Realsense* filmando ao vivo feito através do Notebook Windows com processador Intel I7 e memória RAM de 8 GB pertencente ao Projeto URCA. A Figura 43 registra um dos testes de campo realizados com o Notebook, seguindo os mesmos parâmetros de referência utilizados nos testes com o Mini PC:

Figura 43: Testes de campo em tempo real com Notebook Windows

Fonte: do Autor

- ➢ Processamento com Notebook Windows: a Figura 44 identifica a imagem extraída pelo programa de detecções processada pelo Notebook Windows para a filmagem feita ao vivo do carro que passava a 20 km/h.

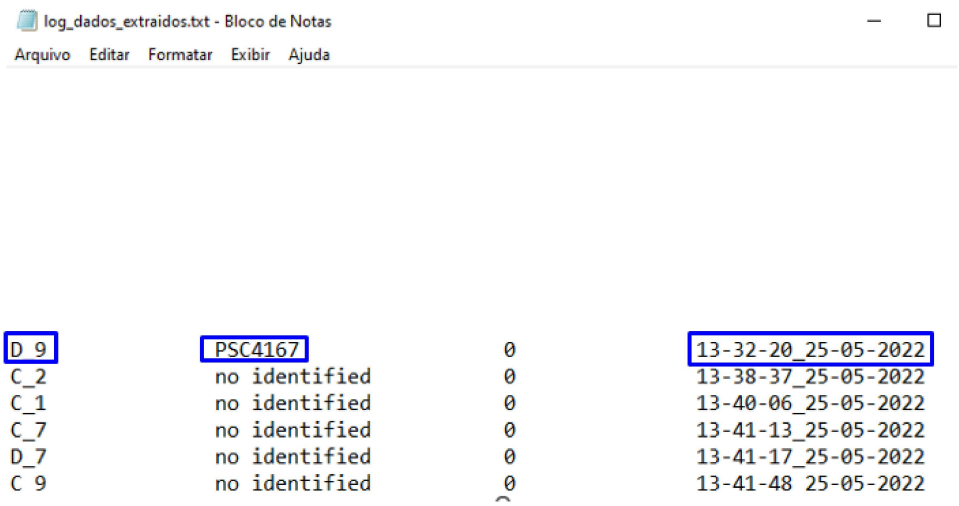
Figura 44: Imagem extraída para filmagem ao vivo a 20 km/h



Fonte: do Autor

- ✓ Resultado: conforme mostra a Figura 45, a placa do carro que passou a 20 km/h, foi lida corretamente. Por meio da Figura 45 também pode-se ver que foi gerado um log das saídas do programa, que identifica que também houve outras detecções no mesmo teste, mas sem sucesso de identificação da placa. O carro passou com outras velocidades superiores a 20 km/h no mesmo teste, mas a mesma placa não pode ser identificada através do Notebook Windows com filmagem de carros ao vivo para velocidades superiores a 20 km/h.

Figura 45: Resultado do processamento com entrada de filmagem ao vivo com Notebook Windows



Fonte: do Autor

Também pode-se observar que o processamento foi feito por meio da identificação do mesmo ID, data e hora – destacadas com um retângulo azul nas Figuras 44 e 45–, onde há uma diferença de 11 segundos na hora que, na Figura 43 corresponde ao momento da geração da imagem do quadro com o carro detectado através do *trigger* e na Figura 44 é a hora da saída da extração da placa gerada com o processamento do Notebook.

4.1.4. Testes com *Macbook* com *MacOS*

Para os mesmos fins citados na seção 4.1.3, também foi realizado um teste de campo com processamento com entrada de filmagem ao vivo da *Realsense* feito através do *Macbook Air* com sistema operacional *MacOS* que possui processador Intel I5 e memória RAM de 8 GB pertencente ao Projeto URCA. A Figura 46 registra um dos testes de campo realizados com o Notebook, seguindo os mesmos parâmetros de referência utilizados nos testes com o Mini PC:

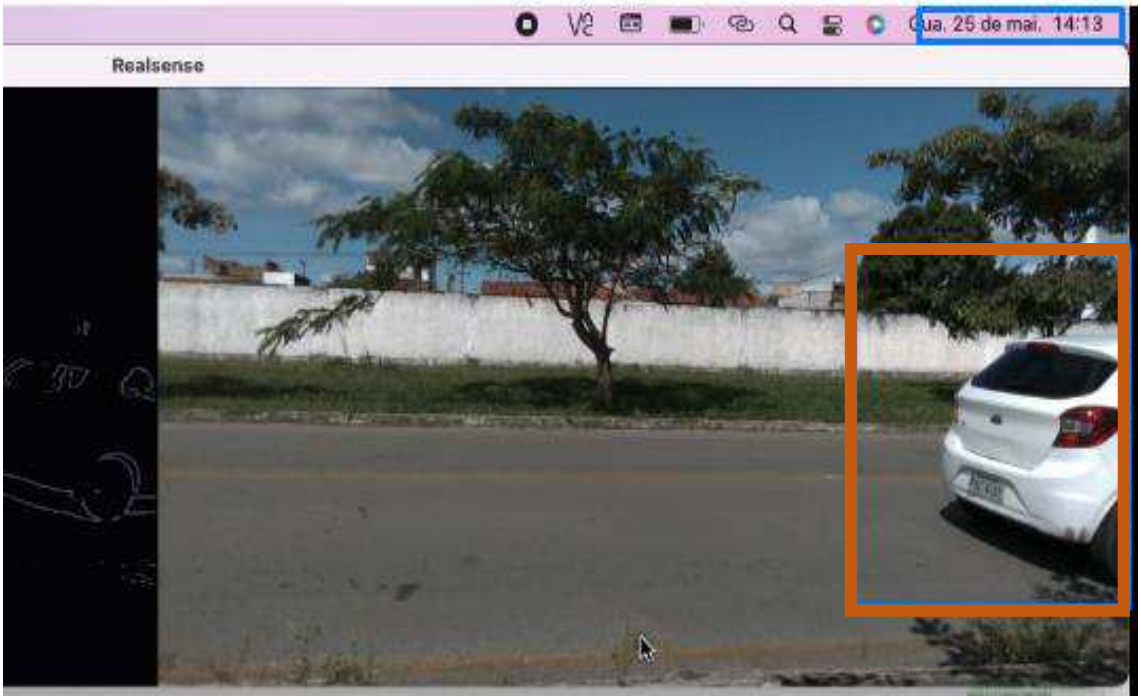
Figura 46: Testes de campo com entrada de filmagem ao vivo com *Macbook*

Fonte: do Autor

- ✓ Resultados: o teste foi realizado com o carro passando com velocidades respectivamente a 30 km/h e a 20 km/h, mas em nenhuma das vezes houve a detecção do carro ou o quadro gerado foi tardio sem apresentar imagem de carro algum e, por conta disto, não foi possível acontecer a extração nem da placa nem da quantidade de pessoas.

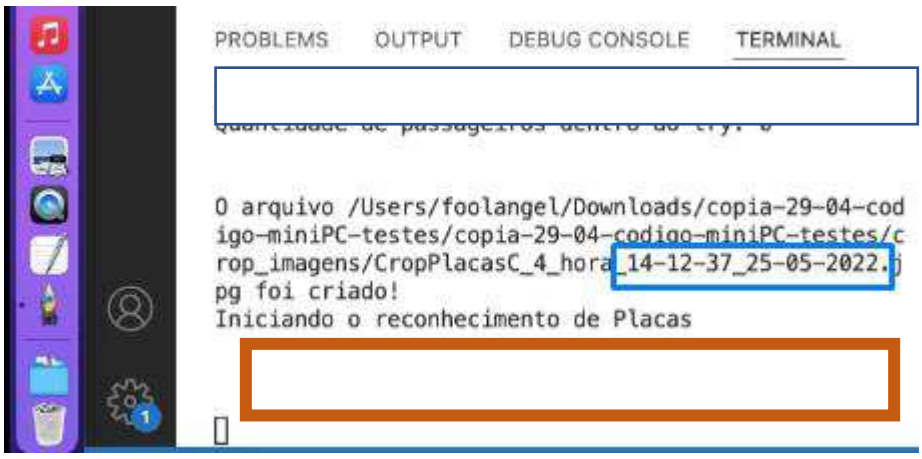
As Figuras 47 e 48 ilustram este processo, em que o destaque laranja na primeira Figura mostra uma imagem de um dos quadros do vídeo que filmava ao vivo, em que o carro passa a 20 km/h e a segunda mostra o processamento do programa por meio do *Macbook* que não obteve retorno de detecção de nenhuma placa. Já os destaques em azul nas duas Figuras identifica que o processamento foi feito poucos segundos depois.

Figura 47: Processamento em tempo real com *Macbook*



Fonte: do Autor

Figura 48: Resultado do processamento para entrada de filmagem ao vivo com *Macbook*



Fonte: do Autor

4.2 Testes com entrada de vídeo gravado

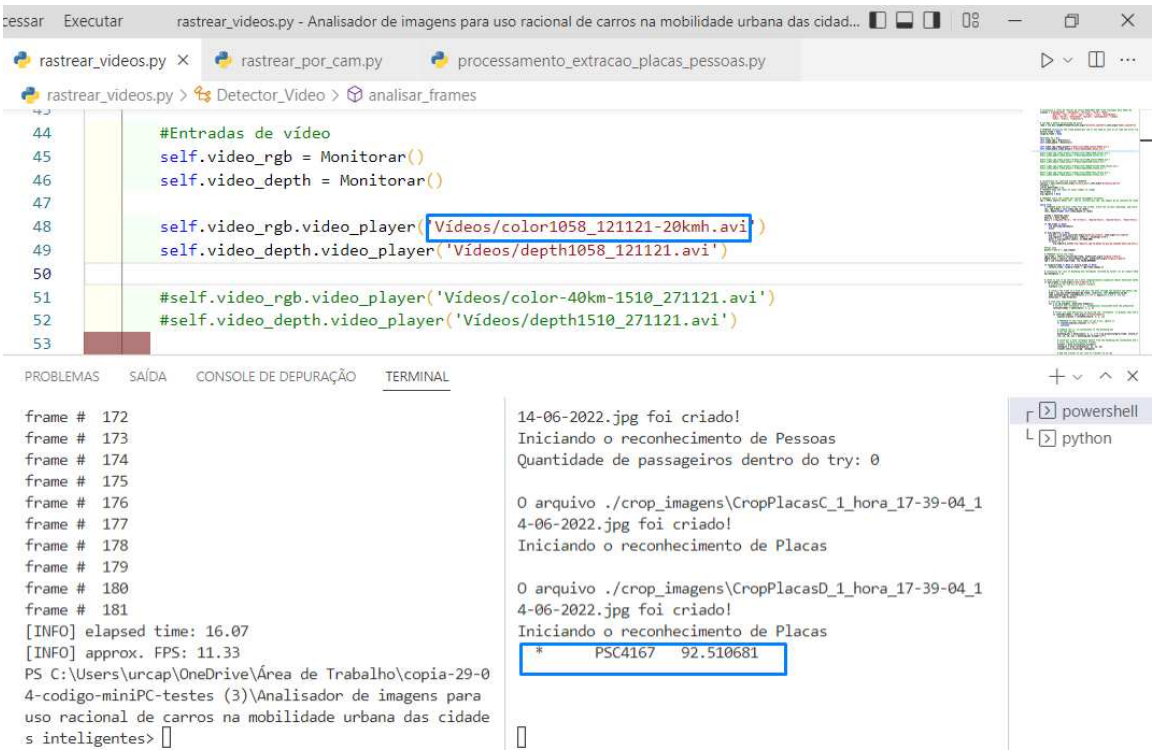
Como pôde ser visto, o processamento para atingir os requisitos da Gestão do Projeto URCA para detecção de placas e quantidade de pessoas a velocidades mais altas utilizando a filmagem da câmera ao vivo dos carros é muito alto. Como uma forma alternativa de alcançar resultados eficientes priorizando alcançá-los para carros trafegando a velocidades de até 60 km/h nas avenidas, considerou-se uma forma alternativa de processamento, desta vez, não com resposta sequencial ao momento da passagem do carro, mas fazendo a troca da entrada do programa que tinha o fluxo da câmera *Realsense* ao vivo, por uma entrada agora com vídeos gravados com a mesma câmera, mas de forma assíncrona, com estas gravações feitas nos mesmos moldes dos testes de filmagem ao vivo para as velocidades de 20, 40 e 60 km/h.

