CENTRO PAULA SOUZA

FATEC SANTO ANDRÉ

Mecatrônica Industrial

Henrique Batista Botelho

SISTEMA DE MONITORAMENTO DE ELEVADORES

Santo André

2022

Henrique Batista Botelho

SISTEMA DE MONITORAMENTO DE ELEVADORES

Trabalho de Conclusão de Curso apresentado ao curso de Mecatrônica Industrial da Faculdade de Tecnologia de Santo André, orientado pelo Prof. Me. Paulo Tetsuo Hoashi, como requisito parcial para a obtenção do título de Tecnólogo em Mecatrônica Industrial.

Santo André

2022

FICHA CATALOGRÁFICA

B748s

Botelho, Henrique Batista

Sistema de monitoramento de elevadores / Henrique Batista Botelho. - Santo André, 2022. – 50f: il.

Trabalho de Conclusão de Curso – FATEC Santo André. Curso de Tecnologia em Mecatrônica Industrial, 2022.

Orientador: Prof. Paulo Tetsuo Hoashi

1. Mecatrônica. 2. Projeto. 3. ESP32-CAM. 4. Tecnologia. 5. Programação. 6. Linguagem de programação. 7. Segurança. 8. Pessoas. 9. Elevadores. 10. Sistema de monitoramento. I. Sistema de monitoramento de elevadores.

629.89

Fatec CENTRO PAULA SOUZA GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Santo André

LISTA DE PRESENÇA

Santo André, 14 de dezembro de 2022.

LISTA DE PRESENÇA REFERENTE À APRESENTAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO COM O TEMA: **"SISTEMA DE MONITORAMENTO DE ELEVADORES"** DOS ALUNOS DO 6º SEMESTRE DESTA U.E.

BANCA

PRESIDENTE:
PROF. PAULO TETSUO HOASHI _____

MEMBROS:
PROF. LUIZ VASCO PUGLIA _____

PROF. PEDRO ADOLFO GALANI _____

ALUNO:

HENRIQUE BATISTA BOTELHO _____

AGRADECIMENTO

Gostaria de agradecer este trabalho de conclusão de curso, a primeiramente a Deus por caminhar ao meu lado me dando força e saúde nos desafios enfrentados. A mim, pela dedicação, empenho, esforço, tempo, saúde física e psicológica aplicada durante o decorrer do curso, o qual enfrentei desafios adversos e me superei para a conclusão deste. À Beatriz Miranda, minha psicóloga, a qual ao longo de cada sessão me deu estímulos e doses de apoio, ânimo e confiança para concluir este trabalho. Ao Murilo Zanini de Carvalho e a todos os meus professores, em especial ao meu orientador Paulo Tetsuo Hoashi e ao Fernando Garup Dalbo, que me auxiliaram durante o processo de realização deste projeto. Aos meus colegas de classe, em especial o Gabriel Rodrigues da Silva, que esteve presente durante a realização deste trabalho. E à minha namorada Stephanie da Silva Pereira pelo apoio e auxílio ao longo da elaboração deste projeto.

“O essencial é invisível aos olhos.”

O PEQUENO PRÍNCIPE

RESUMO

Este presente trabalho objetiva-se ao monitoramento por imagem da quantidade de pessoas em uma cabina de elevador. A metodologia deste projeto baseia-se na apresentação da fundamentação teórica, onde há o apontamento das principais influências deste trabalho, permeando a utilização da ESP32-CAM, programada através da linguagem de programação Python, no programa Visual Studio Code, com as bibliotecas OpenCv, Numpy e Urllib, com base na Norma Técnica Brasileira 16042. Como resultado, houve a exposição das linhas de linguagem executadas com a realização da configuração do sistema para quantificação de pessoas no elevador, que em caso de excedência, emitirá um aviso de lotação aparecerá alertando os responsáveis.

Palavras-chave: Monitoramento. Elevador. ESP32-CAM. Programação. Python.

ABSTRACT

This present work is aimed at image monitoring of the number of people in an elevator cabin. The methodology of this project is based on the presentation of the theoretical foundation, where the main influences of this work are indicated, permeating the use of the ESP32-CAM, programmed through the Python programming language, in the Visual Studio Code program, with the OpenCv libraries, Numpy and Urllib, based on the Brazilian Technical Standard 16042. As a result, the lines of language executed with the configuration of the system for quantification of people in the elevator were exposed, which, in case of excess, will issue a warning of capacity and will appear alerting those responsible.

Keywords: Monitor. Elevator. ESP32-CAM. Language Program. Python.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A ideia do projeto surgiu com a necessidade da prevenção de aglomeração de pessoas em locais fechados, com a finalidade de evitar a propagação do vírus da COVID-19. Com a pandemia sobre controle, o projeto foi orientado a outras aplicações, como a quantificação de pessoas numa cabine de elevador. Há uma grande incidência no mal uso deste meio de transporte, onde os usuários excedem a capacidade máxima permitida, colocando em risco a segurança própria e do funcionamento do mecanismo, ocasionando acidentes, manutenções demasiadas e sobrecusto.

Com isto, este projeto busca controlar o limite máximo de pessoas dentro da cabine de elevador, a fim de mitigar os riscos e custos adicionais com manutenções sobrepujantes, quantificando pessoas através de uma ESP32-CAM programada na linguagem Python.

1.1 Motivação

A princípio o projeto tinha como intuito o monitoramento para auxiliar na inspeção de aglomerações, uma vez que em 2020, com o princípio da pandemia de COVID-19, muitas medidas tiveram que ser praticadas a fim de evitar a disseminação em massa do vírus. Entretanto, com o passar dos anos, houve o controle da pandemia, findando a necessidade do rígido distanciamento social.

Com isso, este projeto focou-se na utilização de elevadores. Este meio de transporte foi primeiramente planejado em 1793, de acordo com Arbache (2019), e permeia as rotinas humanas desde então. Auxiliando na locomoção de usuários e objetos a atingir grandes alturas. Entretanto, para assegurar um deslocamento seguro, alguns parâmetros devem ser seguidos, como o controle de carga e quantidade de pessoas por metro quadrado de cabine.

Devido a isso, há a necessidade da criação de um sistema que contabilize a quantidade de pessoas e excedentes (caso houver) em cada viagem que a cabine de elevador realizar, mitigando os problemas ocasionados por sobrecargas e posteriores sobrecustos com manutenções excedentes.

1.2 Objetivo

Este sistema deverá ser aplicado em cabines de elevadores para contabilizar a quantidade de pessoas através de imagem no espaço aferido, e se necessário, viabilizando a segurança dos passageiros e do mecanismo, atenuando possíveis falhas e acidentes por carga excedente. Havendo a possibilidade de implementação em escolas, centros comerciais, casas de festas e eventos em contagem de pessoas.

1.3 Metodologia

Este trabalho está organizado na seguinte estrutura:

- Fundamentação teórica: destaca-se o conhecimento necessário a adquirir para que seja possível entender e realizar o projeto.
- Metodologia: descreve-se o planejado e desenvolvimento do projeto.
- Resultados obtidos: apresenta-se todos os testes feitos, com seus resultados para que fosse possível alcançar o funcionamento do sistema.
- Considerações finais: demonstra-se o entendido final dos autores com o projeto como um todo.
- Apêndice: apresenta-se os materiais adicionais que compõe o trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Para a realização do trabalho houve o embasamento em pesquisas bibliográficas realizadas em meio digital que expõem os componentes importantes para a elaboração deste projeto.

2.1 COVID-19

De acordo com Souza (2020) o Coronavírus (CoV) é uma família de vírus a qual caracteriza-se pela dificuldade respiratória, sendo ele zoonótico. A primeira tipologia deste foi identificada em 1960, por June Almeida, na Escócia. Em 2002, houve a primeira aparição do SARS, que infectou numerosas pessoas em todos os continentes entre os anos de 2003 e 2004, levando 10% dos acometidos a óbito.

A doença provocada pelo vírus SARS-CoV2, o qual ficou conhecido como Covid-19 tornou-se um problema de saúde pública mundial, espalhando-se em todos os continentes em menos de um mês após o primeiro caso confirmado, sendo este em 31 de dezembro de 2019, na cidade de Wuhan na China, de acordo com Santos [2021].

De acordo com um artigo publicado pela revista Science, o marco zero da pandemia, está em uma vendedora que trabalhava no mercado de animais na cidade chinesa. Uma equipe de especialista da OMS hospedou-se na cidade para estudar como ocorreu esta contaminação, realizando um artigo indiciando a contaminação humana através do contato com morcegos, entretanto, os especialistas afirmaram que este laudo é inconclusivo, sendo difícil averiguar corretamente o marco inicial da pandemia, segundo o jornal Exame (2021).

O Brasil constatou o primeiro caso de COVID-19 no final de fevereiro de 2020. A declaração de contaminação comunitária atestou-se em março deste mesmo ano, mês também em que houve o registro do primeiro óbito ocasionado pela doença. Logo após os acontecimentos, as lideranças governamentais começaram a reunir-se e apresentar medidas para mitigar a propagação do vírus em massa. A principal recomendação a ser seguida era o isolamento social, evitando aglomerações visto que o contágio acontecia de maneira simples. Como medida econômica houve a criação do Auxílio emergencial, o qual o governo provia fundo monetário para a população carente, de acordo com a Agência Brasil (2021).

Ainda em 2020, os cientistas iniciaram os estudos de uma vacina eficaz contra o vírus. Em janeiro de 2021, o Brasil iniciou a vacinação populacional por grupos prioritários, após a Anvisa aprovar a utilização emergencial da CoronaVac e da vacina Oxford, completando a carteira vacinal até a quarta dose, em congruência com a Agência Brasil (2021). O mês mais letal da pandemia no país foi abril de 2021, o qual registrou 82.266 vítimas, segundo o site Poder360 (2021).

Com o avanço da aplicação vacinal, houve a redução brusca de casos letais, o afrouxamento das medidas de isolamento social, onde em setembro de 2022, não há mais a obrigatoriedade do uso das máscaras em locais fechados, voltando aos poucos aos protocolos sanitários comuns, de acordo com o Governo Brasileiro (2022).

2.2 Elevadores

Segundo Arbache (2019) a história dos elevadores começa desde 1.500 a.C. com os egípcios construindo suas pirâmides e transportando água do rio Nilo. Na Roma antiga, 300 a.C., estes elevadores rudimentares serviam para movimentar itens pesados, eram abertos e compostos de plataforma com guinchos, movidos manualmente por pessoas, animais ou rodas de água, fazendo com que o carro se movesse verticalmente.

