Universidade Federal de Ouro Preto
Escola de Minas
Engenharia de Controle e Automação

Karine Isabelle Marins

VISION: Robô Móvel com Visão Computacional Aplicado à Detecção de Placas de Segurança

Ouro Preto, 2026

Karine Isabelle Marins

VISION: Robô Móvel com Visão Computacional Aplicado à Detecção de Placas de Segurança

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Engenheira de Controle e Automação.

Orientador: Dr. André Luiz Carvalho Ottoni
Coorientador: Dr. José Alberto Naves Cocota Junior

Ouro Preto
2026

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA CONTROLE E AUTOMACAO

FOLHA DE APROVAÇÃO

Karine Isabelle Marins

VISION: Robô Móvel com Visão Computacional Aplicado à Detecção de Placas de Segurança

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Engenharia de Controle e Automação

Aprovada em 2 de março de 2026

Membros da banca

[Doutor] - André Luiz Carvalho Ottoni - Orientador (Universidade Federal de Ouro Preto)
[Doutor] - José Alberto Naves Cocota Júnior - Coorientador (Universidade Federal de Ouro Preto)
[Doutor] - Jadson Castro Gertrudes - Convidado (Universidade Federal de Ouro Preto)
[Doutor] - Agnaldo José da Rocha Reis - Convidado (Universidade Federal de Ouro Preto)

José Alberto Naves Cocota Júnior, coorientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 05/03/2026

Documento assinado eletronicamente por **Jose Alberto Naves Cocota Junior**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 05/03/2026, às 22:01, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **1070342** e o código CRC **231B55FD**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591533 - www.ufop.br

Agradecimentos

Inicialmente, agradeço a Deus por me conceder força, sabedoria e discernimento ao longo desta trajetória. Sua presença foi fundamental nos momentos mais desafiadores, renovando minha esperança e me dando perseverança para seguir em frente.

À minha mãe, Rosana, e ao meu pai, Hilton, registro minha mais profunda gratidão. O amor incondicional, o apoio constante e todos os esforços dedicados à minha formação foram essenciais para que eu alcançasse este objetivo. Vocês são minha base, minha inspiração diária e a razão maior das minhas conquistas.

Ao meu namorado, Anderson, agradeço pelo companheirismo, pela paciência e pelo incentivo ao longo de toda essa jornada. Seu apoio foi indispensável nos momentos de incerteza, trazendo serenidade e confiança para que eu pudesse concluir esta etapa com determinação.

Aos meus orientadores, André Luiz e Cocota, expresso meu sincero agradecimento pelas orientações valiosas, pelo comprometimento e pela confiança depositada em mim durante o desenvolvimento deste trabalho.

Aos professores e colegas da Universidade Federal de Ouro Preto, minha gratidão pela convivência, pelas trocas de conhecimento e pelo ambiente acadêmico estimulante, que contribuiu de maneira significativa para minha formação pessoal e profissional.

Ao Instituto Tecnológico Vale (ITV), à Samarco e à Blip, agradeço pelas oportunidades de crescimento e aprendizado proporcionadas por meio da iniciação científica e do estágio. As experiências vivenciadas nessas instituições foram determinantes para o fortalecimento dos conhecimentos adquiridos na graduação, possibilitando a aplicação prática da teoria e ampliando minha visão profissional.

À Escola de Minas da Universidade Federal de Ouro Preto, manifesto minha gratidão pela formação acadêmica recebida, pela infraestrutura disponibilizada e pelas oportunidades oferecidas, que foram essenciais para a concretização deste estudo.

Quando a mulher negra se movimenta, toda a estrutura da sociedade se movimenta com ela

— Angela Davis.

Resumo

A evolução tecnológica tem impactado significativamente a robótica e os sistemas autônomos, especialmente o desenvolvimento de robôs móveis equipados com visão computacional. Nesse contexto, o uso de Redes Neurais Convolucionais (CNNs) tem se destacado por sua elevada eficiência em tarefas de detecção e classificação de imagens, permitindo que esses sistemas interpretem o ambiente e tomem decisões com maior precisão. Entretanto, um dos principais desafios dessa aplicação consiste em garantir acurácia e desempenho em tempo real em dispositivos embarcados de baixo custo. Diante disso, objetiva-se com esse trabalho desenvolver um protótipo de robô móvel equipado com visão computacional para a detecção automática de placas de segurança, como saída de emergência, risco de choque elétrico e uso obrigatório de EPI. A motivação deste trabalho está relacionada ao aumento da segurança em ambientes industriais, propondo um sistema capaz de realizar a identificação rápida e eficiente de placas de sinalização durante atividades de inspeção. A metodologia foi estruturada em quatro etapas principais: construção da base de dados, treinamento e validação do modelo na plataforma Edge Impulse, desenvolvimento e integração do hardware do robô e realização de testes experimentais para validação do sistema. O modelo foi implementado na plataforma Edge Impulse e a inferência foi realizada por meio de um dispositivo móvel. O sistema foi avaliado em um robô com chassi 4WD, equipado com motores DC, módulo Bluetooth e câmera para aquisição das imagens. Os resultados experimentais demonstraram desempenho satisfatório do sistema proposto. Em testes simulados, o modelo alcançou acurácia de 97,33% e *F1 Score* próximo de 99%. Nos testes de inferência em dispositivo móvel, as médias de confiança das classes foram 0,91 para EPI, 0,98 para Saída de Emergência e 0,89 para Choque. Nos testes dinâmicos com o robô em movimento, as médias foram 0,93 para EPI, 0,96 para Saída de Emergência e 0,93 para Choque, evidenciando robustez e consistência do sistema em diferentes condições de aquisição de imagem.

Palavras-chaves: Redes Neurais Convolucionais, Robótica Móvel, Segurança do Trabalho, Visão Computacional

Abstract

Technological evolution has significantly impacted robotics and autonomous systems, especially in the development of mobile robots equipped with computer vision. In this context, the use of Convolutional Neural Networks (CNNs) has stood out due to their high efficiency in image detection and classification tasks, enabling these systems to interpret the environment and make more accurate decisions. However, one of the main challenges of this application is ensuring accuracy and real-time performance on low-cost embedded devices. Therefore, we developed a mobile robot prototype equipped with computer vision for the automatic detection of safety signs, such as emergency exit, electrical hazard, and mandatory use of personal protective equipment (PPE). The motivation of this work is related to improving safety in industrial environments by proposing a system capable of performing fast and efficient identification of safety signage during inspection activities. The methodology was structured into four main stages: dataset construction, model training and validation using the Edge Impulse platform, robot hardware development and integration, and experimental testing for system validation. The model was implemented on the Edge Impulse platform, and inference was performed through a mobile device. The system was evaluated using a 4WD chassis robot equipped with DC motors, a Bluetooth module, and a camera for image acquisition. Experimental results demonstrated satisfactory performance of the proposed system. In simulated tests, the model achieved an accuracy of 97.33% and an F1 Score close to 99%. In mobile device inference tests, the average confidence scores were 0.91 for PPE, 0.98 for Emergency Exit, and 0.89 for Electrical Hazard. In dynamic tests with the robot in motion, the averages were 0.93 for PPE, 0.96 for Emergency Exit, and 0.93 for Electrical Hazard, demonstrating robustness and consistency under different image acquisition conditions.

Keywords: Convolutional Neural Networks, Computer Vision, Mobile Robotics, Occupational Safety.

Lista de Figuras

Figura 1 – Exemplo de robô móvel quadrúpede da Unitree aplicado em inspeção e segurança industrial. 17
Figura 2 – Anatomia do olho humano 19
Figura 3 – Comparação entre a visão humana e uma câmera fotográfica 19
Figura 4 – Comparação entre neurônio biológico e neurônio artificial. 20
Figura 5 – Exemplo de Rede Expandida em Grafo Computacional. 22
Figura 6 – Exemplo ilustrativo do funcionamento de uma Rede Neural Convolucional 23
Figura 7 – Fluxograma metodológico do trabalho. 29
Figura 8 – Exemplos de imagens da classe **Saída de Emergência**. 31
Figura 9 – Exemplos de imagens da classe **Choque Elétrico**. 32
Figura 10 – Exemplos de imagens da classe **Uso Obrigatório de EPI**. 33
Figura 11 – Importação das imagens na plataforma *Edge Impulse*. 35
Figura 12 – Definição da pasta, tipo de imagem e label na plataforma *Edge Impulse*. 35
Figura 13 – Modelo definido com as imagens na plataforma *Edge Impulse*. 36
Figura 14 – Página de configuração de entrada e saída das imagens na plataforma *Edge Impulse*. 37
Figura 15 – Paramêtros das imagens na plataforma *Edge Impulse*. 37
Figura 16 – Definição dos hiperparâmetros de treinamento na plataforma *Edge Impulse*. 39
Figura 17 – Kit chassi 4WD completo. 40
Figura 18 – Detalhe da estrutura do chassi 4WD e demais componentes 41
Figura 19 – Placa Arduino UNO utilizada como unidade de controle principal. . . . 41
Figura 20 – Módulo ESP32-CAM para captura de imagens e processamento. 42
Figura 21 – Módulo Bluetooth para comunicação sem fio. 42
Figura 22 – Módulo Ponte H utilizado para controle dos motores DC. 43
Figura 23 – Fontes de alimentação utilizadas no robô. 43
Figura 24 – Componentes auxiliares integrados ao circuito do robô. 44
Figura 25 – Esquema geral de montagem e interligação dos componentes do robô móvel. 47
Figura 26 – Montagem inicial do chassi 4WD. 48
Figura 27 – Arduino Uno e conexões com a ponte H 49
Figura 28 – Conexão do módulo Bluetooth ao Arduino com divisor de tensão. . . . 50
Figura 29 – Conexão do buzzer ao Arduino Uno. 51
Figura 30 – Esquema de conexão do módulo ESP32 ao sistema eletrônico do robô. . 52
Figura 31 – Robô montado - vistas frontal e frontal diagonal. 52
Figura 32 – Robô montado - vistas superior e lateral. 53

Figura 33 – Métricas finais de treinamento e validação do modelo. 55
Figura 34 – Matriz de confusão resultante da validação do modelo. 55
Figura 35 – Resultados da fase de teste na plataforma *Edge Impulse* 56
Figura 36 – Matriz de Confusão na Etapa de Testes 57
Figura 37 – QR Code do Edge Impulse, utilizado para acessar o modelo de inferência. 58
Figura 38 – Resultados dos testes 1 a 3 da classe **EPI**. 59
Figura 39 – Resultados dos testes 4 a 6 da classe **EPI**. 59
Figura 40 – Resultados dos testes 7 a 9 da classe **EPI**. 60
Figura 41 – Resultado do teste 10 da classe **EPI**. 60
Figura 42 – Resultados dos testes 1 a 3 da classe **Saída de Emergência**. 61
Figura 43 – Resultados dos testes 4 a 6 da classe **Saída de Emergência**. 61
Figura 44 – Resultados dos testes 7 a 9 da classe **Saída de Emergência**. 62
Figura 45 – Resultado do teste 10 da classe **Saída de Emergência**. 62
Figura 46 – Resultados dos testes 1 a 3 da classe **Choque**. 63
Figura 47 – Resultados dos testes 4 a 6 da classe **Choque**. 63
Figura 48 – Resultados dos testes 7 a 9 da classe **Choque**. 64
Figura 49 – Resultado do teste 10 da classe **Choque**. 64
Figura 50 – Tela principal do aplicativo Arduino Bluetooth Controller utilizado para envio de comandos ao VISION. 66
Figura 51 – Placas EPI para experimentos 67
Figura 52 – Placas Saida de Emergência para experimentos 67
Figura 53 – Placas Choque para experimentos 68
Figura 54 – VISION em testes dinâmicos - Exemplo 1 69
Figura 55 – VISION em testes dinâmicos - Exemplo 2 69
Figura 56 – VISION em testes dinâmicos - Exemplo 3 70
Figura 57 – VISION em testes dinâmicos - Exemplo 4 70
Figura 58 – Testes Dinâmicos 1 a 3: detecção da classe **EPI**. 71
Figura 59 – Testes Dinâmicos 4 a 6: detecção da classe **EPI**. 71
Figura 60 – Testes Dinâmicos 7 a 9: detecção da classe **EPI**. 72
Figura 61 – Teste Dinâmico 10: detecção da classe **EPI**. 72
Figura 62 – Testes Dinâmicos 1 a 3: detecção da classe **Saída de Emergência**. . . 73
Figura 63 – Testes Dinâmicos 4 a 6: detecção da classe **Saída de Emergência**. . . 73
Figura 64 – Testes Dinâmicos 7 a 9: detecção da classe **Saída de Emergência**. . . 74
Figura 65 – Teste Dinâmico 10: detecção da classe **Saída de Emergência**. 74
Figura 66 – Testes Dinâmicos 1 a 3: detecção da classe **Choque**. 75
Figura 67 – Testes Dinâmicos 4 a 6: detecção da classe **Choque**. 75
Figura 68 – Testes Dinâmicos 7 a 9: detecção da classe **Choque**. 76
Figura 69 – Teste Dinâmico 10: detecção da classe **Choque**. 76

Lista de tabelas

Tabela 1 – Resumo quantitativo e divisão estratificada do conjunto de dados. . . . 33
Tabela 2 – Valores de confiança nos testes de inferência para as classes EPI, Saída de Emergência e Choque 65
Tabela 3 – Valores de confiança nos testes dinâmicos para as classes EPI, Saída de Emergência e Choque 77

Sumário

1 INTRODUÇÃO 11
1.1 Objetivos 13
1.2 Justificativas e Relevância 14
1.3 Métodos 14
1.4 Organização e estrutura 15

2 REVISÃO DE LITERATURA 16
2.1 Robótica Móvel 16
2.2 Visão Computacional 18
2.3 Redes Neurais Artificiais 20
2.4 Métricas de Avaliação 24
2.5 Trabalhos Relacionados 26

3 METODOLOGIA 29
3.1 Base de Dados 29
3.1.1 Obtenção das Imagens 29
3.1.2 Rotulagem e Organização das Classes 30
3.1.3 Divisão do Conjunto de Dados em Treino e Teste 33
3.2 Modelo no Edge Impulse 34
3.2.1 Criação do Projeto e Importação dos Dados 34
3.2.2 Configuração do Impulso (*Impulse Design*) 36
3.2.3 Configuração de Hiperparâmetros e Treinamento 38
3.3 Desenvolvimento do Robô 40
3.3.1 Arquitetura de Hardware 40
3.3.2 Arquitetura de Software 44
3.3.3 Montagem do Protótipo 47

4 RESULTADOS 54
4.1 Resultados de Validação 54
4.2 Resultados dos Testes Simulados 56
4.3 Resultados nos Testes Práticos 57

5 CONCLUSÕES 78

Referências 80

APÊNDICES 86

APÊNDICE A – FIRMWARE DO ESP32 PARA STREAMING DE VÍDEO 87

APÊNDICE B – FIRMWARE DO ARDUINO PARA CONTROLE DO ROBÔ VIA BLUETOOTH 90

APÊNDICE C – USO DE FERRAMENTAS DE INTELIGÊNCIA ARTIFICIAL 92

1 Introdução

A evolução tecnológica tem transformado significativamente diversos aspectos da sociedade. A incorporação de tecnologias como inteligência artificial, robótica avançada e sistemas inteligentes tem permitido que as empresas automatizem tarefas repetitivas, aumentem a produção e reduzam custos operacionais (OLIVEIRA; SANTOS; FERREIRA, 2024). Essas inovações têm se expandido para diversas áreas, incluindo a automação industrial e o setor da saúde (TORRES, 2022). No contexto industrial, sistemas automatizados de monitoramento têm sido empregados para aprimorar a segurança e reduzir a exposição dos trabalhadores a condições de risco, contribuindo para ambientes mais seguros e eficientes (HORA, 2024; LAN; AWOLUSI; CAI, 2024).