Tais testes foram feitos com os dois Notebooks do Projeto especificados na seção 3.1 e que possuem poder de processamento menor que o do Mini PC, a fim de identificar se esta forma de processamento alternativa pode ser praticada e eficiente se realizada por máquinas menos robustas. Os Resultados para cada computador serão mostrados a seguir, apresentando as saídas do processamento tendo como entrada os vídeos de gravação de carros monitorados com velocidades de 20, 40 e 60 km/h e as suas respectivas respostas das extrações de informação:

4.2.1. Testes com Notebook com Windows

- Processamento para 20 km/h: A Figura 49 mostra o processamento realizado pelo programa que contém os algoritmos no Notebook Windows, tendo como entrada um vídeo de gravação feito com testes de campo com carro passando pela área monitorada pela *Realsense* a 20 km/h:

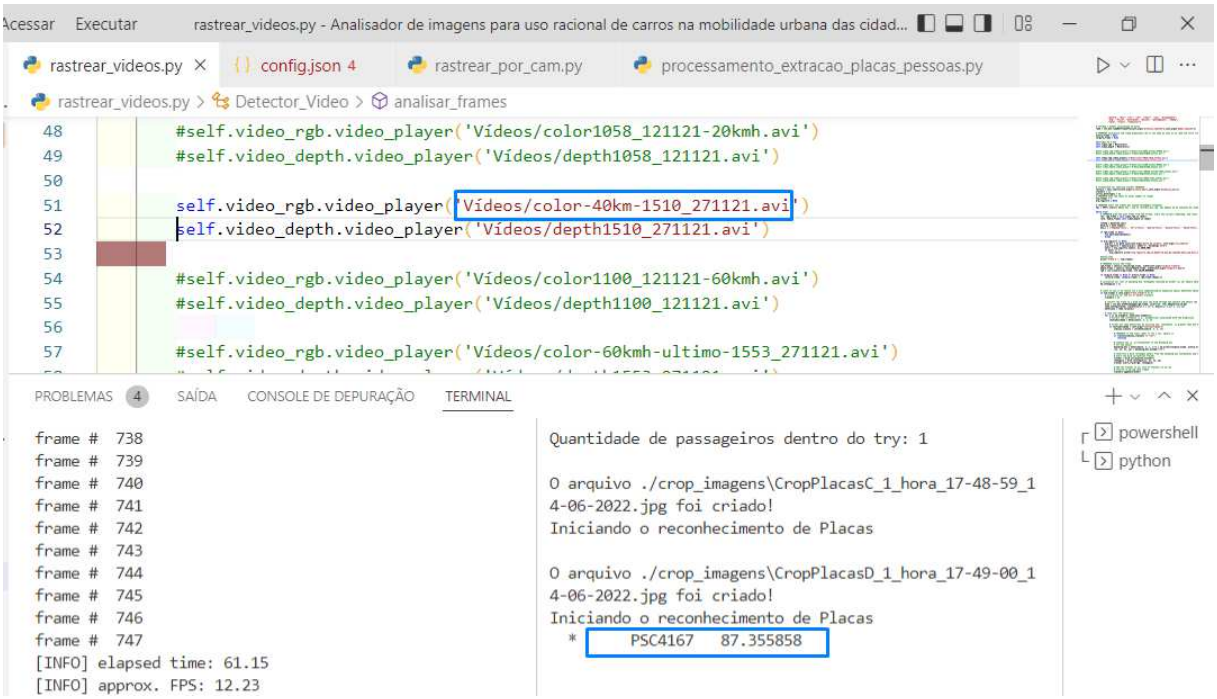
Figura 49: Processamento do Notebook Windows para entrada de vídeo de 20 km/h



Fonte: do Autor

➢ Processamento para 40 km/h: A Figura 50 mostra o processamento realizado no Notebook Windows, tendo como entrada um vídeo de gravação feito com testes de campo com carro passando pela área monitorada pela *Realsense* a 40 km/h:

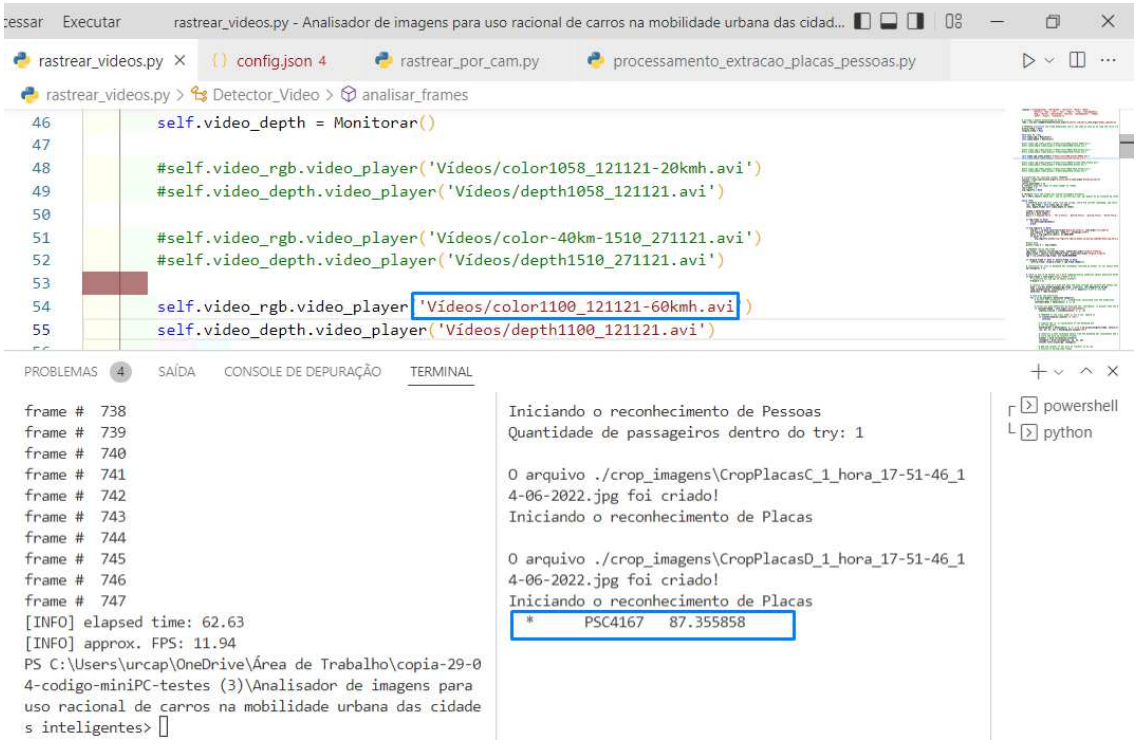
Figura 50: Processamento do Notebook Windows para entrada de vídeo de 40 km/h



Fonte: do Autor

➢ Processamento para 60 km/h: A Figura 51 mostra o processamento realizado no Notebook Windows, tendo como entrada um vídeo de gravação feito com testes de campo com carro passando pela área monitorada pela *Realsense* a 60 km/h:

Figura 51: Processamento do Notebook Windows para entrada de vídeo de 60 km/h



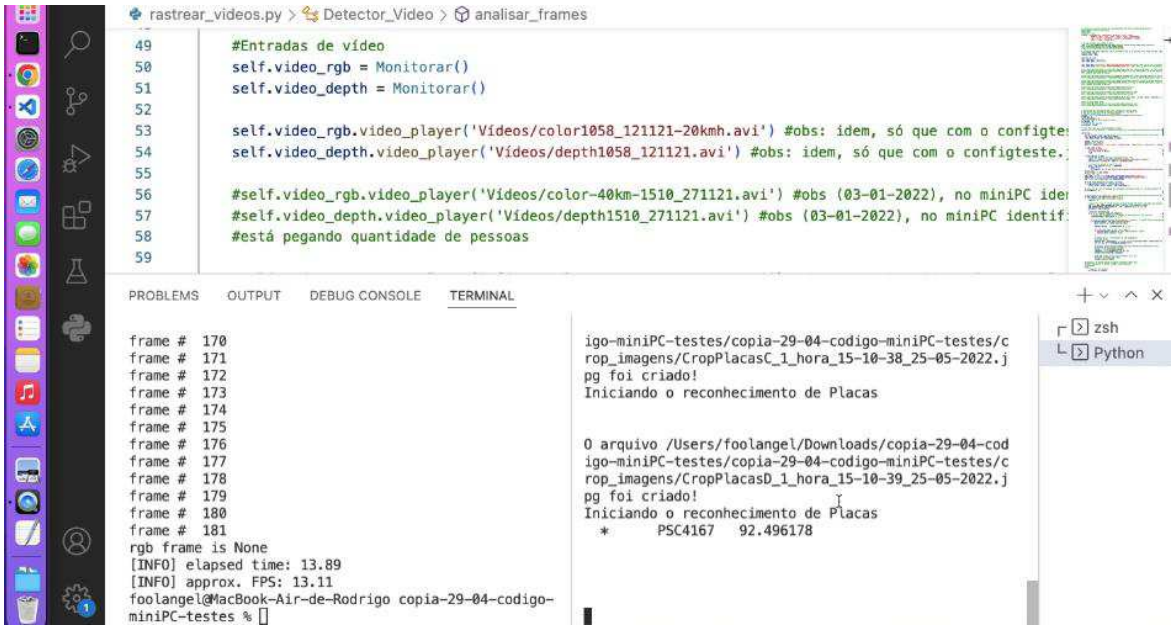
Fonte: do Autor

✓ Resultados: conforme as Figuras 49, 50 e 51, as saídas de identificação de placas para as três velocidades propostas foram feitas corretamente.

4.2.2. Testes com *Macbook* com *MacOS*

➢ Processamento para 20 km/h: A Figura 52 mostra o processamento realizado no *Macbook*, tendo como entrada um vídeo de gravação feito com testes de campo com carro passando pela área monitorada pela *Realsense* a 20 km/h:

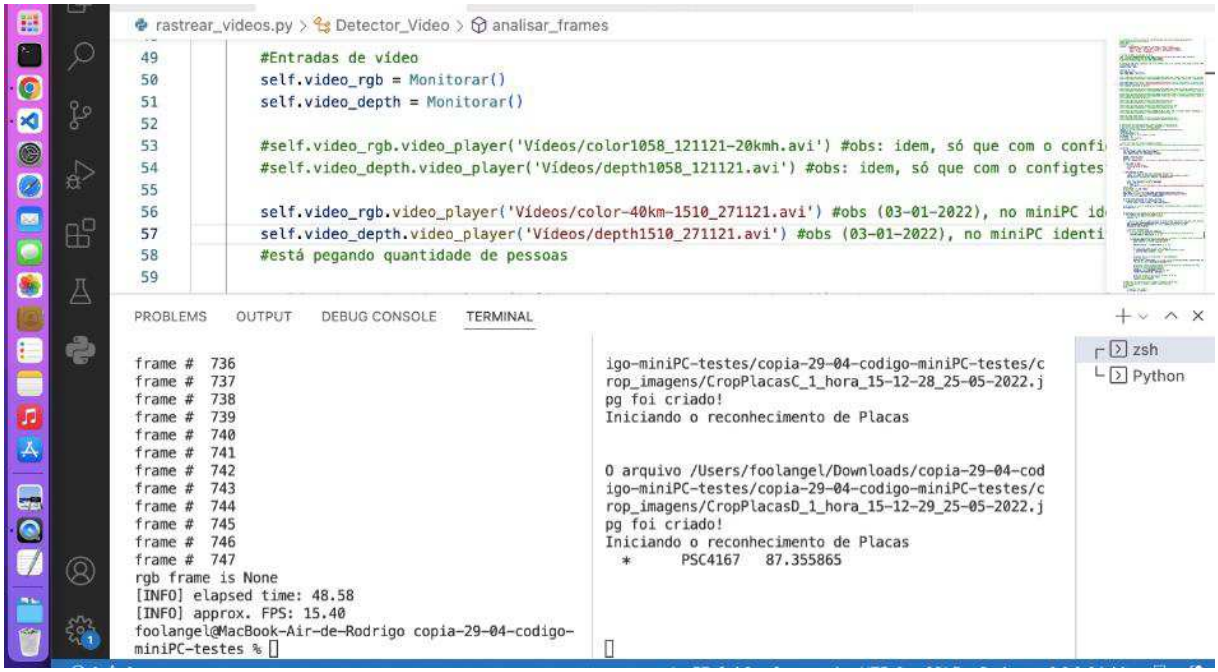
Figura 52: Processamento no *Macbook* para entrada de vídeo de 20 km/h



Fonte: do Autor

- Processamento para 40 km/h: A Figura 53 mostra o processamento realizado no *Macbook*, tendo como entrada um vídeo de gravação feito com testes de campo com carro passando pela área monitorada pela *Realsense* a 40 km/h:

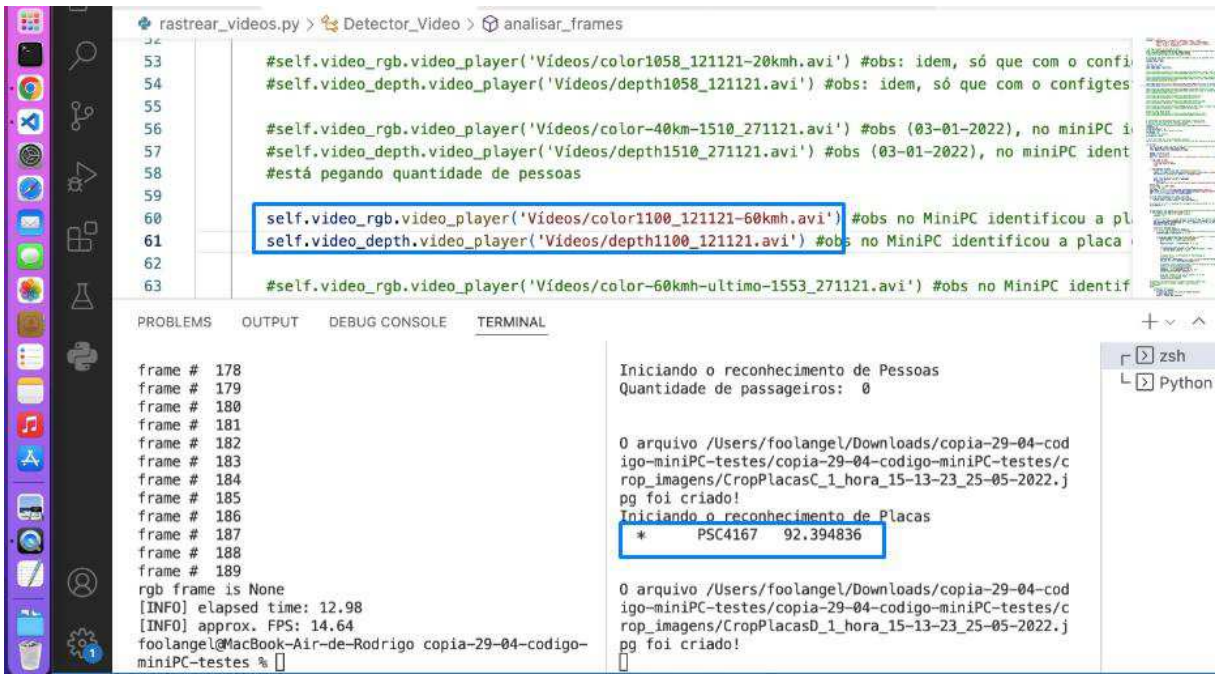
Figura 53: Processamento no *Macbook* para entrada de vídeo de 40km/h



Fonte: do Autor

➢ Processamento para 60 km/h: A Figura 54 mostra o processamento realizado no *Macbook*, tendo como entrada um vídeo de gravação feito com testes de campo com carro passando pela área monitorada pela *Realsense* a 60 km/h:

Figura 54: Processamento no *Macbook* para entrada de vídeo de 60 km/h



Fonte: do Autor

✓ Resultados: conforme as Figuras 52, 53 e 54, as saídas de identificação de placas para as três velocidades propostas foram feitas corretamente.