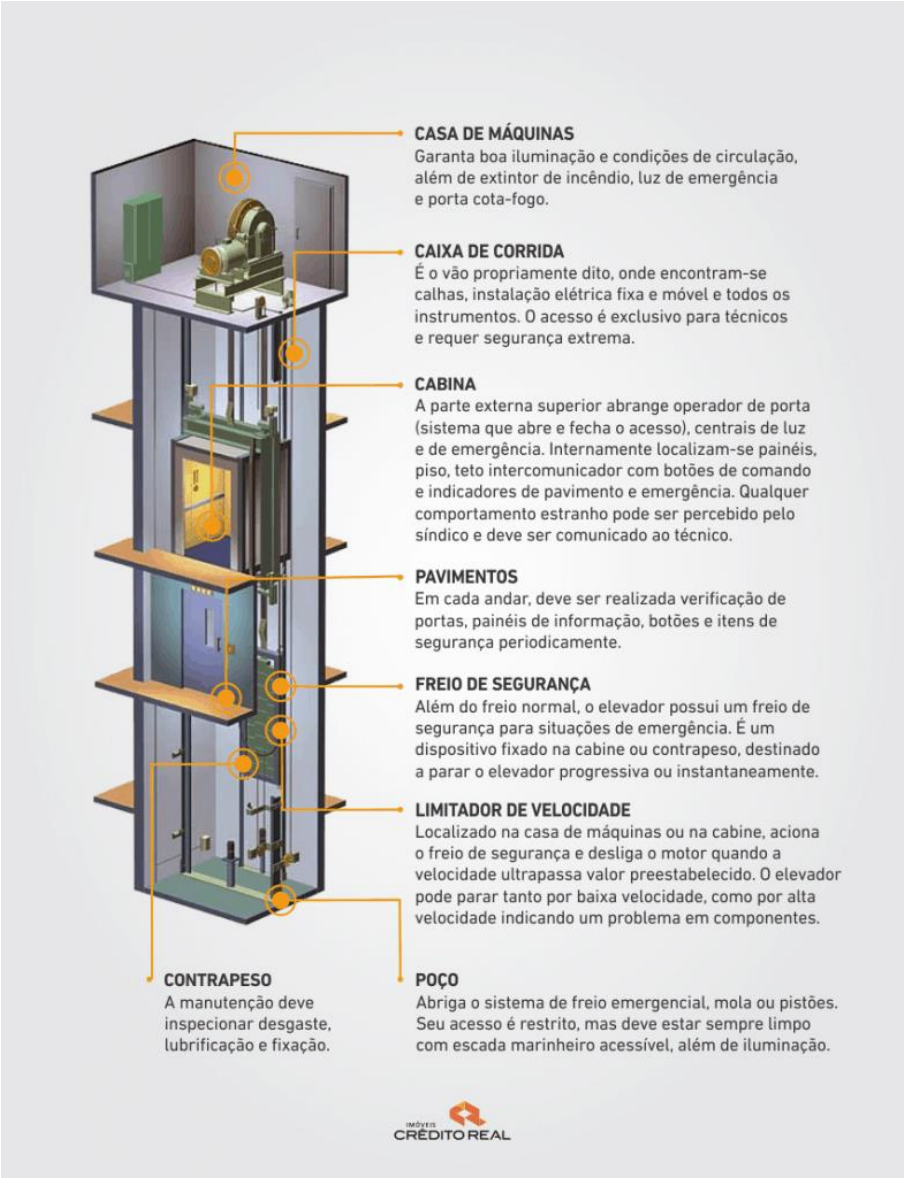
Em 1743 foi criado o primeiro elevador planejado para o transporte exclusivo de pessoas, no palácio de Versalhes na França. Mas não tinha inovações expressivas comparado aos utilizados em Roma. Com a primeira revolução industrial, e as máquinas a vapor, começou a ser produzido elevadores fixos, que carregavam materiais e pessoas. Em 1853 foi incluído por Elisha Graves Otis o primeiro item de segurança para proteger a cabina de queda, em caso de falha do cabo. A empresa Werner von Siemens criou o primeiro elevador elétrico em 1880.

Ainda de acordo com Arbache, os elevadores são compostos de diversos componentes, a casa de máquina é onde ficam as máquinas de tração, quadro de força e os demais componentes para o funcionamento do elevador. A cabine é destinada ao transporte de pessoas. Operador de portas corresponde ao equipamento motorizado sobre a porta da cabine do elevador, realizando a abertura e o fechamento no local correto. A máquina de tração é composta pelo motor, que fornece a potência para a subida e decida da cabina, polias e freio. A caixa de corrida, onde se movimenta a cabine e o contrapeso, é estruturado por paredes verticais, fundo do poço e teto.

Portas de pavimento são as portas situadas em cada pavimento que a cabina faz paradas. Painel de comando é responsável por gerenciar todo o sistema do elevador, tais quais, a estratégia de tráfego, precisão das paradas e velocidade de subida e descida. Quadro de força fornece proteção à instalação elétrica, possibilitando a ativação ou desativação de energia elétrica para o elevador.

O limitador de velocidade é um item de segurança que desliga o motor e ativa o freio de segurança caso o limite de velocidade do carro ultrapasse a velocidade de um limite pré-determinado. O contrapeso forma-se por longarinas e cabeçotes onde são fixados pesos equivalentes ao peso do carro somado de 50% da capacidade licenciada. Correntes de compensação é utilizado para contrabalancear o peso dos cabos de tração quando o contrapeso e a cabine estão em posições opostas. O poço fica localizado na parte inferior da caixa de corrida, instrumentos como limites, polias, para choques e interruptores são instalados aqui. A fonte de emergência aciona automaticamente a iluminação na cabine e a sirene, na falta de energia elétrica. O aparelho seletor, envia um sinal ao quadro de comando indicando o posicionamento do elevador, sentido de movimento e as paradas requisitadas. Cabo de comando são cabos elétricos designados à comunicação entre a cabine, dispositivos da caixa de corrida e quadro de comando como é possível observar na Figura 1.

Figura 1: Componentes do elevador



Fonte: Elevadores (2016).

2.2.1 Sistemas de segurança

Como diz Fabro (2021), os elevadores possuem diversos itens de segurança. Há o freio da máquina tração, o limitador de velocidade e o freio de segurança. O limitador de velocidade é um importante componente de segurança, quando ele identifica que a cabina ultrapassou o limite de velocidade estabelecido, ele aciona mecanicamente o freio de segurança freando o elevador. Caso o comando elétrico dos freios da máquina de tração não pare a cabina, o limitador de velocidade executa um segundo comando, acionando os freios de segurança mecanicamente.

A NBR 16042 (2013), determina a área máxima da cabina do elevador de acordo com a carga nominal, sendo esta apresentada na Quadro 1.

Quadro 1: Área máxima da cabina

Carga nominal (massa) kg	Área máxima da cabina m^2	Carga nominal (massa) kg	Área máxima da cabina m^2
300	0,90	1 000	2,40
375	1,10	1 050	2,50
400	1,17	1 125	2,65
450	1,30	1 200	2,80
525	1,45	1 250	2,90
600	1,60	1 275	2,95
630	1,66	1 350	3,10
675	1,75	1 425	3,25
750	1,90	1 500	3,40
800	2,00	1 600	3,56
825	2,05	2 000	4,20
900	2,20	2 500 [a]	5,00
975	2,35		
[a] Acima de 2 500 kg, acrescentar 0,16 m^2 para cada 100 kg adicionais. Para cargas intermediárias, a área é determinada por interpolação linear.			

Fonte: Associação Brasileira de Normas técnicas, 2013.

A determinação de área mínima se dá pelo número de passageiros, apresentada na Quadro 2.

Quadro 2: Área mínima de cabina

Número de passageiros –	Área útil mínima m^2	Número de passageiros –	Área útil mínima m^2
4	0,79	13	2,15
5	0,98	14	2,29
6	1,17	15	2,43
7	1,31	16	2,57
8	1,45	17	2,71
9	1,59	18	2,85
10	1,73	19	2,99
11	1,87	20	3,13
12	2,01		
Acima de 20 passageiros, acrescentar 0,115 m^2 para cada passageiro adicional.			

Fonte: Associação Brasileira de Normas técnicas, 2013.

Ainda em concordância com a ABNT, o número de passageiros do elevador é calculado pela divisão da carga nominal por 75, com o resultado arredondado para o valor inteiro menor mais próximo. Exemplificando, um elevador com carga nominal de

300kg possui o limite de quatro passageiros, pois 300/75=4.Com isso é possível deduzir o Quadro 3.

Quadro 3: Relação de área mínima e máxima da cabine por peso total e número de pessoas do máximo

Carga nominal (kg)	Número de pessoas	Área mínima (m²)	Área máxima (m²)
300	4	0,79	0,9
375	5	0,98	1,1
400	5	0,98	1,17
450	6	1,17	1,3
525	7	1,31	1,45
600	8	1,45	1,6
630	8	1,45	1,66
...	...	...	...

Fonte: Autoria própria, 2022.

Se houvesse um programa que quantificasse pessoas numa cabina de elevador através da imagem de vídeo haveria controle da quantidade de pessoas, sem exceções como apresentado na Figura 2.

Figura 2: Elevador lotado.

Fonte: Blogspot, (2022).

2.3 Vídeo

De acordo com Monte [2014], o vídeo surgiu através de uma pesquisa realizada pelos irmãos Lumière, onde descreve a habilidade do cérebro humano em armazenar luz na retina, mesmo após a remoção da estimulação original, ou seja, um vídeo com uma taxa de 10 frames, é o suficiente para a sensação de movimento da imagem. Em 1941 surgiu a primeira câmera de vídeo, desenvolvida pela RCA. Inicialmente, essas

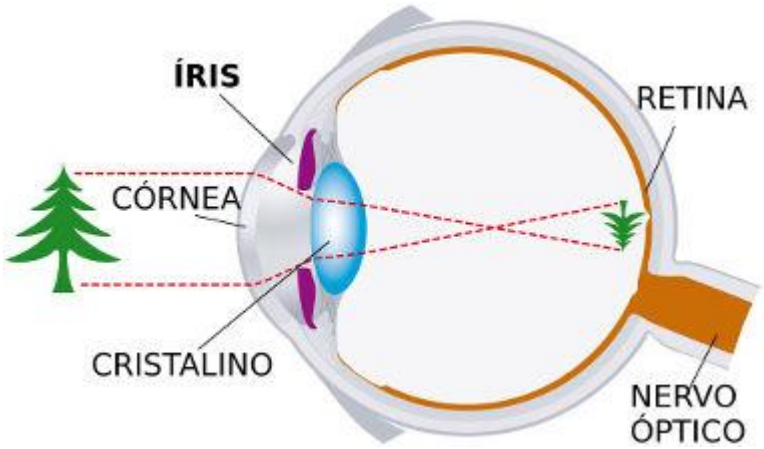
ferramentas eram todas analógicas e possuíam diversos padrões que determinavam o formato de gravação.

A imagem apresentada pelo vídeo é determinada pelo FPS (frame por segundo) que pe uma unidade de medida, ao qual um dispositivo eletrônico realiza a exibição dos quadros por segundo, ou seja, a frequência de exibição dos frames em um determinado tempo, segundo Monte [2014].

2.4 Imagem

Em concordância com Hoashi (2021), a definição de imagem pode ser dada por uma representação, reprodução ou imitação da forma de um objeto, entre outros. Entretanto, a visão computacional possui embasamento científico decorrente de estudos de algoritmos que procuram analisar e entender os conceitos da visão humana, ou seja, como interpretar a imagem, que pode ser interpretado na Figura 3.

Figura 3: Interpretação de imagem pela visão humana



Fonte: Mundo Educação, (2021)

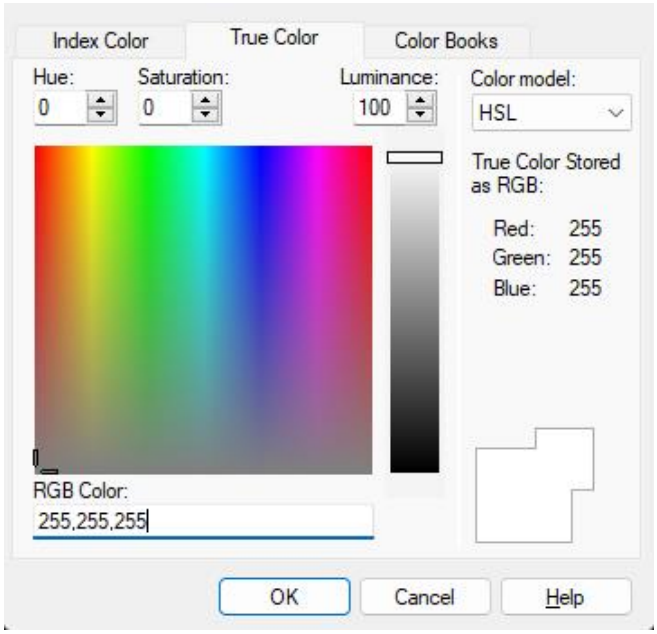
Segundo Hoashi (2021), uma imagem é composta por diversos pixels[1]. Além disso, há dois tipos de imagem, conforme apresentado a seguir:

- Bitmap: Estes são uma matriz de bips, ou seja, a imagem reproduzida é composta por diversos pixels, onde a cada pixel é indicada uma cor e através das coordenadas X e Y, estes são distribuídos em uma malha e formam uma imagem.
- Vetores: São imagens baseadas em polígonos determinados por pontos. Estes pontos são compreendidos pelo computador através das distâncias apresentadas entre eles, com isso, os vetores são infinitamente escaláveis, ou seja, é possível expandir um vetor quando desejável sem diminuição da qualidade da imagem gerada.

[1] Menor componente que constitui uma imagem digital.