Estudos recentes mostram que a Inteligência Artificial (IA) tem se destacado como uma tecnologia promissora na prevenção de acidentes de trabalho. Uma revisão sistemática internacional (SILVA; DUTRA *et al.*, 2023) identificou que a IA, especialmente por meio de técnicas de *Deep Learning* e Redes Neurais Convolucionais (CNNs), tem sido aplicada na detecção de uso de Equipamentos de Proteção Individual (EPIs), monitoramento de fadiga, identificação de zonas de risco e prevenção de colisões em ambientes industriais e na construção civil. Embora o interesse científico tenha crescido nos últimos anos, ainda existem lacunas, principalmente no desenvolvimento de sistemas inteligentes capazes de atuar de forma preventiva e em tempo real. No contexto brasileiro, uma revisão similar (SILVA; FIGUEIREDO; DUTRA, 2023) mostrou que a aplicação da IA na segurança do trabalho é ainda incipiente, com foco maior em prevenção de acidentes de trânsito e setores específicos como a saúde, evidenciando a necessidade de pesquisas que consolidem evidências sobre os benefícios da IA na redução de acidentes ocupacionais.

Nesse cenário, os robôs móveis surgem como uma tecnologia promissora, com potencial para transformar a forma como tarefas de inspeção e monitoramento de segurança são realizadas em ambientes industriais. Esses robôs dispõem de tecnologias de hardware e software que possibilitam a utilização em linhas de produção, depósitos e plantas de manufatura, sendo capazes de realizar tarefas complexas, como detecção de falhas, monitoramento de processos e identificação de situações de risco, de forma autônoma ou controlada remotamente (TORRES, 2022; LAN; AWOLUSI; CAI, 2024; CAVALCANTE *et al.*, 2023). Em geral, visam aumentar a eficiência operacional, reduzir falhas humanas e aprimorar a segurança ocupacional, especialmente em locais de difícil acesso ou com risco elevado de acidentes (ROCHA, 2023; BERARDINUCCI; URGO, 2025).

A incorporação de Inteligência Artificial nesses sistemas é fundamental para tornar as operações mais seguras, eficientes e confiáveis. Entre as tecnologias que viabilizam

essa evolução, destaca-se a Visão Computacional, área que utiliza técnicas de Inteligência Artificial, Aprendizado de Máquina (*Machine Learning*) e Processamento de imagens para extrair informações relevantes de imagens e vídeos. Por meio da identificação de padrões visuais, esses sistemas tornam-se capazes de interpretar o ambiente e apoiar processos de tomada de decisão (SANTOS; FIGUEIRA, 2024). De acordo com Berardinucci e Urgo (2025), a aplicação de visão computacional baseada em redes neurais profundas possibilita que robôs móveis realizem tarefas de inspeção com elevada precisão e em tempo real, ampliando significativamente a eficiência e a confiabilidade da segurança automatizada em ambientes industriais.

Entre as técnicas mais promissoras da Visão Computacional, destacam-se as Redes Neurais Convolucionais (CNNs), um modelo amplamente utilizado no aprendizado de máquina. Segundo Li (2021), as CNNs operam por meio de aprendizado supervisionado, utilizando amostras de treinamento e técnicas de retropropagação para ajustar continuamente seus parâmetros. Esse ajuste reduz a diferença entre a saída gerada e o vetor esperado, garantindo maior precisão nos resultados. Devido à sua eficácia, as CNNs são aplicadas em diversas áreas, como reconhecimento de imagens, biomedicina, controle inteligente e estabilização de vídeo (ZHANG; ZHANG; ZHOU, 2022; LOU; SHI, 2020; LI, 2021). Em especial, elas têm alcançado desempenho de ponta em tarefas de reconhecimento de imagens e detecção de objetos em tempo real, como evidenciado pelos modelos da série YOLO (*You Only Look Once*), amplamente utilizados em sistemas de segurança e inspeção automatizada (LAN; AWOLUSI; CAI, 2024; KRICHEN, 2023).

Recentemente, Lan, Awolusi e Cai (2024) demonstraram a eficácia de algoritmos de *Deep learning* na detecção de Equipamentos de Proteção Individual (EPIs), como capacetes de segurança e coletes refletivos, evidenciando o potencial da visão computacional para o aprimoramento das práticas de segurança no setor industrial. Além disso, sistemas inteligentes de vigilância têm mostrado que algoritmos baseados em Redes Neurais Convolucionais, como YOLOv8 e SSD com MobileNetV2, permitem a detecção e rastreamento de EPIs, incluindo capacetes, coletes, luvas e máscaras, com taxas de acerto superiores a 90% (SNEHALAKSHMI *et al.*, 2024; REDDY *et al.*, 2024). Algumas abordagens também integram múltiplos indicadores de segurança, como a detecção de sonolência em trabalhadores, aumentando a robustez do monitoramento e possibilitando alertas em tempo real, mesmo em ambientes com iluminação variável ou diferentes poses (SNEHALAKSHMI *et al.*, 2024).

Nesse contexto, a Visão Computacional aplicada a robôs móveis, em conjunto com sensores embarcados, possibilita a detecção e a interpretação, em tempo real, de placas e sinalizações de segurança, como saídas de emergência, avisos de risco elétrico e instruções de uso obrigatório de EPI. Segundo XD4Solutions (2025b,a), robôs móveis inteligentes, como o *Unitree Go2*, integram sensores avançados, incluindo LiDAR 3D e câmeras de profundidade, permitindo o mapeamento do ambiente, a navegação autônoma

e o desvio de obstáculos. Além disso, esses sistemas contam com processamento embarcado para a execução de algoritmos de inteligência artificial, viabilizando aplicações voltadas à inspeção, ao monitoramento e à segurança em ambientes complexos. Essas características são fundamentais para o desenvolvimento de soluções automatizadas voltadas à segurança do trabalho e à conformidade com normas regulatórias.

Em virtude dos fatos apresentados, apresenta-se nesta monografia o VISION, um Robô Móvel com Visão Computacional para Detecção de Placas de Segurança, como saída de emergência, risco de choque elétrico e uso obrigatório de EPI, utilizando Redes Neurais Convolucionais. O intuito é integrar soluções de inteligência artificial com hardware embarcado eficiente e acessível, permitindo a detecção precisa dessas sinalizações em tempo real. Esse estudo visa contribuir para os avanços no desenvolvimento de tecnologias voltadas à segurança em ambientes industriais e prediais, promovendo a prevenção de acidentes e a automação dos processos de inspeção visual.

1.1 Objetivos

Objetivo Geral

O objetivo geral deste trabalho é desenvolver e avaliar a aplicação prática de um protótipo de robô móvel, voltado à detecção de placas de segurança, tais como saída de emergência, risco de choque elétrico e uso obrigatório de equipamentos de proteção individual (EPI), utilizando técnicas de visão computacional.

Objetivos Específicos

- Montagem do robô móvel VISION, incluindo a instalação e integração de todos os componentes necessários;
- Desenvolvimento e implementação de sistema de visão computacional capaz de processar e interpretar as imagens capturadas pelo robô;
- Utilização do microcontrolador ESP32 para a execução de testes e validação do sistema de visão computacional;
- Integração do sistema de visão computacional ao robô móvel, visando avaliar de maneira efetiva o desempenho do sistema de detecção de sinais de trânsito em condições realistas.

1.2 Justificativas e Relevância

Segundo Topolsky *et al.* (2019), sistemas de visão computacional para robôs móveis enfrentam desafios significativos devido às limitações de recursos computacionais, exigindo a otimização das tarefas visuais e a escolha de algoritmos de reconhecimento com baixa complexidade para funcionamento em tempo real em hardware embarcado restrito. Essa otimização é essencial para que robôs móveis alcancem desempenho eficaz em aplicações práticas, especialmente em ambientes industriais, onde a sinalização de segurança desempenha papel crítico na prevenção de acidentes e na proteção dos trabalhadores.

No contexto industrial moderno, a capacidade de robôs móveis identificarem automaticamente placas de sinalização e alertas visuais contribui tanto para a redução de falhas humanas quanto para o aumento da eficiência e confiabilidade dos processos de inspeção e monitoramento. Plataformas robóticas avançadas como o *Unitree Go2* (XD4SOLUTIONS, 2025a), um robô quadrúpede inteligente equipado com sensores LIDAR 4D e recursos de IA para percepção e navegação em ambientes complexos, exemplificam a evolução da robótica móvel integrada a sistemas de percepção sofisticados, capazes de mapear e interpretar o ambiente em tempo real.

Entretanto, apesar do avanço de modelos robóticos comerciais e de pesquisa, a aplicação de visão computacional embarcada de baixo custo em plataformas móveis ainda demanda estudos que demonstrem a capacidade de detecção de objetos e sinais em cenários reais, considerando variáveis como iluminação, ângulo e ruído visual. A precisão na identificação de sinalizações de segurança, por exemplo, é fundamental para garantir que soluções autônomas possam ser utilizadas com confiança em ambientes industriais dinâmicos e potencialmente perigosos.

Dessa forma, este trabalho se justifica pela necessidade de desenvolver e validar um sistema embarcado capaz de realizar a detecção de placas de segurança em tempo real, integrado a um robô móvel. A relevância da pesquisa está em contribuir para o aprimoramento de tecnologias voltadas à segurança industrial, oferecendo uma ferramenta de apoio às rotinas de inspeção e monitoramento, ampliando a eficiência e reduzindo riscos aos trabalhadores.

1.3 Métodos

A abordagem utilizada para a detecção de placas de segurança baseia-se na integração de diferentes técnicas e áreas do conhecimento, conforme descrito a seguir:

- **Robótica Móvel**: engloba os princípios de navegação e controle do robô, permitindo a movimentação para a exploração do ambiente.

- **Programação em Sistemas Embarcados**: responsável pela implementação do software embarcado no ESP32, possibilitando a aquisição e o processamento de imagens em tempo real.
- **Visão Computacional - Redes Neurais Convolucionais (CNNs)**: aplicadas para segmentação, detecção e reconhecimento de placas de segurança diretamente no hardware. O modelo de aprendizado profundo é treinado por meio do *Edge Impulse* e acessado via QRCODE, permitindo a execução otimizada do processamento de imagens em tempo real.

1.4 Organização e estrutura

O presente trabalho é composto por cinco (5) capítulos, organizados da seguinte forma:

- **Capítulo 1 - Introdução:** apresenta a contextualização e a motivação do estudo, além de definir seus objetivos e a estrutura do documento.
- **Capítulo 2 - Revisão de Literatura:** discute os principais conceitos, teorias e trabalhos relacionados, fornecendo a base teórica necessária para a compreensão do tema.
- **Capítulo 3 - Metodologia:** descreve detalhadamente o projeto, abrangendo desde sua concepção e construção mecânica até os métodos utilizados para os testes e experimentos.
- **Capítulo 4 - Resultados:** apresenta os resultados finais, bem como uma análise detalhada aplicada à eles e, por fim, uma discussão acerca do obtido.
- **Capítulo 5 - Conclusões:** sintetiza as conclusões do estudo, destacando suas contribuições, limitações e sugestões para pesquisas futuras.

2 Revisão de Literatura

Neste capítulo, serão apresentados os conceitos fundamentais necessários para a compreensão da pesquisa, começando pela teoria que sustenta o trabalho. Após essa fundamentação teórica, serão apresentados os principais trabalhos relacionados que contribuíram para o avanço do tema e serviram de base para o desenvolvimento da pesquisa, proporcionando uma visão geral das abordagens existentes e suas implicações.

2.1 Robótica Móvel

A robótica é uma área multidisciplinar que envolve conceitos de engenharia, computação e eletrônica, dedicada ao desenvolvimento de máquinas programáveis capazes de executar tarefas de forma autônoma ou semiautônoma. Segundo Almeida, Santos Carrara e Yuan (2013) a definição de robô é dada pela *"Robotics Industries Association"(RIA)*. Nesta, define-se um robô como um manipulador reprogramável multifuncional projetado para manusear materiais, peças, ferramentas ou dispositivos especiais, através de movimentos programados para a realização de uma variedade de tarefas. Ainda, com base no padrão internacional ISO 8373 *"Vocabulary"* e na definição dada pela *International Federation of Robotics*, os robôs são um mecanismo acionável programado com um grau de autonomia para realizar locomoção, manipulação ou posicionamento (CASTRO SOUZA; PEDRON; SILVA, 2023).

Um robô móvel pode ser definido como um dispositivo mecânico de transporte automático, constituído por uma plataforma dotada de um sistema de locomoção capaz de navegar em um determinado ambiente, transportando cargas e executando tarefas com um nível variável de autonomia (SECCHI, 2012). A definição de robô móvel está relacionada à sua capacidade de alcançar objetos com mínima intervenção humana. Para isso, esses robôs são equipados com atuadores, como rodas, pernas, articulações e garras, e sensores, como câmeras, radares e *lasers* (*LiDAR*) para perceber o ambiente ao seu redor (RUSSELL; NORVIG, 2022).

Segundo Russell e Norvig (2022), a autonomia de robôs móveis depende diretamente da integração entre sistemas de percepção e mecanismos de tomada de decisão. Nesse contexto, a visão computacional desempenha um papel fundamental ao permitir que o robô interprete o ambiente por meio da captura e do processamento de imagens, possibilitando a identificação de objetos, sinais e padrões visuais. As informações extraídas pelos sistemas de visão podem ser utilizadas como estados de entrada para algoritmos de aprendizado, entre eles o Aprendizado por Reforço (*Reinforcement Learning* – RL), que permite ao robô aprender estratégias de ação a partir da interação com o ambiente. Dessa forma, a combinação entre visão computacional e aprendizado por reforço contribui para o desenvolvimento de robôs móveis mais autônomos, adaptáveis e eficientes, capazes de operar em cenários reais e dinâmicos.

Um exemplo representativo dessa tecnologia é o robô móvel quadrúpede (Figura 1) desenvolvido pela empresa Unitree (XD4SOLUTIONS, 2025b), amplamente utilizado em inspeções industriais, monitoramento de áreas restritas e operações em ambientes potencialmente perigosos. Sua locomoção estável, aliada à integração com sensores avançados, como câmeras e sistemas de visão computacional, permite a navegação autônoma e a identificação de elementos relevantes do ambiente. Além disso, robôs móveis desse tipo também vêm sendo aplicados em contextos de segurança patrimonial e vigilância, realizando rondas autônomas, identificação de sinalizações de segurança e detecção de situações anômalas, contribuindo para a redução de riscos operacionais e para o aumento da segurança dos trabalhadores.

Figura 1 – Exemplo de robô móvel quadrúpede da Unitree aplicado em inspeção e segurança industrial.
Fonte: Adaptado de (XD4SOLUTIONS, 2025b)

2.2 Visão Computacional

A Visão Computacional é uma área da ciência da computação dedicada ao desenvolvimento de métodos e algoritmos que permitem aos computadores interpretar e compreender informações visuais provenientes de imagens ou vídeos. Seu principal objetivo é extrair características relevantes do ambiente, possibilitando a identificação, classificação e localização de objetos, além da tomada de decisões automatizadas com base em dados visuais. Para isso, integra técnicas de processamento de imagens, aprendizado de máquina e inteligência artificial, buscando simular a capacidade humana de percepção visual. Sistemas de Visão Computacional são amplamente aplicados em diversas áreas, como reconhecimento facial, inspeção automática de peças em linhas de produção, monitoramento de ambientes, diagnóstico médico por imagem e navegação de robôs móveis em ambientes estruturados e não estruturados (AZEVEDO; CONCI; LETA, 2022).

Muitos dos conceitos empregados na Visão Computacional são inspirados no sistema visual humano, uma vez que a visão é o principal meio pelo qual os seres humanos percebem e interpretam o mundo ao seu redor (PEDRINI; SCHWARTZ, 2007). No entanto, diferentemente da visão biológica, a Visão Computacional busca modelar esse processo por meio de algoritmos matemáticos e técnicas computacionais.

A Figura 2 apresenta a estrutura do olho humano, frequentemente utilizada como analogia para compreender o funcionamento dos sistemas artificiais de visão. De forma simplificada, o olho atua como um sensor óptico responsável por captar a luz do ambiente, enquanto o cérebro é encarregado do processamento e interpretação dessas informações visuais.(PEDRINI; SCHWARTZ, 2007; AZEVEDO; CONCI; LETA, 2022).