De acordo com os resultados apresentados através dos testes realizados e apresentados nas seções 4.1 e 4.2 foi possível identificar que o requisito a ser satisfeito para se realizar extrações de placas e quantidade de passageiros de carros transitando a altas velocidades (testadas até os 60 km/h), pôde ser feita através da solução proposta, otimizada e integrada proposta neste Trabalho.

4.3 Teste do encapsulamento mecânico

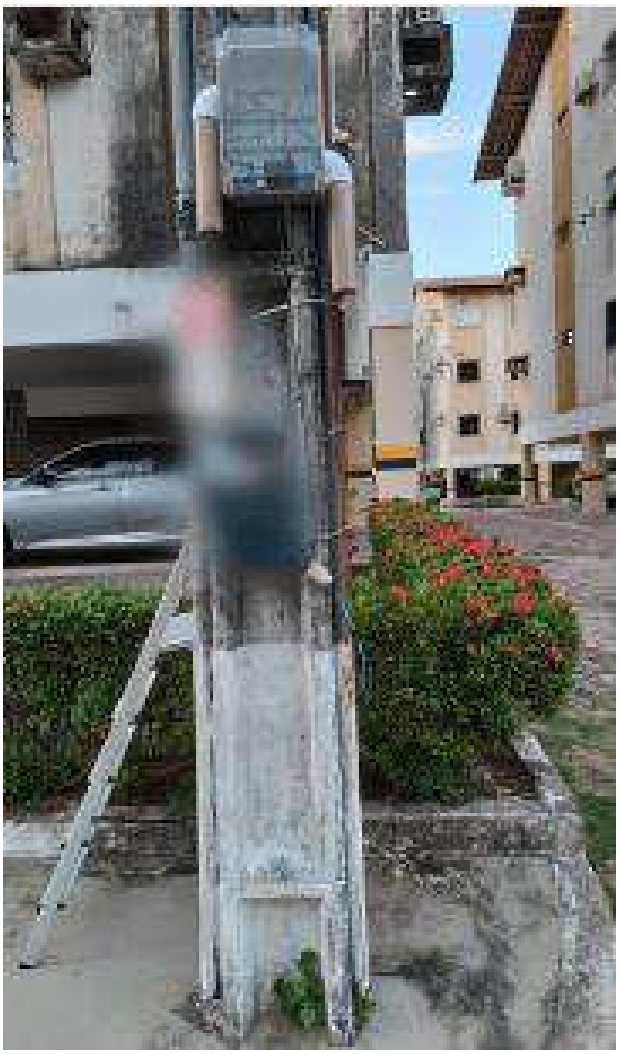
Para testar o encapsulamento mecânico planejado e implementado que fora apresentado na seção 3.3, está sendo feito um teste de durabilidade e resistência do encapsulamento e o grau de proteção que ele oferece ao equipamento da solução URCA. Para fins de teste, primeiramente está sendo utilizada a placa *Raspberry*, a qual está encapsulada pela caixa hermética e sob a ação das duas ventoinhas, com uma tendo a função de exaustor e outra a função de ventilador. A *Raspberry* está ligada 24 horas por dia por algumas semanas e o

conjunto encapsulado está preso a um poste numa localidade onde é possível permitir alimentação de energia e ter acesso a uma rede *Wifi*.

Este teste consiste em verificar a resistência e proteção oferecidos pelo encapsulamento em relação ao calor, às chuvas e também à poluição do ambiente, de forma que estes fatores não sejam capazes de prejudicar o equipamento eletrônico encapsulado.

A Figura 55 mostra o encapsulamento mecânico feito e a sua fixação no local de teste, onde ficará exposto sob a ação dos fatores ambientais e servirá para testes de durabilidade e funcionalidade dos equipamentos eletrônicos então encapsulados.

Figura 55: Fixação do encapsulamento da solução no local de teste

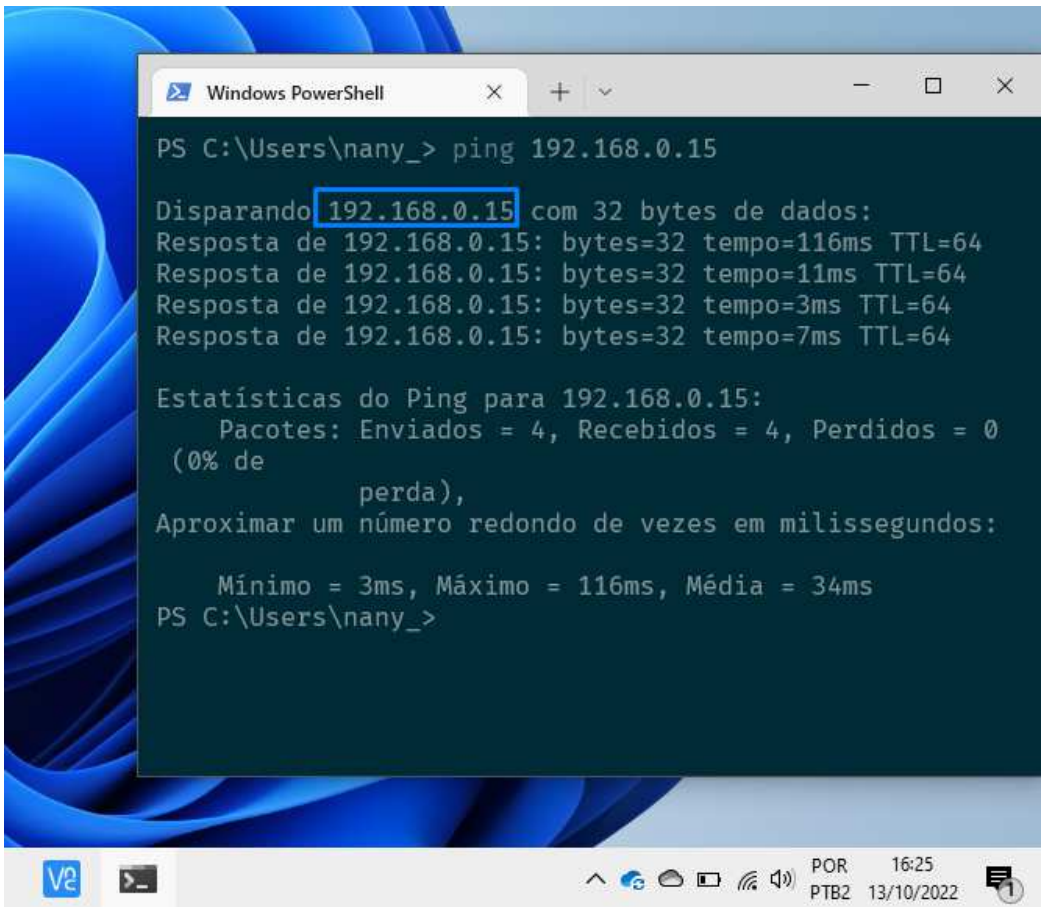


Fonte: do Autor

Para ilustrar o teste de funcionamento da *Raspberry* dentro do encapsulamento mecânico no local de teste, as Figuras 56 a 59 identificam o acesso remoto feito ao minicomputador em diferentes dias e horários, onde está sendo testada a conectividade da placa à rede – através do comando *Ping* (ao IP da *Raspberry*) – feito pelo *shell* do computador pelo qual está sendo feito

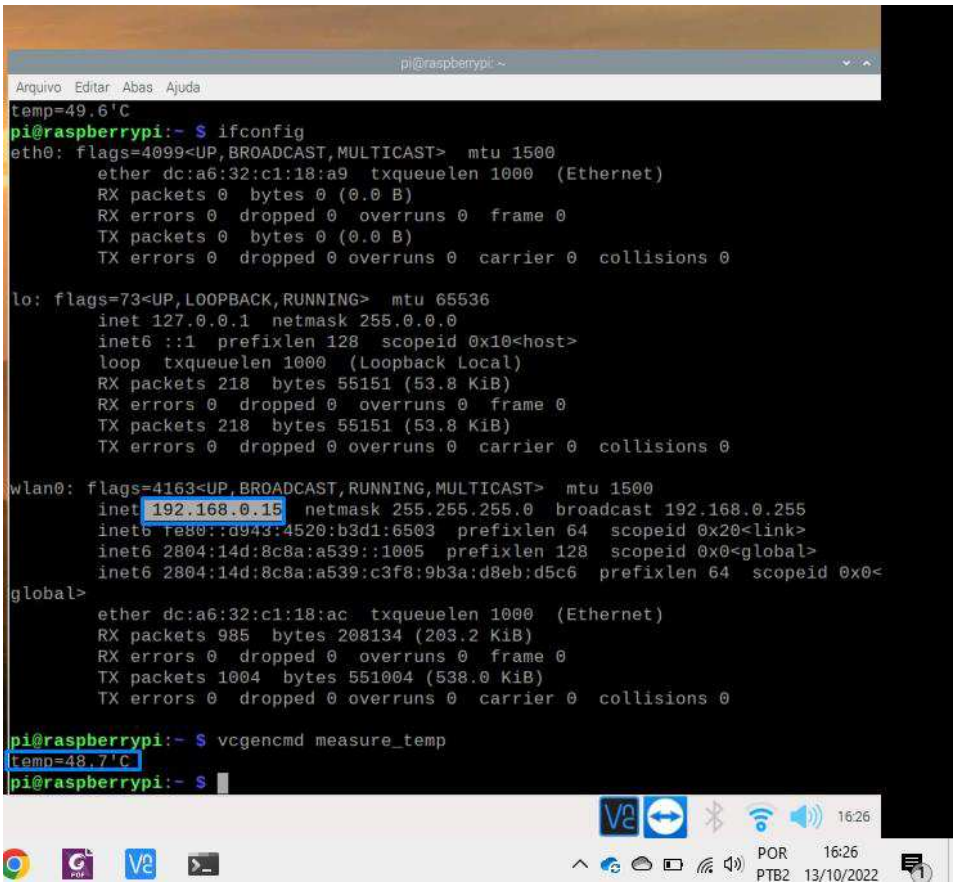
o acesso remoto. Também se verifica a temperatura em graus *Celsius* (ºC) da placa no local de teste, através do comando do sistema *Raspbian* "*vcgcmd measure_temp*", a qual atinge altas temperaturas devido ao calor do local e do próprio aquecimento do equipamento.