A imagem digital é comumente compreendida pela coloração RGB, que é um sistema de cores aditivas composto pelos tons de vermelho (red), verde (green) e azul (blue), estes tons variam de 0 a 255 ou de 0 a FFh, como exemplo cor branca apresentada na Figura 4. O objetivo principal deste sistema, é a reprodução de cores em meios eletrônicos, de acordo com Hoashi (2021).

Figura 4: Sistema RGB - Exemplificação com cor branca.



Fonte: Autoria própria, 2022.

2.5 Internet

De acordo com Bisneto (2003), a criação da internet decorre do lançamento do primeiro satélite ao espaço pela URSS em 1957, o “Sputnik”, sendo este um satélite de comunicação. No mesmo ano, o “Sputnik II” leva o primeiro ser vivo (Terra) ao espaço, a cachorra Laika. Com isso, iniciou a corrida espacial entre Estados Unidos e União das Repúblicas Socialistas Soviéticas, sendo primariamente liderada pelos soviéticos.

Em paralelo ao lançamento do primeiro satélite de comunicação, o presidente norte americano Dwight D. Eisenhover (1953-1961), criou a ARPA (Agência de Desenvolvimento de Projetos Avançados), a qual objetificava-se a evolução dos mecanismos de defesa nacional. No ano seguinte, a NASA – National Aeronautics and Space Administration passa a supervisionar os satélites, com isso a ARPA destina-se ao desenvolvimento de sistema de defesa terrestres, ainda em concordância com Bisneto (2003).

A agência inicia o recrutamento de engenheiro e cientistas em universidades e centro de pesquisa pelo território norte-americano. Surge assim a conceituação da criação de uma rede de computadores, para interligar profissionais da área por todo país, e que propiciassem ao mesmo tempo as mesmas informações a todos os envolvidos no projeto, segundo Bisneto (2003).

Mesmo com todo desenvolvido realizado pela ARPA, o maior colaborador para evolução deste projeto foi Joseph C. R. Licklider (1915-1990), cientista da computação do MIT, criando a teoria de “packet switch”, que consiste na divisão e armazenamento em pedaços da informação militar norte americana, sendo esta distribuídas em diferentes localizações do país, em congruência com Bisneto (2003).

Em 1962, Licklider desenvolveu o conceito de uma rede de computadores espalhadas pelo globo, conectando as pessoas a troca, inserção e remoção de dados sem obstruir ou interferir no trabalho da outra ponta do sistema, sendo esta teoria denominada por “Rede Galáctica”, sendo esta o marco inicial da parceria do cientista com a ARPA, em congruência com Bisneto (2003).

Em 1967, o cientista Lawrence G. Roberts divulga os planos da Rede Galáctica da ARPA, a qual intitulava-se por ARPANET. Com isso, a comunidade descobre que este conceito estava em desenvolvimento em outros centros de pesquisas ao redor do mundo, desta forma, a ARPA convida outros cientistas e entidades para colaborarem neste projeto e solucionar substancial problema da época: A criação de um aparelho de fac-símile compatível com dados binários, o MODEM, em consonância com Bisneto (2003).

Desse modo, com a divulgação dos planos da ARPANET, houve a facilitação da procura de novos esforços angariados para o desenvolvimento do modem. Em 1969, a empresa BBN, realizou a primeira comunicação entre computadores, sendo está entre a Universidade de Stanford, localizada em Massachusetts e a Universidade da California em Los Angeles, propiciando o vasto crescimento da rede. De acordo com Bisneto (2003).

2.6 Python

Em concordância com Silva (2018), Python é uma linguagem de programação interpretada, ou seja, cada linha é compilada e executada de cada vez. Trata-se de um software gratuito, podendo ser utilizado em aplicações de criação de scripts para redes de computadores.

Ainda de acordo com Silva (2018), o matemático Guido Van Rossum, foi o criador desta linguagem. Guido iniciou a carreira realizando trabalho voluntário com a linguagem ABC, desenvolvida por Lambert Meertens, o qual percebeu o potencial de Van Rossum e o admitiu para a implementação deste projeto. Entretanto, o matemático aspirava outros ideais, dedicando-se ao desenvolvimento de Python, tendo este como um hobby.

Em 1999, Guido submeteu uma proposta de financiamento à DARPA (Agência do Departamento de Defesa Norte-Americano). Com isso, ele estabeleceu os seus limites, fazendo com que o programa se popularizasse e em 2001 instituiu-se a Python Software Foundation. Esta é uma fundação sem fins lucrativos, possuindo uma comunidade aberta a quaisquer alterações que são avaliadas e caso esteja em concordância com a maioria dos usuários, esta é implementada na próxima versão, segundo Silva (2018).

Com isso, entende-se que este software está em constante evolução e aprimoramento. As sugestões de aplicação e usabilidade deste tornam-no ainda mais habitual no meio das linguagens de programação do mercado tecnológico, que é promissor.

2.7 IDE Arduino

De acordo com Bittencourt (2022), Arduino IDE (Ambiente de Desenvolvimento Integrado) é um mecanismo de desenvolvimento multiplataforma com linguagem Java. Este é um programa que agrupa características e ferramentas que auxiliam no desenvolvimento de software. Ademais, é um compressor de linguagens de programação GCC (Compilador de Linguagem), decorrentes dos projetos Processing e Wiring, devido a isto, possui capacidade de programar em C e C++.

2.8 Ambiente de Desenvolvimento

De acordo com a Revista NetMagazine (2016), o Visual Studio Code foi desenvolvido pela Microsoft e lançado em 2015. Em concordância com a Revista Remessa Online (2021) o VS Code em suma é um editor que ampara na criação de um código de software, principalmente nas fases de codificação e teste. Este Possui assistência para linguagens como Python, Ruby etc. Além disso, é um programa open source, viabilizando o desenvolvimento comunitário deste software, segundo a Revista Netmagazine (2016).

2.8.1 OpenCV

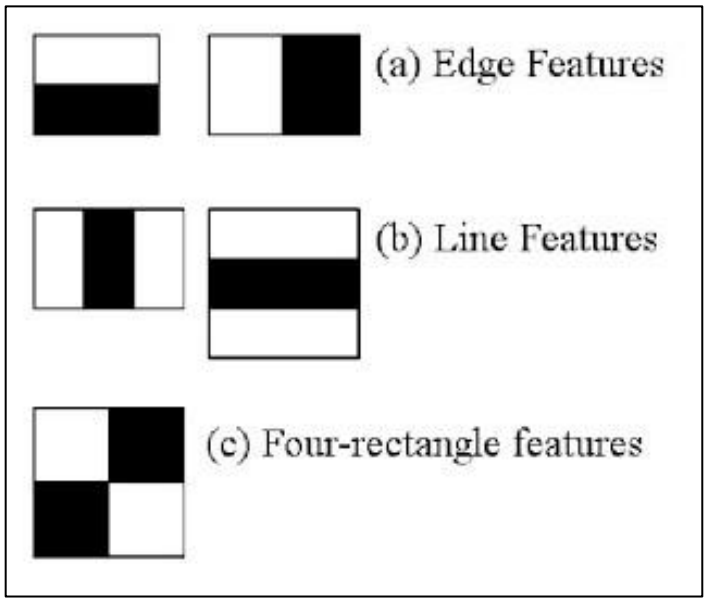
De acordo com Bertoleti (2020), OpenCV (Open Source Computer Vision Library) caracteriza-se por uma biblioteca de visão computacional open-source multiplataforma, sendo esta desenvolvida em 2002 pela Intel. Esta ferramenta agrupa diversos recursos para aplicações de visão computadorizada, compreendendo entre estes: obtenção e aquisição de imagens de câmeras digitais, assim como o tratamento e processando de imagens estáticas e vídeo, sendo estes gravados ou transmissão ao vivo.

2.8.2 Haar Cascade

Em concordância com Cascade (2022) a identificação de objetos usando classificadores em cascata foi proposto por Paul Viola e Michael Jones em seu artigo “Rapid Object Detection using a Boosted Cascade of Simple Features” em 2001, é baseado em recursos Haar, uma abordagem fundamentada em aprendizado de máquina. A função cascata se desenvolve a partir de imagens positivas e negativas, em seguida detecta objetos em outras imagens.

O algoritmo inicialmente necessita de muitas imagens com rostos (imagens positivas) e imagens sem rostos (imagens negativas) para treinar o classificador. Para extrair recursos dele são usados os recursos do Haar mostrados na Figura 5. Cada recurso é um valor único calculado pela subtração da soma dos pixels sob o retângulo branco com a soma dos pixels sob o retângulo preto.

Figura 5: Recursos do Haar

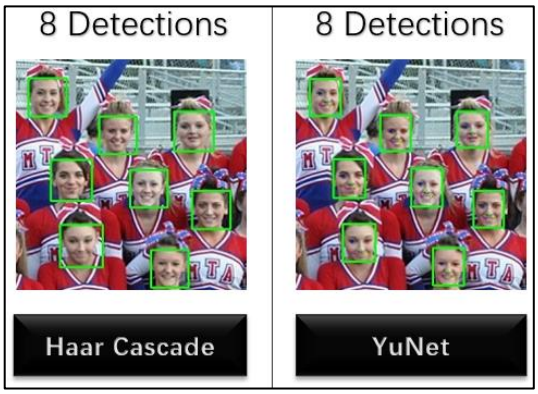


Fonte: Cascade (2022)

Ainda de acordo com Cascade (2022) é aplicado cada recurso em todas as imagens de treinamento. Para cada feição, obtém a melhor limiar que classificará as faces em positivas e negativas. Para alcançar a precisão exigida são selecionadas as características com taxa de erro mínima, cada imagem recebe um peso igual no início, após cada classificação os pesos das imagens mal classificadas são aumentados. O mesmo processo é feito, calculando novas taxas e pesos.

De acordo com Nelson (2022), YuNet classifica-se como um detector facial baseado em Rede Neural Convolucional, desenvolvido por Shiqi Yu em 2018. A diferença deste para o Haar Cascade tradicional é o enfoque em reconhecimento facial e a velocidade na detecção, conforme apresentado na Figura 6.

Figura 6: Diferença de detecção facial entre o Haar Cascade e Yunet



Modelo	Número de Detecções	Consumo de tempo
Classificador em cascata	8	25,81ms
YuNet	8	5,09ms

Fonte: Nelson, 2022

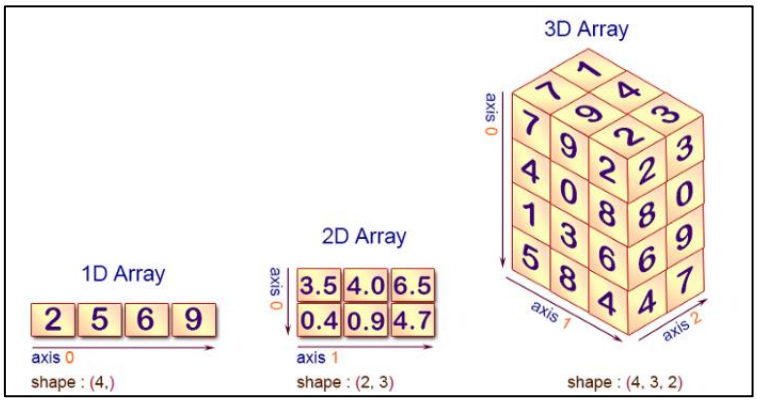
2.8.3 NumPy

Em concordância Mulinari [2022], esta tipologia de biblioteca foi desenvolvida por Travis Oliphant em 2017, sendo este referenciado aos projetos Numeric e Numarray com o propósito de agrupar a comunidade ao redor de um único framework de processamento de arrays[2]. Em concordância com Luiz Santiago Junior (2018), no núcleo desta biblioteca de código está o objeto ndarray, agrupando arrays n-dimensionais de tipos de dados homogêneos com variadas operações sendo realizadas em código compilado para desempenho.

Ainda de acordo com Junior (2018), no Numpy, as dimensões são denominadas por eixos, sendo estes classificados. Exemplificando, a matriz 2-D tem 2 eixos. Conforme apresentado na Figura 7, o eixo 0 estará presente em todas as linhas de cada coluna, e o eixo um realizará o inverso.