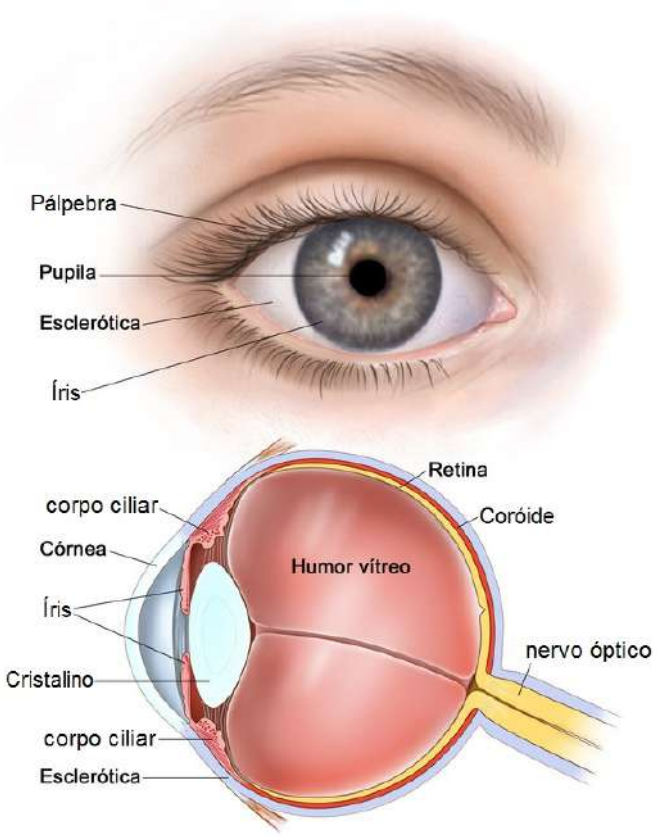


Figura 2 – Anatomia do olho humano
Fonte: Adaptado de (REDAÇÃO SANAR, 2019)

De maneira análoga, em sistemas de Visão Computacional, dispositivos como câmeras realizam a aquisição das imagens, e algoritmos computacionais assumem o papel do processamento, análise e extração de informações relevantes. A Figura 3 ilustra essa analogia entre o sistema visual humano e uma câmera fotográfica, destacando similaridades funcionais entre seus componentes (PEDRINI; SCHWARTZ, 2007).

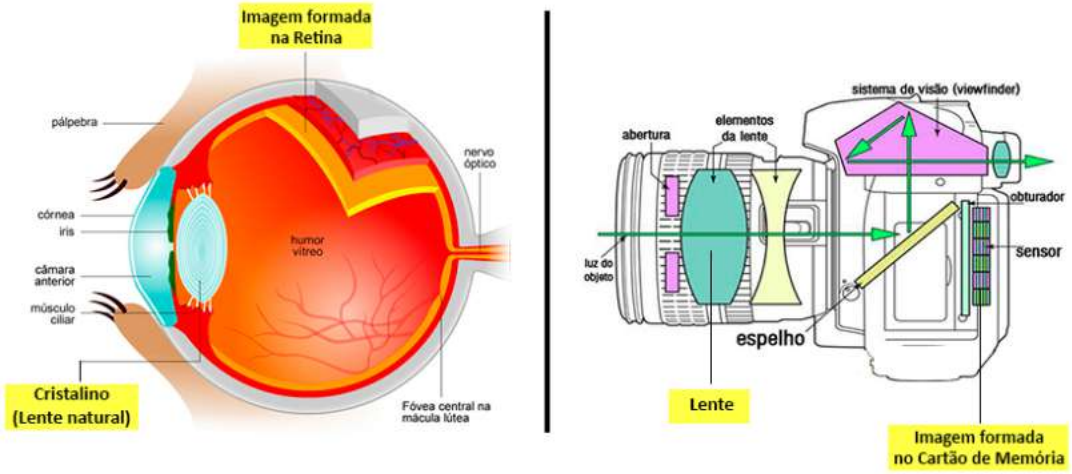
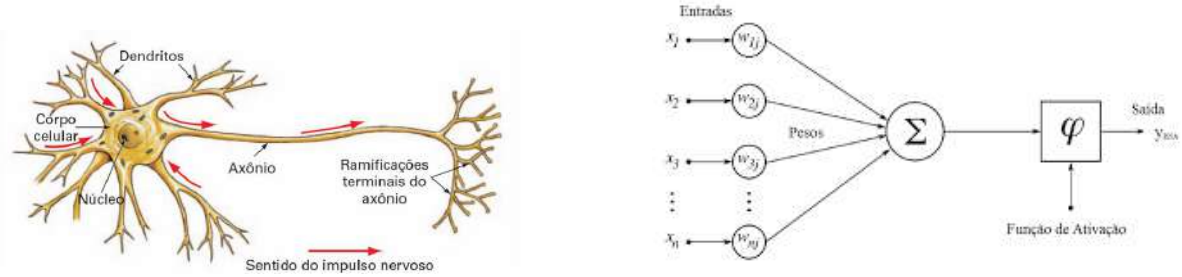


Figura 3 – Comparação entre a visão humana e uma câmera fotográfica
Fonte: Adaptado de (GOOGLE, 2025)

Nesse contexto, a Visão Computacional emprega técnicas como processamento digital de imagens, aprendizado de máquina e redes neurais profundas para simular, de forma computacional, a capacidade humana de interpretar cenas visuais. Essas técnicas permitem o desenvolvimento de aplicações avançadas, como detecção de placas de segurança, reconhecimento de padrões visuais e navegação autônoma de robôs, temas diretamente relacionados a este trabalho.

2.3 Redes Neurais Artificiais

As **Redes Neurais Artificiais (RNAs)** surgiram como uma abstração inspirada no funcionamento do sistema nervoso humano, com o objetivo de desenvolver modelos computacionais capazes de aprender padrões a partir de dados. Embora a semelhança entre neurônios biológicos e artificiais seja apenas conceitual, essa analogia foi fundamental para a formulação dos primeiros modelos matemáticos de aprendizado (GOODFELLOW *et al.*, 2016). A Figura 4 ilustra essa inspiração inicial, comparando a estrutura de um neurônio biológico com um neurônio artificial.



(a) Neurônio biológico (b) Neurônio artificial

Figura 4 – Comparação entre neurônio biológico e neurônio artificial.
Fonte: Adaptado de (GOOGLE, 2025)

Do ponto de vista biológico, o neurônio é composto por dendritos, responsáveis pela recepção dos sinais, pelo corpo celular, onde ocorre a integração das informações, e pelo axônio, que transmite a resposta. Esse mecanismo inspirou diretamente o modelo artificial, no qual as conexões sinápticas são representadas por **pesos**, parâmetros que determinam a influência de cada entrada no processamento da informação (GOODFELLOW *et al.*, 2016).

O desenvolvimento das redes neurais artificiais teve início na chamada primeira onda da Inteligência Artificial, marcada pelos estudos da cibernética entre as décadas de 1940 e 1960. Nesse período, destacam-se as contribuições de McCulloch e Pitts (1943), que propuseram um dos primeiros modelos formais de neurônio artificial, bem como os estudos sobre aprendizagem biológica apresentados por Hebb (1949). Posteriormente, Rosenblatt

(1958) desenvolveu o Perceptron, considerado um dos primeiros modelos implementáveis de rede neural (GOODFELLOW *et al.*, 2016).

De forma geral, um neurônio artificial realiza uma combinação linear das entradas $x_1, x_2, \ldots, x_n$, ponderadas por pesos ajustáveis $w_1, w_2, \ldots, w_n$, acrescida de um termo de viés (*bias*). O resultado dessa combinação é então aplicado a uma função de ativação, conforme descrito na Equação 2.1:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.1}$$

O termo b permite ajustar o limiar de ativação do neurônio, aumentando a flexibilidade do modelo.

À medida que a pesquisa em redes neurais evoluiu, verificou-se que a combinação de múltiplos neurônios organizados em camadas possibilitava a aprendizagem de representações mais complexas. Esse arranjo deu origem às redes neurais multicamadas e, posteriormente, ao conceito de **Aprendizado Profundo** (*Deep Learning*). Redes profundas são caracterizadas pela presença de diversas camadas intermediárias, nas quais cada nível aprende representações progressivamente mais abstratas dos dados (GOODFELLOW *et al.*, 2016; RUSSELL; NORVIG, 2022).

Uma forma rigorosa de interpretar o funcionamento dessas redes é por meio de um **grafo computacional**, no qual cada nó representa uma operação elementar — como soma, multiplicação ou aplicação de funções de ativação — e cada aresta representa o fluxo de dados entre essas operações. Essa representação torna explícita a forma como a informação é transformada ao longo da rede e é fundamental para o processo de treinamento dos modelos (RUSSELL; NORVIG, 2022). A Figura 5 exemplifica essa estrutura.

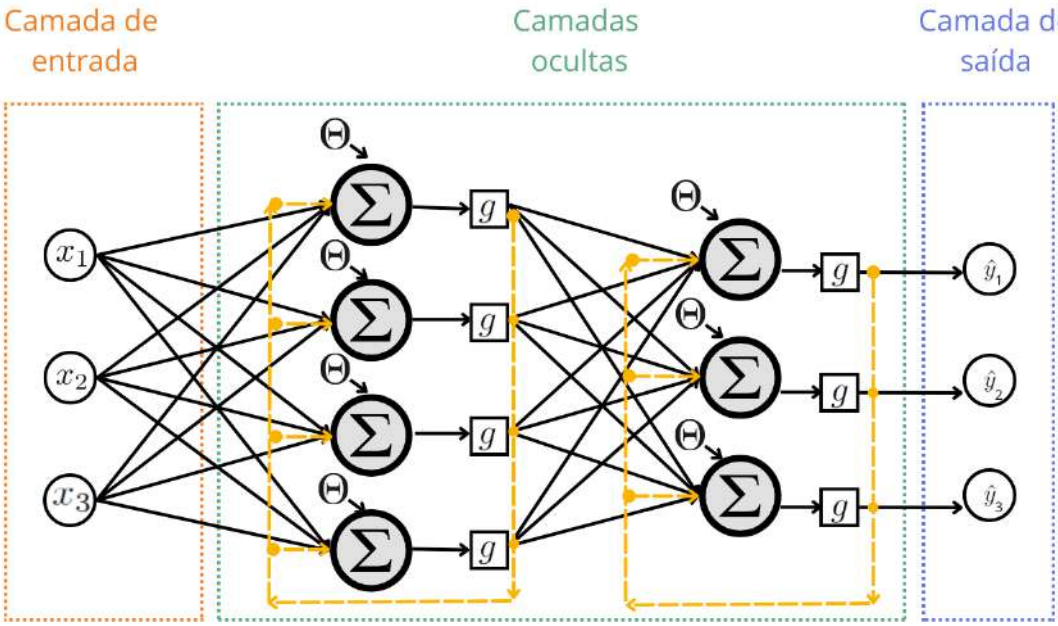


Figura 5 – Exemplo de Rede Expandida em Grafo Computacional.
Fonte: Adaptado de (GOOGLE, 2025)

Com o avanço da capacidade computacional e a disponibilidade de grandes volumes de dados, arquiteturas especializadas passaram a ser desenvolvidas para tarefas específicas. Entre elas, destacam-se as **Redes Neurais Convolucionais (RNCs)**, amplamente utilizadas em aplicações de visão computacional. Essas redes exploram a estrutura espacial dos dados por meio da aplicação de filtros locais, conhecidos como *kernels*, capazes de extrair padrões relevantes em diferentes regiões da imagem (GOODFELLOW *et al.*, 2016; RUSSELL; NORVIG, 2022).

Uma arquitetura típica de RNC é composta principalmente por:

- **Camadas Convolucionais**, responsáveis pela extração de características locais, como bordas, texturas e formas;
- **Camadas de Pooling**, que reduzem a dimensionalidade das representações e conferem maior robustez a pequenas variações espaciais.

Essas camadas formam uma hierarquia composicional, na qual características simples são combinadas para gerar representações mais complexas, permitindo elevado desempenho em tarefas de reconhecimento e classificação de imagens (LECUN; BENGIO; HINTON, 2015). Dessa forma, as Redes Neurais Convolucionais têm sido amplamente aplicadas em problemas como detecção, segmentação e reconhecimento de objetos, incluindo sinais de trânsito, rostos, textos e outros elementos visuais em ambientes reais.

A Figura 6 ilustra as principais etapas do processamento realizado por uma Rede Neural Convolucional.

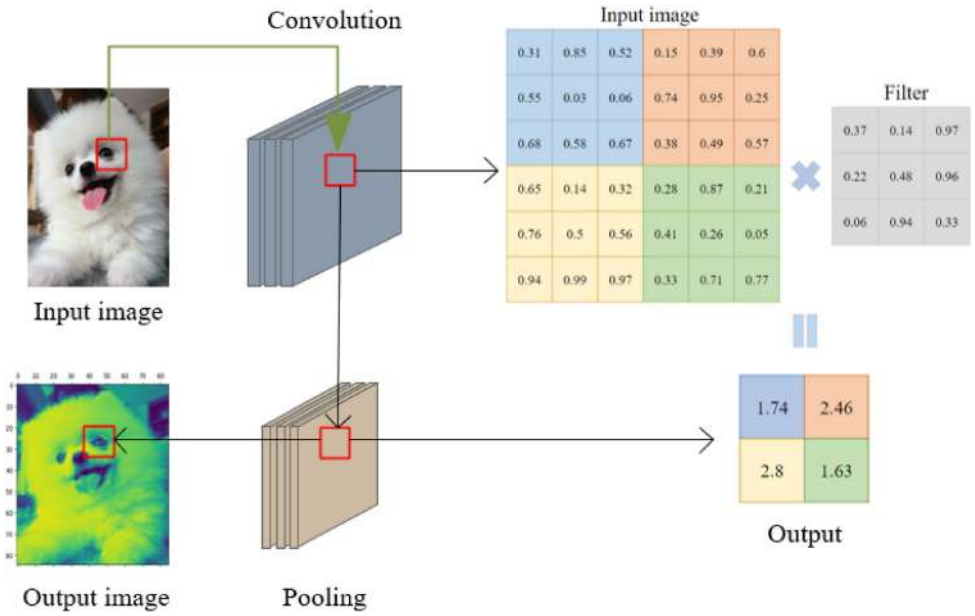


Figura 6 – Exemplo ilustrativo do funcionamento de uma Rede Neural Convolucional
Fonte: Adaptado de (GOOGLE, 2025)

Inicialmente, a imagem de entrada é analisada localmente por meio de um filtro convolucional, representado pelo quadrado vermelho que percorre diferentes regiões da imagem. Esse filtro, também denominado *kernel*, possui valores fixos que são aplicados sobre pequenas janelas da imagem por meio de uma operação de convolução.

Na parte superior direita da figura, observa-se a representação numérica dessa operação. Cada região da imagem de entrada é multiplicada elemento a elemento pelos valores do filtro, e o resultado dessas multiplicações é somado, gerando um único valor de saída. Esse valor indica o grau de correspondência entre o padrão presente na imagem e o padrão que o filtro foi projetado para detectar, como bordas ou contrastes específicos.

A aplicação do mesmo filtro em diferentes posições da imagem gera um **mapa de características**, no qual valores mais elevados indicam regiões onde o padrão foi identificado com maior intensidade. Esse compartilhamento de pesos permite que a rede detecte o mesmo padrão independentemente de sua posição na imagem.

Em seguida, o mapa de características é submetido à etapa de **pooling**, ilustrada na parte inferior da figura. Nessa fase, pequenas regiões do mapa são agregadas, geralmente por meio de operações como máximo ou média, resultando em uma nova representação com menor dimensionalidade. O pooling reduz o custo computacional e torna a rede mais robusta a pequenas variações espaciais, como deslocamentos ou ruídos.

Por fim, o resultado dessas operações é um mapa de saída mais compacto, que preserva as informações mais relevantes extraídas da imagem original. À medida que esse processo é repetido ao longo de múltiplas camadas convolucionais e de pooling, a rede passa a aprender representações cada vez mais abstratas, possibilitando a identificação de

objetos ou classes específicas presentes na imagem.

No contexto deste trabalho, esse tipo de arquitetura é empregado para o reconhecimento automático de placas de segurança, utilizando técnicas de visão computacional embarcada.