Figura 56: Testando conectividade do equipamento encapsulado



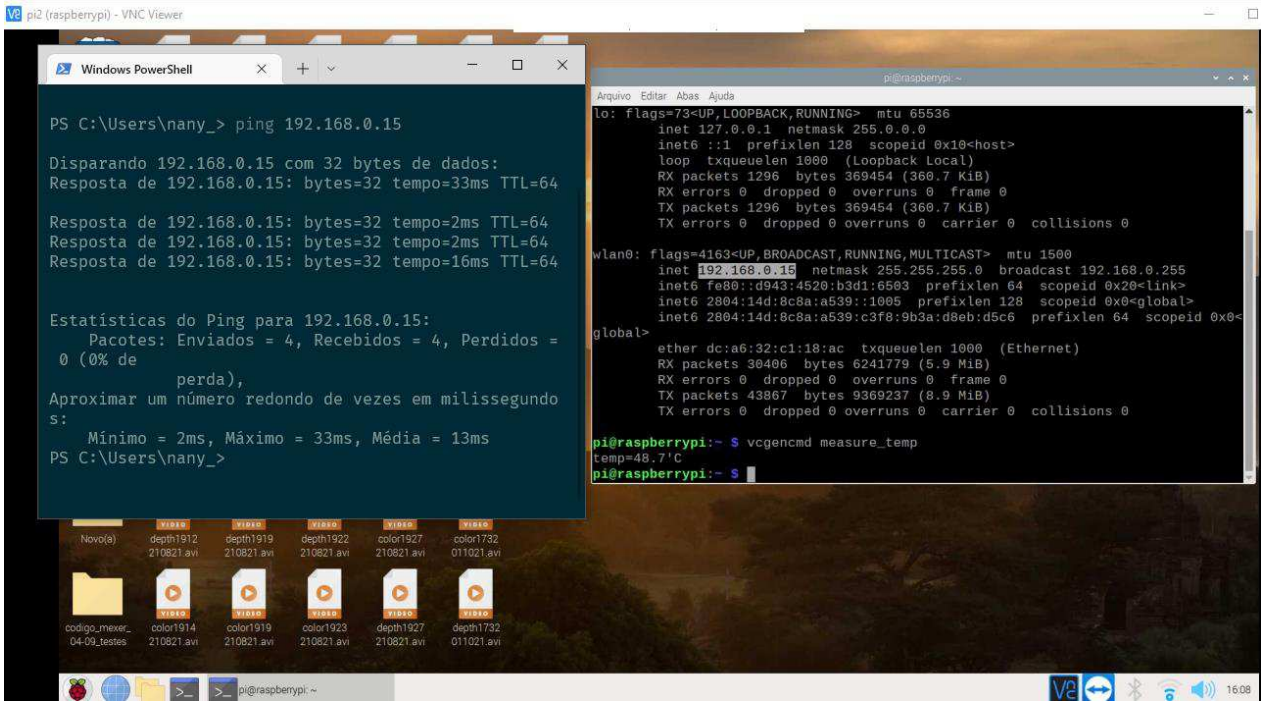
Fonte: do Autor

Figura 57: Acesso remoto ao equipamento encapsulado em funcionamento



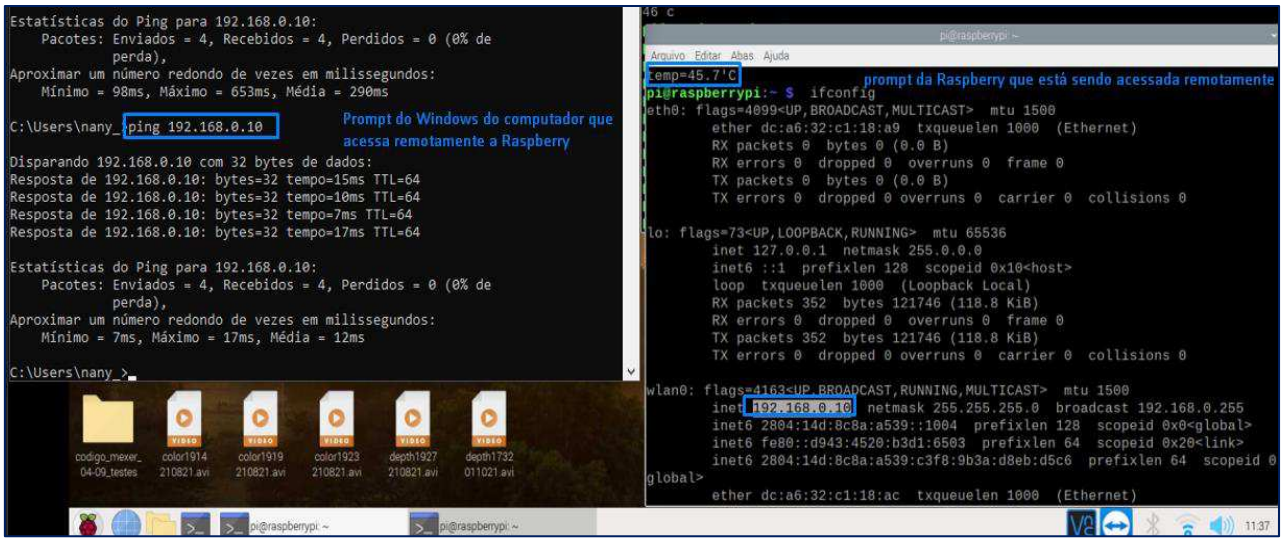
Fonte: do Autor

Figura 58: Teste de conectividade, acesso remoto e verificação de temperatura do equipamento



Fonte: do Autor

Figura 59: Teste de conectividade, acesso remoto e verificação de temperatura do equipamento em horário e dia diferentes



Fonte: do Autor

Este projeto do encapsulamento mecânico utilizando a *Raspberry* continuará sendo testado por mais dias, para verificar a durabilidade e fator de proteção do encapsulamento à exposição do equipamento eletrônico aos diferentes fatores *outdoors* adversos, para que se possa colocar futuramente o mini PC no lugar da placa menor e continuar os testes necessários com mais confiança na proteção e durabilidade entendidos através do encapsulamento projetado. Até o presente momento o encapsulamento tem-se mostrado eficiente, pois, mesmo com mais de um mês fixada em um poste, a *Raspberry* encapsulada tem conexão à rede e pode ser acessada remotamente, conforme visto nas Figuras acima.

4.4 Aplicativo URCARONA v1.1

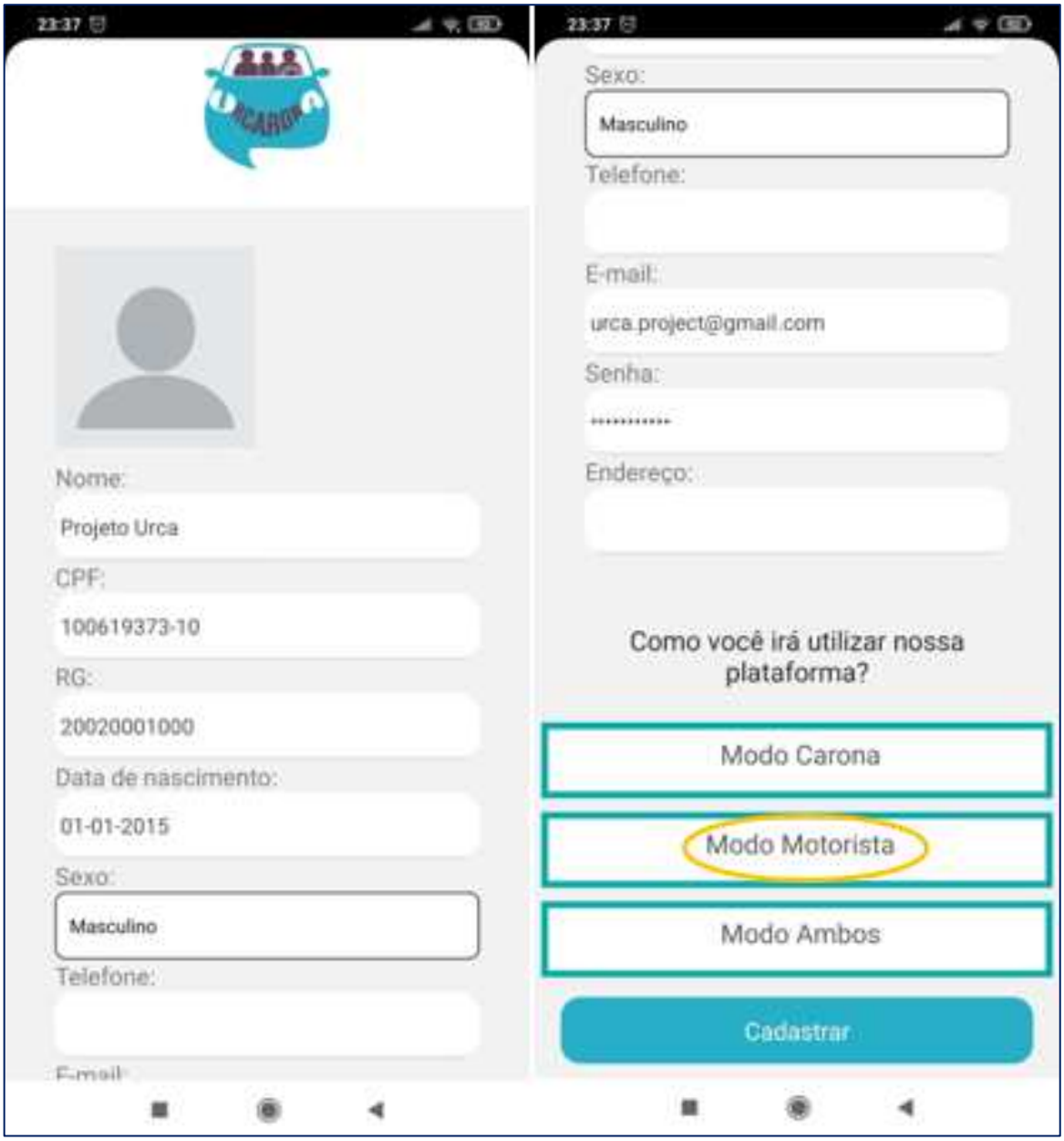
O aplicativo URCARONA encontra-se na sua fase final de desenvolvimento, na versão 1.1, o qual tem sido implementado utilizando o Framework *React Native*, que é a tecnologia responsável por gerar os aplicativos nativos tanto para a plataforma Android quanto para a Ios, através do conjunto de bibliotecas do Framework e da linguagem Javascript.

O URCARONA v1.1 está sendo desenvolvido baseado nos requisitos apresentados por [6], e também é parte integrante da solução URCA proposta. Ele servirá para conectar os usuários cadastrados como motoristas (“Modo Motorista) aos cadastrados como caroneiros (“Modo Carona”). As Figuras 60 a 71 mostram o fluxo das Telas implementadas, cujas funcionalidades podem ser entendidas pela sequência das Figuras.

Figura 60: Tela inicial do aplicativo

Fonte: do Autor

Figura 61: Tela de Cadastro de usuário com o Modo Motorista



Fonte: do Autor

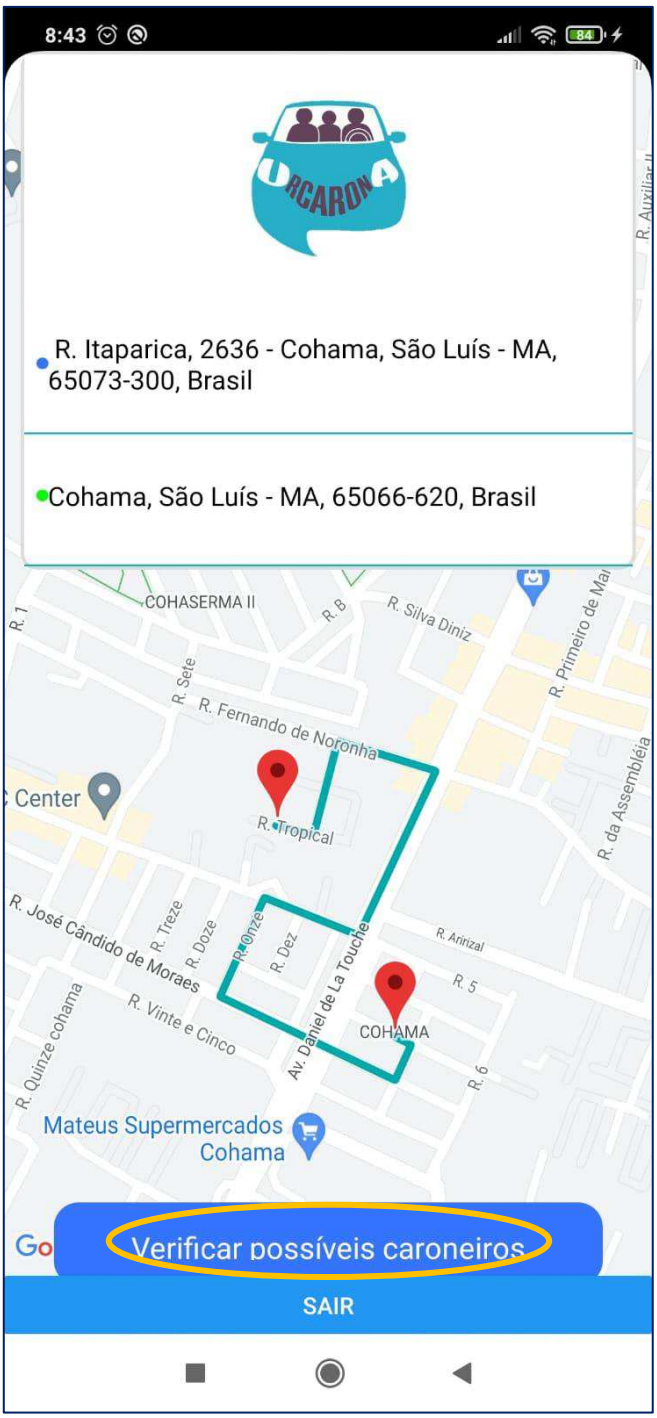
Figura 62: Tela Cadastro do carro para Modo motorista

