UNIVERSIDADE PAULISTA

CAIO DUARTE MARINO	**N6824G6**
DANIEL NOGUEIRA DA SILVA	**N661666**
DOUGLAS CORREA RIBEIRO	**F1793C3**
KAREN PINHEIRO FERNANDES	**N549734**
LUCAS CARDOSO DE MENEZES	**N478438**
MIKAEL BRENNER S ALEXANDRE	**N604786**

IDENTIFICAÇÃO DE PATOLOGIAS ASFÁLTICAS POR MEIO DA VISÃO COMPUTACIONAL

SÃO PAULO

2024

CAIO DUARTE MARINO	**N6824G6**
DANIEL NOGUEIRA DA SILVA	**N661666**
DOUGLAS CORREA RIBEIRO	**F1793C3**
KAREN PINHEIRO FERNANDES	**N549734**
LUCAS CARDOSO DE MENEZES	**N478438**
MIKAEL BRENNER S ALEXANDRE	**N604786**

IDENTIFICAÇÃO DE PATOLOGIAS ASFÁLTICAS POR MEIO DA VISÃO COMPUTACIONAL

Trabalho de conclusão de curso para obtenção do título de graduação em Engenharia de Computação apresentado à Universidade Paulista – UNIP.

Orientador: Prof. Me. Ricardo Bacci

SÃO PAULO

2024

CAIO DUARTE MARINO	**N6824G6**
DANIEL NOGUEIRA DA SILVA	**N661666**
DOUGLAS CORREA RIBEIRO	**F1793C3**
KAREN PINHEIRO FERNANDES	**N549734**
LUCAS CARDOSO DE MENEZES	**N478438**
MIKAEL BRENNER S ALEXANDRE	**N604786**

IDENTIFICAÇÃO DE PATOLOGIAS ASFÁLTICAS POR MEIO DA VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso para obtenção do título de graduação em engenharia de computação apresentado à Universidade Paulista – UNIP.

Orientador: Prof. Me. Ricardo Bacci

Aprovado em:

BANCA EXAMINADORA

_____/ /_____

Prof.

_____/ /_____

Prof.

_____/ /_____

Prof.

CIP - Catalogação na Publicação

IDENTIFICAÇÃO DE PATOLOGIAS ASFÁLTICAS POR MEIO DA VISÃO COMPUTACIONAL / Lucas cardoso de menezes...[et al.]. - 2024.
51 f. : il. color + Banner e slides.

Trabalho de Conclusão de Curso (Graduação) apresentado ao Instituto de Ciência Exatas e Tecnologia da Universidade Paulista, São Paulo, 2024.
Área de Concentração: Engenharia de infraestrutura.
Orientador: Prof. Dr. Ricardo Bacci.

1. Visão computacional. 2. Patologias asfálticas. 3. Engenharia de infraestrutura. 4. Pavimentos flexiveis. 5. Inteligencia artificial. I. de menezes, Lucas cardoso. II. Bacci, Ricardo (orientador).

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da Universidade Paulista com os dados fornecidos pelo(a) autor(a).

RESUMO

A visão computacional é uma área dentro de inteligência artificial que possui grande impacto em vários setores da sociedade. A capacidade da máquina enxergar objetos de interesse em uma imagem e produzir uma resposta de classificação ou detecção de elementos é de suma importância no contexto de automação. As técnicas de visão computacional permitem, através de etapas de tratamento de imagem e do uso de classificadores, oferecer respostas a diversos problemas que se apresentam.

Esta monografia objetiva, levantar, diagnosticar e propor tratamento de patologias asfálticas por meio da visão computacional, que analisará imagens dos mais diversos pavimentos, para identificação de patologias presentes no mesmo, com o intuito de informar o melhor tratamento, assim criando um cenário mais eficaz e pragmático no trabalho de manutenção de vias de pavimento.

Pois, no Brasil, o transporte de cargas e passageiros ocorre predominantemente através de rodovias de pavimento flexível, sendo 60% de toda mercadoria brasileira utilizando o transporte rodoviário, o que torna o país dependente das mesmas com boas condições. É preciso redefinir prioridades, e implementar medidas que garantam de fato a recuperação da economia, a retomada da oferta de empregos e a segurança em todos os sentidos. Dessa forma, faz-se necessário melhorar a qualidade de nossas rodovias, reduzindo, assim, os acidentes e as mortes, o consumo de combustível e o desgaste dos veículos, entre outros desperdícios. É preciso determinar causas patológicas do pavimento urbano, verificando técnicas para caracterização das condições de superfície dos pavimentos, condições de drenagem e tráfego atuante. Buscando diagnósticos que permita estabelecer as condições reais de conservação das vias, através da realização de levantamentos visuais, dando enfoque na posição do poder público diante do problema.

Palavras-chave: Visão computacional; patologias asfálticas; diagnósticos

ABSTRATIC

Computer vision is an area within artificial intelligence that has a major impact on various sectors of society. The machine's ability to see objects of interest in an image and produce a classification or element detection response is of paramount importance in the context of automation. Computer vision techniques make it possible, through image processing steps and the use of classifiers, to provide answers to various problems that arise.

This monograph aims to survey, diagnose and propose treatment for asphalt pathologies using computer vision, which will analyze images of a wide variety of sidewalks in order to identify the pathologies present, with the aim of informing the best treatment, thus creating a more effective and pragmatic scenario in the maintenance of sidewalk roads.

In Brazil, freight and passenger transportation takes place predominantly on flexible sidewalk roads, with 60% of all Brazilian goods being transported by road, which makes the country dependent on sidewalks in good condition. There is a need to redefine priorities and implement measures that really guarantee the recovery of the economy, the resumption of jobs and safety in every sense. It is therefore necessary to improve the quality of our roads, thereby reducing accidents and deaths, fuel consumption and wear and tear on vehicles, among other wastes. It is necessary to determine the pathological causes of urban sidewalks, verifying techniques for characterizing sidewalk surface conditions, drainage conditions and traffic. Diagnoses must be made to establish the real conditions of road maintenance, by carrying out visual surveys, focusing on the position of public authorities in the face of the problem.

Keywords: Computer vision; asphalt pathologies; diagnostics

LISTA DE ILUSTRAÇÕES

Figura 1 - Placa ESP32.....17
Figura 2 - Placa Raspberry Pi 19
Figura 3 - Câmera model 3 12 megapixels Raspberry Pi.....24
Figura 4- Placa Raspberry Pi 3 Model B 26
Figura 5 - Representação do algoritmo de detecção de faces 33
Figura 6 - Detecção de pessoas em sala de aula 35
Figura 7 - Placa ESP 32 em sua configuração.....40
Figura 8 - Configuração do Raspberry PI 3.....41
Figura 9 - Instalação da câmera.....41
Figura 10 - Comunicação do ESP 32 e o sensor ultrassônico 42
Figura 11 - Montagem do Hardware 44
Figura 12 - Capturando imagem.....46
Figura 13 - Identificação de uma patologia.....48

SUMÁRIO

1 INTRODUÇÃO 10
1.1 Justificativa 11
1.2 Objetivos 11
1.2.1. Objetivo Geral 11
1.2.2. Objetivos Específicos 11
1.3 Estrutura da Monografia 12
1.4 Metodologia 12
2 REFERENCIAL TEÓRICO 13
2.1 Hardware 13
2.1.1 Microcontroladores 15
2.1.2 Microcontroladores usados no projeto 17
2.1.3 Funcionamento do Raspberry Pi 20
2.1.4 Sensores 21
2.1.5 Evolução das câmeras 22
2.1.6 Câmera Raspberry Pi 24
2.2. Software 28
2.2.1. IA (Inteligência Artificial) 28
2.2.2. Python 30
2.2.3. OpenCV 32
2.2.4. HaarCascade 34
2.2.5. YOLO 36
2.2.6. Protocolo de comunicação MQTT 36
3 DESENVOLVIMENTO 38
3.1 Configuração do hardware 40
3.1.1 Configurando o ESP32 40
3.1.2 Configurando o RaspBerry PI 3 41
3.1.3 Montagem da solução 44
3.2 Configuração do fluxo de identificação das patologias 46
3.2.1 Google Storage 47
3.2.2 VertexAI 49
3.2.3 Modelo de Identificação com YOLO 50
4 CONCLUSÃO 51
5 REFERÊNCIAS BIBLIOGRÁFICAS 53

1 INTRODUÇÃO

No concerto da modernidade, onde a inteligência artificial rege a batuta do progresso, a visão computacional desponta como uma virtuosa solista, tecendo melodias de eficiência e engenhosidade para as cidades inteligentes. Essa área da inteligência artificial, dotada da maestria de extrair informações de imagens, compõe uma sinfonia de possibilidades para as metrópoles do futuro.

No compasso da automação, a visão computacional se apresenta como uma prima bailarina, dotando as máquinas da capacidade de enxergar e interpretar o mundo ao seu redor. Imagens se transformam em partituras repletas de dados, permitindo a identificação de objetos, a classificação de elementos e a geração de respostas precisas a problemas diversos.

Em nosso solo brasileiro, onde o transporte pulsa nas veias das rodovias, a visão computacional se apresenta como um maestro talentoso, pronto para reger a sinfonia da manutenção asfáltica. Através da análise minuciosa de imagens de pavimentos, essa tecnologia mapeia patologias com maestria, indicando o tratamento ideal para cada caso. Essa melodia de eficiência garante a longevidade das estradas, reduz custos e compõe um futuro mais seguro para o transporte.

Assim, a visão computacional se ergue como uma virtuosa solista no concerto das cidades inteligentes, tecendo melodias de eficiência, sustentabilidade, segurança, inclusão e bem-estar. Essa tecnologia rege a sinfonia do progresso, compondo um futuro mais próspero e harmonioso para as metrópoles do mundo.

1.1 Justificativa

A escolha do tema fundamenta-se na relevância e na urgência de abordar questões relacionadas à preservação de vias cardeais das cidades. O tipo de via flexível apresenta condições específicas que podem impactar negativamente a integridade das vias.

Ao abordar essa temática, busca-se contribuir para o avanço do conhecimento na área de engenharia, com foco em soluções práticas e sustentáveis para mitigar os efeitos adversos causado pelas patologias, Souza, R. A. et al. (2020) Adptado.