2.4 Métricas de Avaliação

A avaliação do desempenho de modelos de classificação é uma etapa fundamental no desenvolvimento de sistemas baseados em aprendizado de máquina (GÉRON, 2019). Para essa finalidade, diversas métricas podem ser empregadas, dentre as quais destacam-se a matriz de confusão, a acurácia, a precisão, a revocação (*recall*) e o F1-score, que possibilitam uma análise abrangente da performance do classificador.

Matriz de Confusão

A matriz de confusão é uma das ferramentas mais completas para avaliação de classificadores (GÉRON, 2019). Ela permite visualizar o número de predições corretas e incorretas realizadas pelo modelo, organizadas por classe.

Em suma, a matriz é composta por quatro elementos fundamentais:

- **Verdadeiros Positivos (VP)**: amostras corretamente classificadas como pertencentes à classe em análise;
- **Verdadeiros Negativos (VN)**: amostras corretamente classificadas como pertencentes às demais classes;
- **Falsos Positivos (FP)**: amostras incorretamente classificadas como pertencentes à classe em análise (Erro do Tipo I);
- **Falsos Negativos (FN)**: amostras incorretamente classificadas como pertencentes a outras classes (Erro do Tipo II).

Acurácia

A acurácia representa a proporção de predições corretas em relação ao total de amostras avaliadas (GÉRON, 2019), sendo definida como:

$$Acuracia = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.2}$$

Embora seja uma métrica intuitiva e amplamente utilizada, a acurácia pode não ser adequada em cenários com desbalanceamento de classes, pois não distingue os diferentes tipos de erro.

Precisão

A precisão mede a proporção de predições positivas que realmente pertencem à classe positiva (GÉRON, 2019), sendo dada por:

$$Precisao = \frac{TP}{TP + FP} \tag{2.3}$$

Essa métrica é especialmente relevante quando se deseja minimizar a ocorrência de falsos positivos.

Revocação (Recall)

A revocação, também chamada de sensibilidade ou taxa de verdadeiros positivos (TPR), indica a proporção de instâncias positivas corretamente identificadas pelo modelo (GÉRON, 2019):

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

Essa métrica é importante em aplicações nas quais a não detecção de um evento positivo pode gerar consequências significativas.

F1-Score

O F1-score combina precisão e revocação em uma única métrica, sendo definido como a média harmônica entre ambas (GÉRON, 2019):

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.5}$$

Por utilizar a média harmônica, o F1-score penaliza valores muito baixos de uma das métricas, garantindo que o desempenho seja considerado elevado apenas quando precisão e revocação apresentarem resultados satisfatórios.

Função Softmax

A função *Softmax* é amplamente utilizada em problemas de classificação multiclasse em redes neurais, especialmente na camada de saída de modelos de aprendizado profundo. Sua principal finalidade é converter os valores de saída da rede, conhecidos como *logits*, em

probabilidades normalizadas que somam 1, permitindo a interpretação da saída como uma distribuição de probabilidade sobre as classes (GOODFELLOW *et al.*, 2016; EDGE IMPULSE, 2024).

Matematicamente, considerando um vetor de entrada $\mathbf{z} = [z_1, z_2, \ldots, z_K]$, a função Softmax para uma classe j é definida como:

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \quad \text{para } j = 1, \ldots, K \tag{2.6}$$

onde K representa o número total de classes e e^{z_j} corresponde à exponencial do logit associado à classe j.

2.5 Trabalhos Relacionados

O uso de visão computacional embarcada em robôs móveis tem sido amplamente explorado para tarefas de reconhecimento e detecção de sinais, textos e objetos em diferentes contextos de aplicação. Diversos estudos investigam a integração entre plataformas robóticas móveis e técnicas de processamento de imagens, evidenciando a relevância dessa abordagem para sistemas inteligentes de inspeção, monitoramento e assistência.

No contexto do reconhecimento de textos e sinalizações, o trabalho de Ismael e Saeed (2024) apresentou o desenvolvimento de um robô móvel de quatro rodas equipado com visão computacional e técnica de OCR (Reconhecimento Óptico de Caracteres) para detecção e reconhecimento de textos presentes em placas, especialmente em ambientes considerados perigosos, como zonas de combate. O sistema foi implementado utilizando Arduino UNO, comunicação via módulo Bluetooth HC-05 e processamento das imagens com OpenCV e biblioteca Tesseract. Os resultados demonstraram correspondência de 100% entre o texto detectado e o texto presente nas imagens capturadas em tempo real, evidenciando a viabilidade do uso de visão computacional embarcada para identificação de sinalizações.

Avançando para abordagens baseadas em aprendizado profundo, Ragab *et al.* (2024) desenvolveram um robô móvel autônomo equipado com sistema de visão fundamentado no algoritmo YOLO *(You Only Look Once)* para detecção de objetos em tempo real. Diferentemente da abordagem baseada em OCR, o processamento das imagens não ocorre diretamente no robô, sendo transferido via Wi-Fi para um laptop, onde o algoritmo realiza a detecção. Posteriormente, os resultados são enviados para uma Raspberry Pi responsável pelo controle dos motores e dos movimentos do robô. Embora o foco principal tenha sido a manipulação de objetos, o estudo demonstrou elevada precisão na identificação visual por meio de redes neurais convolucionais, reforçando a eficiência de modelos de detecção baseados em aprendizado profundo quando aplicados a plataformas móveis.

No âmbito de aplicações práticas voltadas à segurança e monitoramento, Sepeeh *et al.* (2024) desenvolveram e validaram um robô móvel autônomo destinado a tarefas de vigilância. O sistema integra câmeras de alta resolução, sensores de movimento (PIR), sensores ultrassônicos e conectividade IoT, permitindo navegação autônoma, desvio de obstáculos, detecção de movimento e monitoramento remoto em tempo real. Os resultados experimentais demonstraram desempenho confiável em diferentes ambientes, com transmissão de dados para plataforma em nuvem e capacidade de operação contínua sem intervenção humana. O estudo evidencia o potencial da integração entre visão computacional, sensores embarcados e comunicação sem fio para soluções escaláveis de segurança automatizada.

De maneira complementar, Widayaka *et al.* (2023) implementaram técnicas de visão computacional no robô móvel RHI-MOLA, desenvolvido como plataforma de aprendizagem em robótica. O sistema utiliza uma Raspberry Pi para processamento de imagens e um microcontrolador baseado em Arduino para o controle do robô. A detecção de objetos é realizada por meio do modelo de cores HSV, permitindo o alinhamento do robô em relação ao alvo identificado. Para estabilização do posicionamento, foi empregado um controlador, possibilitando que o robô mantivesse sua posição relativa ao objeto detectado. Os resultados demonstraram a viabilidade da segmentação por cores para tarefas de alinhamento, embora o desempenho dependa das condições de iluminação do ambiente, evidenciando o potencial da visão computacional embarcada tanto para aplicações práticas quanto educacionais.

Abordando a percepção visual em ambientes mais complexos, Yang (2023) propuseram um algoritmo de posicionamento e desvio de obstáculos baseado em visão computacional e inteligência artificial aplicado a robôs de inspeção. A proposta surge como alternativa aos métodos tradicionais de inspeção de linhas de transmissão, que exigem a atuação direta de trabalhadores em campo, frequentemente expostos a condições perigosas e ambientes de alta tensão. O método utiliza uma rede neural convolucional aprimorada (CNN) para identificação de obstáculos em tempo real, considerando a distribuição e o formato dos objetos no terreno, além de empregar o método do campo de potencial artificial para otimização da trajetória. Os resultados indicaram desempenho superior em comparação com métodos tradicionais, demonstrando a eficiência de abordagens baseadas em aprendizado profundo para percepção visual e navegação autônoma de robôs móveis.

Observa-se, portanto, que os trabalhos analisados reforçam a importância da integração entre robótica móvel e visão computacional para tarefas de detecção, reconhecimento e interpretação do ambiente. Contudo, a maior parte das pesquisas concentra-se na detecção de objetos genéricos, reconhecimento de obstáculos ou aplicações específicas como vigilância e manipulação. Poucos trabalhos focam especificamente em placas de segurança normativas. Diferentemente dessas abordagens, o presente trabalho direciona-se à aplicação

da visão computacional em um robô móvel para detecção e reconhecimento de placas de sinalização de segurança, com foco em ambientes controlados, contribuindo para aplicações voltadas à conscientização e apoio à segurança em ambientes internos.

3 Metodologia

A metodologia aplicada neste trabalho foi estruturada em quatro fases principais, organizadas de forma a conduzir o desenvolvimento do sistema desde a coleta de dados até sua validação prática. As fases compreendem: **(i) Base de Dados**, **(ii) Modelo no *Edge Impulse***, **(iii) Desenvolvimento do Robô** e **(v) Resultados**. Cada uma delas possui ações e etapas específicas que orientam sua execução, assegurando clareza, rigor e reprodutibilidade ao processo metodológico. A Figura 7 apresenta a sequência lógica das atividades realizadas.

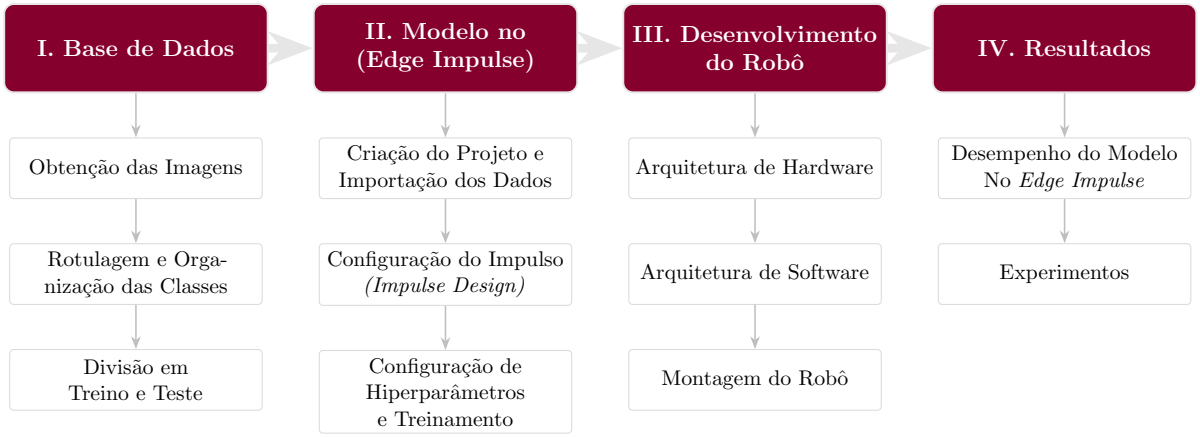


Figura 7 – Fluxograma metodológico do trabalho.
Fonte: Própria autora (2025)

3.1 Base de Dados

A fase inicial da metodologia consistiu na construção da **Base de Dados**, elemento fundamental para o treinamento e a validação do modelo de visão computacional. O processo de aquisição foi pautado na necessidade de garantir a robustez e a capacidade de generalização do modelo perante variações. Assim, as imagens foram selecionadas visando diversidade em cenários, resoluções, ângulos de captura e condições de iluminação.

3.1.1 Obtenção das Imagens

O acervo de imagens foi obtido a partir de fontes públicas disponíveis na internet, priorizando plataformas com licenças permissivas e alta qualidade de material fotográfico. As principais fontes utilizadas foram:

- **Google Images**[1]: Empregada para a aquisição de um **volume substancial de ima-**

[1] Disponível em: https://images.google.com/.(GOOGLE, 2025)

gens, garantindo a diversidade contextual e a representação de diferentes condições de captura.

- **Unsplash**[2]: Utilizada como fonte complementar para obter imagens de alta resolução e qualidade profissional, o que contribuiu para a nitidez e o detalhamento do *dataset*.

Após a aquisição, o conjunto de dados foi submetido a uma etapa de curadoria rigorosa, visando a eliminação de duplicações e imagens que pudessem comprometer a integridade do treinamento.

3.1.2 Rotulagem e Organização das Classes

Após a importação das imagens, foi realizada a etapa de **rotulagem**, responsável por identificar e categorizar cada amostra de acordo com o tipo de sinalização representada. Com base nos objetivos do trabalho, o conjunto de dados foi organizado em três classes principais: *saida_emergencia*, *choque* e *epi*.

Classe 1: *saida_emer*

A classe *saida_emer* foi composta por imagens que apresentam as placas de Saída de Emergência, com ampla variação visual, incluindo diferentes formatos de placas, presença ou ausência de setas direcionais, variações de tipografia e distintas condições de iluminação. Essa diversidade é importante para possibilitar o reconhecimento robusto da sinalização em ambientes reais, que geralmente apresentam cenários heterogêneos.

Na Figura 8 são apresentados exemplos representativos das imagens que compõem essa classe.

[2] Disponível em: https://unsplash.com/.(UNSPLASH, 2025)

Figura 8 – Exemplos de imagens da classe **Saída de Emergência**.

Fonte: Adaptado de (GOOGLE, 2025) e (UNSPLASH, 2025)

Classe 2: *choque*

A classe *choque* englobou imagens relacionadas à sinalização de risco elétrico, amplamente utilizada em ambientes industriais. O conjunto apresenta variações no formato do símbolo, no estilo do pictograma e nas cores predominantes, refletindo diferentes padrões gráficos encontrados na prática.

Essas variações visuais são relevantes para garantir que o modelo generalize corretamente a identificação desse tipo de sinalização. Exemplos representativos dessa classe são apresentados na Figura 9.

Figura 9 – Exemplos de imagens da classe **Choque Elétrico**.
Fonte: Adaptado de (GOOGLE, 2025) e (UNSPLASH, 2025)

Classe 3: *epi*

A classe *epi* reuniu imagens de sinalizações relacionadas ao uso obrigatório de equipamentos de proteção individual, incluindo símbolos de capacete, óculos, luvas e outros itens de segurança. As imagens apresentam diversidade quanto ao estilo gráfico, às cores e à composição visual.

Essa variedade contribui para que o modelo aprenda a identificar corretamente os símbolos associados ao uso de EPI em diferentes contextos visuais. Exemplos representativos dessa classe são apresentados na Figura 10.

Figura 10 – Exemplos de imagens da classe **Uso Obrigatório de EPI**.
Fonte: Adaptado de (GOOGLE, 2025) e (UNSPLASH, 2025)

3.1.3 Divisão do Conjunto de Dados em Treino e Teste

O conjunto de dados rotulado foi organizado em dois subconjuntos, destinados ao **treinamento** e ao **teste**, adotando-se a proporção de **80%** e **20%** das imagens, respectivamente. Essa divisão foi realizada de forma manual para cada classe, assegurando que cada categoria mantivesse a mesma proporção em ambos os subconjuntos. Esse tipo de balanceamento é essencial para preservar a representatividade das amostras e garantir que a avaliação do modelo seja conduzida de maneira justa, a partir de dados não utilizados durante o treinamento.

Com a organização e a divisão do conjunto de dados concluídas, a Tabela 1 apresenta o resumo quantitativo do *dataset*, bem como a distribuição das imagens por classe nos conjuntos de treinamento e teste.

Tabela 1 – Resumo quantitativo e divisão estratificada do conjunto de dados.

Classe	**Total de Imagens**	**Treinamento (80%)**	**Teste (20%)**
saida_emergencia	129	103	26
choque	120	96	24
epi	124	99	25
Total Geral	**373**	**298**	**75**

Observa-se que todas as classes mantiveram a proporção previamente definida, o

que contribui para um processo de treinamento equilibrado e para uma avaliação mais confiável do desempenho do modelo.

3.2 Modelo no *Edge Impulse*

A segunda fase do trabalho abrangeu o processo de definição e treinamento do modelo de *Deep Learning*. Para essa etapa, existem diversas plataformas e ferramentas que possibilitam o desenvolvimento de soluções baseadas em IA, como o *OpenCV*, o *TensorFlow* e o *PyTorch*. No entanto, optou-se por utilizar a plataforma ***Edge Impulse***[3], por se tratar de um ambiente especificamente voltado para aplicações em sistemas embarcados.