[2] Estrutura multidimensional que permite o armazenamento de dados na memória do computador, de maneira que cada item localizado nesta estrutura pode ser encontrado por meio de indexação

Figura 7: Matriz em dimensões



Fonte: Santiago Junior, 2018

Com o funcionamento e desenvolvimento baseado na estrutura de dados ndarray, esta ferramenta apresenta operações diversas, como: tratamento e limpeza de dados, geração de subconjuntos e filtragens, estatísticas descritivas, manipulação de dados relacionais, entre outros. Ademais, no NumPy há a possibilidade da realização de operações e funções matemáticas em arrays, dispensando a escritura de laços, geração de números aleatórios entre outros, em concordância com Mulinari [2022].

2.8.4 Urllib

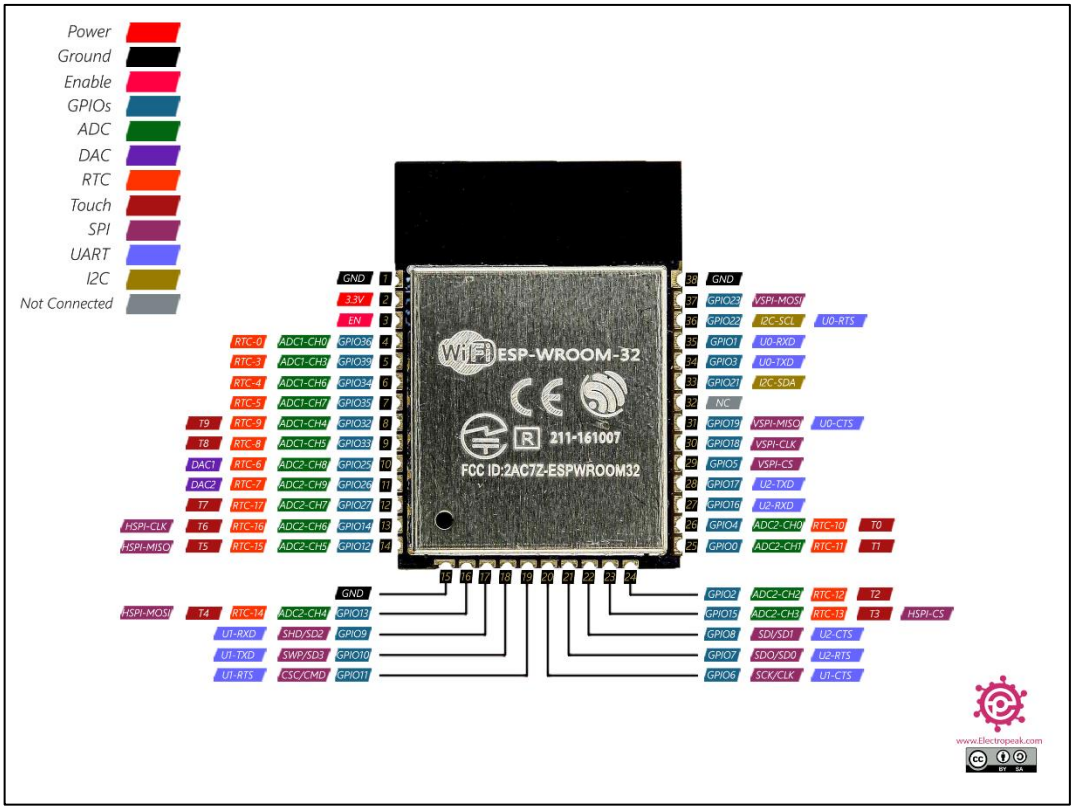
De acordo com Python (2022), Urllib é um agrupamento que armazena diversos módulos para execução de URLs, possibilitando o download e análise de dados, como apresentado abaixo:

- Urllib.request: abertura;
- Urlllib.parse: definição das funções de manipulação de URLs e partes componentes;
- Urllib.error: análise;
- Urllib.robotparser: verificação de arquivos URL.

2.9 ESP 32

Segundo a ESP32 Overview [2015], o ESP32 é um microcontrolador com Wi-Fi e Bluetooth integrados tornando possível o desenvolvimento de diversas aplicações. O módulo ESP32 foi projetado para dispositivos móveis, eletrônicos vestíveis e para dispositivos IoT (Internet of Things), podendo operar em ambientes industriais entre –40°C e 125°C. Na **Erro! Fonte de referência não encontrada.**, é possível observar o aspecto físico do microcontrolador, como também sua pinagem.

Figura 8: ESP32



Fonte: Módulo ESP32 (2021)

2.9.1 ESP32-CAM

Segundo Robocore [2022], ESP32-CAM é uma placa desenvolvida pela ESPRESSIF, resultante da junção de um módulo ESP32 com uma câmera OV2640 de 2MP e um slot para cartão MicroSD. Com este dispositivo é possível capturar e armazenar fotos e vídeos, mesmo em ambientes com baixa luminosidade, pois este conta com LED de flash embutido.

3 METODOLOGIA

Para a elaboração deste projeto foi utilizada a metodologia apresentada no Quadro 4.

Quadro 4: Metodologia aplicada no trabalho

	CAPTURA DA IMAGEM	TRATAMENTO E PROCESSAMENTO DA IMAGEM
AMBIENTE DE DESENVOLVIMENTO	IDE ARDUINO	VISUAL STUDIO CODE
LINGUAGEM	C++	PYTHON
HARDWARE	ESPCAM 32	COMPUTADOR

Fonte: Autoria própria, 2022.

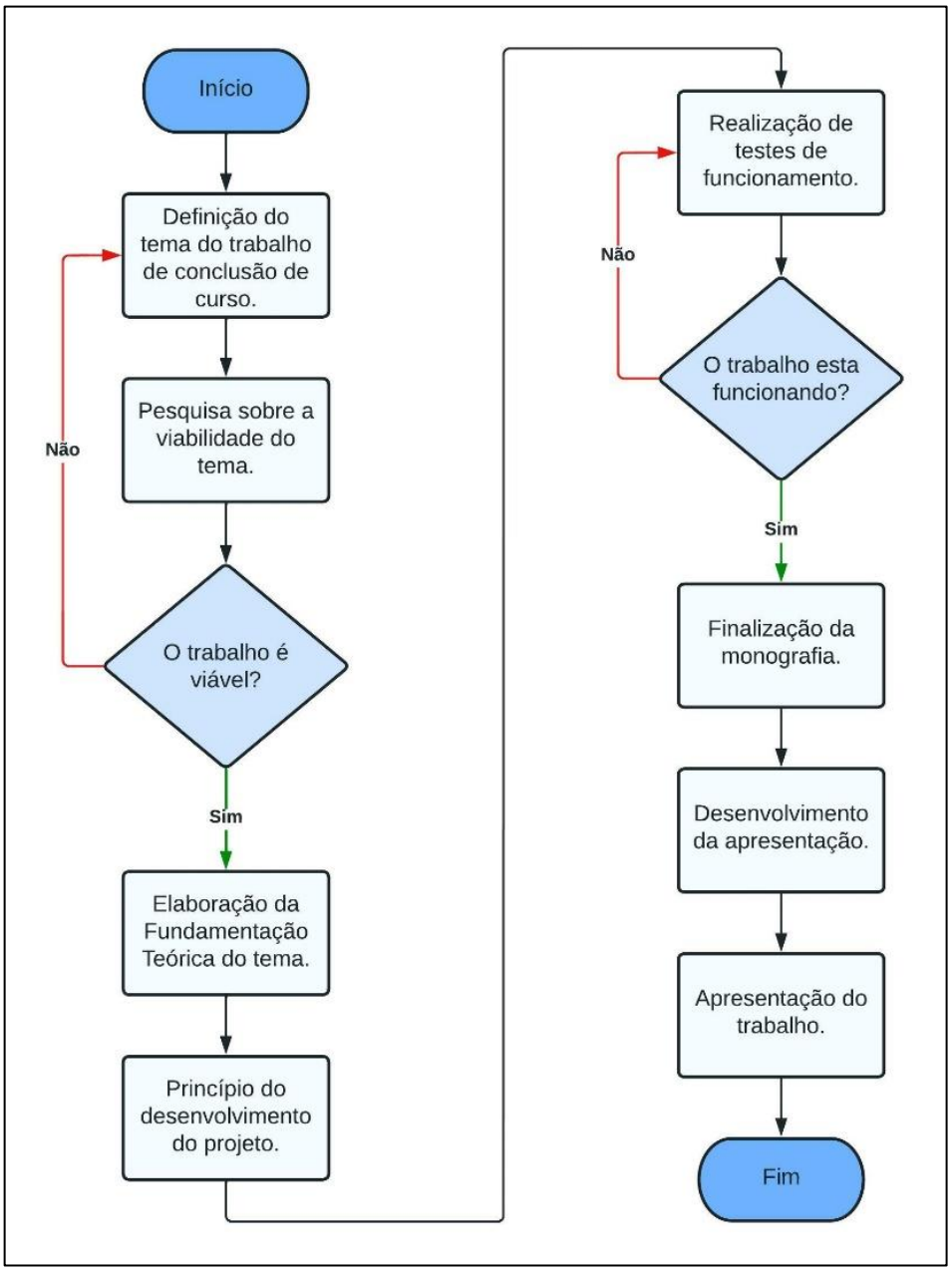
4 DESENVOLVIMENTO

Conforme apresentado durante a fundamentação teórica, este trabalho consiste na monitoração quantitativa de pessoas em uma cabina de elevador. Nos tópicos abaixo haverá a exposição da realização do programa.

4.1 Diagrama de Blocos

No diagrama de blocos apresentado na Figura 9 está representado o fluxo de orientação para elaboração deste projeto.

Figura 9: Diagrama de blocos

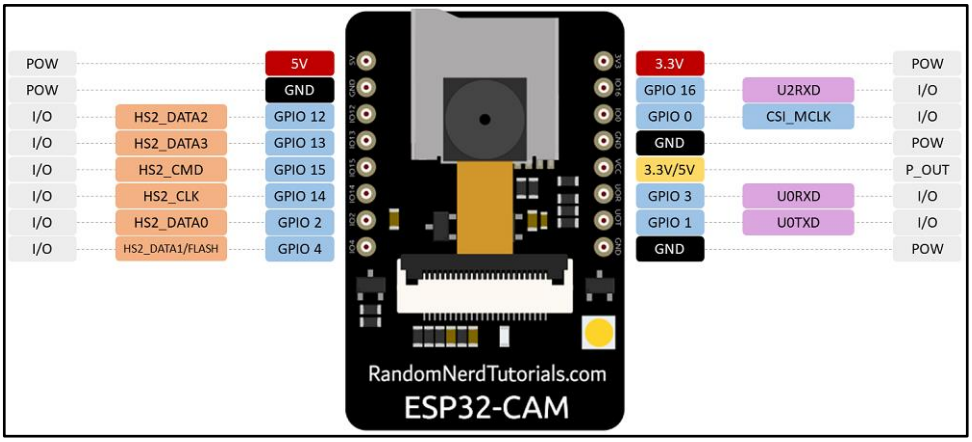


Fonte: Autoria própria, 2022.

4.2 Tecnologias Utilizadas no Projeto

Para a realização do projeto foi utilizado o ESP32-CAM, com a pinagem, apresentada na Figura 10, com a câmera OV2640.