Fonte: do Autor

Figura 63: Tela de Login com usuário cadastrado

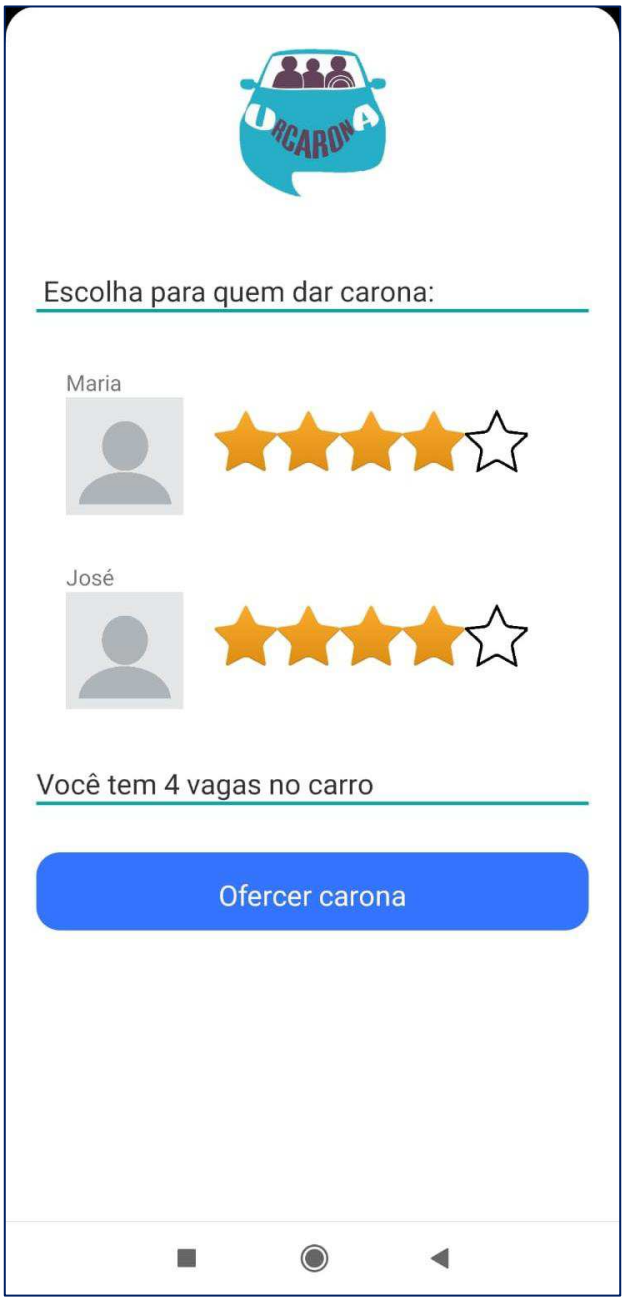
Fonte: do Autor

Figura 64: Tela Home do aplicativo para Modo Motorista



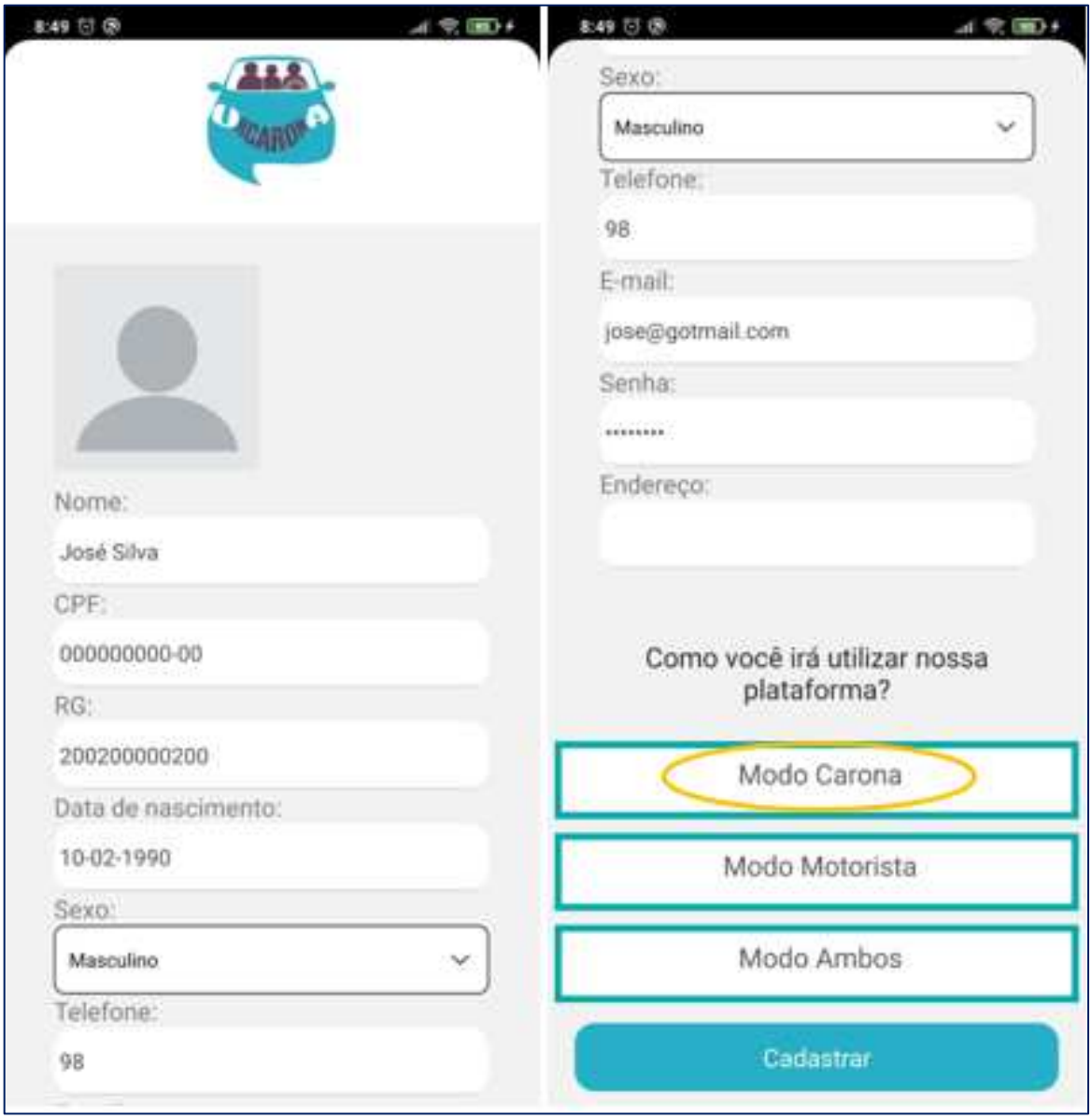
Fonte: do Autor

Figura 65: Tela para o Motorista escolher para quem dar carona



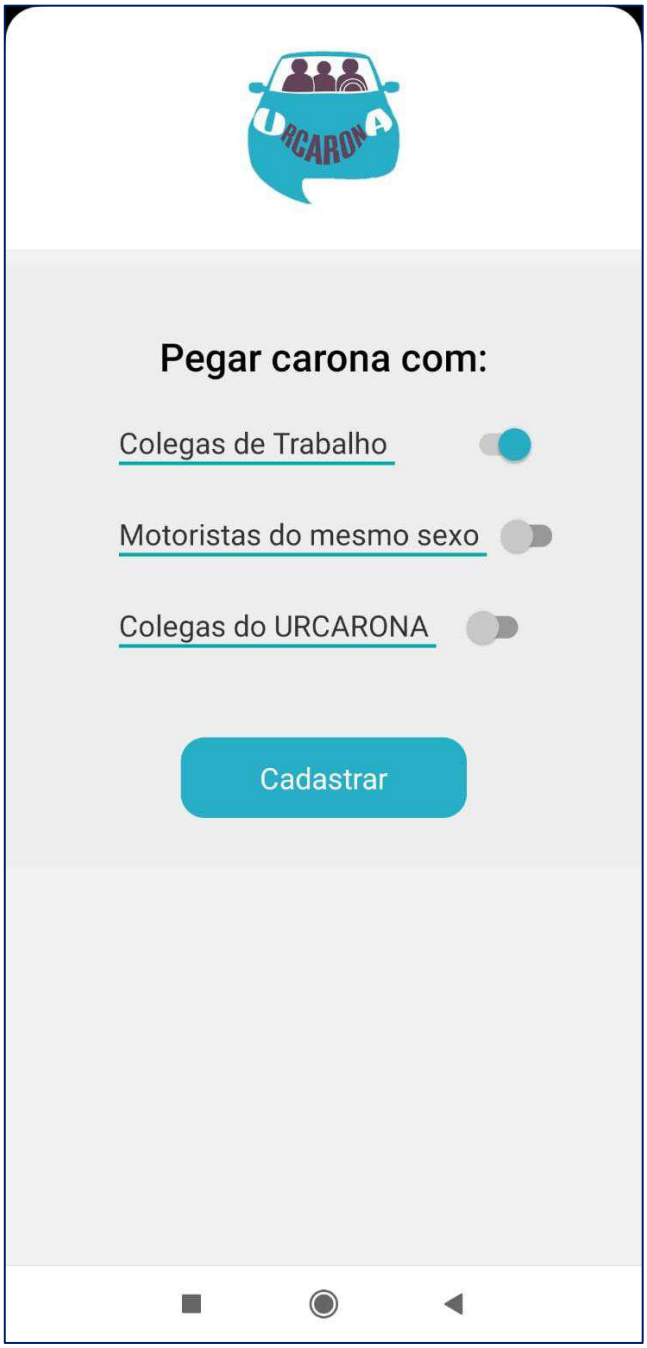
Fonte: do Autor

Figura 66: Tela de Cadastro de usuário com o Modo Carona



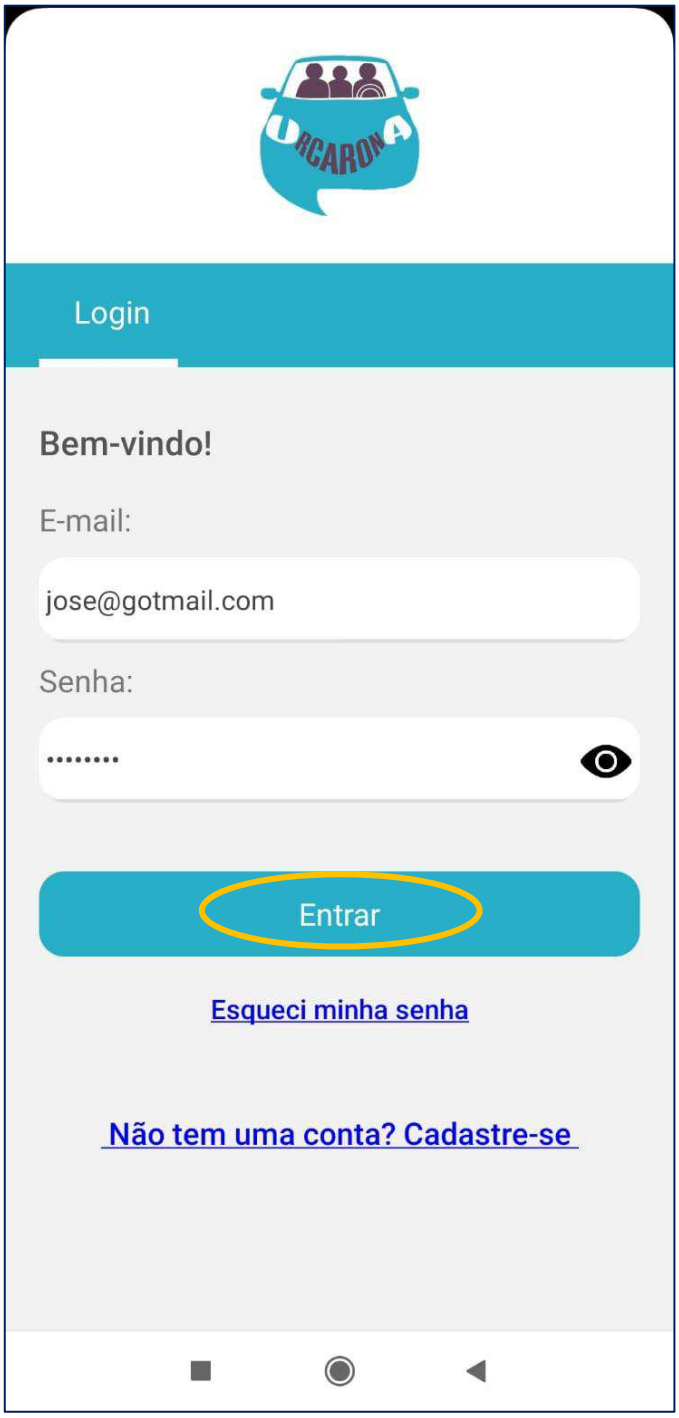
Fonte: do Autor

Figura 67: Tela Cadastro preferências para Modo Carona



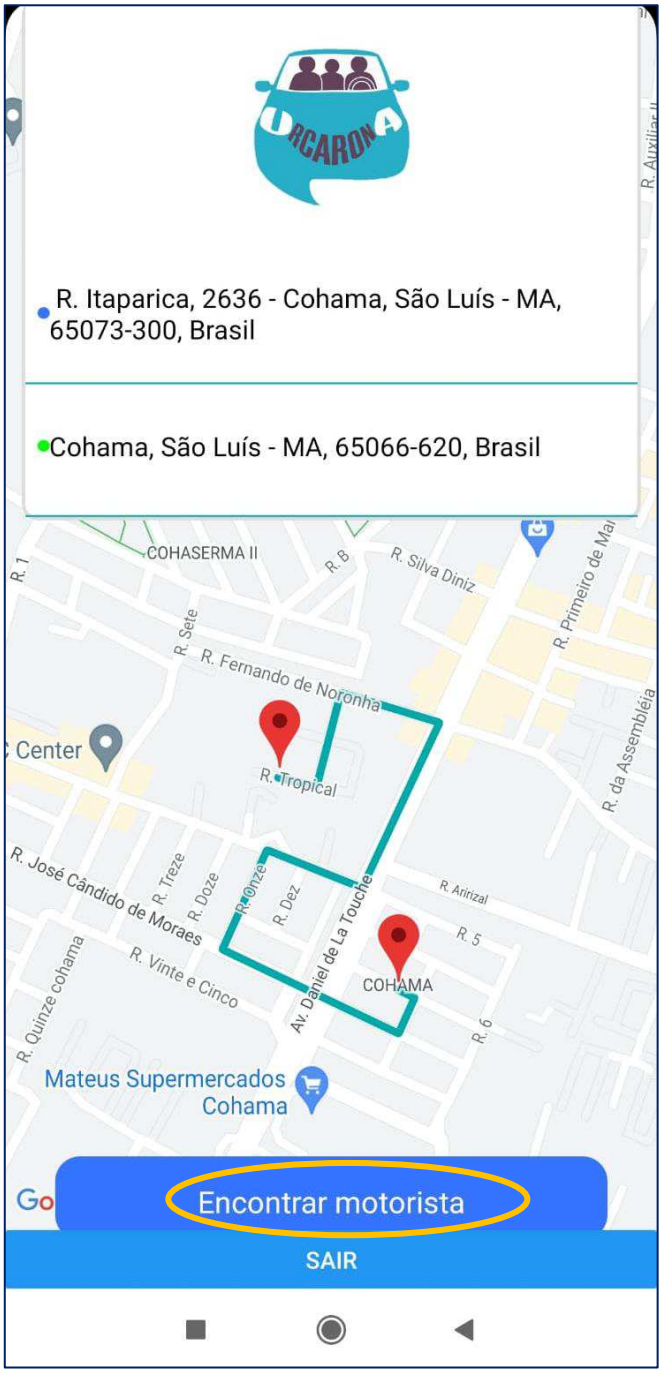
Fonte: do Autor

Figura 68: Tela de Login com usuário cadastrado no Modo Carona



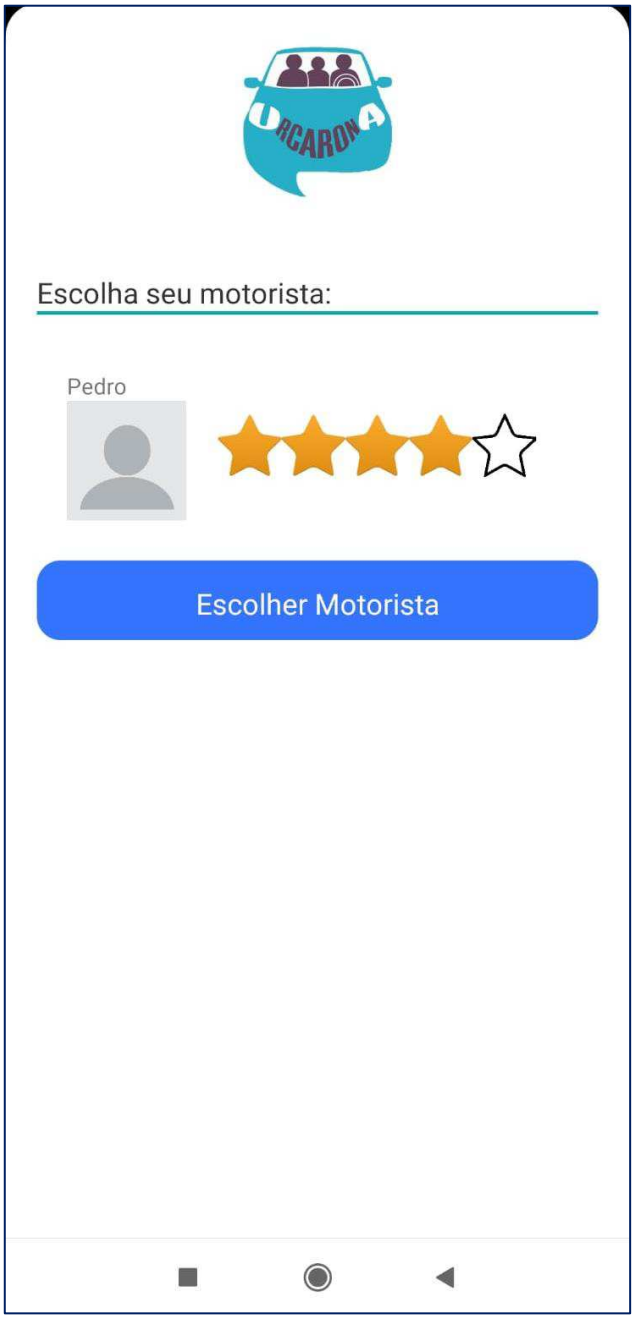
Fonte: do Autor

Figura 69: Tela Home do aplicativo para Modo Carona



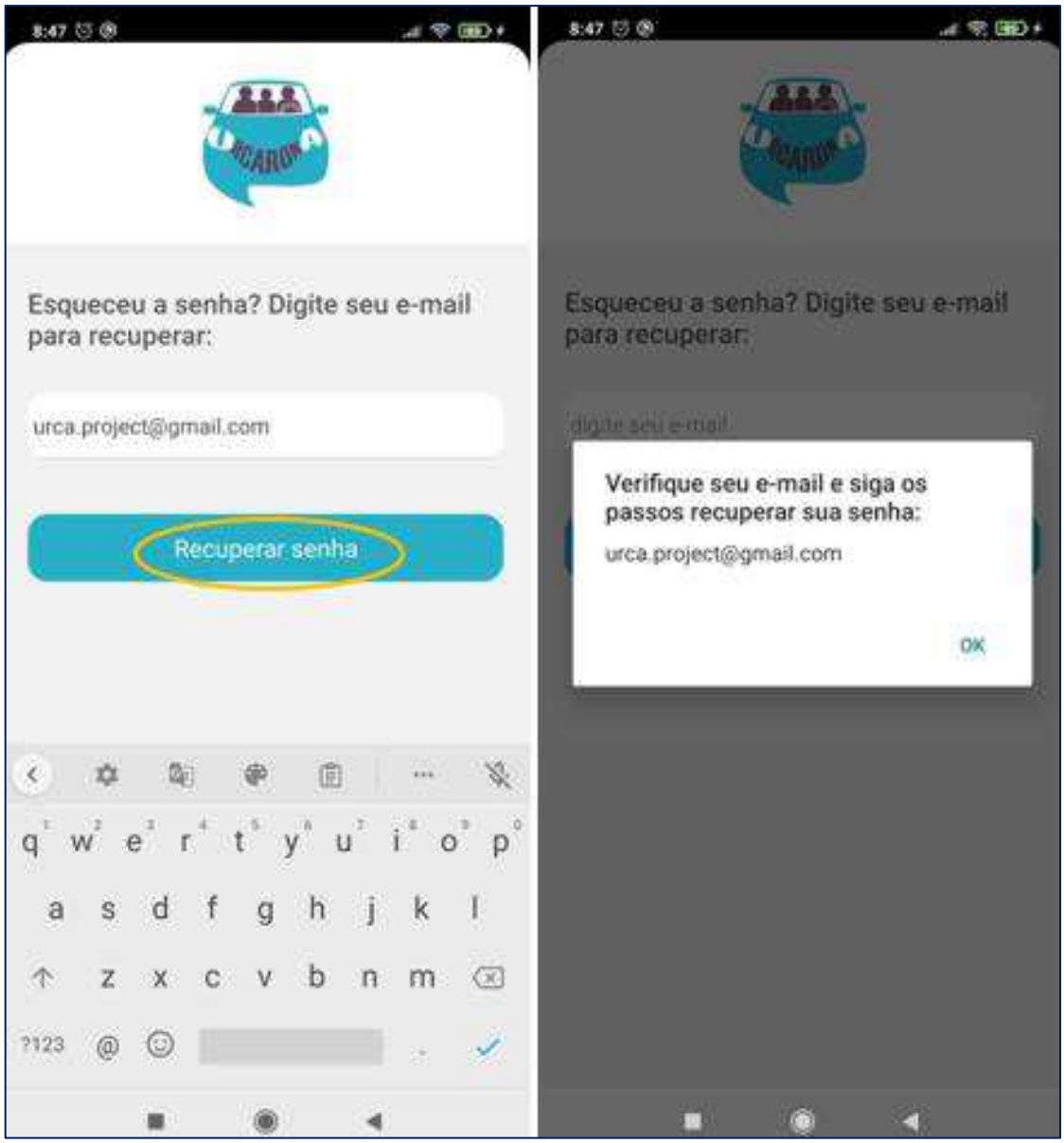
Fonte: do Autor

Figura 70: Tela para o Caroneiro escolher com quem pegar carona



Fonte: do Autor

Figura 71: Tela Esqueceu a senha



Fonte: do Autor

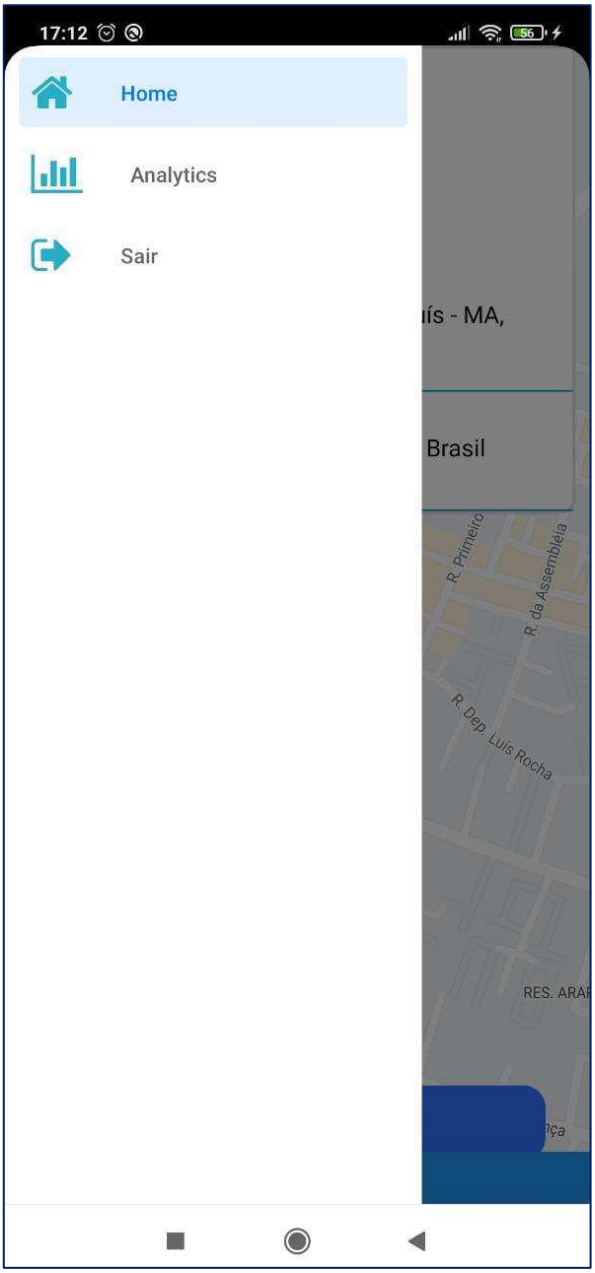
4.5 Integração do Aplicativo URCARONA ao *Analytics* do Projeto

E como mais um resultado entre os produzidos por este Trabalho, está a integração do aplicativo URCARONA ao *Analytics* do Projeto. O URCA *Analytics* é mais uma aplicação dentre as desenvolvidas nas fases anteriores do Projeto e que integra a solução URCA. Ela foi desenvolvida por [9] e foi planejada com o intuito de fornecer uma interface amigável para analisar as informações extraídas pelas aplicações de capturas e processamento que