O *Edge Impulse* facilita o desenvolvimento de modelos de Inteligência Artificial para dispositivos com recursos computacionais limitados, oferecendo ferramentas integradas que abrangem todo o ciclo de desenvolvimento, desde a importação e organização dos dados até a configuração, treinamento e validação do modelo. Além disso, sua interface intuitiva e seu fluxo de trabalho guiado tornam o processo mais acessível, mesmo para usuários com menor experiência em Machine Learning.

Dessa forma, a escolha da plataforma contribuiu para tornar o desenvolvimento mais ágil e estruturado, permitindo a implementação eficiente de aplicações de visão computacional, como a identificação de placas e sinais de segurança em dispositivos embarcados.

3.2.1 Criação do Projeto e Importação dos Dados

A fase inicial do desenvolvimento envolveu a criação do projeto e, posteriormente, a importação do conjunto de dados. Na interface da plataforma, foi necessário definir o método de obtenção das imagens como sendo o upload de imagens existentes (Figura 11).

[3] https://www.edgeimpulse.com/

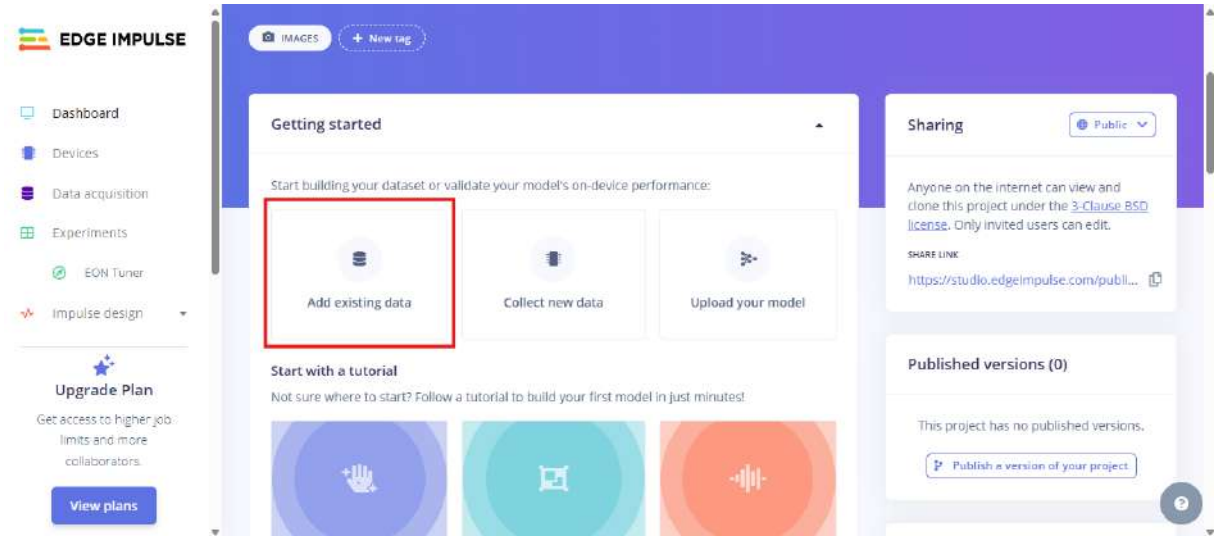


Figura 11 – Importação das imagens na plataforma *Edge Impulse*.
Fonte: Própria autora (2025)

Em seguida, na área de importação da plataforma (Figura 12), foi selecionado o **modo de envio por pastas**, permitindo o carregamento simultâneo de múltiplas imagens.

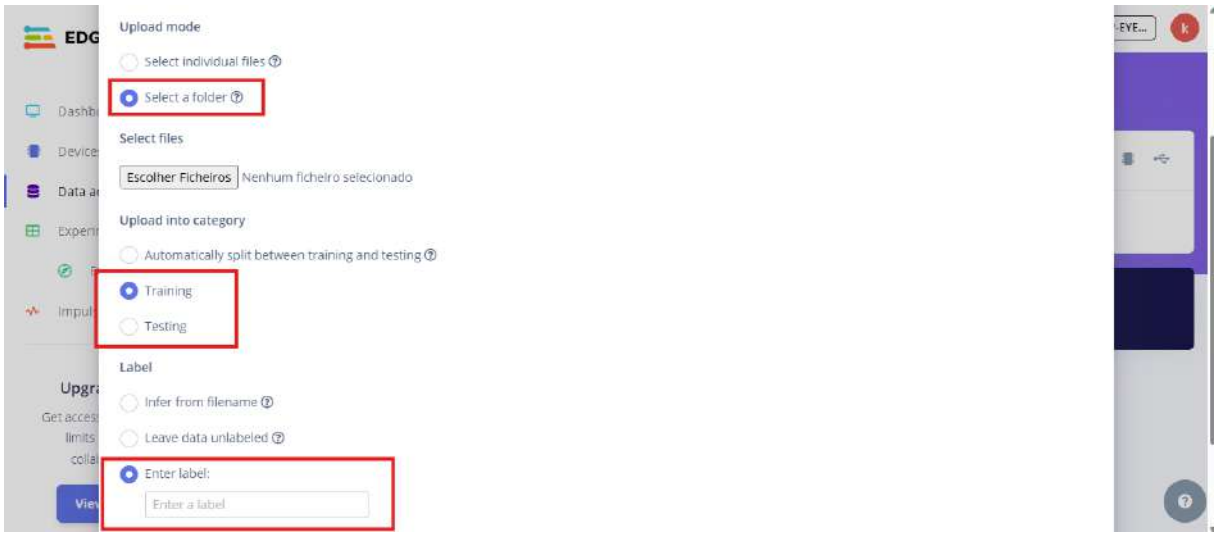


Figura 12 – Definição da pasta, tipo de imagem e label na plataforma *Edge Impulse*.
Fonte: Própria autora (2025)

Durante o processo de importação, a cada pasta enviada era definida a sua categoria, sendo especificado se o conjunto de imagens correspondia ao **treinamento** ou ao **teste**.

Por fim, na etapa de rotulagem(*label*), cada pasta de imagens foi associada ao seu respectivo rótulo, assegurando a identificação correta das classes e concluindo a preparação do conjunto de dados para o treinamento supervisionado do modelo.

Assim, obteve-se 373 imagens, sendo 298 em treino e 75 em teste, conforme apresentado na Figura 13.

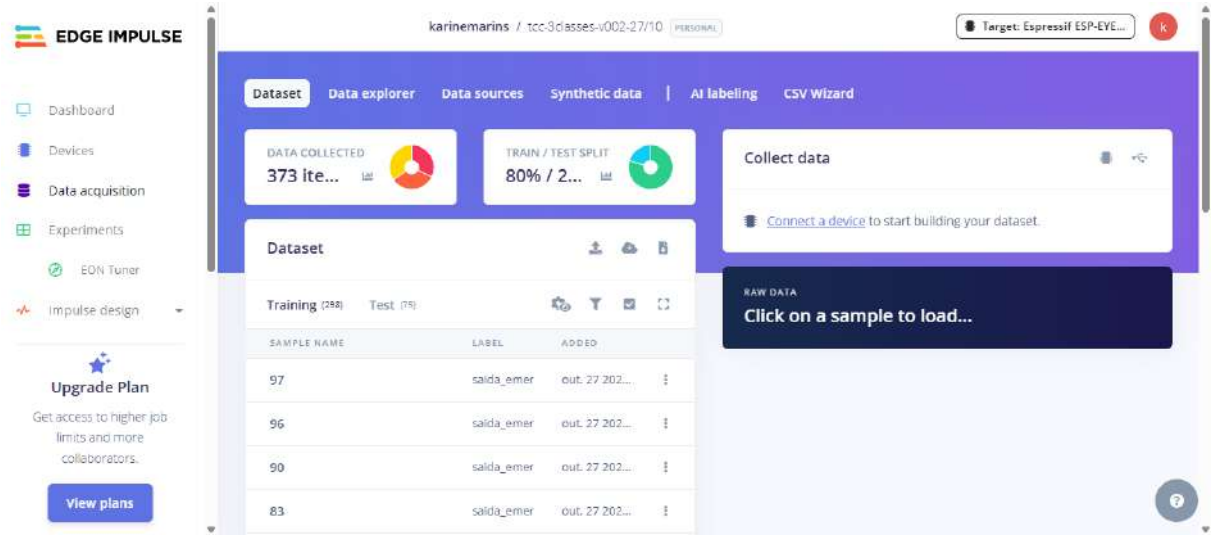


Figura 13 – Modelo definido com as imagens na plataforma *Edge Impulse*.
Fonte: Própria autora (2025)

3.2.2 Configuração do Impulso (*Impulse Design*)

Concluida a etapa de importação de dados, fez-se a configuração do modelo a fim iniciar o treinamento. Na seção *Impulse* (Figura 14), o *pipeline* de processamento e aprendizado foi configurado, definindo as etapas de entrada e saída dos dados.

- **Resolução de entrada**: Configuração das imagens e padronização das dimensões (96×96 pixels, conforme o modelo escolhido).
- **Bloco de Processamento (*Processing Block*)**: Utilização do bloco *Image* para a determinação de como as imagens seriam pré-processadas (escolha entre RGB ou Grayscale).
- **Bloco de Aprendizado (*Learning Block*)**: Seleção do bloco *Transfer Learning* para o treinamento da Rede Neural.

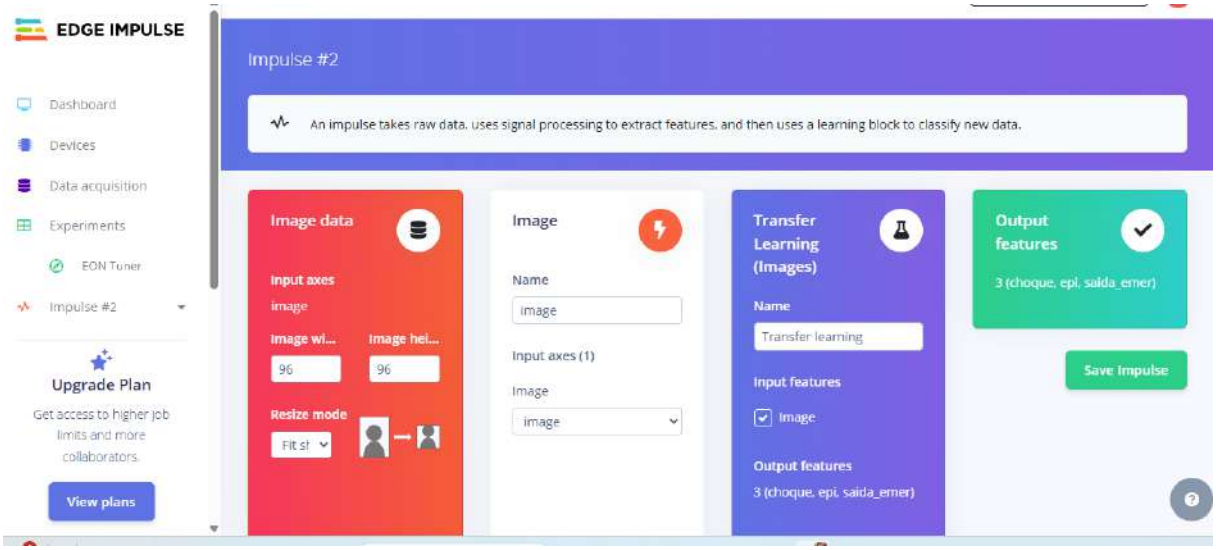


Figura 14 – Página de configuração de entrada e saída das imagens na plataforma *Edge Impulse*.

Fonte: Própria autora (2025)

Em seguida, foram definidos os **parâmetros de pré-processamento das imagens**. Para tal, optou-se pela utilização do **espaço de cores RGB** (*color depth*), conforme ilustrado na Figura 15. Essa escolha permite a preservação das informações cromáticas originais das imagens, aspecto relevante para a correta identificação das diferentes sinalizações, que frequentemente se distinguem por cores específicas.

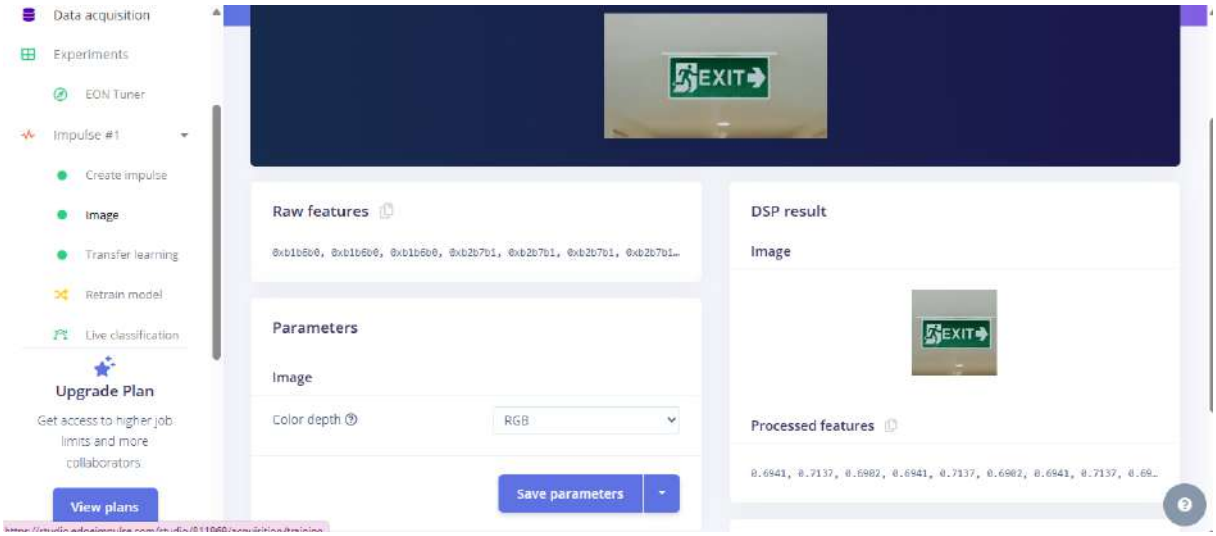


Figura 15 – Paramêtros das imagens na plataforma *Edge Impulse*.
Fonte: Própria autora (2025)

3.2.3 Configuração de Hiperparâmetros e Treinamento

O modelo de arquitetura neural adotado foi o *MobileNetV2*, selecionado em função de sua eficiência estrutural e adequação a sistemas embarcados. A MobileNetV2 foi proposta por Sandler *et al.* (2018) como uma arquitetura convolucional otimizada para dispositivos móveis, projetada para reduzir o número de parâmetros e o custo computacional em comparação com redes tradicionais mais profundas. Segundo os autores, a arquitetura mantém desempenho competitivo em tarefas de visão computacional mesmo com menor complexidade, sendo especialmente indicada para aplicações com restrição de processamento e memória (SANDLER *et al.*, 2018).

A aplicação prática da MobileNetV2 em sistemas embarcados é evidenciada em diferentes estudos. Em Rojas-Romero e Smith Caicedo Saenz (2022), a arquitetura foi ajustada para a detecção de uso de máscara facial durante a pandemia de COVID-19, sendo implementada em uma Raspberry Pi para realizar classificação de imagens em tempo real. O estudo demonstrou que, mesmo em uma plataforma de baixo custo e recursos limitados, a MobileNetV2 manteve alto nível de precisão e desempenho satisfatório de inferência, evidenciando sua viabilidade para aplicações embarcadas.

De forma complementar, aplicações da MobileNetV2 também são observadas em contextos robóticos. No trabalho apresentado em Supriyadi, Bachtiar e Wibowo (2023), a arquitetura foi empregada para a detecção de traves em robôs de futebol da competição *Indonesian Robot Contest (ERSOW)*. O sistema foi desenvolvido para auxiliar o robô na identificação da posição do gol durante a partida, compensando erros acumulados provenientes de sensores. O modelo demonstrou elevada taxa de detecção, com acurácia de 93% e taxa média de processamento de 49 FPS, evidenciando a viabilidade da MobileNetV2 para aplicações em tempo real embarcadas em sistemas robóticos móveis.