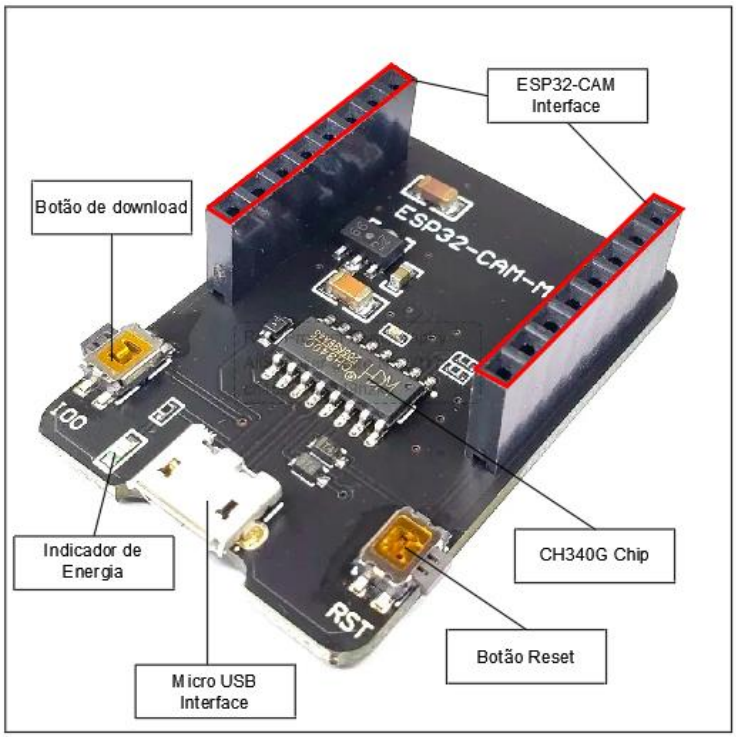
Figura 10: Pinagem ESP32 - CAM



Fonte: ESP32-CAM AI-Thinker Pinout Guide, [2020].

. E o módulo conversor USB Serial CH340G, com a pinagem evidenciada na Figura 11, para ESP32-CAM.

Figura 11: Pinagem do Módulo Conversor

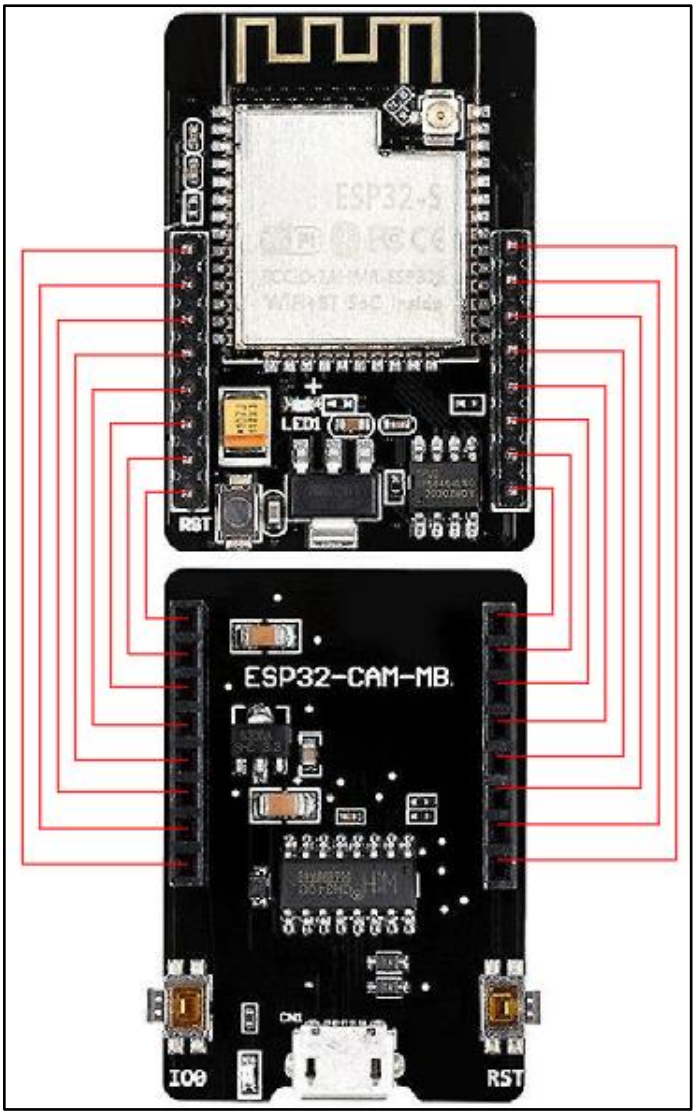


Fonte: Casa da Robótica modificado por autor, 2022.

4.3 Arquitetura do Sistema

A arquitetura deste projeto consiste na ligação do módulo conversor USB Serial CH340G com o ESP32 – CAM, e conforme apresentado na Figura 12 e Figura 13.

Figura 12: Ligação entre o Módulo Conversor e ESP32-CAM



Fonte: Autoria própria, 2022.

Figura 13: Ligação entre Módulo Conversor e ESP32-CAM

Fonte: Autoria própria, 2022.

4.4 Testes

Durante a construção do projeto, houve a realização dos seguintes testes:

1. Teste da inicialização do ESP 32;
2. Teste do programa dos frames;
3. Teste com frames no VS Code;
4. Teste do tratamento de imagem;
5. Teste do resultado.

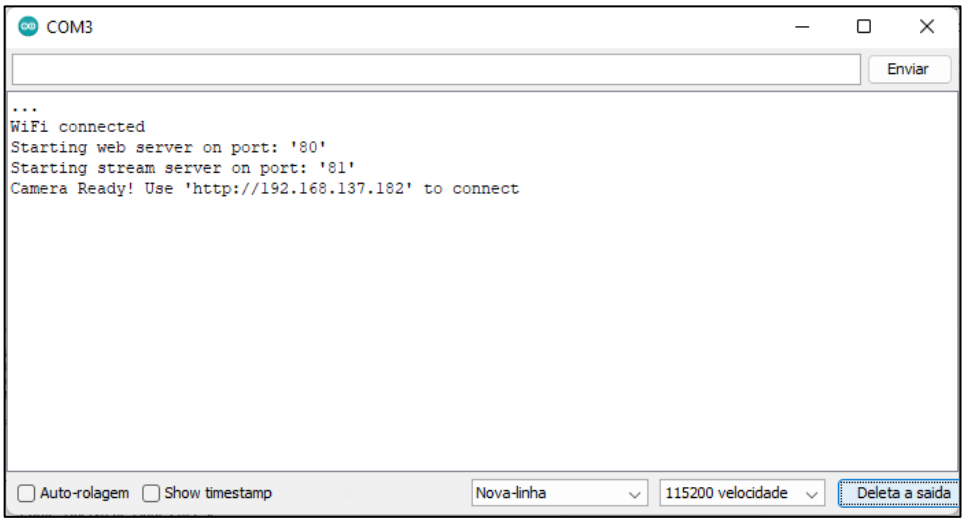
Sendo estes exemplificados e apresentados abaixo.

4.4.1 Inicialização do ESP 32

A inicialização do módulo ESP32 partiu de um programa exemplo disponibilizado pela IDE do Arduino, com esse exemplo torna-se visível a imagem gerada através da câmera. Dentro da interface de programação é necessário realizar a instalação do pacote ESP32 no gerenciador de placas, após a instalação é possível encontrar o exemplo ‘CameraWebServer’, conforme descrito no Apêndice C.

Com o ESP32-Cam conectado ao provedor de internet local, gerando assim o código IP, onde irá ocorrer a transmissão da imagem gerada pela câmera, conforme apresentado na Figura 14.

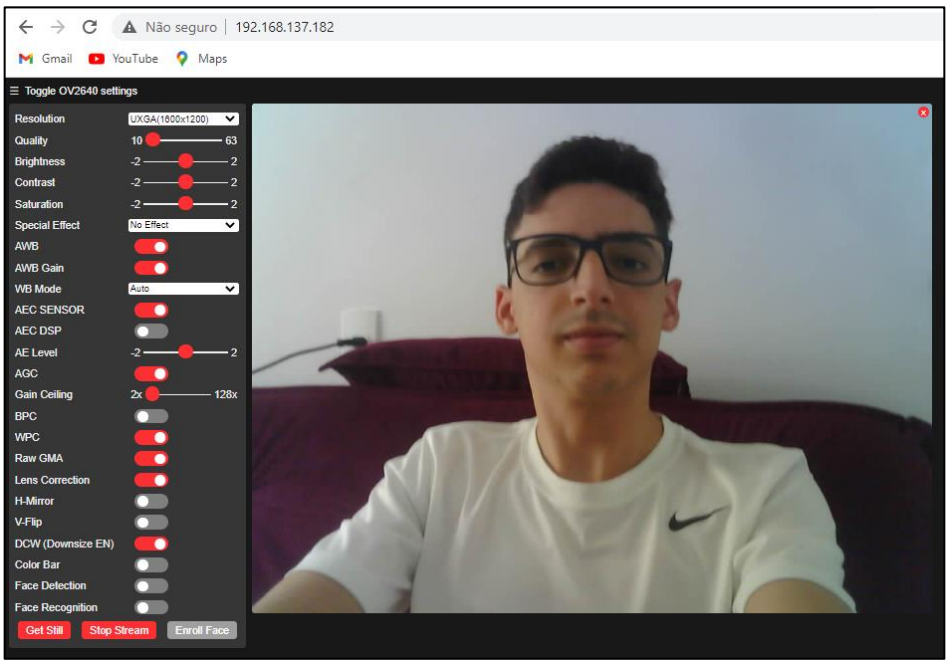
Figura 14: Monitor Serial



Fonte: Autoria própria, 2022.

Logo após a geração do código, este foi copiado na barra de busca do Google, abrindo a tela da Figura 15 a seguir.

Figura 15: Imagem gerada pela câmera



Fonte: Autoria própria, 2022.

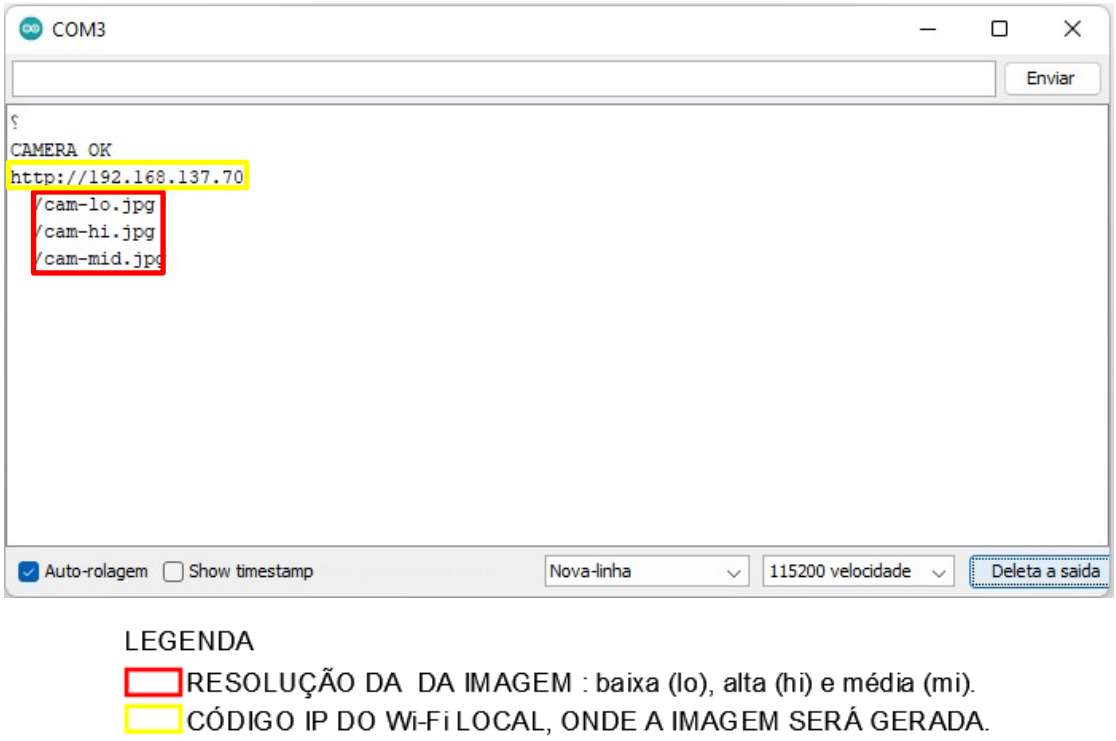
Entretanto, por mais que o programa registre a imagem, não foi possível realizar o tratamento dessa imagem no VS Code. Com isso, houve a gravação de outro programa no módulo, gerando um frame isolado no código IP do WI-FI na rede local.

4.4.2 Captura de Imagem

O processo para captação de imagem deste projeto iniciou-se no Arduino, com um programa publicado por Electronics (2022), utilizado para realizar a geração de uma imagem no código IP, sendo este apresentado no Apêndice D.