1.2 Objetivos

1.2.1. Objetivo Geral

Esta monografia tem por objetivo realizar um levantamento, fornecer a identificação das patologias asfálticas através do uso da tecnologia da visão computacional.

1.2.2. Objetivos Específicos

Para se atingir o Objetivo Geral, é necessário:

- Realizar uma revisão bibliográfica, em forma de um referencial teórico, sobre as tecnologias que serão utilizadas, abordando aspectos técnicos;
- Propor soluções técnicas, sustentáveis e economicamente viáveis para as patologias identificadas, considerando as especificidades de cada caso estudado; e
- Avançar no conhecimento sobre o tema, por meio da documentação dos resultados, da discussão crítica sobre as práticas adotadas e das recomendações para futuros estudos e intervenções.

1.3 Estrutura da Monografia

Esta Monografia é composta por dois capítulos, a saber:

- O primeiro capítulo trata da introdução do tema de estudo, destacando sua importância a justificativa, os objetivos e a sistematização da pesquisa;
- O segundo capítulo relata o referencial teórico de forma geral, informando as tecnologias que serão usadas para o protótipo.

1.4 Metodologia

A monografia se vale de uma revisão bibliográfica, em forma de um Referencial Teórico, sobre os hardwares e softwares que serão utilizados no protótipo.

De acordo com Cervo e Bervian (2005, p. 60) a pesquisa bibliográfica procura explicar um problema a partir de referências publicadas em artigos, livros, dissertações e teses. Pode ser realizada independentemente ou como parte da pesquisa descritiva ou experimental. Em ambos os casos, busca-se conhecer e analisar as contribuições culturais ou científicas do passado sobre determinado assunto, tema ou problema (CERVO; BERVIAN, 2005, p. 60).

2 REFERENCIAL TEÓRICO

2.1 Hardware

O hardware é a parte física de um computador ou dispositivo eletrônico, seu funcionamento é essencial para processar informações, armazenar dados e realizar demandas de qualquer tarefa. Trabalhando em conjunto com o software executando as instruções lógicas, que é a parte lógica composta por sistema operacional, programas e aplicativos.

A história do hardware surgiu com os seus primeiros dispositivos no Ábaco, na Mesopotâmia, onde utilizava contas em hastes para realizar cálculos básicos. Essa ferramenta milenar foi fundamental para o desenvolvimento do comércio e da matemática.

No século XVII, Blaise Pascal deu um passo importante na criação da máquina de Pascal, uma calculadora mecânica que facilitava cálculos complexos. Já em meados do século XIX, Charles Babbage concedeu a Máquina Analítica, considerada a precursora dos computadores modernos, com a capacidade de processar dados e armazenar informações em cartões perfurados.

Dando um salto no tempo, a década de 1940 marcou o início da era das válvulas, com a construção do ENIAC (*Electronic Numerical Integrator and* Computer*)*, o primeiro computador eletrônico para uso geral. Apesar do seu tamanho colossal e fragilidade, o ENIAC abriu caminho para o desenvolvimento de computadores mais poderosos e eficientes da época. Em 1947, a invenção do transistor revolucionou o Hardware, substituiu as válvulas volumosas e frágeis. Com isso, possibilitou a criação dos dispositivos menores, com mais rapidez e confiáveis, sendo eles, por exemplo: O IBM 1401, um computador de grande porte com uso de transistores na década de 1950.

Com o surgimento dos Circuitos Integrados (CIs) na década de 1960, os dispositivos eletrônicos foram miniaturizados e passaram a integrar diversos transistores em um único chip de silício. impulsionaram a criação de computadores pessoais acessíveis ao público de modo geral. Modelos como o Apple II e o IBM PC se tornaram ícones da época, popularizando e

transformando a computação da melhor forma para trabalharmos, métodos de estudos e diversão.

As últimas décadas presenciaram uma convergência de tecnologias e uma explosão na demanda por Hardware. A *World Wide Web* e a computação em nuvem transformaram a maneira de acessamos e compartilhamos informações.

O futuro do Hardware promete ainda mais miniaturização, processamento mais rápido e inteligência artificial aprimorada. E diversos dispositivos interconectados que moldaram a maneira que vivemos e nos comunicamos. Chips e softwares especializados em processamento de linguagem natural, aprendizado de máquina e visão computacional estão abrindo novos caminhos para as aplicações e inovações.

2.1.1 Microcontroladores

Um microcontrolador MCU: (*Microntroller unit*) é um computador em um único chip de circuito integrado (IC) de um semicondutor MOSFET. Na termologia moderna, é semelhante ou equivalente a um sistema em chip (SoC: *System-on-a-Chip*), pode incluir um microcontrolador como um de seus componentes. Contém uma ou mais de uma CPU junto com memória e periféricos de entrada / saída (I/O: input/output) programáveis

Os microcontroladores são projetados para aplicações embarcadas, em contraste com os microprocessadores usados em computadores pessoais ou em outras aplicações de uso geral que consiste em chips discretos. São usados em dispositivos controlados automaticamente, como sistemas de controle, eletrodomésticos, ferramentas elétricas, controles remotos, entre outros sistemas embarcados. No contexto de internet das coisas (IoT), os microcontroladores são econômicos e populares em coleta de dados, detectando e atuando no mundo físico como os dispositivos de ponta.

Alguns microcontroladores podem usar palavras de quatro bits, operando em frequências baixas como 4 KHz, para um baixo consumo de energia (miliwatts ou microwatts). Eles geralmente têm a capacidade de reter a funcionalidade enquanto aguardam um evento, como o pressionamento de um botão ou uma interrupção. O consumo de energia durante o repouso (*clock* da CPU e da maioria dos periféricos desligados) pode ser apenas nano watts, tornando muitos deles adequados para aplicações de bateria de longa duração. Outros microcontroladores podem cumprir funções de críticas em desempenho, onde podem precisar agir mais como um processador de sinal digital (DSP), com velocidades de *clock* e consumo de energia mais altos.

As origens do microprocessador e do microcontrolador são de acordo com a invenção do MOSFET. Inventado por Mohamed M. Atalla e Dawon Kahng em Bell Labs em 1959, e demonstrado pela primeira vez apenas no ano se 1960. Neste mesmo período, Atalla propôs o conceito integrado que era um chip de circuito integrado fabricado a partir do MOSFET. Onde os chips atingiram a maior densidade de transistores e menos custos de fabricação do que os chips com transistores bipolares.

Os primeiros microprocessadores *multichips*, o *Four-Phase Systems* AL1 em 1969 e o Garrett Reserach MP944 em 1970, foram desenvolvidos com diversos chips MOS LSI. O lançamento do microprocessador de único chip foi o Intel 4004 em 1970. Desenvolvido por Federico Faggin, usando sua tecnologia MOS de porta silício, juntamente com os engenheiros da Intel Marcian Hoff e Stan Mazor.

2.1.2 Microcontroladores usados no projeto

ESP32

ESP32 é uma placa de desenvolvimento de hardware aberto é um microcontrolador de baixo custo e alto desempenho energético, desenvolvido pela Espressif Systems, baseada em um processador dual-core de 32 bits e equipada com 520 KB de memória flash.

Ela se destaca por sua versatilidade, oferecendo suporte a Wi-Fi, Bluetooth e diversas outras tecnologias de conexão. Equipado com um processador dual-core de até 240 MHz, 520 KB de RAM, 4 MB de memória flash interna e uma ampla gama de periféricos, como UART, SPI, I2C e câmera, o ESP32 é perfeito para projetos de IoT, sua principal área de aplicação. Além disso, pode ser utilizado em projetos de robótica, automação residencial, entretenimento, entre outros.

Outra característica da placa é que ela conta com 34 pinos GPIO, sendo 22 digitais e 12 analógicos. Os pinos digitais podem ser configurados como entrada ou saída, enquanto os pinos analógicos são usados para ler sinais analógicos, como temperatura, tensão elétrica e pressão.

Figura 1 - Placa ESP32

Fonte: <https://www.pngwing.com/pt/search?q=esp32>

Raspberry Pi

O Raspberry é unidade processável do tamanho de um cartão de crédito, que pode ser conectado a um monitor ou televisor compatível com HDMI, é um hardware integrado de uma placa única, a principal ideia do Raspberry é ensinar computação e ciências similares através de uma computação real. É possível trabalhar com periféricos integrados a placa, que é composta por um *system on chip*, modelo que inclui um processador de 700MHz, GPU Vídeo Core IV, e 512 GB de memória RAM.

Para quem deseja montar um computador e utilizá-lo com intuito de diversas finalidades o Raspberry é ideal para atender sua demanda, podendo ser montando com monitor, mouse, teclado e outros acessórios. É importante ressaltar, que ele é menos potente do que um computador completo, ou seja, para uso mais específicos a placa não é adequada. No entanto, o minicomputador é procurado por profissionais da computação, servindo para alguns exemplos: Desenvolver robôs; exercitar programação; projetos; desenvolver um mini servidor e muitos mais.

O Raspberry Pi foi criado pela Fundação Raspberry Pi, no Reino Unido, a desenvolvedora é uma organização sem fins lucrativos que tem como principal foco a promoção da ciência por um preço acessível. O primeiro modelo do minicomputador foi lançado em 2012, e foi nomeado de Raspberry Pi 1 Model B. Ele conta com *System-on-a-Chip* (SoC) da Broadcom. A placa vinha com CPU, SDRAM, GPU, DSP e uma porta USB.

Figura 2 - Placa Raspberry Pi

Fonte: Imagem própria

Muitas versões foram lançadas depois do primeiro minicomputador, sendo eles:

- Raspberry Pi 1 Model B;
- Raspberry Pi 1 Model A;
- Raspberry Pi 1 Model B+;
- Raspberry Pi 1 Model A+;
- Raspberry Pi 2 Model B;
- Raspberry Pi Zero;
- Raspberry Pi Zero W;
- Raspberry Pi Zero 2 W;
- Raspberry Pi 3 Model B;
- Raspberry Pi 3 Model B+:
- Raspberry Pi 3 Model A+;
- Raspberry Pi 4 Model B;
- Raspberry Pi 400.