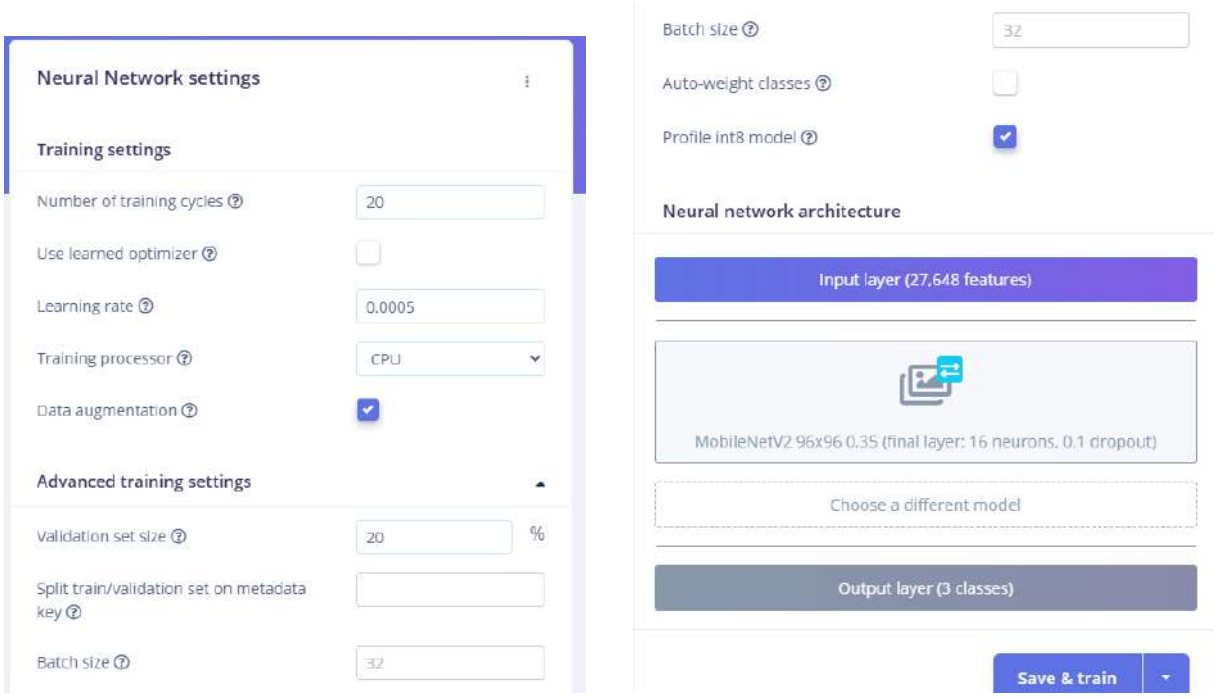
Os hiperparâmetros de treinamento foram definidos na seção *Training settings* da plataforma *Edge Impulse*. A configuração adotada corresponde àquela que apresentou melhor desempenho após a realização de experimentos preliminares, com base nas métricas de validação do modelo.

- **Número de ciclos de treinamento (*epochs*)**: 20;
- **Taxa de aprendizado (*learning rate*)**: 0,0005;
- **Tamanho do lote (*batch size*)**: 32;
- **Conjunto de validação (*validation set*)**: 20% do conjunto de dados de treinamento (60 imagens), reservado para o monitoramento do desempenho do modelo durante o treinamento e para o ajuste dos hiperparâmetros.

Como estratégia para aumentar a robustez do modelo, foi ativada a opção de aumento de dados (*data augmentation*), com o objetivo de verificar se a geração de variações artificiais das imagens de treinamento poderia melhorar o desempenho da rede. Essa técnica permitiu a criação de diferentes transformações nas imagens originais, ampliando a diversidade do conjunto de dados e reduzindo a possibilidade de *overfitting*. Os resultados obtidos indicaram uma melhora no desempenho do modelo após a aplicação do aumento de dados, evidenciando maior capacidade de generalização. O treinamento foi realizado utilizando processamento em CPU (*training processor*), conforme as configurações disponíveis na plataforma.

Visando à adequação do modelo ao ambiente embarcado, foi adotado o perfil int8 optimized, que aplica quantização em 8 bits. Essa técnica reduz o tamanho do modelo e o consumo de recursos computacionais, tornando a rede mais leve e eficiente, além de contribuir para menor tempo de inferência em dispositivos com capacidade limitada (EDGE IMPULSE, 2024).

A Figura 16 ilustra a interface de definição dos principais hiperparâmetros utilizados durante o treinamento do modelo.



(a) Configuração dos hiperparâmetros de treinamento.

(b) Parâmetros adicionais definidos na plataforma *Edge Impulse*.

Figura 16 – Definição dos hiperparâmetros de treinamento na plataforma *Edge Impulse*.
Fonte: Própria autora (2025)

3.3 Desenvolvimento do Robô

A terceira fase do projeto envolveu o desenvolvimento do robô e a integração do sistema embarcado, incluindo a montagem do chassi, integração dos componentes eletrônicos e implementação do software para controle, comunicação e processamento das informações, permitindo a execução do modelo de aprendizado de máquina e a coordenação dos atuadores de forma modular e expansível.

3.3.1 Arquitetura de Hardware

A arquitetura de hardware do robô é composta por módulos interconectados, cada um com função específica, garantindo o funcionamento completo do sistema. A seguir, são apresentados os principais componentes do robô, com suas respectivas imagens e descrições.

Kit Chassi 4WD

O kit Chassi 4WD, apresentado nas Figuras 17 e 18 (ELETROGATE, 2026b), constitui a base estrutural do robô, fornecendo suporte físico para todos os componentes eletrônicos e garantindo a mobilidade do sistema. Contém os seguintes itens:

- 01 x Estrutura do chassi em acrílico;
- 04 x Motores DC 3-6V com redução;
- 04 x Rodas de 65 mm;
- 01 x Porta para 4 pilhas;
- Peças diversas para montagem do chassi (parafusos, suportes e conectores).

Figura 17 – Kit chassi 4WD completo.
Fonte: ELETROGATE (2026b)

Figura 18 – Detalhe da estrutura do chassi 4WD e demais componentes
Fonte: ELETROGATE (2026b)

Arduino UNO

O Arduino UNO é um microcontrolador capaz de atuar em projetos de baixo custo. Apresentado na Figura 19 (MAKERHERO, 2026), é responsável neste projeto pelo controle dos motores, leitura de sensores e gerenciamento das entradas e saídas digitais do sistema. Ele foi escolhido por possuir um número adequado de *General-Purpose Input/Output(GPIOs)*, compatibilidade com drivers de motores e facilidade de programação.

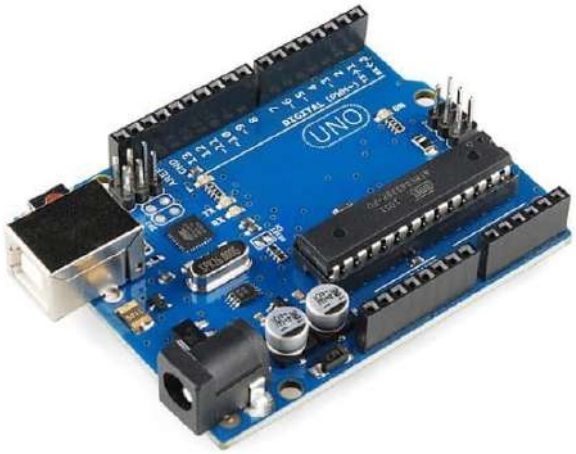


Figura 19 – Placa Arduino UNO utilizada como unidade de controle principal.
Fonte: MAKERHERO (2026)

ESP32 WROVER-DEV-CAM

O ESP32 WROVER-DEV-CAM, exposto na Figura 30 (RS ROBÓTICA, 2026), é responsável pela captura de imagens. Possui conectividade Wi-Fi integrada, permitindo

comunicação remota e transmissão de dados.

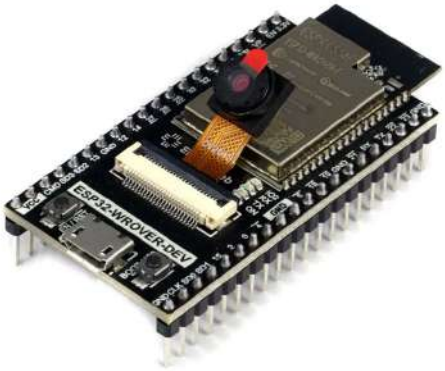


Figura 20 – Módulo ESP32-CAM para captura de imagens e processamento.
Fonte: RS ROBÓTICA (2026)

Módulo Bluetooth

O módulo Bluetooth (Figura 21)(ROBÓTICA EDUCACIONAL, 2026) permite a comunicação sem fio entre o robô e dispositivos externos, como smartphones ou computadores. Ele é utilizado para envio de comandos e recebimento de dados, tornando o controle remoto do robô possível.

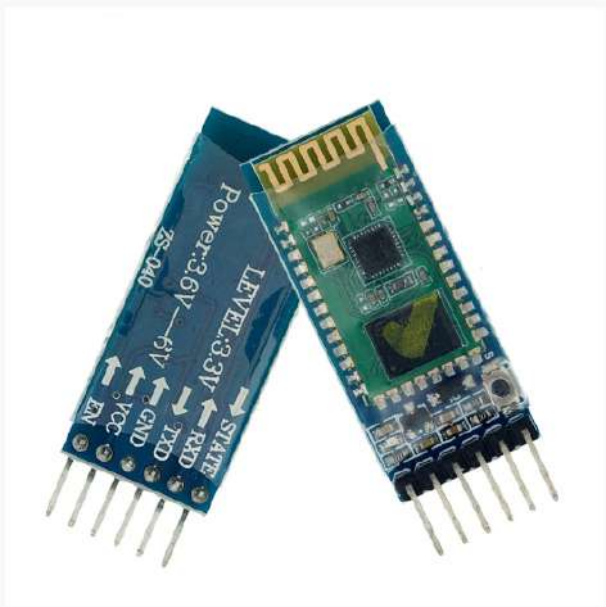


Figura 21 – Módulo Bluetooth para comunicação sem fio.
Fonte: ROBÓTICA EDUCACIONAL (2026)

Ponte H

A ponte H é um componente eletrônico utilizado para controlar a direção e a velocidade dos motores DC. Ela permite que o Arduino ou outro microcontrolador controle

os motores em ambos os sentidos (horário e anti-horário) possibilitando manobras precisas do robô. A Figura 22(VLADCONTROL, 2026) apresenta o módulo utilizado neste projeto.

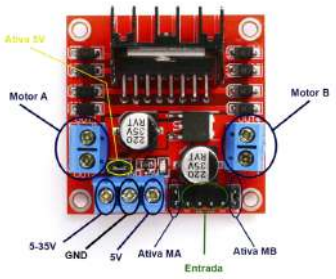


Figura 22 – Módulo Ponte H utilizado para controle dos motores DC.
Fonte: VladControl (2026)

Fonte de Alimentação

O robô é alimentado por um conjunto de 2 baterias de 3.7 V de lítio conectada à ponte H (Figura 23a)(GOOGLE, 2025), além de um conjunto de 4 pilhas AA (Figura 23b)(ELGIN, 2026), fornecendo energia para todos os componentes eletrônicos, incluindo Arduino, ESP32, motores e módulos auxiliares.

(a) Bateria de Lítio.
Fonte: GOOGLE (2025)

(b) Conjunto de 4 pilhas AA.
Fonte: ELGIN (2026)

Figura 23 – Fontes de alimentação utilizadas no robô.

Protoboard, Resistores e Buzzer

A protoboard (Figura 24a) (ROBOCORE, 2026) é utilizada para realizar conexões temporárias e testes de circuitos. Neste sistema, inclui-se resistores de 1 kΩ (Figura

24b)(COPEL ELETRÔNICA, 2026) e buzzer para sinalização sonora (Figura 29)(ELETROGATE, 2026a).

(a) Protoboard utilizada para conexões.
Fonte: ROBOCORE (2026)

(b) Resistor de 1 kΩ.
Fonte: COPEL ELETRÔNICA (2026)

(c) Buzzer para sinalização sonora.
Fonte: ELETROGATE (2026a)

Figura 24 – Componentes auxiliares integrados ao circuito do robô.

3.3.2 Arquitetura de Software

Captura e Transmissão de Imagens

O sistema de captura e transmissão de imagens é responsável por prover ao robô móvel a capacidade de percepção visual do ambiente. Essa funcionalidade é implementada por meio do módulo ESP32 WROVER-DEV-CAM (Figura 30), que integra microcontrolador, interface Wi-Fi e sensor de câmera, atuando como um nó de sensoriamento visual distribuído dentro da arquitetura do sistema. Dessa forma, o dispositivo realiza a aquisição das imagens do ambiente e as disponibiliza em tempo real para monitoramento remoto e posterior processamento pelo modelo de visão computacional.

O firmware desenvolvido para o ESP32 tem como objetivo principal configurar o hardware da câmera, estabelecer a conexão com a rede sem fio e disponibilizar um servidor HTTP embarcado para transmissão contínua de vídeo. Assim, o microcontrolador passa a operar como um servidor de streaming, enviando quadros de imagem no formato MJPEG para qualquer dispositivo conectado à mesma rede local.

O Código apresentado no apêndice A apresenta o firmware completo implementado no módulo.

Inicialmente, nas linhas 1 e 2, são incluídas as bibliotecas `esp_camera.h` e `WiFi.h`, responsáveis, respectivamente, pelo controle do sensor de imagem e pela comunicação em rede. Essas bibliotecas constituem a base funcional do sistema de visão embarcado.

Na sequência, nas linhas 4 e 5 são definidas as credenciais de acesso à rede Wi-Fi por meio das variáveis `ssid` e `password`. Essas informações permitem que o ESP32 se

conecte à rede local e obtenha um endereço IP, tornando-se acessível como servidor de vídeo.

Entre as linhas 7 e 20 são definidos os pinos utilizados na interface com a câmera. Esse conjunto de diretivas estabelece o mapeamento entre os sinais físicos do sensor e os pinos do ESP32. Os sinais D0 a D7 formam o barramento paralelo de dados da imagem, enquanto VSYNC, HREF e PCLK são responsáveis pelo sincronismo da leitura dos pixels. Os pinos SIOD e SIOC realizam a comunicação de controle via protocolo SCCB, e o sinal XCLK fornece o clock externo necessário ao funcionamento do sensor.

A biblioteca `esp_http_server.h`, incluída na linha 22, possibilita a criação de um servidor HTTP embarcado. Em seguida, na linha 23, é declarada a função `stream_handler()`, que será responsável pelo envio dos quadros de imagem ao cliente.

A função `startCameraServer()`, implementada entre as linhas 25 e 41, realiza a configuração e inicialização do servidor HTTP. Nesse trecho, é criada uma rota associada ao endereço raiz do servidor, de modo que, ao acessar o IP do ESP32 em um navegador, a função de streaming seja automaticamente acionada.

As linhas 43 a 48 definem constantes utilizadas na transmissão das imagens em formato MJPEG. Esses cabeçalhos HTTP indicam ao navegador que múltiplas imagens JPEG serão enviadas sequencialmente, caracterizando um fluxo contínuo de vídeo.

A função `stream_handler()`, descrita entre as linhas 50 e 68, é responsável pela transmissão propriamente dita. A cada iteração do laço, um frame é capturado por meio da função `esp_camera_fb_get()`, enviado ao cliente utilizando pacotes HTTP e, em seguida, liberado com `esp_camera_fb_return()` para reutilização do buffer. Esse processo ocorre de forma contínua, garantindo o streaming de vídeo em tempo real.

A inicialização geral do sistema ocorre na função `setup()`, apresentada entre as linhas 70 e 107. Inicialmente, é configurada a comunicação serial para depuração. Em seguida, é estruturada a variável `camera_config_t`, na qual são associados os pinos definidos anteriormente aos sinais da câmera, além da definição de parâmetros operacionais, como frequência do clock de 20 MHz, formato de pixel JPEG, resolução QVGA, qualidade de compressão e número de buffers de frame. Esses parâmetros foram escolhidos visando um equilíbrio entre qualidade de imagem, taxa de transmissão e capacidade de processamento do dispositivo.

Ainda na função `setup()`, nas linhas 102 a 105, o ESP32 realiza a conexão com a rede Wi-Fi, permanecendo em espera até que a comunicação seja estabelecida. Esse procedimento garante que o servidor só seja iniciado quando houver conectividade de rede disponível.