Com o ESP32-Cam conectado ao provedor de internet local e gerando assim o código IP, onde irá ocorrer a transmissão da imagem gerada pela câmera, conforme apresentado na Figura 16.

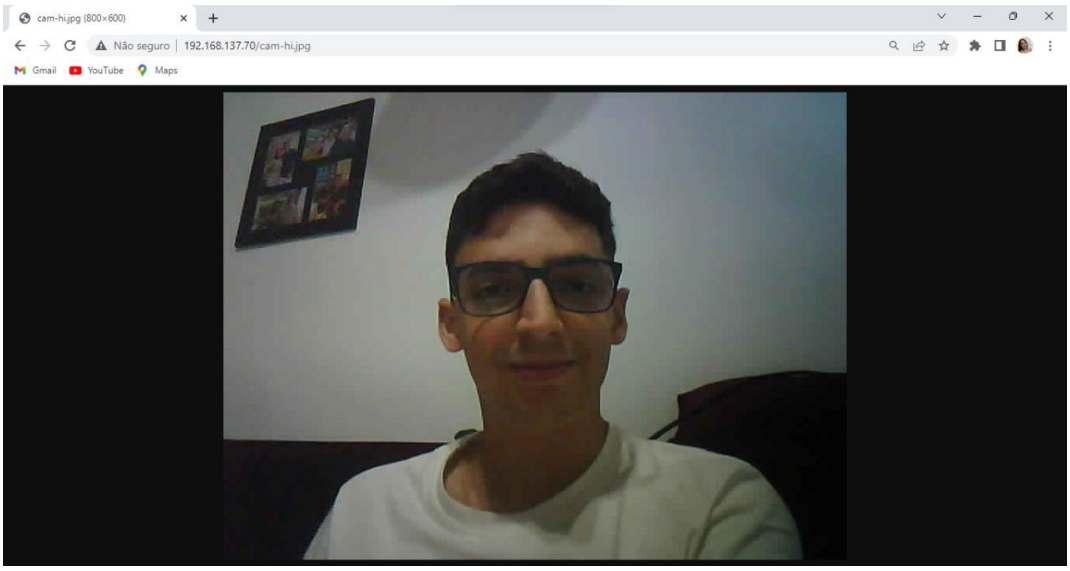
Figura 16: Monitor serial



Fonte: Autoria própria, 2022.

Logo após a geração do código, este foi copiado na barra de busca do Google com a resolução já escolhida, de acordo com a Figura 16, fornecendo a imagem da Figura 17 logo após.

Figura 17: Frame gerado pela câmera



Fonte: Autoria própria, 2022.

Com esse código gerado, há a aplicação no programa VSCODE com a biblioteca urllib.request previamente instalada, possibilitando a importação do código url ao programa, como representado na Figura 18.

Figura 18: Biblioteca urllib no VS Code

```
import cv2
import urllib.request
import numpy as np

url = "http://192.168.137.70/cam-hi.jpg"
```

Fonte: Autoria própria, 2022.

4.4.3 Processamento de Imagem

Com a imagem registrada no VSCODE, e as bibliotecas NumPy e OpenCv anteriormente instaladas, os seguintes códigos foram inseridos, a fim da realização do tratamento dos frames captados. A seguir há a apresentação dos códigos no programa e a exemplificação destes.

Figura 19: Linhas 1 a 7 do programa no VS Code

```
1  import cv2
2  import urllib.request
3  import numpy as np
4
5  url = "http://192.168.137.70/cam-hi.jpg"
6  file_address = 'opencv-master\data\haarcascades\haarcascade_frontalface_default.xml'
7
```

Fonte: Autoria própria, 2022.

Quadro 5: Explicação da função de cada linha de programação da Figura 19.

LINHA	FUNÇÃO	VARIAVEL
1	Importação das bibliotecas "cv2", "NumPY" e "urllib"	-
2		
3		
5	Está apresentado a URL do código IP gerado pela rede no programa Arduino.	url
6	Evidencia o endereço do arquivo utilizado para reconhecimento facial, sendo este um material da aula do curso de Robótica	file_address

Fonte: Autoria própria, 2022.

Figura 20: Linhas 8 a 18 do programa no VS Code

```
8   while (1):
9       req = urllib.request.urlopen(url)
10      arr = np.asarray(bytearray(req.read()), dtype=np.uint8)
11      frame = cv2.imdecode(arr, -1)
12      frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
13
14      face_classifier = cv2.CascadeClassifier(file_address)
15      faces = face_classifier.detectMultiScale(frame_gray, 1.05, 5)
16
17      count = 0
18
```

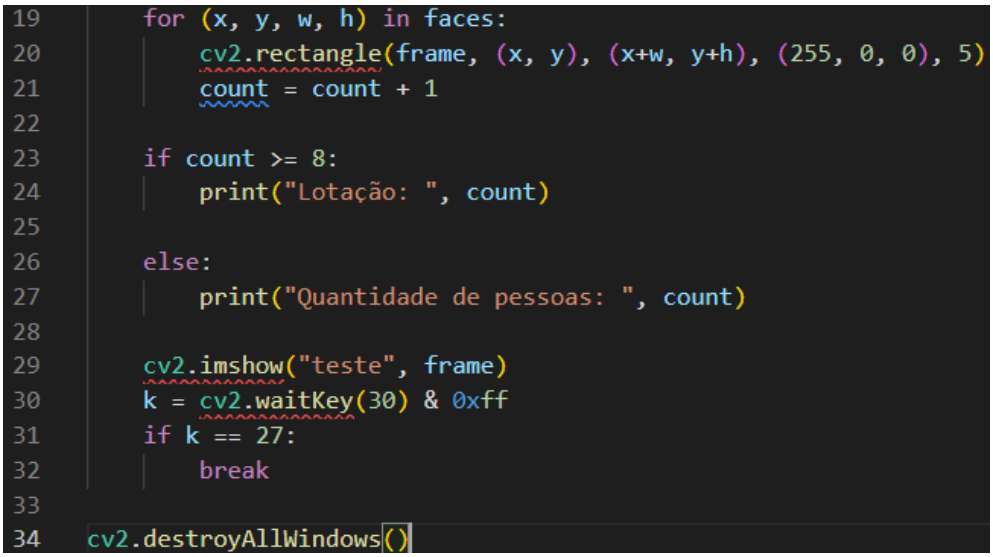
Fonte: Autoria própria, 2022.

Quadro 6: Explicação da função de cada linha de programação da Figura 20.

LINHA	FUNÇÃO	VARIAVEL	OBSERVAÇÃO
8	Evidencia que, enquanto o programa estiver ligado, ele deve executar os demais comandos dentro do "while".	-	-
9	Realiza abertura e gravação do conteúdo do "url"	req	-
10	Há a conversão do material armazenado na variavel "req" para matriz	arr	-
11	Realiza a transformação dos dados em formato de imagem	frame	-
12	Transforma a imagem que está no "frame" do modo de cor BGR para monocromo cinza	frame_gray	-
14	Carrega o arquivo dentro da váriavel "file_Adress" para a realização da identificação facial	face_classifier	-
15	Há a utilização do comando CV2.CascadeClassifier.detectMultiScale para a execução da identificação facial dentro da imagem gerada em cinza com as configurações de scale factor, ajustado em 1.05, reduzindo em 5% e minNeighbors, que está ajustado em 5	faces	Scale factor: redimensiona o tamanho da imagem e MinNeighbors é o parametro que identifica o rosto de acordo com a qualidade da imagem
17	Inicia o contador em 0.	count	-

Fonte: Autoria própria, 2022.

Figura 21: Linhas 19 a 34 do programa no VS Code



```
19      for (x, y, w, h) in faces:
20          cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 5)
21          count = count + 1
22
23      if count >= 8:
24          print("Lotação: ", count)
25
26      else:
27          print("Quantidade de pessoas: ", count)
28
29      cv2.imshow("teste", frame)
30      k = cv2.waitKey(30) & 0xff
31      if k == 27:
32          break
33
34  cv2.destroyAllWindows()
```

Fonte: Autoria própria, 2022.

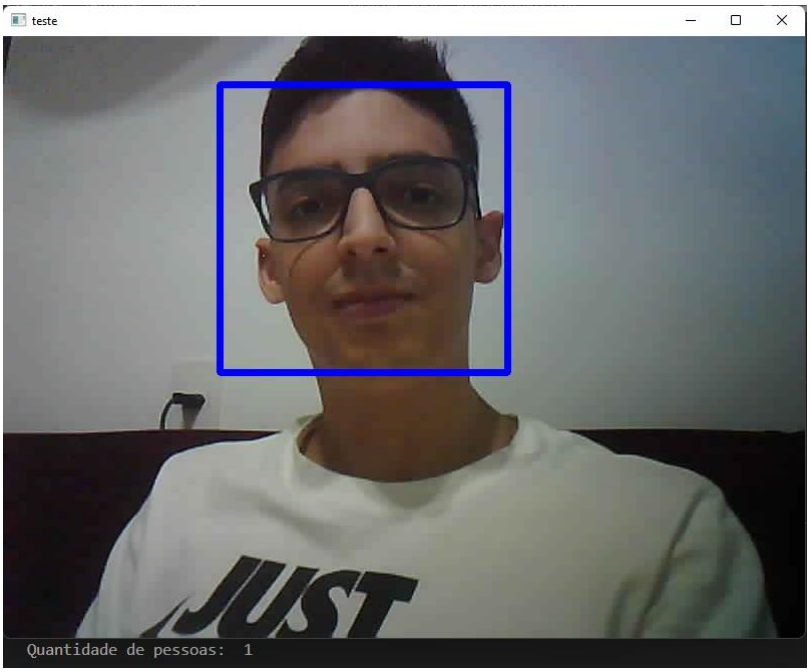
Quadro 7: Explicação da função de cada linha de programação da Figura 21

LINHA	FUNÇÃO	VARIAVEL	OBSERVAÇÃO
19	Inicia o comando "for" que executa um retangulo e quantifica os rostos encontrados dentro da imagem frame, na cor azul com espessura de 5 bites.	-	-
20		-	-
21		-	-
23	Executa o comando "if", se o contador estiver maior ou igual a 8, mostrando a mensagem "lotação: " e a quantidade de pessoas contadas.	-	-
24		-	-
26	Entende que se o contador obtiver número inferior a 8, ele mostrará a mensagem "quantidade de pessoas: " e a quantidade de rostos apurados	-	-
27		-	-
29	O programa irá colher a imagem gravada na variavel "frame" e mostrá-la como "teste	-	-
30	Aguarda 30 milisegundos e escreve 0xFF na varíavel	k	0XFF representa de 0 à 255, pois é o número limite para as teclas do teclado do computador
31	Se "k" for igual a 27, o programa sai do "while"	-	A chave 27 é igual a tecla "esc"
32		-	
34	Executa o fechamento de todas as janelas abertas	-	-

Fonte: Autoria própria, 2022.

Com o advento da construção do programa, houve a realização do tratamento de imagem e a consecutiva quantificação de pessoas através do módulo ESP 32 CAM, sendo está a principal tecnologia utilizada no projeto, conforme apresentado na Figura 22.

Figura 22: Teste com o monitoramento quantitativo da imagem



Fonte: Autoria própria, 2022.

5 RESULTADOS OBTIDOS

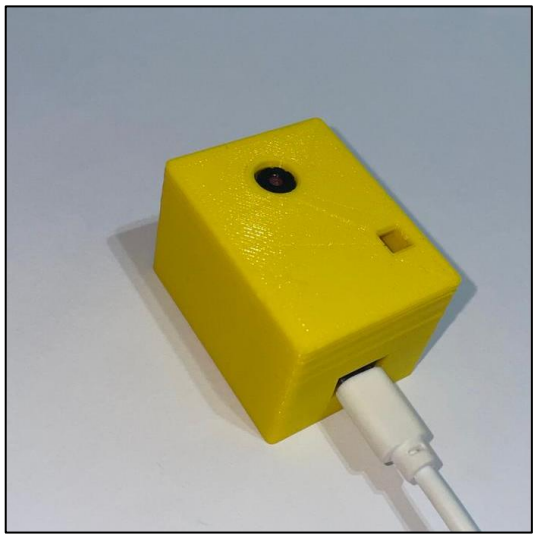
Com os resultados obtidos, este trabalho consiste na quantificação monitorada de pessoas numa cabina de elevador. Este faz-se necessário visto que é um parâmetro de segurança no funcionamento do elevador. Com o teste realizado na cabina do elevador obteve-se o resultado apresentado na Figura 23 a seguir.