Utilizaremos o Raspberry Pi 3 Model B neste projeto.

2.1.3 Funcionamento do Raspberry Pi

O RPI é basicamente um computador estabelecido em uma placa única que conta com memória RAM, GPU, CPU, placa de vídeo e entradas como UBS e HDMI, ele não conta uma memória interna precisando de um micro SD, áudio e vídeo, GPIO de 10 pinos que é um de seus diferenciais sabendo que essa entrada facilita o controle e automação de dispositivos eletrônicos, bem como para monitores LCD e câmeras. Também conta com uma saída P2 para conexão de fones de ouvido, são componentes que possibilitam o seu funcionamento como um computador comum.

O seu sistema operacional atual é o Raspberry Pi OS, que é baseado em Debian (Linux). Outros sistemas operacionais também são compatíveis com o dispositivo, como Ubuntu, Arch Linux e Debian. O ARM do Windows funciona nas versões mais modernas do minicomputador. O dispositivo possui uma conectividade Wi-Fi, Bluetooth e uma placa de rede.

Sua alimentação varia de acordo com o modelo do minicomputador. Em resumo, o RPI pode ser carregado por uma fonte de alimentação com entrada micro USB, Usb tipo C, ou por um pino conector semelhante ao de notebooks.

2.1.4 Sensores

Os sensores são dispositivos utilizados para captação de dados provenientes de variações físicas, para cada tipo de variação é utilizado um tipo específico de sensor (Groover, 2011), fatores com temperatura, pressão e agressividade no ambiente podem influenciar nas medidas. Características como a exatidão, precisão e incerteza, a sensibilidade deve ser observada para que se tenha confiabilidade dos dados colhidos. A faixa de leitura adequada ao processo devem ser considerados para que se obtenham os dados mantendo suas devidas características. Após processar estes dados, conforme a necessidade de cada processo passe-se a ter informações relativas da etapa monitorada.

No caso do sensor de nível por radar, a exatidão é influenciada por outros fatores como o tipo de aplicação, modelo de antena, software de processamento de ecos etc.

2.1.4.1 Sensor de nível por ultrassônico:

O funcionamento do dispositivo ultrassônico se faz por meio da geração das ondas ultrassônicas a partir de transdutores piezelétricos que, possuem oscilador a cristal, sendo capaz de produzir sinais na faixa do ultrassom, para ser usado como transdutor, o quartzo e a turmalina, cristais piezelétricos naturais devem ser cortados de forma que um campo elétrico alternado, quando neles aplicado produzam variações em sua espessura. Descoberto por Pierre e Jacques Curie em 1880, o efeito piezelétrico consiste na variação das dimensões físicas de certos materiais sujeitos a campos elétricos. O contrário também ocorre, ou seja, a aplicação de pressões.

Por exemplo, pressões acústicas que causam variações nas dimensões de materiais piezelétricos provocam o aparecimento de campos elétricos nele. Sua aplicação em sensores de nível que utilizam ultrassom por SONAR, também conhecido por técnica do eco, são muito utilizados em plantas industriais e tem funcionamento similar ao utilizado pelos sensores de micro-ondas por radar. Utilizando a modulação por chaveamento da portadora, cuja frequência é na faixa de ultrassom de 16kHz a 20MHz, que propagam tomando por referência a velocidade do som 344 m/s no ar a 20°C.

2.1.5 Evolução das câmeras

As câmeras de monitoramento percorrem um longo vinho desde suas origens até os sistemas sofisticados de hoje. Essa jornada é marcada por avanços tecnológicos que transformaram a forma de como vigiamos e protegemos o mundo ao nosso redor.

Em 1946 Marie Van Der Zande cria o primeiro sistema e nasce a vigilância por vídeo. Na década de 1950 são criadas as câmeras analógicas de CFTV (Circuito Fechado de Televisão) se tornam populares para o uso industrial e militar.

No ano de 1970 chamado por Armazenamento e Popularização o advento das fitas cassetes é bem informativo e permite a gravação de imagens impulsionando o uso do CFTV em ambientes residenciais e comerciais. Já na década de 1980 foram criadas câmeras com avanços de luminosidade, isto é, microchips possibilitam a captação de imagens em ambientes com pouca luz. Na década seguinte evoluímos com as câmeras blindadas e infravermelhos, onde são introduzidas por sistemas compactos permitindo a gravação de imagens por várias câmeras simultaneamente.

No início do século XXI as câmeras oferecem maior nitidez, como um zoom aprimorado e armazenamento em discos rígidos. Os custos dos sistemas de segurança diminuem, tornando-os acessíveis pata um público mais amplo. Em 2010, o reconhecimento facial se torna uma grande realidade, iniciando em empresas e bancos.

Automação e Visão Computacional se faz presente de 2020 até o momento atual, fazendo integração de câmeras com inteligência artificial, possibilitando alguns recursos como:

- Reconhecimento fácil e de objetos;
- Análise de comportamento;
- Alerta em tempo real para ameaças;
- Visão computacional avançada para rastreamento de movimentos e detecção de anomalias;
- Zonas de segurança virtuais;
- Armazenamento em nuvem e integração com outros sistemas de segurança;
- Aprendizado de máquina, *Deep Learning* e IA Embarcada;
- Análise preditivas aprimorada para prever crimes e incidentes;
- Automação inteligente para tomadas de decisões autônomas em emergências;
- Segurança cibernética robusta contra-ataques hackers;
- Personalização e customização para atender às necessidades e orçamentos específicos de cada usuário.

A evolução das câmeras de monitoramento demonstra a busca incessante por segurança e eficiência. Com novas tecnologias no horizonta, o futuro promete sistemas ainda mais inteligentes, autônomos e personalizados, moldando o panorama da vigilância e proteção nos próximos anos.

2.1.6 Câmera Raspberry Pi

Existem agora vários módulos de câmeras oficiais do Raspberry Pi. O modelo original de 5 megapixels foi lançado em 2013, foi seguido por um módulo de câmera de 8 megapixels 2 que foi lançado em 2016. Todas essas câmeras vêm em versões de luz visível e infravermelho, enquanto o Módulo de Câmera 3 também vem como um modelo FoV padrão ou largo para um total de quatro variantes diferentes.

Além disso, uma câmera de alta qualidade de 12 megapixels com variantes de montagem CS ou M12 para uso com lentes externas foi lançada em 2020 e 2023, respectivamente.

A Câmera Módulo 3 é o modelo mais recente que foi lançada, possuindo 12 megapixels, será utilizada em nosso projeto para capturar uma boa imagem sobre patologia. Ela é compatível coma nossa placa de Raspberry Pi 3 model B, no qual já possui sua própria entrada para conexão e configuração.

Figura 3 - Câmera model 3 12 megapixels Raspberry Pi

Fonte: <https://www.raspberrypi.com/products/camera-module-3/ >

Para instalar a câmera, o cabo flexível é inserido no conector rotulado Câmera no Raspberry Pi, que está localizado entre as portas Ethernet e HDMI. O cabo deve ser inserido com os contatos prateados voltados para a porta HDMI. Para abrir o conector, pixels as guias na parte superior do conector para cima e, em seguida, em direção à porta Ethernet. O cabo fez deve ser inserido firmemente no conector, com cuidado para não dobrar o *flex* em um ângulo muito agudo. Para fechar o conector, empurre a parte superior do conector em direção à porta HDMI e para baixo, mantendo o cabo flexível no lugar.

Dependendo do modelo da câmera, pode vir com um pequeno pedaço de filme plástico azul translucido cobrindo a lente. Ele vem pata uma forma de proteção a lente enquanto está sendo enviada.

Para preparar o Software: Antes de prosseguir, recomenda-se garantir que o firmware da GPU e os aplicativos estejam todos atualizados, mantendo as instruções do sistema operacional. Em seguida, siga as instruções relevantes do software libcamera e a biblioteca Python e Picamera2.

2.1.7 Raspberry Pi 3 model B

O modelo do Raspberry Pi 3 model B, atende as necessidades do projeto por vantagens que vão nos ajudar com as patologias e fissuras asfálticas. Ele é um pequeno computador, poderoso e extremamente acessível.

O Raspberry Pi 3 model B, tem adaptador Wi-Fi e Bluetooth 4.1, evitando que tenha que adicionar por fora adaptadores assim liberado as portas USB para outras funcionalidades, como por exemplo a ligação de um HD externo.

Essa placa possui 1G de memória RAM, com adaptador para cartão micro SD e GPU Videocore IV 3D. O Raspberry Pi 3 mantém total compatibilidade com o Raspberry Pi 2, não somente em termos de aplicação como também em layout, já que temos a posição de todos os conectores. Com ele podemos rodar distribuições Linux como o Raspbian e Ubuntu, além do Windows 10 IoT.

Figura 4- Placa Raspberry Pi 3 Model B

Fonte: Imagem própria

Especificações Técnicas:

- Raspberry Pi 3
- Modelo: B
- Marca: Raspberry Pi
- Broadcom BCM2837, QUAD Core 64-bit ARMv8 Cortex-A53 1.2GHz
- 1 GB RAM
- Rede BCM43438 wireless 802.11n Lan e Bluetooth 4.1 (BLE) integrados
- 100 base Ethernet
- GPIO header de 40 pinos
- HDMI Full Size
- 4 USB 2.0
- CSI câmera port para conectar a uma câmera Raspberry Pi
- DSI display port para conectar a um display touch-screen Raspberry Pi
- Saída Estéreo 3.5mm 4 polos e vídeo composto
- Micro SD Slot – pata instalação do sistema e armazenamento de dados
- 5V/2.5 A DC Power input via Micro USB
- Material: Metais / Placa de Fenolite / Componentes Eletrônicos
- Cor: Verde
- Origem:UK
- Tamanho: 85 mm Largura x 56mm Profundidade x 17mm Altura
- Peso: 65g

2.2. Software

Software é um conjunto de instruções, dados ou programas utilizados para operar computadores e executar tarefas específicas. Diferente do hardware, que se refere aos componentes físicos de um computador, o software é intangível e atua como o cérebro do sistema, ditando como o hardware deve funcionar. Existem várias categorias de software, cada uma desempenhando funções distintas. O software de sistema, por exemplo, inclui sistemas operacionais como Windows, macOS e Linux, que gerenciam os recursos de hardware e fornecem serviços essenciais para outros softwares, funcionando como uma ponte entre o usuário, o hardware e outros aplicativos.