geram os dados que são armazenados no banco de dados do Projeto. Desta forma, o *Analytics* é responsável por fazer o tratamento destes dados, classificando-os de acordo com regras previamente definidas em diferentes ocorrências [9] para, assim, poder sintetizar os dados como informações úteis que demonstrem as funcionalidades das aplicações integradas da solução proposta.

Por fim, para que essas informações se tornassem de mais fácil acesso, uma vez que elas ficavam disponíveis somente pelo acesso à máquina com o banco de dados, foi feita a integração desta base de dados ao aplicativo, apresentando-os também de forma visual em uma das Telas do *app*, a fim de que o acesso seja disponibilizado de maneira mais fácil, para quem vier a ser usuário do aplicativo URCARONA.

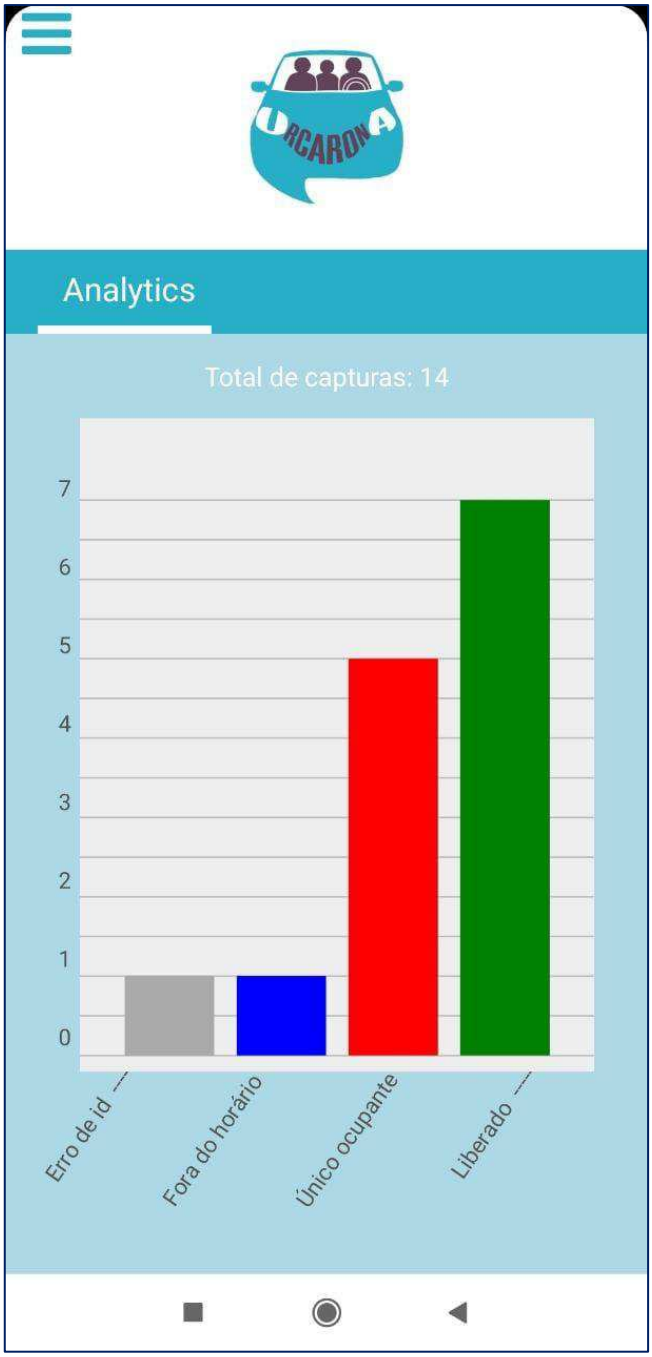
As Figuras 72 e 73 mostram a integração do URCA *Analytics* ao aplicativo, onde a primeira mostra a disponibilização da funcionalidade a partir da Tela Home, e a segunda mostra a classificação das capturas por ocorrências, conforme planejado em [9] e mostrado sucintamente neste Trabalho na seção 2.6.

Figura 72: Acesso à Tela do *Analytics*



Fonte: do Autor

Figura 73: Tela com gráficos do *Analytics*



Fonte: do Autor

5. CONCLUSÕES

Por fim, após a conclusão de todas as etapas anteriores que, percorridas uma a uma, formam o processo de efetivação deste Trabalho e através dos Resultados apresentados na seção anterior, pôde-se verificar que os objetivos propostos foram alcançados.