Por fim, a linha 107 aciona a função `startCameraServer()`, iniciando o servidor de streaming de vídeo. A função `loop()`, apresentada na linha 110, permanece vazia, uma

vez que, após a inicialização, o funcionamento do sistema passa a depender das rotinas internas do servidor HTTP e da biblioteca da câmera, que operam de forma assíncrona.

Assim, o sistema embarcado implementa um pipeline completo de aquisição e transmissão de imagens, no qual o ESP32 atua como dispositivo de borda responsável pela captura visual, enquanto o processamento de alto nível, como a detecção e classificação de sinais de trânsito, pode ser realizado externamente. Essa abordagem caracteriza uma arquitetura distribuída de visão computacional, adequada para aplicações embarcadas de baixo custo e operação em tempo real.

Controle de locomoção do Robô

O Arduino (Figura 19) é responsável pelo controle dos atuadores do robô móvel, executando os comandos de movimentação recebidos por comunicação Bluetooth. O firmware implementado permite que o robô se desloque em diferentes direções e acione um sinal sonoro, funcionando como a camada de controle de baixo nível do sistema.

O código-fonte completo do firmware encontra-se apresentado no Apêndice B.

Inicialmente, é incluída a biblioteca `SoftwareSerial.h` (linha 1 do Código B.1), responsável por permitir a criação de uma porta serial virtual para comunicação com o módulo Bluetooth. Em seguida, o objeto `bt` é instanciado (linha 4), configurado nos pinos digitais 3 (RX) e 4 (TX), possibilitando a recepção dos comandos enviados remotamente.

Na sequência, são definidas diretivas `#define` (linhas 7 a 10) associando os pinos digitais do Arduino aos sinais de controle da ponte H responsável pelos motores. Os pinos `IN1` e `IN2` controlam o motor do lado esquerdo (sentido frente e ré), enquanto `IN3` e `IN4` controlam o motor do lado direito. Também é definido o pino `BUZZER` (linha 13), utilizado para acionamento do sinal sonoro.

Na função `setup()` (iniciada na linha 15), as comunicações seriais são inicializadas (linhas 16 e 17), sendo a serial padrão utilizada para depuração e a serial `bt` para comunicação Bluetooth. Em seguida, os pinos associados aos motores e ao buzzer são configurados como saídas digitais (linhas 19 a 23). Por fim, a função `parar()` é chamada (linha 25) para garantir que o robô inicie em estado seguro, com todos os motores desligados.

A função `loop()` implementa a lógica principal de operação (linha 28). O código verifica continuamente se há dados disponíveis no módulo Bluetooth por meio de `bt.available()` (linha 29). Quando um caractere é recebido, ele é armazenado na variável `cmd` (linha 30) e exibido na serial para depuração (linha 31). Em seguida, é utilizada uma estrutura `switch-case` (linhas 33 a 40), que associa cada comando a uma ação específica de movimentação ou sinalização sonora.

Os comandos de movimentação incluem avanço (`'F'`), ré (`'B'`), giro à esquerda (`'L'`), giro à direita (`'R'`) e parada (`'S'`), correspondendo às chamadas das funções de

controle de motores (linhas 34 a 38). O comando `'Y'` (linha 39) aciona a função `buzzer()`, responsável por emitir um pulso sonoro curto.

As funções `frente()`, `re()`, `esquerda()` e `direita()` (linhas 44 a 70) implementam combinações específicas de sinais digitais nos pinos da ponte H, determinando o sentido de rotação de cada motor. Já a função `parar()` (linhas 72 a 77) desativa todos os sinais de controle, interrompendo o movimento do robô. Por fim, a função `buzzer()` (linhas 79 a 83) ativa temporariamente o buzzer por meio de um atraso controlado, produzindo um sinal sonoro breve.

Dessa forma, o Arduino atua como o módulo responsável pelo controle direto dos atuadores, recebendo comandos de alto nível via Bluetooth e convertendo-os em sinais elétricos adequados para a movimentação do robô.

3.3.3 Montagem do Protótipo

A construção do robô foi executada de forma modular, integrando as interfaces mecânica e eletrônica. A arquitetura do sistema priorizou a organização dos componentes para facilitar a manutenção e a estabilidade dos sinais elétricos. O esquema geral de interligação, que detalha a topologia da rede de componentes, é apresentado na Figura 25.

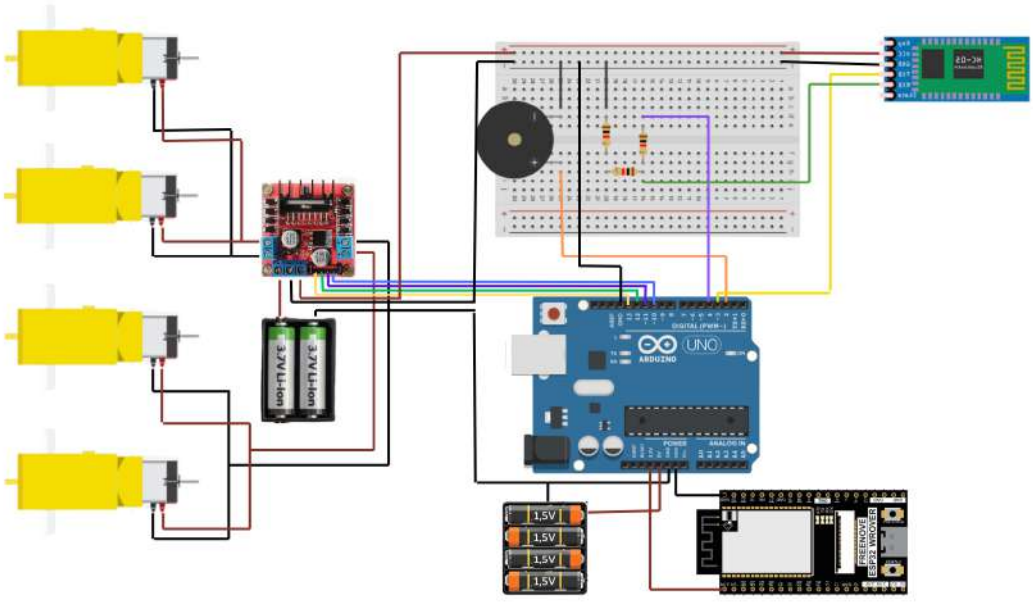


Figura 25 – Esquema geral de montagem e interligação dos componentes do robô móvel.
Fonte: Própria autora (2026)

A Figura 25 ilustra a integração entre os motores de corrente contínua (CC), a ponte H para o acionamento de potência, o microcontrolador Arduino Uno como unidade

central de processamento, além dos módulos de comunicação sem fio e sinalização sonora. A seguir, detalha-se a montagem dividida por subsistemas funcionais.

Sistema Mecânico e de Tração

A estrutura física do robô utiliza um chassi do tipo *4WD* (*Four-Wheel Drive*), composto por uma base de acrílico onde foram fixados quatro motores CC acoplados a rodas independentes. O gerenciamento elétrico desses motores é realizado por um driver de potência (Ponte H L298N), que permite o controle bidirecional da rotação.

Os motores foram agrupados lateralmente: os atuadores do lado direito foram conectados ao canal A da ponte H, enquanto os do lado esquerdo foram vinculados ao canal B, conforme a Figura 26. Esta configuração é essencial para a técnica de direção diferencial, possibilitando manobras de avanço, recuo e rotação sobre o próprio eixo através da variação de polaridade e sinais PWM (*Pulse Width Modulation*).

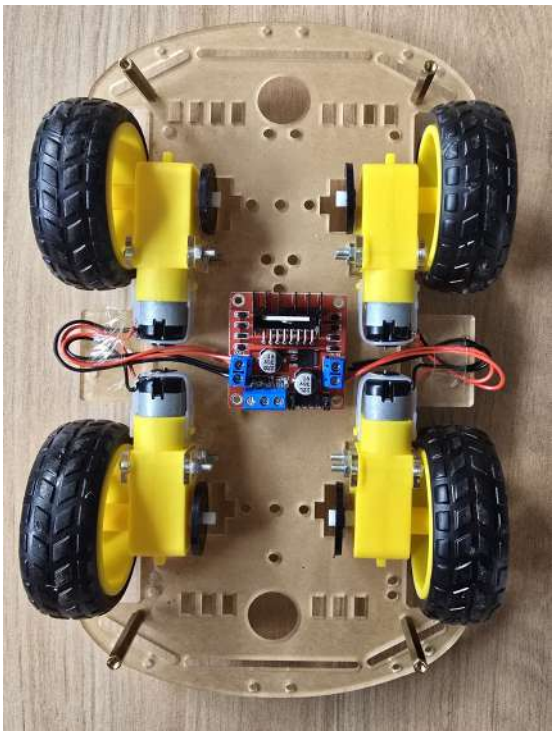


Figura 26 – Montagem inicial do chassi 4WD.
Fonte: Própria autora (2026)

A alimentação do sistema de potência provém de duas células de lítio de 3,7 V associadas em série, resultando em uma tensão nominal de 7,4 V. Esta fonte é dedicada exclusivamente à ponte H para evitar que ruídos eletromagnéticos provenientes dos motores interfiram na lógica de controle.

Sistema de Controle

O processamento central é realizado por uma placa Arduino Uno. Conforme detalhado na Figura 27, as portas digitais 10, 11, 12 e 13 foram configuradas como saídas para os pinos de controle (IN1 a IN4) da ponte H. Esta interface permite a execução dos algoritmos de locomoção descritos no Código B.1.

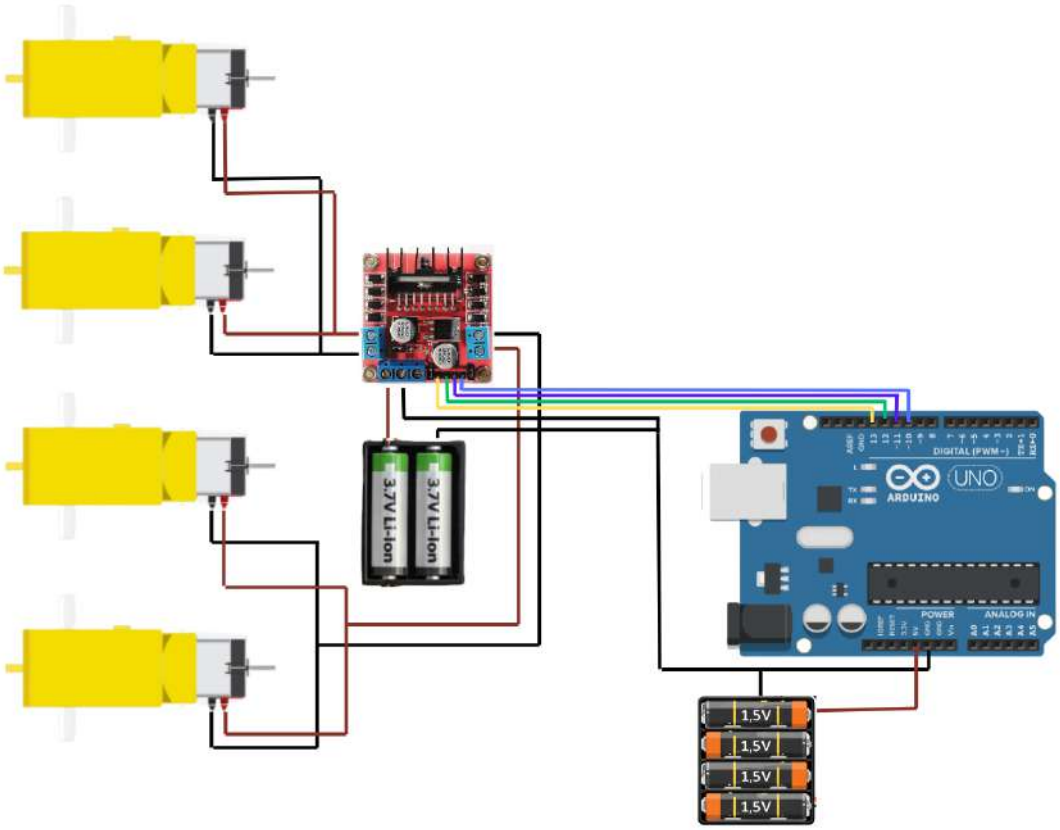


Figura 27 – Arduino Uno e conexões com a ponte H
Fonte: Própria autora (2026)

Diferente do sistema de potência, o Arduino é alimentado por uma fonte independente de 6 V conectada ao pino *VIN*, garantindo estabilidade de tensão para o processador e para os sensores periféricos.

Sistema de Visão, Comunicação e Feedback

Para a interface de comunicação sem fio, utilizou-se o módulo Bluetooth HC-05, estabelecendo o enlace de dados via protocolo UART. A montagem exigiu a implementação de um divisor de tensão resistivo entre o Arduino Uno e o módulo, visando a compatibilização dos níveis lógicos de 5 V e 3,3 V, respectivamente, garantindo assim a integridade do hardware contra sobretensões (Figura 28).

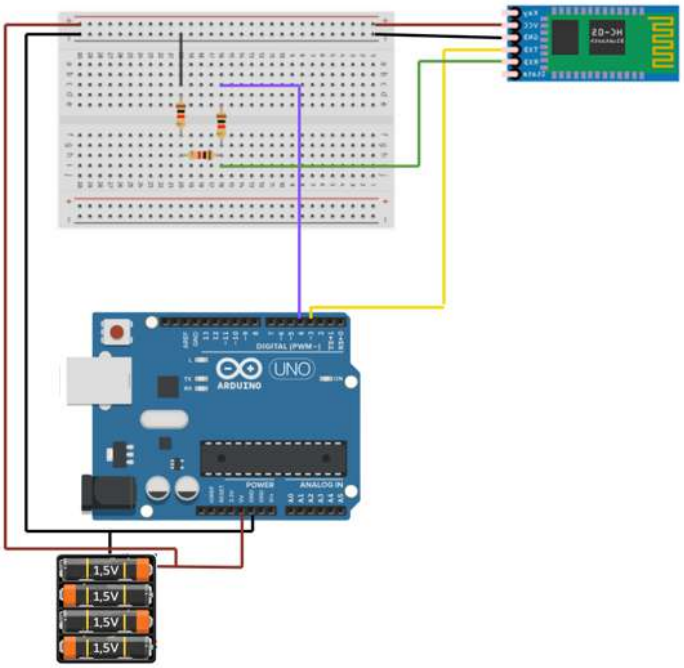


Figura 28 – Conexão do módulo Bluetooth ao Arduino com divisor de tensão.
Fonte: Própria autora (2026)

Adicionalmente, foi integrado ao sistema um buzzer com a finalidade de fornecer sinais sonoros, auxiliando na sinalização dos diferentes estados de operação do robô. O buzzer foi conectado à porta digital 2 do Arduino Uno (Figura 29), sendo acionado conforme a lógica definida no código B.1, permitindo a emissão de alertas sonoros durante a execução das tarefas do sistema.

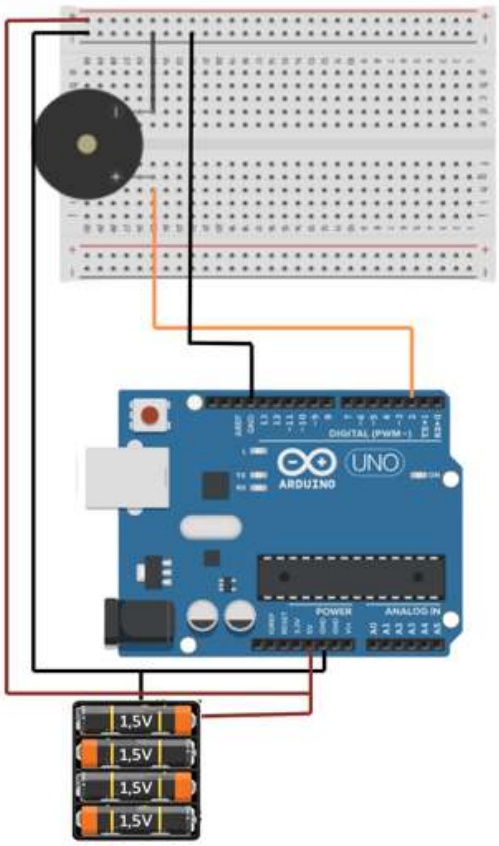


Figura 29 – Conexão do buzzer ao Arduino Uno.
Fonte: Própria autora (2026)

Ademais, o sistema eletrônico do robô incorpora um módulo ESP32-CAM, responsável pela aquisição e transmissão de imagens em tempo real. Esse módulo atua como o sistema de visão embarcado do robô, realizando a captura das imagens por meio de um sensor de câmera e a transmissão dos dados via rede sem fio, conforme a lógica definida no código A.1.