Além disso, há o software de aplicação, composto por programas que permitem aos usuários realizar tarefas específicas, como processadores de texto, planilhas, navegadores de internet e softwares de edição de imagem. Estes são projetados para ajudar os usuários a cumprir funções específicas de produtividade ou entretenimento. Outro tipo crucial é o software de desenvolvimento, que inclui ferramentas usadas por programadores para criar, depurar, manter e dar suporte a outros softwares e aplicações, como ambientes de desenvolvimento integrado, linguagens de programação e frameworks.

2.2.1. IA (Inteligência Artificial)

O termo "Inteligência Artificial" foi criado por John McCarthy, um cientista da computação americano. Ele cunhou o termo em 1956 durante a Conferência de Dartmouth, que é amplamente considerada como o ponto de partida formal para o campo da IA.

Basicamente, a inteligência artificial (IA) é um campo da ciência da computação que se concentra na criação de sistemas capazes de realizar tarefas que normalmente requerem inteligência humana. Atividades como reconhecimento de fala, tomada de decisão, tradução de idiomas, e reconhecimento de padrões. A IA utiliza algoritmos avançados, redes neurais artificiais e aprendizado de máquina para analisar grandes volumes de dados, aprender com eles e fazer previsões ou tomar decisões baseadas nessas análises.

“Os algoritmos são a força vital da inteligência artificial, definindo a lógica pela qual os problemas são resolvidos.” (RUSSEL, Stuart. NORVIG, Peter Inteligência Artificial: Uma Abordagem Moderna. 2010)

Existem dois tipos principais de IA: A IA fraca - ou estreita - que é projetada para realizar uma tarefa específica, como assistentes virtuais (Siri ou Alexa) ou sistemas de recomendação (Netflix, Amazon), e IA forte - ou geral - que é uma forma hipotética de IA capaz de realizar qualquer tarefa cognitiva humana. A IA tem aplicações em diversos campos, como saúde, finanças, transporte e entretenimento, e continua a evoluir rapidamente, levantando também discussões sobre ética, privacidade e impacto no emprego.

Para nosso projeto, a IA tem exercido uma influência profunda no campo do reconhecimento de imagem, transformando a maneira como sistemas e dispositivos interpretam e analisam dados visuais. Este avanço é amplamente atribuído ao desenvolvimento de redes neurais Convolucionais (CNNs), que são uma classe de redes neurais especialmente projetadas para processamento de imagens. As CNNs automatizam a extração de características das imagens, o que resulta em um reconhecimento mais preciso e eficiente. A IA também tem sido crucial na detecção de objetos em imagens e vídeos. Algoritmos de aprendizado profundo conseguem identificar e localizar vários objetos dentro de uma única imagem, tornando esta tecnologia valiosa em áreas como vigilância, automação industrial e veículos autônomos.

2.2.2. Python

O Python é uma linguagem de programação amplamente empregada em diversos setores, incluindo aplicações web, desenvolvimento de software, ciência de dados e aprendizado de máquina (ML). Sua popularidade entre os desenvolvedores decorre de sua eficiência, facilidade de aprendizado e capacidade de operar em múltiplas plataformas. O software do Python é gratuito e se integra facilmente a diferentes sistemas, simplificando o processo de desenvolvimento.

Os benefícios do Python são diversos. Sua sintaxe básica, assemelhada ao inglês, torna os programas fáceis de compreender. Além disso, os desenvolvedores podem aumentar sua produtividade ao escrever menos linhas de código em comparação com outras linguagens. Python também se integra facilmente a outras linguagens populares, como Java, C e C++, proporcionando flexibilidade aos desenvolvedores. A comunidade Python ativa, composta por milhões de desenvolvedores em todo o mundo, oferece suporte rápido sempre que surgem problemas. Além disso, há uma variedade de recursos educacionais disponíveis online, incluindo vídeos, tutoriais, documentação e guias para desenvolvedores.

"Python é um experimento em quanta liberdade os programadores precisam. Muita liberdade e ninguém consegue ler o código dos outros; pouca liberdade e a expressividade é colocada em risco”. (ROSSUM, Guido Van)

Essa linguagem também oferece portabilidade para uma variedade de sistemas operacionais, como Windows, MacOS, Linux e Unix. Quanto às bibliotecas, elas consistem em coleções de código frequentemente utilizadas, que os desenvolvedores podem incorporar em seus programas Python para evitar a necessidade de escrever código do zero. Python é fornecido com a Biblioteca Padrão, contendo uma ampla gama de funções reutilizáveis. Além disso, existem mais de 137 mil bibliotecas Python disponíveis para diversas aplicações, abrangendo desde desenvolvimento web até ciência de dados e aprendizado de máquina (ML).

Decidimos utilizar essa linguagem pois funciona muito bem em conjunto com a biblioteca OpenCV para o reconhecimento das patologias. Primeiramente, o Python oferece uma sintaxe clara e legível, facilitando o desenvolvimento e a manutenção do código, aspecto crucial em projetos complexos como este. Além disso, a ampla gama de ferramentas fornecidas pelo OpenCV permite o processamento eficiente de

imagens e a detecção precisa de características específicas, essenciais para identificação das patologias. A combinação dessas duas tecnologias possibilita o desenvolvimento rápido de soluções desde a aquisição das imagens até a análise detalhada dos defeitos, agilizando todo o processo. Por fim, a interoperabilidade do Python com outras bibliotecas e frameworks comuns em visão computacional e aprendizado de máquina amplia ainda mais as capacidades de análise e tomada de decisões baseadas em dados.

2.2.3. OpenCV

OpenCV (Open Computer Vision Library), é uma poderosa biblioteca de código aberto utilizada para visão computacional e aprendizado de máquina. Desenvolvida para oferecer uma infraestrutura robusta e versátil, o OpenCV simplifica o processo de desenvolvimento de aplicações que envolvem análise e processamento de imagens, bem como reconhecimento de padrões em vídeos e imagens estáticas.

Uma das principais vantagens do OpenCV é a sua abrangência de funcionalidades. A biblioteca possui uma vasta gama de módulos dedicados ao processamento de imagens e vídeos, incluindo operações como filtragem, segmentação, transformações geométricas e muito mais. Além disso, essa biblioteca oferece suporte para manipulação de estruturas de dados complexas e operações de álgebra linear, por exemplo, que é fundamental para muitos algoritmos.

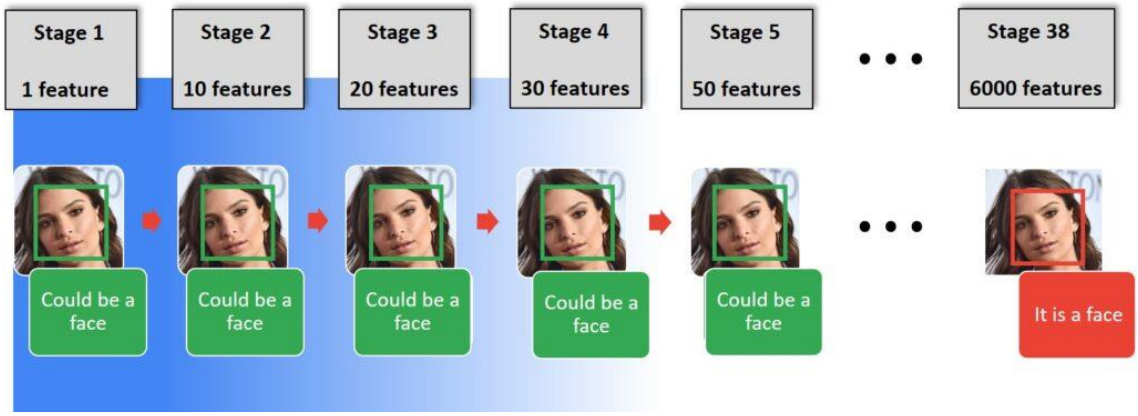
A interface gráfica do usuário (GUI) integrada ao OpenCV proporciona uma experiência de desenvolvimento mais intuitiva, permitindo a criação de aplicativos com sistemas de janelas independentes, controle de mouse e teclado, entre outras funcionalidades. Isso facilita a interação com os algoritmos de visão computacional e o desenvolvimento de aplicações visuais sofisticadas.

Além de suas capacidades técnicas impressionantes, o OpenCV é amplamente reconhecido e utilizado por empresas de renome em todo o mundo. Gigantes da tecnologia como Google, Yahoo, Microsoft, IBM e Toyota, entre outras, confiam no OpenCV para desenvolver soluções inovadoras em áreas como reconhecimento facial, identificação de objetos, análise de vídeos e muito mais. Essa adoção generalizada é um testemunho da confiabilidade e eficácia dessa biblioteca como uma ferramenta essencial para aplicações de visão computacional em diversos setores da indústria.

Entendemos que utilizar essa biblioteca para nosso algoritmo de reconhecimento de patologias asfálticas seria o ideal, porque ele oferece uma ampla gama de ferramentas eficientes para processamento de imagens, como detecção de bordas e segmentação, que vão ser essenciais para identificar fissuras e buracos. O OpenCV é altamente otimizado para desempenho, permitindo processamento rápido, e isso se encaixa perfeitamente com a proposta do projeto. Além disso, suporta várias linguagens de programação e sistemas operacionais, oferecendo flexibilidade no

desenvolvimento e na implementação. A extensa documentação e a ativa comunidade que utiliza essa biblioteca garante acesso a diversos recursos educativos e suporte contínuo, facilitando a resolução de problemas.

Figura 5 - Representação do algoritmo de detecção de faces



Fonte: <https://datahacker.rs/008-how-to-detect-faces-eyes-and-smiles-using-haar-cascade-classifiers-with-opencv-in-python/>

2.2.4. HaarCascade

HaarCascade é um algoritmo utilizado para detecção de objetos em imagens ou vídeos. Este algoritmo foi desenvolvido por Paul Viola e Michael Jones em 2001. Ele é amplamente utilizado para detecção de faces em imagens, mas também pode ser utilizado para detectar objetos como carros, motos, olhos, bocas etc.