Neste Trabalho, foi feita a integração das diferentes aplicações desenvolvidas por [6], [7], [8] e [9], que se apresentavam de forma separada até então: através do encapsulamento mecânico planejado e montado (apresentado na seção 4.3), que, de forma compacta, fornece proteção aos equipamentos e materiais especificados para poderem executar os programas no ambiente *outdoor* que, por sua vez, tem os equipamentos protegidos rodando as aplicações com o processamento otimizado – desenvolvido nesta Pesquisa, apresentado na seção 3.6 – e cujos dados gerados, por sua vez, poderão estar ao alcance dos usuários que se conectarem ao aplicativo URCARONA que está integrado aos dados do URCA *Analytics* e cuja integração também foi realizada neste processo de Gestão (apresentada na seção 4.4),.

Desta forma, com relação aos requisitos de extração para diferentes velocidades levantados na seção 2.7, pôde-se concluir que os testes realizados com o processamento otimizado e o equipamento especificado tornaram possível a obtenção de extração de placas e quantidade de passageiros para carros transitando a altas velocidades (considerando até 60 km/h), tendo o processamento realizado pelo Mini PC como o único capaz de poder gerar as saídas e extração de informações desejadas, realizando-se os testes de campo com a entrada de vídeo com *frames* sendo capturados e processados ao vivo para as velocidades mais altas. Já os outros computadores (*Raspberry*, Notebook Windows e *Macbook*) – de poder de processamento menor –, foram capazes de fornecer resultados efetivos de extração de informações quando a aplicação executada neles teve a entrada por vídeo gravado, mesmo sendo de vídeos feitos com carros transitando a até 60 km/h.

Também destacam-se como Resultados os de Produção Intelectual, presentes nos Apêndices, que trazem o Registro da Marca URCARONA emitido pelo INPI sob o número de processo 917496850 (Apêndice A), cuja marca fora utilizada como logomarca do aplicativo URCARONA v1.1 implementado e também o Registro de *Software* da versão 1.1 da aplicação desenvolvida pelos integrantes do Projeto, intitulado “Analisador de imagens para uso racional de carros na mobilidade urbana das cidades inteligentes” registrado sob o número de Processo BR512022003259-7 (Apêndice B). Também será submetido o pedido de Registro de Software da versão 1.1 do aplicativo URCARONA que está em fase final de desenvolvimento.

5.1. Dificuldades encontradas

Uma das dificuldades encontradas durante a execução deste Trabalho foi a logística para realização dos testes de campo, que precisaram ser realizados diversas vezes, mas muitas vezes foram adiados ou pela existência de poucas pessoas – necessárias para manipulação do material ou para dirigir o carro com velocidade conhecida – que estivessem com disponibilidade no horário que se fazia mais propício para a realização dos testes ou mesmo pelas condições climáticas, em que as chuvas muito frequentes na região impediam a manipulação dos materiais, inclusive os eletrônicos. Também dentro da logística dos testes, esteve a dificuldade de encontrar um local seguro e ao mesmo tempo com bom movimento de carros para a aplicação dos testes.

5.2. Aprendizados

Fica o aprendizado de que o trabalho em equipe é muito importante quando se quer ir longe. Hoje, a solução apresentada pelo Projeto URCA é fruto de trabalhos realizados por diferentes pessoas cooperadoras e diferentes integrantes que, mesmo tendo efetivado suas Pesquisas, ainda contribuem com o Projeto, seja na realização de testes ou no suporte nas pesquisas por melhores soluções dentro dos desafios propostos.

Também de que as tecnologias atuais estão ao nosso alcance e têm muito a oferecer, as quais nós devemos e podemos utilizá-las e explorá-las para promover soluções diversas para o bem estar, seja uma solução que alcance uma localidade ou toda uma região, mas o potencial de inovação a ser alcançado através da Pesquisa e outras iniciativas é real – assim como o Projeto URCA que, lançando mão da Pesquisa e da absorção de novas tecnologias visa diminuir um problema atual, mesmo dentro da proposta de ser de baixo custo.

5.3. Trabalhos futuros

Algumas considerações sobre os testes de campo podem ser pontuadas como trabalhos futuros:

- Melhorar o arquivo de configuração do programa de captura focado na funcionalidade de extrair quantidade de ocupantes, parar poder gerar massa de dados nesse quesito com valores mais próximos do real quanto os resultados obtidos de extração de placas;

- Realizar testes com entrada por vídeo para monitoramento de carros aleatórios com diferentes velocidades;

- Verificar viabilidade de detecções de carros a 60 km/h cuja identificação de quantidade de pessoas e de placas possam ser conjugadas por uma única câmera utilizando um ângulo

mínimo que sirva para as duas funcionalidades. Até o momento, o ângulo da câmera para identificação de placas dos carros em altas velocidades precisa ser maior que o especificado para identificação de pessoas, o que comprometeu e diminuiu o campo de visão apropriado para captura de quantidade de passageiros (de modo transversal ao vidro lateral do carro), visto que as duas formas de capturas têm sido feitas por uma mesma câmera;

- Com relação à funcionalidade de identificação de pessoas, procurar identificar a média de transparência de películas dos carros até um valor adequado para o qual seja possível a identificação de passageiros dentro do carro pela câmera IR, bem como analisar como ficará o desempenho desta para dias com temperaturas mais altas

- Otimizar ainda mais o sistema de refrigeração do encapsulamento mecânico;

- Fazer a demonstração do protótipo em funcionamento com todas as aplicações que foram integradas neste Trabalho, para poder apresentar a solução para os órgãos reguladores de trânsito tais como a Agência Estadual de Mobilidade Urbana e Serviços Públicos (MOB-MA).

REFERÊNCIAS

[1] SINDIPEÇAS; ABIPEÇAS. **Relatório da Frota Circulante de 2022**. Disponível em:<https://www.sindipecas.org.br/sindinews/Economia/2022/RelatorioFrotaCirculante_2022.pdf >. Acessado em 12 de maio de 2022.

[2] OLIVEIRA, C. H. R.; SILVA, R. M. L.; SILVA, L. H. G. F.; ***Urban Mobility over Internet of Things to Smart Cities***. ICWN'15 - The 14th International Conference on Wireless Networks, 2015.

[3] PORTAL AUTO. **Quais são os gases emitidos pelos automóveis?** Disponível em: <http://portalauto.com.br/geral/emissao-de-gases/>. Acessado em 10 de Maio de 2022.

[4] MOBILIZE BRASIL. **Cresce o custo dos congestionamentos no Brasil**. Disponível em: <https://www.mobilize.org.br/noticias/11851/cresce-o-custo-dos-congestionamentos-no-brasil.html>. Acessado em 13 de maio de 2022.

[5] FIRJAN. **O custo dos deslocamentos nas principais áreas urbanas do Brasil**. Disponível em: <http://www.firjan.com.br/publicacoes/publicacoes-de-economia/o-custo-dos-deslocamentos-nas-principais-areas-urbanas-do-brasil-1.htm>. Acessado em 14 de Maio de 2022.

[6] COSTA, A. P. F. **URCA – Uso Racional de Carros nas Cidades Inteligentes**. Dissertação (Mestrado em Engenharia da Computação e Sistemas) – PECS, Universidade Estadual do Maranhão. São Luís, p.108. 2020.

[7] COSTA, A. P. F. **Solução tecnológica de captura e armazenamento de dados para o projeto URCA**. Monografia (Graduação em Engenharia da Computação), Universidade Estadual do Maranhão. São Luís, p.72. 2017.

[8] THOMAZ, V. F. **Solução tecnológica de reconhecimento de imagens para o projeto URCA de Uso Racional de Carros nas Cidades Inteligentes**. Monografia (Graduação em Engenharia da Computação), Universidade Estadual do Maranhão. São Luís, p.74 2017.

[9] CHAVES JR, L. C. **Implantação do Projeto URCA – Uso Racional de Carros nas Cidades Inteligentes**. Dissertação (Mestrado em Engenharia da Computação e Sistemas) – PECS, Universidade Estadual do Maranhão. São Luís, p.91. 2021.

[10] CHIMBA, D. CAMP, J. ***High Occupancy Vehicle (HOV) Detection System Testing***. Relatório Final – Universidade Estadual do Tennessee e Universidade Vanderbilt. Nashville-Tennessee-EUA. p.79. 2018.

[11] ROCHA, I. A. MEYER, M. F. BALASSIANO, M. BALASSIANO, R. **Caronaê UFRJ - unificando e ampliando as caronas na Cidade Universitária/UFRJ**. Disponível em: <https://anais.eneds.org.br/index.php/eneds/article/view/419/>. Acessado em 14 de Outubro de 2022.

[12] CRUZ, T. **Cidades Inteligentes: a Tecnologia Como Solução de Problemas Urbanos**. Disponível em: <https://www.vivadecora.com.br/pro/cidades-inteligentes/amp/>. Acessado em 18 de Maio de 2022.

[13] GREENVIEW. ***Smart Cities***. Disponível em: <https://greenviewgv.com.br/smart-cities/>. Acessado em 18 de Maio de 2022.

[14] JOÃO, B. N.; SOUZA, C. L.; SERRALVO, F. A. **Revisão sistemática de cidades inteligentes e internet das coisas como tópico de pesquisa**. Disponível em: <https://www.scielo.br/j/cebape/a/mBqjGxPSbRKPsXcS99z8LrD/?lang=pt> Acessado em 19 de Maio de 2022.

[15] UNITED NATIONS. ***World Population Prospects: The 2017 Revision.*** Disponível em: <https://www.un.org/development/desa/publications/world-population-prospects-the-2017-revision.html> Acessado em 19 de Maio de 2022.

[16] HAYKIN, Simon. **Redes Neurais: princípios e prática**; td: Paulo Martins Engel. 2ª ed. Porto Alegre: Bookman, 2001.

[17] GONÇALVES, A. R. **Redes Neurais Artificiais**. Unicamp. Faculdade de Engenharia Elétrica e de Computação.

[18] CALIS, J. O. G. **Aplicação de Redes Neurais Convolucionais para reconhecimento automático de placas de veículos**. Monografia (Graduação em Engenharia da Computação), Universidade Estadual Paulista. São José do Rio Preto, p. 68. 2018.

[19] CEDRO TECHNOLOGIES. **OpenCV: Uma breve introdução à visão computacional com python**. Disponível em:< https://blog.cedrotech.com/opencv-uma-breve-introducao-visao-computacional-com-python>. Acessado em 24 de Maio de 2022.

[20] MARENGONI, M; STRINGHINI, D. **Tutorial: Introdução à Visão Computacional usando OpenCV.** Universidade Presbiteriana Mackenzie. Faculdade de Computação e Informática.

[21] MELO, N. H; SANTOS, M. M. D. **Estudo de técnicas para reconhecimento ótico de caracteres e seu uso para detecção de placas de identificação automotivas brasileiras**. Revista de Engenharia e Tecnologia. V.10, No. 3, Dez/2018.

[22] GONZÁLEZ, M. **O que é OCR: como funciona uma leitura automatizada de documentos?** Disponível em: <https://blog.idwall.co/o-que-e-ocr-leitura-automatizada-documentos/>. Acessado em 1 de Junho de 2022.

[23] OPENALPR. **OpenALPR**. Disponível em: <https://github.com/openalpr/openalpr>. Acessado em 2 de Junho de 2022.

[24] VIVA O LINUX. **Reconhecimento de placas de veículos com OpenALPR**. Disponível em: <https://www.vivaolinux.com.br/artigo/Reconhecimento-de-placas-de-veiculos-com-OpenALPR>. Acessado em 28 de Maio de 2022.

[25] ERAD - Escola Regional de Alto Desempenho, 11ª, 2011, Porto Alegre. Tutoriais e Mini-Cursos: **Fundamentos das Arquiteturas para Processamento Paralelo e Distribuído**. Porto Alegre: Sociedade Brasileira de Computação, Instituto de Informática da UFRGS, 2011. 22-59.

[26] Controle Net Tecnologia. **O que é mais importante num processador: Núcleos, frequência ou threads?** Disponível em: <https://www.controle.net/faq/o-que-e-mais-importante-num-processador-nucleos-frequencia-ou-threads>. Acesso em: 03 de outubro de 2022.

[27] FONSECA, F. M. A. **Texturas com Relevo utilizando Iluminação por Pixel e Processamento Paralelo**. Dissertação (Mestrado do Departamento de Informática) – Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro,2004.

[28] INTEL. ***Intel® RealSense™ Technology***. Disponível em: <https://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html >. Acesso em: 28 de outubro de 2020.

[29] INTEL. ***Intel RealSense Depth Camera D435***. Disponível em: <https://www.intelrealsense.com/depth-camera-d435/>. Acesso em: 28 de outubro de 2020.

[30] CHAVES JR, L. C. **Implantação do Projeto URCA – Uso Racional de Carros nas Cidades Inteligentes**. Qualificação (Mestrado em Engenharia da Computação e Sistemas) – PECS, Universidade Estadual do Maranhão. São Luís, p.63. 2020.

[31] RASPBERRY PI. **Raspberry Pi 4 Model B**. Disponível em: <https://static.raspberrypi.org/files/product-briefs/200521+Raspberry+Pi+4+Product+Brief.pdf>. Acesso em 29 de outubro de 2020.

[32] ROBOCORE. **Raspberry Pi 4 8GB - Model B Anatel**. Disponível em: <https://www.robocore.net/placa-raspberry-pi/raspberry-pi-4-8gb>. Acesso em 29 de Outubro de 2020.

[33] MERCADO LIVRE. **Caixa De Passagem 300x220x148 Km03212=stx322/a Alta Transpar**. Disponível em: <https://produto.mercadolivre.com.br/MLB-881522381-caixa-de-passagem-300x220x148-km03212stx322a-alta-transpar-_JM?>. Acesso em 17 de Fevereiro de 2021.

[34] GRUPO AVANTIA. **5 dicas para posicionar câmeras de segurança**. Disponível em: <https://revistasegurancaeletronica.com.br/5-dicas-para-posicionar-cameras-de-seguranca/>. Acesso em: 30 de outubro de 2020.