Figura 23: Teste com o monitoramento no elevador

Fonte: Autoria própria, 2022.

Para a proteção do módulo ESP32 utilizado foi elaborada e modelada uma caixa impressa em 3D, obtendo também uma tampa de fechamento, como apresentado na Figura 24, o desenho técnico de ambas encontradas nos Apêndices A e B.

Figura 24: Caixa para o ESP32



Fonte: Autoria própria, 2022.

Conforme o resultado obtido, conclui-se que este possibilita que um programa quantifique pessoas por reconhecimento facial, por meio de vídeo -stream- através de um módulo de câmera, programado em linguagem Python, podendo ser utilizado como auxílio para impedir que os indicadores de segurança exigidos pela ABNT NBR 16042 sejam violados, ocasionando avarias e acidentes escusáveis no funcionando desta tipologia de transporte de passageiros vertical.

6 CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi realizar o monitoramento quantitativo de pessoas numa cabina de elevador, visto que este é um parâmetro de segurança que se faz necessário o assessoramento em concordância com a ABNT NBR 16042.

Inicialmente, houve a realização dos estudos e pesquisas em torno do tema, para a partir destes definir o modelo de tecnologia e arquitetura a ser utilizada no projeto. Para a execução do monitoramento, optou-se pelo módulo ESP 32 CAM com OV2640, sendo este programado na linguagem Python.

Logo após, houve a realização de uma programação através da IDE do Arduino para que este obtivesse os frames captados pela câmera. Ademais, houve a utilização do VS Code no tratamento da imagem gerada pelos frames em código IP, tornando-a em monocromo cinza, aplicando as bibliotecas: NumPy, URLLIB e OpenCV. Com isso, foi possível realizar a quantificação de pessoas numa cabina, gerando um retângulo em cada rosto reconhecido pela ferramenta através dos frames captados e já programados.

Com isso, conclui-se que através de estudos, pesquisas e orientações a cerca do tema, mercado comercial e tecnologias foi possível realizar o monitoramento com reconhecimento facial através de vídeo -stream- para auxiliar no controle de pessoas numa cabina de elevador ou em outros locais fechados. A partir deste princípio é possível adquirir êxitos e aperfeiçoar a segurança no transporte vertical de passageiros em curto/médio prazo.

A seção a seguir, evidenciará melhorias e possíveis trabalhos futuros, acerca do monitoramento e quantificação por imagem de vídeo.

7 PROPOSTAS FUTURAS

No que tange a melhorias futuras, este trabalho oferece algumas possibilidades, sendo essas apresentadas abaixo.

- Inclusão do sistema ao quadro de comando de funcionamento do elevador:

Atualmente o sistema realiza a medição da quantidade de pessoas no elevador, entretanto seria interessante que esta ferramenta se comunicasse com o quadro de comando de funcionamento do transporte vertical ao exceder o limite de pessoas a qual a cabine foi dimensionada, fazendo com que o elevador não execute nenhum comando até que a quantidade de pessoas esteja dentro do limite permitido.

- Aprimoramento do programa:

Este monitoramento realiza o reconhecimento facial somente se a pessoa estiver de frente a câmera e sem acessórios faciais, sendo esta, uma possível melhoria fazendo com que este programa reconheça o indivíduo em qualquer posição e acessórios faciais, como máscara protetora facial, óculos de sol e chapéus. Assim como este quantifica as pessoas dentro do programa, havendo a possibilidade de inserir a contagem na imagem.

- Inclusão de um alerta sonoro:

Este programa realiza o monitoramento, entretanto seria de grande valia que fosse emitido alerta sonoro aos passageiros e aos órgãos controladores fazendo com que os parâmetros sejam estritamente atendidos.

REFERÊNCIAS BIBLIOGRÁFICAS

ARBACHE, Rodrigo. História do elevador. 2019. Disponível em: https://meuelevador.com/historia-do-elevador-como-surgiu-o-elevador/. Acesso em: 25 set. 2022.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. ABNT NBR 16042: Elevadores elétricos de passageiros. 1 ed. Rio de Janeiro: Abnt, 2013. 176 p.

A TORRADEIRA Internet. s. d. Disponível em: https://www.livinginternet.com/i/ia_myths_toast.htm. Acesso em: 05 nov. 2022.

BEHERA, Girija Shankar. Face Detection With Haar Cascade. 2020. Disponível em: https://towardsdatascience.com/face-detection-with-haar-cascade-727f68dafd08. Acesso em: 30 nov. 2022.

BERTOLETI, Pedro. OpenCv. 20202. Disponível em: https://www.newtoncbraga.com.br/index.php/microcontroladores/143tecnologia/17799-opencv-o-que-e-onde-usar-e-como-instalar-na-raspberry-pi-mic412.html. Acesso em: 15 nov. 2022.

BISNETO, Pedro Luiz O. Costa. A História da Internet. 2003. Disponível em: http://www.pedroom.com.br/portal/vitae/download/cientificos/03%20A%20Historia%20da%20Internet.pdf. Acesso em: 25 set. 2022.

BITTENCOURT, Sinésio. O que é arduino: tudo o que você precisa saber. Tudo o que você precisa saber. 2022. Disponível em: https://www.hostgator.com.br/blog/o-que-e-arduino/. Acesso em: 15 nov. 2022.

BRASIL. AGÊNCIA BRASIL. Primeiro caso de covid no Brasil completa um ano. 2021. Disponível em: https://agenciabrasil.ebc.com.br/saude/noticia/2021-02/primeiro-caso-de-covid-19-no-brasil-completa-um-ano. Acesso em: 18 set. 2022.

CASA DA ROBOTICA. Módulo Conversor. Disponível em: https://www.casadarobotica.com/placas-embarcadas/esp/shield/modulo-conversor-shield-usb-serial-ch340-para-esp32-cam-esp32. Acesso em: 18 nov. 2022.

CASCADE Classifier. 2022. Disponível em: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html. Acesso em: 29 nov. 2022.

COVID-19: dia em que mais brasileiros morreram foi em 29 de março. 2021. Disponível em: https://www.poder360.com.br/coronavirus/covid-19-dia-em-que-mais-brasileiros-morreram-foi-em-29-de-marco/. Acesso em: 18 set. 2022.

ELECTRONICS, How To. Detecção e identificação de objetos baseados em CAM ESP32 com OpenCV. Detecção e identificação de objetos com câmera ESP32 e OpenCV. 2022. Disponível em: https://how2electronics.com/esp32-cam-based-object-detection-identification-with-opencv/. Acesso em: 11 nov. 2022.

ENGETAX. Como funciona um elevador. 2016. Disponível em: https://engetax.com.br/como-funciona-o elevador/#:~:text=Ele%20utiliza%20o%20mesmo%20princ%C3%ADpio,%C3%A9%20necess%C3%A1ria%20para%20o%20deslocamento.. Acesso em: 30 nov. 2022.

ESP32-CAM AI-Thinker Pinout Guide: GPIOs Usage Explained. GPIOs Usage Explained. [2020]. Disponível em: https://randomnerdtutorials.com/esp32-cam-ai-thinker-pinout/. Acesso em: 20 nov. 2022.

ESP32 Overview | Espressif Systems. [2015]. Disponível em: <https://www.espressif.com/en/products/socs/esp32>. Acesso em: 3 out. 2022.

EXAME. Novo estudo mostra origem do primeiro paciente de covid no mundo**.** 2021. Disponível em: https://exame.com/mundo/estudo-origem-covid/. Acesso em: 18 set. 2022.

FABRO, Nathalia. Elevadores: 10 dúvidas, mitos e verdades sobre o equipamento. 10 dúvidas, mitos e verdades sobre o equipamento. 2021. Disponível em: https://revistacasaejardim.globo.com/Curiosidades/noticia/2021/09/elevadores-10-duvidas-mitos-e-verdades-sobre-o-equipamento.html. Acesso em: 15 nov. 2022.

HARES. A história das câmeras de vigilância. [2022]. Disponível em: https://www.haresconsultoria.com.br/a-historia-das-cameras-de-vigilancia/. Acesso em: 25 set. 2022.

HOASHI, Paulo T.. Introdução a visão computacional: imagem. Santo André: -, 2021. 13 slides, color.

INTRODUÇÃO AO VISUAL STUDIO CODE. -: Net Magazine, v. 127, 2016. Mensal. Disponível em: https://www.devmedia.com.br/introducao-ao-visual-studio-code/34418. Acesso em: 15 nov. 2022.

MÓDULO ESP32. 2021. Disponível em: https://electropeak.com/learn/wp-content/uploads/2021/08/ESP32Pinout-FullPinout.png. Acesso em: 29 nov. 2022.

MONTE, Caio. Surgimento do vídeo. -, [2014]. 11 slides, color. Disponível em: https://slideplayer.com.br/slide/1355904/. Acesso em: 20 dez. 2022.

MULINARI, Bruna. Numpy Python. [2022]. Disponível em: https://harve.com.br/blog/programacao-python-blog/numpy-python-o-que-e vantagens-e-tutorial-inicial/. Acesso em: 15 nov. 2022.

NELSON, Phil. Face Detection Cascade Classifier Vs Yunet. 2022. Disponível em: https://opencv.org/opencv-face-detection-cascade-classifier-vs-yunet/. Acesso em: 30 nov. 2022.

ORTEP. Guia do CFTV: tudo o que você precisa saber. [2022]. Disponível em: https://www.ortep.com.br/guia-do-cftv-tudo-o-que-voce-precisa-saber/. Acesso em: 25 set. 2022.

OTAVIO CHASE. Sistemas embarcados. 2007. Disponível em: http://www.lyfreitas.com.br/ant/pdf/Embarcados.pdf. Acesso em: 25 out. 2022.

PUC GOIAS (Goias). Internet das coisas. 2020. Disponível em: https://ead.pucgoias.edu.br/blog/internet-das-coisas. Acesso em: 18 set. 2022.

PYTHON. Módulos de manipulação de URL. Disponível em: https://docs.python.org/ptbr/3/library/urllib.html#:~:text=urllib%20%C3%A9%20um%20pacote%20que,para%20abrir%20e%20ler%20URLs. Acesso em: 15 nov. 2022.

Remessa Online, 26 out. 2021. Mensal. Disponível em: https://www.remessaonline.com.br/blog/visual-studio-code-confira-as-principais-funcoes-da-ferramenta/. Acesso em: 15 nov. 2022.

ROBOCORE. ESP32-CAM - ESP32 com Câmera. [2022]. Disponível em: <https://www.robocore.net/wifi/esp32-cam-esp32-com-camera>. Acesso em: 3 out. 2022.

SAMORA, Hélio. Como surgiu e qual a utilidade do IoT. 2020. Disponível em: https://www.industria40.ind.br/artigo/20246-como-surgiu-e-qual-a-utilidade-da-iot. Acesso em: 18 set. 2022.

SANTIAGO JUNIOR, Luiz. Entendendo a biblioteca Numpy. 2018. Disponível em: https://medium.com/ensina-ai/entendendo-a-biblioteca-numpy-4858fde63355. Acesso em: 15 nov. 2022.