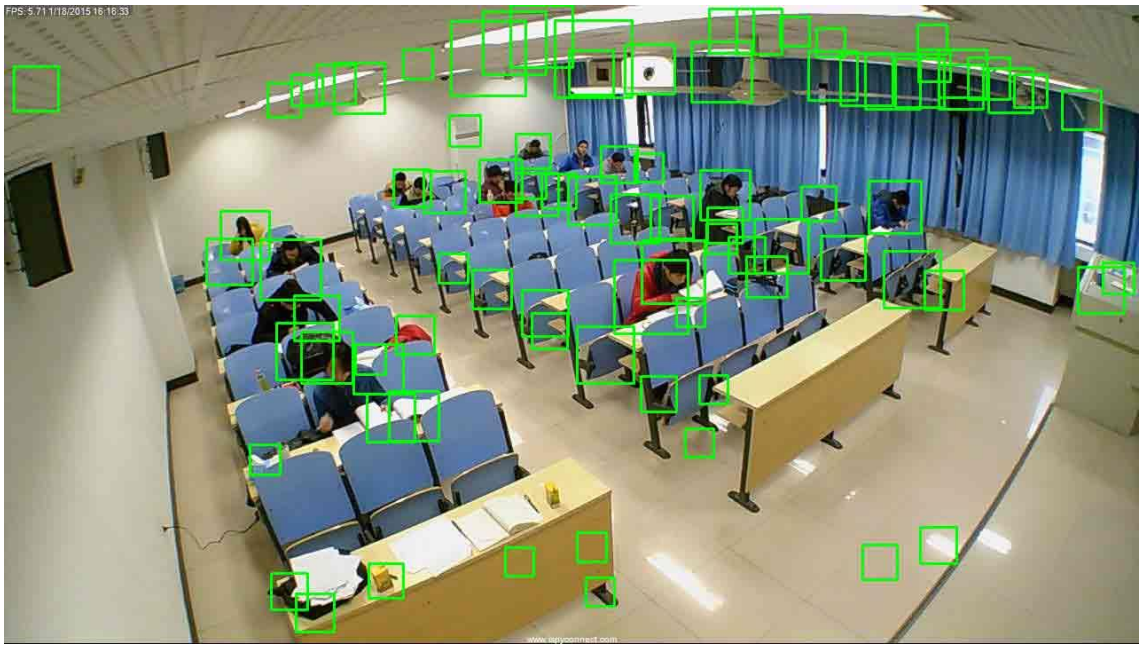
Este algoritmo funciona utilizando um conjunto de características chamadas de "Haar features", que são padrões visuais simples, como bordas, linhas e pontos de transição de intensidade em uma imagem. O algoritmo utiliza essas características para treinar um classificador que pode distinguir entre regiões da imagem que contêm o objeto de interesse.

O HaarCascade deve passar por um tipo de treinamento que geralmente envolve duas fases principais: ter um grande banco de imagens contendo objetos de interesse e outro banco que não contenha o objeto de interesse. A segunda fase consiste no treinamento de um classificador, como, por exemplo, o classificador de AdaBoost, que utiliza essas características para distinguir entre regiões que contêm o objeto de interesse e as regiões que não contêm.

Uma vez treinado, o HaarCascade pode ser utilizado para detectar os objetos de interesse em novas imagens de forma rápida e eficiente. No entanto, é importante lembrar que, para que isso aconteça, o algoritmo deve ser treinado com um grande número de imagens com e sem os objetos de interesse para essa detecção.

Decidimos usar esse algoritmo para nosso projeto devido à sua alta eficiência e precisão na detecção de padrões específicos, podendo utilizar como exemplo o nosso objetivo de detecção: patologias. Esse algoritmo é otimizado para processamento em tempo real, essencial para analisar grandes volumes de dados visuais rapidamente. Ele também é fácil de implementar, com extensa documentação e exemplos disponíveis. Além disso, ele faz parte do OpenCV que é a biblioteca que estamos utilizando, integrando-se bem com outras ferramentas de processamento de imagem, oferecendo uma solução completa e eficaz.

Figura 6 - Detecção de pessoas em sala de aula



Fonte: <https://stackoverflow.com/questions/29100541/people-detection-with-haar-cascade>

2.2.5. YOLO

Uma das ferramentas de Visão Computacional que tem ganhado considerável atenção nos últimos anos é o YOLO: *You Only Look Once*.

Desde sua introdução em 2015, o YOLO rapidamente se destacou como uma técnica inovadora. Através de uma abordagem completamente nova, ele conseguiu alcançar uma precisão igual ou superior a outros métodos de detecção de objetos da época, mas com uma velocidade de detecção significativamente maior. Enquanto as técnicas mais precisas precisavam de aproximadamente 0.5 segundos (ou mais) para processar uma imagem, o YOLO era capaz de detectar com a mesma precisão em menos de 0.05 segundos. Isso possibilitou seu uso em aplicações de tempo real, operando a uma taxa de até 30 quadros por segundo.

Outro fator que contribuiu para o sucesso do YOLO é que ele é totalmente de código aberto e sem restrições de licença. Ou seja, tanto o código-fonte, quanto a arquitetura da rede neural e os pesos pré-treinados podem ser usados por qualquer pessoa de qualquer forma.

Atualmente, essa técnica é considerada o estado da arte em detecção de objetos em tempo real. Em abril de 2020, quase cinco anos após o lançamento de sua primeira versão, foi oficialmente lançada a quarta versão do YOLO. Até a data de sua publicação, o YOLOv4 é o detector de objetos com maior acurácia capaz de ser executado em tempo real, de acordo com os testes realizados usando o DATASET MS COCO.

2.2.6. Protocolo de comunicação MQTT

MQTT (*Message Queuing Telemetry Transport*) é um protocolo de mensagens leve e assíncrono, idealizado para conectar dispositivos com recursos limitados, como sensores e atuadores, em redes de baixa largura de banda. Ele se baseia no modelo de publicação-assinatura, o que significa que os dispositivos (clientes) podem publicar mensagens em tópicos específicos, e outros clientes podem se inscrever nesses tópicos para receber as mensagens.

O mesmo funciona utilizando alguns recursos, como, o broker que é coração do sistema MQTT é o broker, um servidor que gerencia a publicação e a assinatura de mensagens. Ele mantém um registro dos tópicos e dos clientes que estão interessados em cada tópico. Em conjunto do publicador**,** um cliente que envia uma

mensagem para um tópico específico é chamado de publicador. A mensagem contém um *payload* (carga útil) com os dados a serem transmitidos. E o assinante**,** um cliente que se inscreve em um tópico para receber mensagens é chamado de assinante. Os assinantes podem especificar padrões de tópicos para receber mensagens de vários tópicos relacionados.

Suas principais características são:

- **Leveza:** Projetado para dispositivos com recursos limitados, o MQTT tem um overhead de comunicação muito baixo.
- **Assíncrono:** As mensagens são entregues de forma assíncrona, o que significa que o publicador não precisa esperar por uma confirmação de entrega.
- **Publicação-assinatura:** O modelo de publicação-assinatura desacopla os publicadores dos assinantes, permitindo uma arquitetura flexível e escalável.
- **Qualidade de serviço (QoS):** O MQTT oferece diferentes níveis de QoS para garantir a entrega das mensagens, desde a entrega "ao menos uma vez" até a entrega "exatamente uma vez".
- **Retenção de mensagens:** O broker pode reter as últimas mensagens publicadas em um tópico, permitindo que os assinantes que se conectam posteriormente recebam essas mensagens.

3 DESENVOLVIMENTO

O plano de desenvolvimento para a solução de detecção de patologias no asfalto começou com a definição dos requisitos e as ferramentas a serem utilizadas, seguidos das etapas de implementação. Inicialmente, decidimos usar o ESP32, um microcontrolador compacto, junto com um sensor ultrassônico para monitorar as variações de distância do asfalto. Esse sensor seria responsável por detectar anomalias na superfície do pavimento, com base nas variações de distância que indicam possíveis patologias. O ESP32 foi configurado para enviar essas informações para o Raspberry Pi, utilizando o protocolo MQTT, que seria o responsável por receber e interpretar esses dados, além de controlar a câmera conectada ao Raspberry.

A próxima etapa foi configurar o Raspberry Pi para, ao receber a mensagem do ESP32, disparar a captura de uma imagem através da câmera conectada. Para isso, desenvolvemos um script em Python que usa a biblioteca libcamera para capturar as imagens do asfalto sempre que um alerta de variação de distância fosse enviado. Essa captura de imagem precisava ser eficiente e com alta qualidade, para garantir que as patologias pudessem ser identificadas com precisão.

Uma vez que a imagem fosse capturada, o próximo passo seria enviá-la para o Google Storage, para que ela pudesse ser processada na nuvem. Para isso, integramos a API do Google Storage no código do Raspberry Pi, permitindo que as imagens fossem carregadas automaticamente para um diretório específico na nuvem sempre que fossem capturadas. Uma vez carregada, a imagem acionaria uma função no Google Cloud Functions, um serviço de computação sem servidor, configurado para processar as imagens assim que fossem enviadas para o Google Storage.

Essa função do Google Cloud seria responsável por chamar o modelo YOLO treinado, que foi implantado no Vertex AI. O modelo, que já foi treinado para detectar patologias específicas do asfalto, receberia a imagem carregada no Google Cloud Storage, realizaria a detecção e retornaria as patologias encontradas. A função do Google Cloud então formataria a resposta do modelo e geraria um e-mail com os resultados da análise, incluindo a imagem detectada e o tipo de patologia identificada. O envio do e-mail seria feito utilizando o serviço de envio de e-mails do Google ou por meio de uma biblioteca externa como o SendGrid, dependendo da configuração adotada para o projeto.