[35] LEROY MERLIN. **Protetores para câmeras**. Disponível em: <https://www.leroymerlin.com.br/protetores-para-cameras>. Acesso em: 30 de outubro de 2020 fonte:

[36] INTEL REALSENSE. ***Open-Source Ethernet Networking for Intel® RealSense™ Depth Cameras***. Disponível em: <https://dev.intelrealsense.com/docs/open-source-ethernet-networking-for-intel-realsense-depth-cameras>. Acesso em: 31 de outubro de 2020

[37] VISUAL STUDIO CODE. **Visual Studio Code**. Disponível em: <https://code.visualstudio.com/>. Acesso em 15 de Maio de 2022.

[38] TREINAWEB. **VS Code - O que é e por que você deve usar?** Disponível em: <https://www.robocore.net/placa-raspberry-pi/raspberry-pi-4-8gb>. Acesso em 15 de Maio de 2022.

[39] VAN ROSSUN, G. ***Python reference manual***. Department of Computer Science. CWI. 1995.

[40] LUBANOVIC, B. I***ntroducing Python: Modern Computing in Simple Packages***. O'Reilly Media, Inc., 2014.

[41] TREINAWEB. **Por que o Python é a linguagem mais adotada na área de Data Science**. Disponível em: <https://insightlab.ufc.br/por-que-o-python-e-a-linguagem-mais-adotada-na-area-de-data-science>. Acesso em 23 de Maio de 2022.

[42] REFACTORING. **Observer em Python**. Disponível em: < https://refactoring.guru/pt-br/design-patterns/observer/python/example>. Acessado em 3 de Outubro de 2022.

[43] Watchdog 2.1.5 documentation. **API Reference**. Disponível em: <https://python-watchdog.readthedocs.io/en/stable/api.html>. Acessado em 5 de Outubro de 2022.

[44] SIQUEIRA, F. **Sistemas Operacionais - Threads**. Disponível em: <https://sites.google.com/site/proffernandosiqueiraso/aulas/6-thread>. Acessado em 3 de Outubro de 2022.

APÊNDICE A

06/03/2020 INPI

BRASIL | Acesso à informação | Participe | Serviços | Legislação | Canais

Instituto Nacional da Propriedade Industrial
Ministério da Economia

Consulta à Base de Dados do INPI

[Início | Ajuda?]

» Consultar por: No.Processo | Marca | Titular | Cód. Figura] 1/0

Marca

Nº do Processo: **917496850**

Marca: URCARONA

Situação: Registro de marca em vigor

Apresentação: Mista

Natureza: De Serviço

Classificação de Produtos / Serviços

Classe de Nice	Situação da Classe	Especificação
NCL(11) 42	Vide Situação do Processo	Plataforma computacional como serviço [PaaS];

Classificação Internacional de Viena- CFE(4)

Código	Descrição
18.1.9	Automóveis
27.7.1	Algarismos apresentando um algarismo especial

Titulares

	Nome
Titular(1):	UNIVERSIDADE ESTADUAL DO MARANHÃO - UEMA

Representante Legal

	Nome
Procurador:	NÃO DEFINIDO

Titulares

	Nome
Titular(1):	UNIVERSIDADE ESTADUAL DO MARANHÃO - UEMA

Representante Legal

	Nome
Procurador:	NÃO DEFINIDO

Datas

Data de Depósito	Data de Concessão	Data de Vigência
12/06/2019	18/02/2020	18/02/2030

Prazos para prorrogação de registro de marca

	Prazo Ordinário	Prazo Extraordinário
Início	19/02/2029	19/02/2030
Fim	18/02/2030	18/08/2030

Petições

Pgo	Protocolo	Data	Img	Serviço	Cliente	Delivery	Data
✓	800190482206	24/12/2019	-	372	UNIVERSIDADE ESTADUAL DO MARANHÃO - UEMA		-
✓	850190179503	12/06/2019	-	389	UNIVERSIDADE ESTADUAL DO MARANHÃO - UEMA		-

APÊNDICE B

REPÚBLICA FEDERATIVA DO BRASIL
MINISTÉRIO DA ECONOMIA
INSTITUTO NACIONAL DA PROPRIEDADE INDUSTRIAL
DIRETORIA DE PATENTES, PROGRAMAS DE COMPUTADOR E TOPOGRAFIAS DE CIRCUITOS INTEGRADOS

Certificado de Registro de Programa de Computador

Processo Nº: **BR512022003259-7**

O Instituto Nacional da Propriedade Industrial expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de 02/01/2020, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Título: Analisador de imagens para uso racional de carros na mobilidade urbana das cidades inteligentes versão 1.1

Data de criação: 02/01/2020

Titular(es): UNIVERSIDADE ESTADUAL DO MARANHÃO - UEMA

Autor(es): CARLOS HENRIQUE RODRIGUES DE OLIVEIRA; ANA PAULA FERREIRA COSTA; LUIZ CARLOS CHAVES LIMA JUNIOR; AURELIANNY ALMEIDA DA CUNHA

Linguagem: PYTHON

Campo de aplicação: CO-04; IF-01; SV-01; TC-02

Tipo de programa: GI-01; IA-01; LG-01; TC-03; TC-04

Algoritmo hash: SHA-512

Resumo digital hash:
ef486772a22b249b072c2827f1eb402a294c51962131e538960510b5172b85f3ba2800516eddb8978ed59f5b469c200caf a2fb2b2823ecd3f45c2ed0b6400011

Derivação autorizada: Sim, Na condição de inventor e responsável pelo projeto de inovação Uso Racional de Carros (URCA), autorizo a derivação do programa de computador Analisador de imagens para uso racional de carros na mobilidade urbana das cidades inteligentes, registrado no INPI sob o número 512018000813-5 que utiliza linguagem de programação Java para submissão de novo pedido de registro de programa de computador Analisador de imagens para uso racional de carros na mobilidade urbana das cidades inteligentes versão 1.1 com novo código e em nova linguagem de programação em Python. São Luís- MA, 24/11/2022.

_____ Prof. Dr. Carlos Henrique Rodrigues de Oliveira

Expedido em: 29/11/2022

Aprovado por:
Carlos Alexandre Fernandes Silva

APÊNDICE C

```
import requests
from datetime import datetime
import base64
from PIL import Image
from PIL import ImageOps
import time
from watchdog.observers import Observer
from watchdog.events import PatternMatchingEventHandler
from time import sleep

if __name__ == "__main__":
    patterns = ["*.jpg"] #todas as extensões de arquivos
    ignore_patterns = None
    ignore_directories = False
    case_sensitive = True
    #passando os parâmetros de configuração necessários para o
manipulador de eventos:
    manipulador_eventos = PatternMatchingEventHandler(patterns,
ignore_patterns, ignore_directories, case_sensitive)

    #métodos do Observer:
    def ao_criar(event):
        print("\n"f"O arquivo {event.src_path} foi criado! ")
        crop_caminho = event.src_path
        sleep(2)  # ou 1


        ################################      EXTRAÇÃO DE
PLACAS    ######################

        if crop_caminho.startswith('./crop_imagens\CropPlacas') :
            print ('************************* Iniciando o reconhecimento
de Placas **********************')
            sleep(1)

            #caminho do arquivo a ser wnviado para o ALPR:
            IMAGE_PATH = (crop_caminho)
            SECRET_KEY = '**********************' #chave da conta no
servidor do ALPR

            with open(IMAGE_PATH, 'rb') as image_file:
                img_base64 = base64.b64encode(image_file.read())
#converter a imagem para um formato que o servidor aceita
            url =
'https://api.openalpr.com/v2/recognize_bytes?country=br&topn=1&secret_key
=%s' % (SECRET_KEY)
            response = requests.post(url, data=img_base64) #envio da
imagem e obtenção da resposta

            try:
                i = 0
                if len(response.json().get('results')) == 0:
                    print ("------------------------ erro na
identificação da placa ---------------------- Resposta retornada está
vazia \n")
```

```
                    '''
                    linha_informacao[0]="no identified"
                    '''

                #se houve resultado a partir da imagem vinda do Trigger,
filtrar a placa identificada:
                else:
                    #print("Resposta obtida: \n", response.json())
#resposta em formato json
                    for plate in response.json().get('results'):
                        i += 1
                        x1 = plate['coordinates'][0]["x"]
                        y1 = plate['coordinates'][0]["y"]
                        x2 = plate['coordinates'][2]["x"]
                        y2 = plate['coordinates'][2]["y"]

                        #mostrar a placa recortada
                        img = Image.open(IMAGE_PATH)
                        retangulo = (x1-10, y1-10, x2+10,
y2+10)
                        cropped_placa = img.crop(retangulo)
                        cropped_placa.show()

                        for candidate in plate['candidates']:
                            prefix = "-"

                            if candidate['matches_template']:
                                prefix = "*"
                            print("%s placa identificada: %9s - %3.2f" %
(prefix, candidate['plate'], candidate['confidence']), "%")


            except TypeError:
                print("---------------- Erro na identificação da placa
sobre a imagem. Por favor, reiniciar o script ----------------")


        ###################################      EXTRAÇÃO DE
PESSOAS      ###################################

        if crop_caminho.startswith('./crop_imagens\CropPessoas')
:
            try:
                sleep(2) #ou 1
                img = Image.open(crop_caminho)
                gray = ImageOps.grayscale(img)
                print ('\n ********************** Iniciando o
reconhecimento de Pessoas **********************')

                box = (350, 300, 800, 380)
                #box = (200,100,650,200) #em outro carro
                gray = gray.crop(box)
                #gray.show()

                # Pega as dimensões de largura e altura da imagem
                w = gray.size[0]
```

```
                h = gray.size[1]

                # Iniciando variáveis para varredura de pixel
                razao = 0
                pixelVetor = []
                preto = 0
                branco = 0

                # Valores mínimos e máximo para reconhecimento de pixel
                # estático
                #kminPessoa = 150
                #kmaxPessoa = 255

                # definido para uma pessoa genérica
                kminPessoa = 30
                kmaxPessoa = 255

                # Transforma a imagem em um vetor
                if (not gray == "L"):
                    for y in range(0, h):
                        for x in range(0, w):
                            offset = y * w + x
                            xy = (x, y)
                            pixel = gray.getpixel(xy)
                            pixelVetor.append(pixel)
                    #print(pixelVetor)

                # Transforma o vetor em 0 e 255
                for i in range(0, len(pixelVetor)):
                    if (pixelVetor.__getitem__(i) >= kminPessoa and
pixelVetor.__getitem__(i) < kmaxPessoa):
                        pixelVetor.__delitem__(i)
                        pixelVetor.insert(i, 255)
                        branco += 1
                    else:
                        pixelVetor.__delitem__(i)
                        pixelVetor.insert(i, 0)
                        preto += 1


                    t = (w * h)
                    razao = branco / t
                    Passageiros = 0

                # Contagem de Passageiro
                if (t < 59500):
                    if (razao <= 0.07):
                        Passageiros = 0

                    elif (razao >= 0.07 and razao <= 0.235):
                        Passageiros = 1

                    elif (razao > 0.235 and razao < 0.35):
                        Passageiros = 2
```

```
                elif (t >= 59500 and t < 73000):

                    if (razao <= 0.07):
                        Passageiros = 0

                    elif (razao >= 0.07 and razao < 0.28):
                        Passageiros = 1

                    elif (razao >= 0.28 and razao < 0.35):
                        Passageiros = 2

                elif (t >= 73000 and t <= 77000):
                    if (razao <= 0.07):
                        Passageiros = 0

                    elif (razao >= 0.07 and razao < 0.16):
                        Passageiros = 1

                    elif (razao >= 0.16 and razao <= 0.28):
                        Passageiros = 2

                    elif (razao > 0.28):
                        Passageiros = 0

                elif (t > 77000 and t <= 100000):
                    if (razao <= 0.07):
                        Passageiros = 0

                    elif (razao >= 0.07 and razao <= 0.35):
                        Passageiros = 1

                    elif (razao > 0.35 and razao <= 0.4):
                        Passageiros = 2

                elif (t > 100000):
                    if (razao <= 0.07):
                        Passageiros = 0

                    elif (razao >= 0.07 and razao <= 0.25):
                        Passageiros = 1

                    elif (razao > 0.25 and razao <= 0.35):
                        Passageiros = 2

                #linha_informacao[1] = Passageiros #para o arquivo txt

                qtde_passageiros = Passageiros
                print ("Quantidade de passageiros:" , qtde_passageiros)

            except TypeError:
                print("----------------Erro na Extração de pessoas ------
----------")

        #qtde_passageirosBD = qtde_passageiros

    def ao_deletar(event):
```

```
        print("\n" f"O arquivo {event.src_path} foi deletado!")

    '''
    def ao_modificar(event):
        print("")

    def ao_mover(event):
        print(f"O arquivo {event.src_path} foi movido para
{event.dest_path}")'
    '''

    #cada um dos métodos acima definidos serão realizados pelo
manipulador de eventos:
    manipulador_eventos.on_created = ao_criar #será chamado quando um
arquivo ou diretório for criado
    manipulador_eventos.on_deleted = ao_deletar #será chamado quando um
arquivo ou diretório for deletado

    ''' #ações cuja utilização não foi necessária
    manipulador_eventos.on_modified = ao_modificar #será chamado quando
um arquivo ou diretório for modificado
    manipulador_eventos.on_moved = ao_mover #será chamado quando um
arquivo ou diretório for movido p/ outro diretório
    '''

    #pasta onde estão todas as imagens de placas e pessoas
    #cujos crops (recortes) foram feitos pelo Trigger:
    caminho = "./crop_imagens"
    pesquisar_subpastas=True
    observer = Observer()
    observer.schedule(manipulador_eventos, caminho ,
recursive=pesquisar_subpastas)

    #iniciando o observer dos arquivos:
    observer.start()
    try:
        while True:
                time.sleep(1)
    except KeyboardInterrupt: #para parar/interromper o script, apertar
'ctrl+c'
        observer.stop()
    observer.join()
```