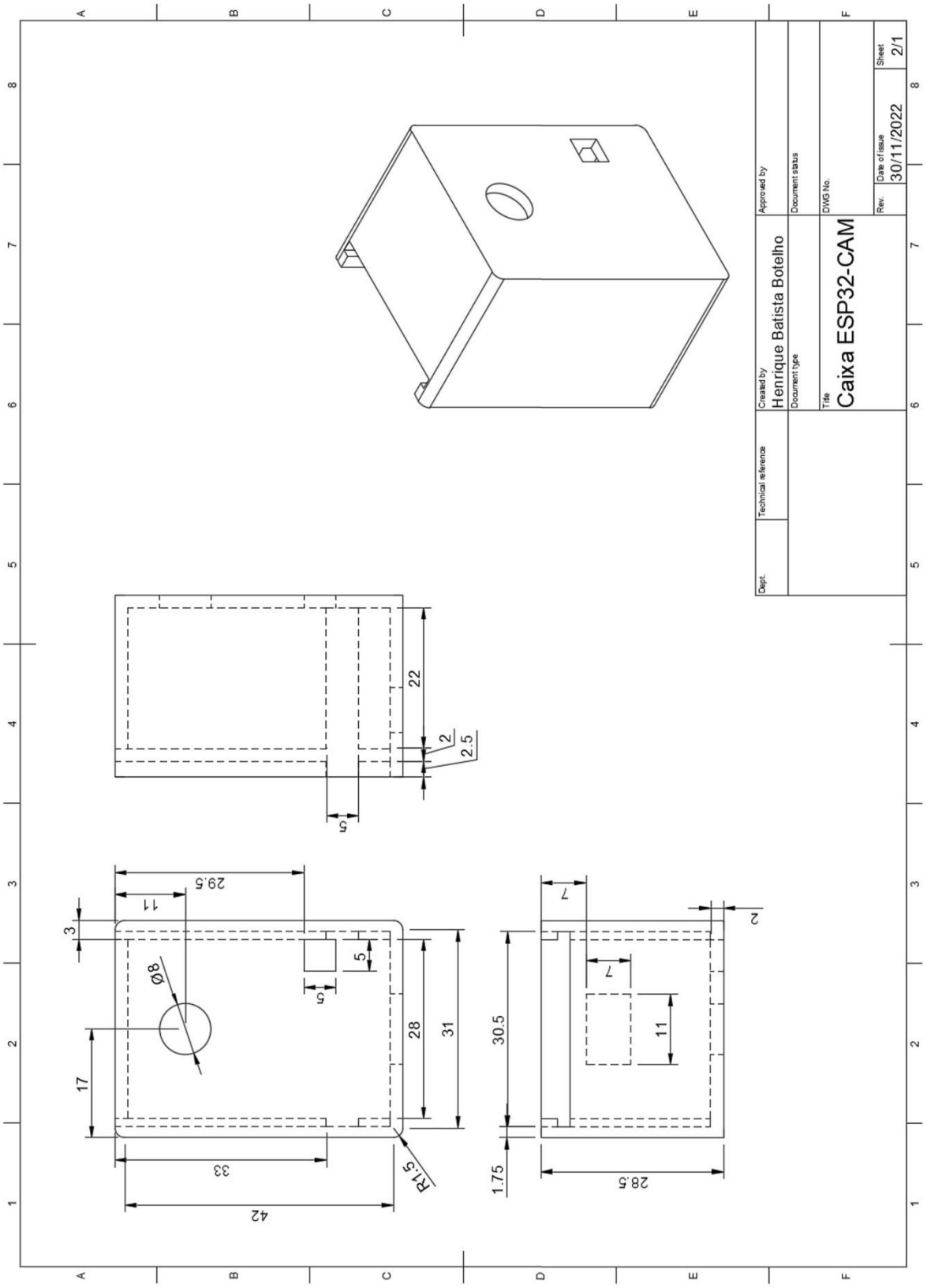
SANTOS, Rui. ESP32-CAM Video Streaming Web Server. 2016. Disponível em: https://randomnerdtutorials.com/esp32-cam-video-streaming-web-server-camera-home-assistant/. Acesso em: 02 set. 2022.

SANTOS, Vanessa Sardinha dos. Covid-19. [2021]. Disponível em: https://mundoeducacao.uol.com.br/doencas/covid-19.htm. Acesso em: 18 set. 2022

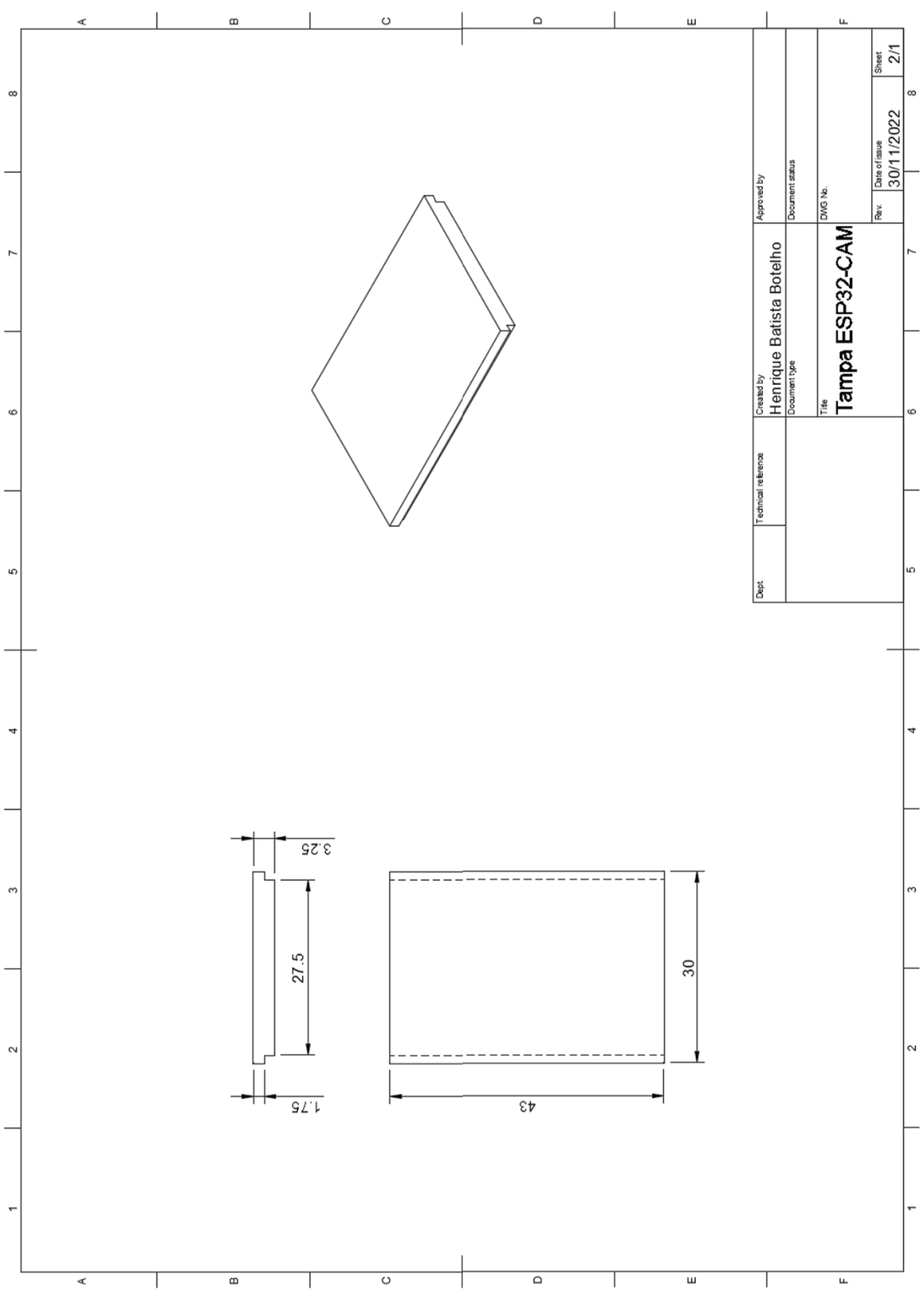
SILVA, Danilo Morais da. Python: História e Ascendência. Programar, [s. l], v. 59, n. -, p. 96-99, fev. 2018. Mensal. Disponível em: Revista_PROGRAMAR_59. Acesso em: 06 nov. 2022.

SILVA, Gizele. Oque é biblioteca. 2022. Disponível em https://coodesh.com/blog/dicionario/oquebiblioteca/#:~:text=Ela%20%C3%A9%20uma%20cole%C3%A3o%20de,mais%20agilidade%20e%20menos%20erros. Acesso em: 15 nov. 2022.

APÊNDICE A – CAIXA PARA O ESP32-CAM



APÊNDICE B – TAMPA TRASEIRA



APÊNDICE C – PROGRAMA CAMERAWEBSERVER

```
#include "esp_camera.h"
#include <WiFi.h>

//
// WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
//            Ensure ESP32 Wrover Module or other board with PSRAM is selected
//            Partial images will be transmitted if image exceeds buffer size
//

// Select camera model
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM

#include "camera_pins.h"

const char* ssid = "Henrique";
const char* password = "Henrique";

void startCameraServer();

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;

  // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
  //                     for larger pre-allocated frame buffer.
  if(psramFound()){
    config.frame_size = FRAMESIZE_UXGA;
    config.jpeg_quality = 10;
    config.fb_count = 2;
  } else {
    config.frame_size = FRAMESIZE_SVGA;
    config.jpeg_quality = 12;
    config.fb_count = 1;
  }

#if defined(CAMERA_MODEL_ESP_EYE)
  pinMode(13, INPUT_PULLUP);
  pinMode(14, INPUT_PULLUP);
#endif

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit saturated
  if (s->id.PID == OV3660_PID) {
    s->set_vflip(s, 1); // flip it back
    s->set_brightness(s, 1); // up the brightness just a bit
    s->set_saturation(s, -2); // lower the saturation
  }
  // drop down frame size for higher initial frame rate
  s->set_framesize(s, FRAMESIZE_QVGA);

#if defined(CAMERA_MODEL_M5STACK_WIDE) || defined(CAMERA_MODEL_M5STACK_ESP32CAM)
  s->set_vflip(s, 1);
  s->set_hmirror(s, 1);
#endif

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
  // put your main code here, to run repeatedly:
  delay(10000);
}
```

APENDICÊ D – CÓDIGO PARA O ESP32

```
#include <WebServer.h>
#include <WIFI.h>
#include <esp32cam.h>

const char* WIFI_SSID = "Henrique";
const char* WIFI_PASS = "Henrique";

WebServer server(80);


static auto loRes =
esp32cam::Resolution::find(320, 240);
static auto midRes =
esp32cam::Resolution::find(350, 530);
static auto hiRes =
esp32cam::Resolution::find(800, 600);
void serveJpg()
{
  auto frame = esp32cam::capture();
  if (frame == nullptr) {
    Serial.println("CAPTURE FAIL");
    server.send(503, "", "");
    return;
  }
  Serial.printf("CAPTURE OK %dx%d
%db\n", frame->getWidth(), frame-
>getHeight(),
                static_cast<int>(frame->size()));

  server.setContentLength(frame->size());
  server.send(200, "image/jpeg");
  WiFiClient client = server.client();
  frame->writeTo(client);
}

void handleJpgLo()
{
  if
(!esp32cam::Camera.changeResolution(loR
es)) {
    Serial.println("SET-LO-RES FAIL");
  }
  serveJpg();
}

void handleJpgHi()
{
  if
(!esp32cam::Camera.changeResolution(hiR
es)) {
    Serial.println("SET-HI-RES FAIL");
  }
  serveJpg();
}

void handleJpgMid()
{
  if
(!esp32cam::Camera.changeResolution(mi
dRes)) {
    Serial.println("SET-MID-RES FAIL");
  }
  serveJpg();
}


void  setup(){
  Serial.begin(115200);
  Serial.println();
  {
    using namespace esp32cam;
    Config cfg;
    cfg.setPins(pins::AiThinker);
    cfg.setResolution(hiRes);
    cfg.setBufferCount(2);
    cfg.setJpeg(80);

    bool ok = Camera.begin(cfg);
    Serial.println(ok ? "CAMERA OK" :
"CAMERA FAIL");
  }
  WiFi.persistent(false);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASS);
  while (WiFi.status() != WL_CONNECTED)
{
    delay(500);
  }
  Serial.print("http://");
  Serial.println(WiFi.localIP());
  Serial.println("  /cam-lo.jpg");
  Serial.println("  /cam-hi.jpg");
  Serial.println("  /cam-mid.jpg");

  server.on("/cam-lo.jpg", handleJpgLo);
  server.on("/cam-hi.jpg", handleJpgHi);
  server.on("/cam-mid.jpg", handleJpgMid);

  server.begin();
}

void loop()
{
  server.handleClient();
}
```