Esse fluxo completo envolveu a integração de diversas tecnologias e serviços, como o MQTT, a captura de imagens no Raspberry Pi, o armazenamento no Google Drive e Google Cloud Storage, o uso da Vertex AI para a detecção de patologias com YOLO, e o envio de e-mails para comunicar os resultados. Com esse planejamento, conseguimos garantir que cada etapa da solução fosse automatizada e interligada, criando um sistema eficiente e funcional para a detecção de patologias no asfalto.

3.1 Configuração do hardware

3.1.1 Configurando o ESP32

A configuração inicial envolveu a instalação da entrada "ESP32 by Espressif Systems" na IDE Arduino e a seleção da porta COM para estabelecer a comunicação com o dispositivo. Para implementar as funcionalidades do sistema, foram utilizadas as bibliotecas WiFi.h, PubSubClient.h e Arduino.h. A biblioteca WiFi.h foi responsável por gerenciar a conexão via Wi-Fi do ESP32, utilizando um SSID e uma senha definidos. Já a PubSubClient.h foi empregada para permitir a comunicação com um broker MQTT, configurado por meio de parâmetros como servidor, porta, nome de usuário, senha e tópico. Por fim, a biblioteca Arduino.h forneceu as definições e funções necessárias para o desenvolvimento do programa.

Os pinos trigPin e echoPin, definidos como 5 e 18, respectivamente, foram utilizados para controlar e realizar a leitura do sensor ultrassônico. O ESP32 teve como função principal monitorar as variações de distância captadas pelo sensor e, quando uma distância limite foi atingida, enviar essa informação ao Raspberry Pi 3 via MQTT.

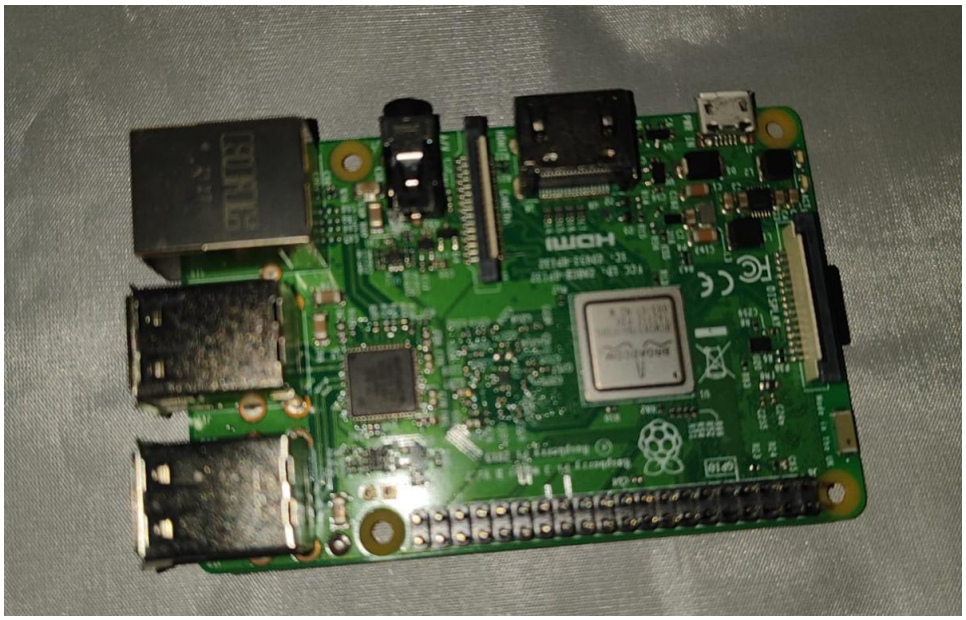
Figura 7 - Placa ESP 32 em sua configuração

Fonte: Imagem própria

3.1.2 Configurando o RaspBerry PI 3

O Raspberry Pi 3 teve início de seu funcionamento ao receber os dados enviados pelo ESP32, essas informações correspondem às variações de distância detectadas pelo sensor ultrassônico, que, ao atingir um valor limite predefinido, acionam a câmera acoplada ao Raspberry Pi. Após ser acionada, a câmera captura fotos, que são enviadas para uma pasta específica.

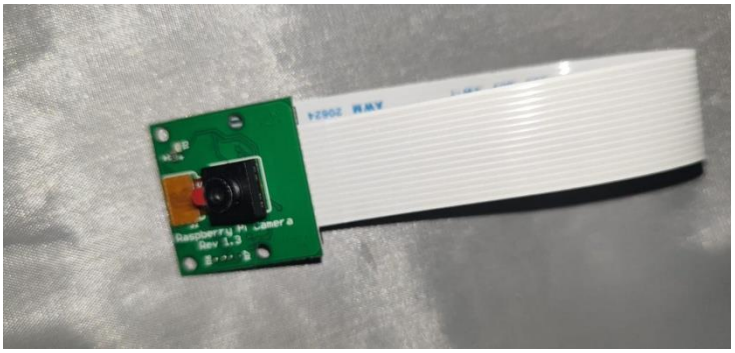
Figura 8 - Configuração do Raspberry PI 3



Fonte: Imagem própria

A instalação e configuração da câmera foram realizadas utilizando a biblioteca **libcamera**. O comando sudo apt install libcamera-apps foi utilizado para instalar os aplicativos necessários, e a captura de fotos foi implementada por meio do comando libcamera-still, chamado dentro da função take_photo.

Figura 9 - Instalação da câmera

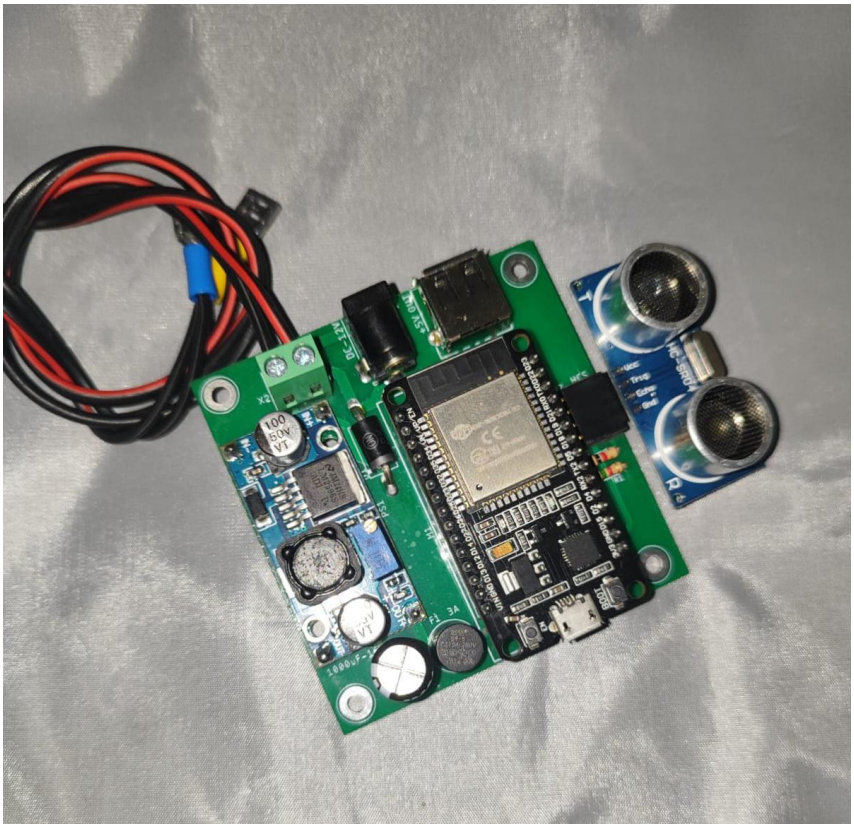


Fonte: Imagem própria

Para o funcionamento completo do sistema, foram importadas as bibliotecas **paho**, **os**, **subprocess**, **json**, **time**. Essas bibliotecas foram essenciais para manipulação de arquivos, execução de comandos, processamento de JSON, além de viabilizar a comunicação via MQTT. Variáveis globais foram definidas para gerenciar o controle de captura de fotos e as configurações de diretórios.

No fluxo de comunicação, o ESP32 publica as informações obtidas pelo sensor ultrassônico no broker MQTT. A função on_connect no Raspberry Pi estabelece a conexão com o broker, permitindo que ele receba e processe os dados publicados no tópico sensor/ultrassonic. Quando o valor limite de distância é atingido, uma *flag* é acionada, ativando a função take_photo, que utiliza o comando libcamera-still para capturar fotos.

Figura 10 - Comunicação do ESP 32 e o sensor ultrassônico



Fonte: Imagem própria

Também foi uma integração com a API do Google Drive para integrar o sistema a nuvem, garantindo autenticação segura, automação do upload de arquivos e

organização eficiente no armazenamento na nuvem. A API desempenha um papel crucial ao permitir que diferentes sistemas se comuniquem, facilitando o gerenciamento e a transferência de arquivos entre o Raspberry Pi e o Google Drive, melhorando a eficiência geral do sistema.

Para implementar a API, utilizamos a importação de bibliotecas e pacotes que possibilitam o funcionamento rápido e eficiente. Foram utilizadas as seguintes bibliotecas: os, para manipulação de arquivos e diretórios no sistema operacional; google.oauth2.service_account, para autenticação segura com o Google; googleapiclient.discovery, que permite a interação com as APIs do Google; e googleapiclient.http, responsável por manipular e realizar o upload de arquivos por meio da API HTTP do Google Drive.

Algumas variáveis globais foram definidas para gerenciar os recursos e simplificar o processo de integração. Entre elas estão: SERVICE_ACCOUNT_FILE, que armazena o caminho para o arquivo JSON usado na autenticação do serviço Google; sourceFilePath, o diretório onde estão localizadas as fotos a serem enviadas; TargetfolderId, o ID da pasta de destino no Google Drive; e SCOPES, que define as permissões necessárias para a interação com a API do Google Drive.

A função authenticate foi implementada para autenticar o sistema e retornar as credenciais necessárias para acessar a API do Google Drive. Já a função upload_file_to_drive foi projetada para enviar os arquivos para o Google Drive. Ela recebe dois argumentos: file_path, que aponta o caminho das fotos a serem enviadas, e folder_id, que identifica a pasta de destino na nuvem. Durante sua execução, a função cria um objeto de upload utilizando MediaFileUpload, envia os arquivos para o Google Drive e, ao final, imprime o ID do arquivo enviado, confirmando a conclusão do processo. Essa solução automatizada simplifica o gerenciamento de arquivos e melhora a integração do sistema com a nuvem.

3.1.3 Montagem da solução

A montagem da solução integra hardware, software e serviços em nuvem para criar um processo automatizado de detecção de patologias no asfalto. Tudo começa com o ESP32, onde o sensor ultrassônico é instalado e configurado para monitorar a superfície do asfalto. Ele é calibrado para detectar variações específicas de distância, enviando alertas ao Raspberry Pi por meio do protocolo MQTT. O Raspberry Pi, por sua vez, está conectado a uma câmera estrategicamente posicionada, configurada para capturar imagens em alta qualidade no momento em que recebe o alerta do ESP32.

Figura 11 - Montagem do Hardware

Fonte: Imagem própria

Assim que a imagem for capturada, o Raspberry Pi executa scripts do Python para processá-la e enviá-la automaticamente ao Google Drive utilizando a API oficial do serviço. No Google Drive, cada novo upload aciona uma função do Google Cloud Functions configurada para monitorar esses eventos. A função obtém a imagem e a armazena no Google Cloud Storage, um serviço essencial para a integração com a Vertex AI.

O modelo YOLO, previamente treinado e implantado na Vertex AI, é então acionado para analisar a imagem, identificando e classificando as patologias presentes. A saída do modelo, contendo as classes detectadas e suas localizações, é formatada em um relatório detalhado. Este relatório é anexado à imagem processada e enviado por e-mail utilizando uma biblioteca de envio, garantindo que os resultados sejam compartilhados em tempo real com os responsáveis pela análise ou manutenção. Assim, a solução une dispositivos físicos e serviços em nuvem para fornecer um sistema eficiente, escalável e automatizado de detecção de patologias.

3.2 Configuração do fluxo de identificação das patologias

Assim que a imagem é salva no Google Storage, um "trigger de trabalho" é configurado para iniciar automaticamente um pipeline no Vertex AI. Esse pipeline pode incluir a pré-processamento da imagem e o envio dela para análise.

O Vertex AI encaminha a imagem para o modelo de identificação de patologias, implementado com YOLO. Esse modelo analisa a imagem em busca de padrões ou anomalias específicas, gerando um relatório ou alerta com os resultados da análise.

Esses gatilhos interligados garantem a automação do processo, desde a coleta de dados com o sensor ultrassônico e a câmera até a análise avançada no modelo de ML. A integração eficiente entre hardware e serviços na nuvem possibilita respostas rápidas e precisas às condições monitoradas.

Figura 12 - Capturando imagem

Fonte: Imagem própria

3.2.1 Google Storage

O Google Cloud Storage é uma solução de armazenamento em nuvem desenvolvida pela empresa Google, ideal para guardar dados não estruturados de diferentes tipos e tamanhos. Ele faz parte da robusta infraestrutura da Google Cloud Platform e foi projetado para atender às demandas de empresas que precisam de segurança, acessibilidade e escalabilidade no gerenciamento de arquivos.

Essa ferramenta elimina a necessidade de servidores físicos ou mídias externas, que possuem espaço limitado e estão sujeitas a falhas, como danos ao hardware ou corrupção de dados. Ao utilizar o Google Storage, temos a garantia de armazenar informações importantes, como documentos sensíveis e backups, de forma confiável e acessível, com possibilidade de acesso remoto a partir de qualquer dispositivo conectado à internet.

Além de proteger os dados contra perdas, o Google Storage também facilita a gestão de grandes volumes de informações, sendo uma alternativa eficiente e moderna para empresas que buscam otimizar seus processos e reduzir custos com infraestrutura física.

Foi utilizado o Google Storage como parte fundamental do nosso fluxo de trabalho com a VertexAI. Nesse processo, as imagens capturadas por nossa câmera e processadas por nosso algoritmo são enviadas diretamente para o armazenamento. Assim que as imagens são guardadas, o Google Storage aciona o VertexAI, que executa nosso modelo de inteligência artificial de forma automatizada.

Essa integração garante um fluxo contínuo e otimizado para o processamento de dados, unindo armazenamento seguro e computação avançada.

Figura 13 - Identificação de uma patologia

Fonte: Imagem própria

3.2.2 VertexAI

O Vertex AI é uma plataforma avançada do Google Cloud, lançada em maio de 2021, projetada para facilitar o desenvolvimento e a implementação de sistemas de machine learning. Com essa solução, os desenvolvedores podem criar, treinar e implementar modelos de forma simplificada e eficiente, eliminando boa parte da complexidade técnica que tradicionalmente envolve projetos de machine learning. Um de seus principais diferenciais é a redução significativa do número de linhas de código necessárias para programar, o que acelera o processo e permite maior foco na solução do problema.

Essa plataforma é ideal para projetos que desejam automatizar processos e extrair padrões úteis de seus dados, mas que enfrentam desafios devido à complexidade técnica ou à quantidade de etapas necessárias para criar e testar modelos tradicionais. Ao integrar ferramentas poderosas e uma interface amigável, o Vertex AI proporciona agilidade no desenvolvimento de soluções de inteligência artificial, contribuindo para aumentar a eficiência operacional e a inovação nos negócios.

Optamos por utilizar o Vertex AI devido à sua simplicidade, que se destaca em comparação com as outras plataformas, apresentando uma complexidade técnica menor. Integramos essa ferramenta ao nosso modelo YOLO, permitindo que ambos trabalhem em conjunto de forma eficiente. Nesse fluxo, as imagens armazenadas no Google Cloud Storage são processadas pelo Vertex AI e pelo YOLO, que detectam e identificam a patologia asfáltica presente na imagem. Essa integração garante agilidade e precisão no reconhecimento e análise das condições do asfalto.

3.2.3 Modelo de Identificação com YOLO

O YOLO (You Only Look Once) é uma técnica que encontra objetos em imagens de forma rápida. Ele olha a imagem toda de uma vez e consegue identificar e localizar diferentes objetos, como pessoas, carros ou problemas no asfalto, tudo em um único passo. Isso torna o YOLO muito eficiente para usar em tempo real.

Nosso modelo YOLO foi treinado e implementado para funcionar com a VertexAI, o que permitiu a integração eficiente entre a detecção de patologias asfálticas e a infraestrutura de computação em nuvem. Ao ser alimentado com imagens de pavimentos, que vem direto do nosso Google Storage, o modelo é capaz de identificar e classificar diferentes tipos de patologias, como fissuras, buracos e ondulações, retornando as classes detectadas de maneira automatizada.

A implementação na VertexAI proporcionou escalabilidade e facilidade no gerenciamento dos dados, permitindo que o modelo fosse executado de maneira otimizada, mesmo com grandes volumes de imagens. A utilização dessa plataforma possibilitou uma integração eficiente com outras ferramentas e pipelines de machine learning, acelerando o processo de análise e tomada de decisões.

Com essa solução, a detecção das patologias ocorre em tempo real, sem a necessidade de intervenção manual, garantindo maior agilidade na identificação dos problemas e no planejamento das manutenções necessárias para preservar a qualidade das vias.

4 CONCLUSÃO

A infraestrutura viária é um pilar fundamental para o desenvolvimento social e econômico de qualquer nação. Entretanto, a degradação dos pavimentos asfálticos, causada por diversos fatores como tráfego intenso, intempéries e negligência na manutenção, representa um desafio significativo para a segurança dos usuários, a fluidez do transporte e a economia.

Neste contexto, a Visão Computacional (VC) surge como uma ferramenta inovadora para auxiliar na gestão eficiente da infraestrutura viária. Através de técnicas de processamento de imagens e aprendizado de máquina, a VC permite a identificação, análise e classificação de patologias em pavimentos asfálticos com alto grau de precisão e eficiência.

A implementação da VC na gestão de pavimentos oferece diversos benefícios tangíveis, tais como:

Diagnóstico Preciso e Rápido: A VC possibilita o diagnóstico precoce e preciso de patologias como fissuras, buracos, deformações e descascamentos, permitindo intervenções tempestivas e eficazes;

Otimização da Manutenção: A identificação precisa das patologias direciona os esforços de manutenção para áreas específicas, otimizando o tempo e os recursos disponíveis;

Redução de Custos: A prevenção de danos graves e a otimização da manutenção contribuem para a redução dos custos totais com reparos e reconstruções de pavimentos;

Aumento da Vida Útil dos Pavimentos: A manutenção preventiva prolonga a vida útil dos pavimentos, diminuindo a necessidade de obras frequentes e dispendiosas;

Melhoria da Segurança no Trânsito: Pavimentos em boas condições reduzem o risco de acidentes e garantem um transporte mais fluido e seguro para todos os usuários;

Promoção da Sustentabilidade: A redução do consumo de materiais para reparos e a diminuição da geração de resíduos contribuem para um futuro mais sustentável.

A implementação da VC na gestão de pavimentos ainda enfrenta alguns desafios, como a necessidade de treinamento de pessoal especializado, a padronização de protocolos de coleta de dados e a integração com sistemas de gestão de infraestrutura. No entanto, o potencial dessa tecnologia é imenso e abre um leque de possibilidades para inovações futuras, como:

Desenvolvimento de Sistemas de Monitoramento em Tempo Real: Monitorar o estado dos pavimentos em tempo real permitirá a identificação imediata de problemas e a tomada de medidas preventivas imediatas;

Criação de Sistemas Autônomos de Reparo: Robôs autônomos equipados com VC podem realizar reparos em pavimentos de forma eficiente e segura, minimizando a necessidade de intervenção humana;

Desenvolvimento de Materiais de Pavimentação Inteligentes: Materiais que se adaptam às condições climáticas e se "autorreparam" podem revolucionar a forma como construímos e mantemos nossas vias.

A sinergia entre a Visão Computacional e a gestão de pavimentos asfálticos representa um passo crucial para a construção de cidades resilientes e sustentáveis. Ao investir em pesquisa e desenvolvimento nessa área, podemos garantir infraestruturas viárias mais seguras, eficientes e duráveis, contribuindo para o bem-estar da sociedade e o desenvolvimento econômico do país.

5 REFERÊNCIAS BIBLIOGRÁFICAS

FREITAS, C. C.; CORTEZIA, D. A. D. Estudo das principais patologias em pavimento – estudo de caso – trecho GO 241 – UHE Serra da Mesa, Minaçu-GO, 2020. Disponível em: <https://repositorio.pucgoias.edu.br/jspui/bitstream/123456789/738/1/TCC%20II%20-%20CAIO%20E%20DANILO%20FINAL%2014_12_2020.2.pdf> Acesso em: 21/05/2024.

SOUTO, E. V.; MORESCO, B. H.; GOLTZ, C. J. Patologias em pavimentos asfálticos: estudo de caso na rua Dr. Renato Figueiro Varela em Nova Xavantina – MT, 2019. Disponível em: <https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://publicacoes.unifimes.edu.br/index.php/interacao/article/download/824/848&ved=2ahUKEwi5ouG66qKGAxVnrpUCHc4xDmMQFnoECCwQAQ&usg=AOvVaw0p_qSdvxOrWNPW-bAqjoAq>. Acesso em: 21/05/2024.

BATISTA, P. M. S. Análise de manifestações patológicas de pavimentos asfálticos em trechos de bairros no município de Paulo Afonso– BA., Delmiro Goulveia - AL, 2021. Disponível em: <https://www.repositorio.ufal.br/bitstream/123456789/7982/1/An%C3%A1lise%20de%20manifesta%C3%A7%C3%B5es%20patol%C3%B3gicas%20de%20pavimentos%20asf%C3%A1lticos%20em%20trechos%20de%20bairros%20no%20munic%C3%ADpio%20de%20Paulo%20Afonso%20-%20BA.pdf>. Acesso em: 21/05/2024.

PEREIRA, J. K. O. Uso de visão computacional para reconhecimento de imagens de frutas em imagens RGB, 2021. Disponível em: <https://repositorio.ufc.br/bitstream/riufc/64490/1/2022_tcc_jkopereira.pdf>. Acesso em: 22/05/2024.

MACHADO, R. C. Sistema de visão computacional para mapeamento de obstáculos em uma mesa de provas de robôs. Uberlândia – MG, 2017. Disponível em: <https://repositorio.ufu.br/bitstream/123456789/22130/3/SistemaVis%C3%A3oComputacional.pdf>. Acesso em: 22/05/2024.

FREITAS, J. M. M. Sensor de nível por micro-ondas e tecnologia RADAR-FMCW. 2023. Dissertação (Mestrado em Engenharia Elétrica) - Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Itajubá, Itajubá, MG.

RODRIGUES, P. H. B. Técnicas de visão computacional para classificação de peças. 2023. Trabalho de Conclusão de Curso (Graduação em Engenharia) - Mackenzie, São Paulo - SP, 2021. Disponível em: <https://dspace.mackenzie.br/items/1cc2806d-38df-42ac-88d2-a2f16d0c4012>. Acesso em: 22/05/2024.

DEPARTAMENTO NACIONAL DE INFRAESTRUTURA DE TRANSPORTES - DNIT. Manual de Restauração de Pavimentos Asfálticos. Brasília, DF: DNIT, 2019.

RASPBERRY PI. Raspberry Pi 3 Model B+ Product Brief. 2018. Disponível em: <https://datasheets.raspberrypi.com/rpi3/raspberry-pi-3-b-plus-product-brief.pdf.> Acesso em: 23/05/2024.

RASPBERRY PI. Camera Module 3 Product Brief. 2017. Disponível em: <https://datasheets.raspberrypi.com/camera/camera-module-3-product-brief.pdf.> Acesso em: 23/05/2024.

SPARKFUN ELECTRONICS. HCSR04 Datasheet. Disponível em: <https://cdn.sparkfun.com/assets/b/3/0/b/a/DGCH-RED_datasheet.pdf.> Acesso em: 23/05/2024.

MQTT Documentation: A documentação oficial do MQTT fornece uma descrição detalhada do protocolo e suas especificações. Disponível em: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>

AWS - O que é MQTT: Uma explicação concisa sobre o MQTT com exemplos práticos. Disponível em: <https://aws.amazon.com/pt/what-is/mqtt/>

Automação Industrial - Protocolo MQTT: Um artigo abrangente sobre o MQTT, cobrindo seus fundamentos e aplicações. Disponível em: <https://www.automacaoindustrial.info/mqtt/>

APÊNDICE A - Código para configuração do ESP32 para controle do sensor e comunicação MQTT

```
'
#include <WiFi.h>
#include <PubSubClient.h>
#include <Arduino.h>

const char* SSID = "LCM-Tw";
const char* PASSWORD = "@Iron%#11";
const char* mqtt_server = "192.168.5.159";
const int mqtt_port = 1883;
const char* mqtt_username = "tccunip";
const char* mqtt_password = "123";
const char* topic = "sensor/ultrasonic";

//WiFiClient wifiClient;
//PubSubClient MQTTclient(wifiClient);

const int trigPin = 5;
const int echoPin = 18;
#define SOUND_SPEED 0.034
long duration;
float distanceCm;

void setup() {
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Connect to Wi-Fi
  WiFi.begin(SSID, PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Connect to MQTT
  MQTTclient.setServer(mqtt_server, mqtt_port);
  while (!MQTTclient.connected()) {
    Serial.println("Connecting to MQTT...");
    if (MQTTclient.connect("ESP32Client", mqtt_username, mqtt_password)) {
      Serial.println("connected");
    } else {
      Serial.println("failed, rc=");
      Serial.println(MQTTclient.state());
```

```
      delay(2000);
    }
  }
}

void loop() {
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  distanceCm = duration * SOUND_SPEED / 2;

  //Serial.print("Distance (cm): ");
  //Serial.println(distanceCm);

  // Publish the distance to the MQTT topic
  String msg = "{\"distance\": " + String(distanceCm) + "}";
  MQTTclient.publish(topic, msg.c_str());

  if (distanceCm < 20) {
    delay(30000); // Delay for 60 seconds
  } else {
    delay(1000); // Default delay between measurements
  }
}
'
```

```
## [1] "\n#include <WiFi.h>\n#include <PubSubClient.h>\n#include <Arduino.h>\n\nconst char* SSID = \"LC
```

APÊNDICE B - Código para configuração do RASPBERRY PI para recebimento das distâncias via MQTT, ativação da câmera e transferência as fotos para o Googel Drive

```
# Pacotes

import os
import time
import subprocess
import json
import logging
from google.oauth2 import service_account
from googleapiclient.discovery import build
from googleapiclient.http import MediaFileUpload
#import paho.mqtt.client as mqtt

# Variáveis globais
SERVICE_ACCOUNT_FILE = '/home/tccunip/python_scripts/.json'  # Substitua pelo caminho do seu arquivo JS
PHOTO_DIRECTORY = '/home/tccunip/fotos'
TARGET_FOLDER_ID = ''  # Substitua pelo ID da sua pasta no Google Drive
taking_pictures = False

# Configurações do log

logging.basicConfig(level=logging.DEBUG, format = '%(asctime)s - %(levelname)s - %(message)s',
                    handlers=[
                         logging.FileHandler('app.log'),
                         logging.StreamHandler()
                    ])
logger = logging.getLogger(__name__)

logger.debug('Iniciando app')
logger.info('Conectando ao broker MQTT')

# Função para autenticar no Google Drive
def authenticate():
    try:
        creds = service_account.Credentials.from_service_account_file(SERVICE_ACCOUNT_FILE, scopes=['ht
        return creds
        logger.debug(f'Autentiação API Google realizada com sucesso')
    except Exception as e:
        logger.error(f'Erro ao autenticar, verifique as variáveis globais: {str(e)}')

# Função para enviar um arquivo para o Google Drive
```

```python
def upload_file_to_drive(file_path, folder_id):
    try:
        service = build('drive', 'v3', credentials=authenticate())
        file_metadata = {'name': os.path.basename(file_path), 'parents': [folder_id]}
        media = MediaFileUpload(file_path, mimetype='image/jpeg', resumable=True)
        file = service.files().create(body=file_metadata,
                                      media_body=media,
                                      fields='id').execute()
        logger.debug(f'Foto enviado para a cloud')
    except Exception as e:
        logger.error(f'Erro ao enviar a imagem via API {str(e)}')

# Função que ativa a câmera e tira a foto
def take_photo(file_path):
    try:
        subprocess.run(["libcamera-still", "-o", file_path, "--nopreview"], check=True, timeout=25)
        foto=+1
        logger.debug(f'Diferença de distância detectada, foto tirada e salva')
    except Exception as e:
        logger.error(f'Erro ao tirar a foto, verifique o funcionamento da câmera {str(e)}')

# Função para conectar no contexto do ESP32
def on_connect(client, userdata, flags, rc):
    try:
        client.subscribe("sensor/ultrasonic")
        logger.debug(f'Conectado no contexto, recebendo informações da distância')
    except Exception as e:
        logger.error(f'Erro ao se conectar ao contexto, verfique as configuração do ESP {str(e)}')

# Função de contexto, para executar as ações
def on_message(client, userdata,msg):
    try:
        global taking_pictures
        data = json.loads(msg.payload.decode()) #decodifica o JSON
        distancia =  float(data["distance"]) # extrai o valor da distancia
        if distancia < 20:
            if not taking_pictures:
                taking_pictures = True
            if taking_pictures:
                    foto = 1
                    while foto <= 3:
                        timestamp = time.strftime("%Y%m%d-%H%M%S")
                        file_path = os.path.join(PHOTO_DIRECTORY, f'foto_{timestamp}_{foto}.jpg')
                        take_photo(file_path)
                        upload_file_to_drive(file_path, TARGET_FOLDER_ID)
                        foto+=1
        else:
            if taking_pictures:
                taking_pictures = False
    except Exception as e:
        logger.error(f'Erro na função de contexo, verifique à aplicação {str(e)}')

#client = mqtt.Client()
```

```
#client.on_connect = on_connect
#client.on_message = on_message
#client.connect("localhost", 1883, 60)
#client.loop_forever()
```