Fisicamente, o ESP32-CAM foi integrado ao sistema de forma independente do controle de locomoção, sendo alimentado diretamente pelo arduino. A Figura 30 apresenta o esquema de conexão do módulo ESP32-CAM ao sistema eletrônico do robô, destacando sua integração ao protótipo.

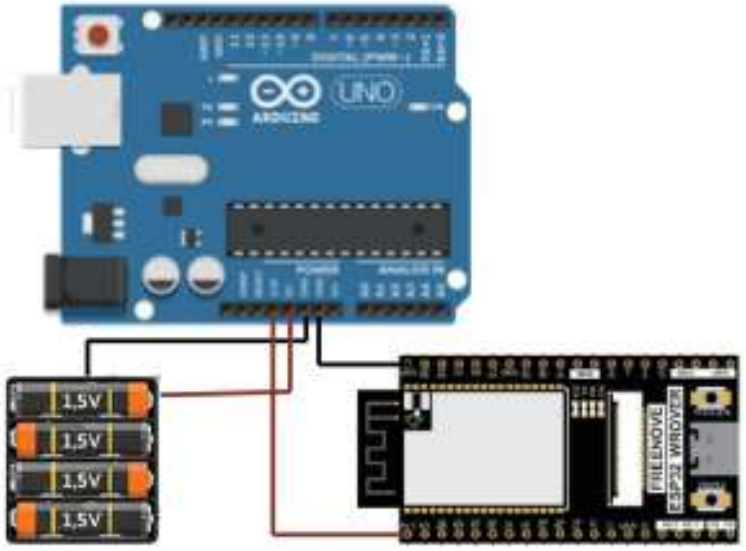
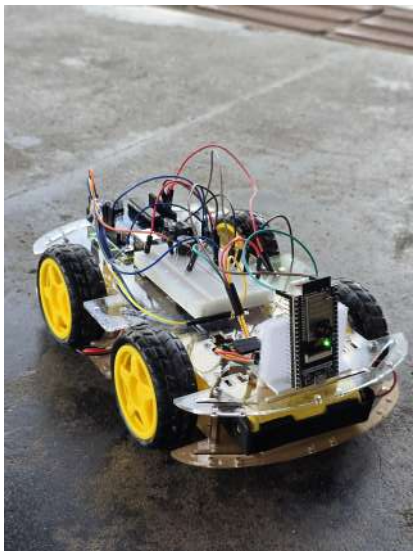
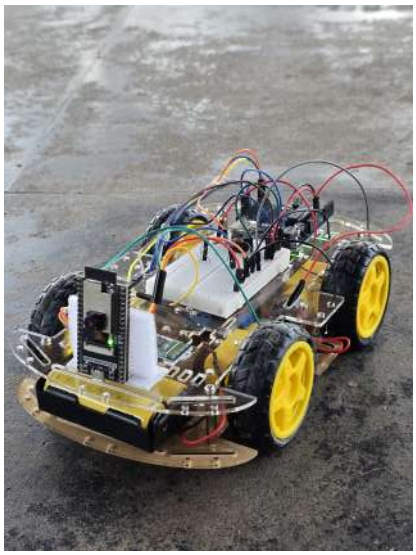


Figura 30 – Esquema de conexão do módulo ESP32 ao sistema eletrônico do robô.
Fonte: Própria autora (2026)

Deste modo, obteve-se o robô VISION completo, com todos sistemas em pleno funcionamento, conforme apresentado nas Figuras 31 e 32. Esta montagem completa representa o VISION pronto para testes e execução das funcionalidades programadas.

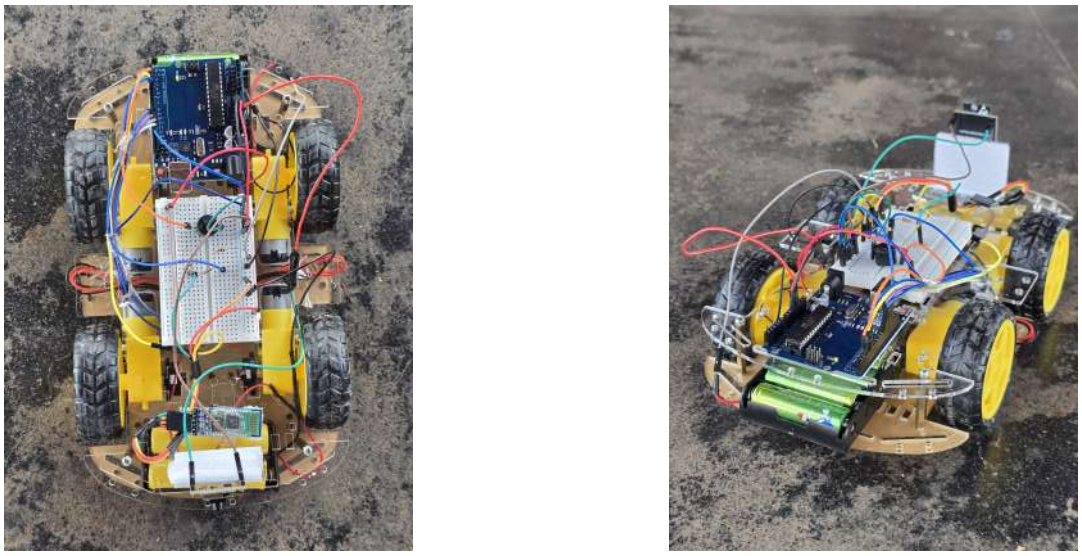


(a) Frente.



(b) Frente diagonal.

Figura 31 – Robô montado - vistas frontal e frontal diagonal.
Fonte: Própria autora (2026)



(a) Visão por cima. (b) De lado.

Figura 32 – Robô montado - vistas superior e lateral.
Fonte: Própria autora (2026)

4 Resultados

Este capítulo apresenta os resultados obtidos nas diferentes etapas de avaliação do sistema de visão computacional desenvolvido. Inicialmente, são discutidos os **Resultados de Validação** do modelo treinado na plataforma Edge Impulse, incluindo tanto os indicadores globais de desempenho quanto as análises detalhadas por classe, permitindo verificar a eficiência do processo de treinamento. Em seguida, são apresentados os **Resultados dos Testes Simulados** realizados com o auxílio de um dispositivo móvel, em um cenário controlado, sem a operação do robô VISION. Por fim, são analisados os **Resultados dos Testes Práticos** com robô VISION, avaliando seu desempenho em condições reais de uso e comprovando a aplicabilidade da solução proposta.

4.1 Resultados de Validação

A validação do modelo de visão computacional foi realizada a partir das métricas disponibilizadas pela plataforma *Edge Impulse*, escolhidas por permitirem avaliar o desempenho global do classificador e seu comportamento individual para cada classe. Foram considerados indicadores como acurácia, *loss*, precisão, *recall* e *F1-score*, possibilitando uma análise quantitativa da capacidade de generalização do modelo treinado.

Desempenho Global do Modelo

A Figura 33 apresenta as métricas finais obtidas ao término do processo de treinamento e validação do modelo. Observa-se que o classificador atingiu **acurácia de 90,0%**, isto é, 90% das amostras do conjunto de validação foram corretamente classificadas, indicando um bom desempenho global. O valor de ***loss* igual a 0,32**, ou seja, o erro médio entre as saídas previstas pelo modelo e os valores reais, evidencia um processo de aprendizado estável, sem indícios de sobreajuste.

Além disso, o modelo apresentou **precisão de 93%**, o que significa que a maioria das classificações realizadas corresponde à classe correta. O ***recall* de 90%** indica que grande parte das amostras relevantes foi corretamente identificada pelo modelo. Por fim, o ***F1-score* global de 90%** confirma o equilíbrio entre precisão e *recall*, reforçando a consistência dos resultados obtidos na etapa de validação.

Figura 33 – Métricas finais de treinamento e validação do modelo.
Fonte: Própria autora (2025)

Análise por Classe: Matriz de Confusão e F1-score

A matriz de confusão, apresentada na Figura 34, é empregada para avaliar o desempenho do modelo de classificação, permitindo a comparação entre as classes previstas e as classes reais do conjunto de validação. Essa representação possibilita identificar, de forma objetiva, os acertos e erros do classificador para cada classe analisada.

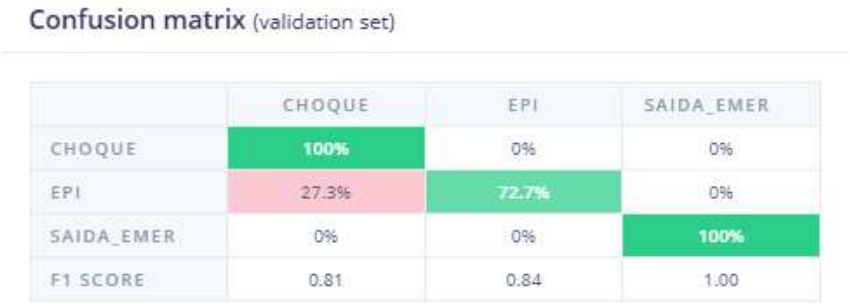


Figura 34 – Matriz de confusão resultante da validação do modelo.
Fonte: Própria autora (2025)

A análise da matriz de confusão evidencia um desempenho consistente do modelo na classificação das três classes avaliadas: *Choque*, *EPI* e *Saída de Emergência*. Observa-se que as classes *Choque* e *Saída de Emergência* apresentaram taxa de acerto de 100%, não sendo confundidas com as demais categorias, o que indica uma boa separabilidade das características aprendidas pela rede neural.

Em contrapartida, a classe *EPI* apresentou desempenho inferior, com aproxima-

damente 72,7% de acerto, sendo parcialmente confundida com a classe *Choque*. Esse comportamento pode estar associado à similaridade visual entre essas classes ou à menor diversidade de amostras representativas no conjunto de dados, impactando a capacidade discriminativa do modelo.

Os valores de F1 Score reforçam essa interpretação, com pontuação máxima para a classe *Saída de Emergência* (1,00), seguida da classe *Choque* (0,81) e da classe *EPI* (0,74). De modo geral, os resultados indicam um bom equilíbrio entre precisão e *recall*, embora apontem possibilidades de aprimoramento, especialmente no que se refere à redução das confusões envolvendo a classe *EPI*.

4.2 Resultados dos Testes Simulados

A fase de **Teste** avalia a capacidade de generalização do modelo utilizando 20% da base de dados original, composta por imagens inéditas que não foram vistas durante o treinamento ou validação. O objetivo dessa etapa é verificar como o modelo se comporta ao ser exposto a dados diferentes daqueles utilizados no processo de aprendizado, assegurando que o modelo não esteja apenas memorizando os dados, mas sim generalizando bem para novos exemplos.

A Figura 35 apresenta o resultado de teste para o modelo elaborado, demonstrando que obteve-se uma **acurácia de 97,33%**, o que indica um bom desempenho de generalização. A *F1 Score* também se manteve elevada, com valor de 99% para todas as classes.

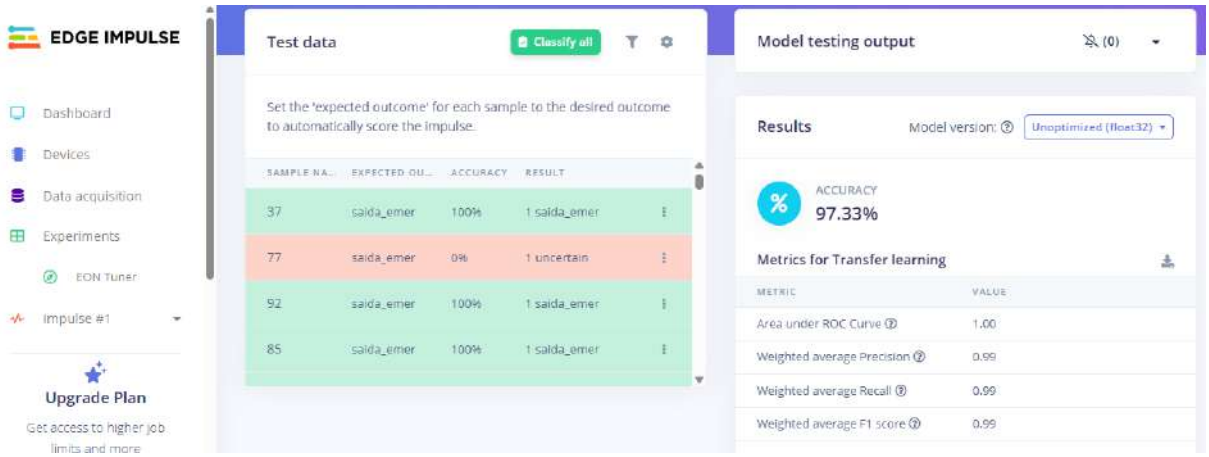


Figura 35 – Resultados da fase de teste na plataforma *Edge Impulse*
Fonte: Própria autora (2025)

A análise da matriz de confusão, apresentada na Figura 36, revela que a maioria das classificações ocorreu corretamente, com o modelo acertando 100% das amostras da classe *Choque* e obtendo 96,2% de acertos na classe *Saída de Emergência*. Já a classe *EPI* obteve 96% de precisão, com uma pequena confusão ocorrendo com a classe *Choque*.

Confusion matrix

	CHOQUE	EPI	SAIDA_EMER	UNCERTAIN
CHOQUE	100%	0%	0%	0%
EPI	4%	96%	0%	0%
SAIDA_EMER	0%	0%	96.2%	3.8%
F1 SCORE	0.98	0.98	0.98	

Figura 36 – Matriz de Confusão na Etapa de Testes

4.3 Resultados nos Testes Práticos

A etapa de testes práticos foi realizada para a validação funcional do sistema por meio de ensaios experimentais em condições próximas à aplicação real do robô VISION. Essa fase foi organizada em duas etapas principais:

1. **Testes com o celular:** realizados sem a utilização do robô, apenas para avaliar o desempenho do modelo de reconhecimento de placas.
2. **Testes dinâmicos com o robô:** realizados com o robô em funcionamento, para validar o sistema completo em condições reais de operação.

Em ambas as etapas, os resultados apresentados correspondem aos valores de confiança fornecidos pelo modelo, obtidos a partir da função *Softmax* apresentada na seção 2. Essa função converte os valores brutos produzidos pelo modelo em probabilidades normalizadas, permitindo interpretar a classe com maior valor como a predição final, associada ao seu respectivo nível de confiança. Para os testes, foi predefinido um valor mínimo de confiança igual a 0,6 (60%). Assim, predições com valor inferior a esse limiar foram classificadas como incertas, não sendo consideradas como reconhecimentos válidos.

Etapa 1: Testes com o celular

Para a primeira etapa, foi definida uma base de dados composta por 30 imagens de placas, utilizada para verificar a capacidade do modelo em identificar corretamente as classes antes dos testes dinâmicos. Assim, foram realizados 10 testes para cada classe (**EPI**, **Saída de Emergência** e **Choque**). Os testes foram conduzidos utilizando apenas um celular Samgung S24 Ultra, sem a presença do robô, para validar previamente o desempenho do modelo de visão computacional.

O acesso ao modelo treinado foi feito por meio do QR Code disponibilizado pelo *Edge Impulse*. Ao escanear o QR Code com o celular, acessou-se a interface de inferência

do modelo, permitindo enviar imagens capturadas a partir da câmera e obter, em tempo real, a classificação da imagem. Essa abordagem permitiu verificar a capacidade do modelo em identificar corretamente cada classe antes de realizar os testes dinâmicos com o robô.

A Figura 37 apresenta a plataforma *Edge Impulse* com o QR Code, indicando como o acesso ao modelo foi realizado.

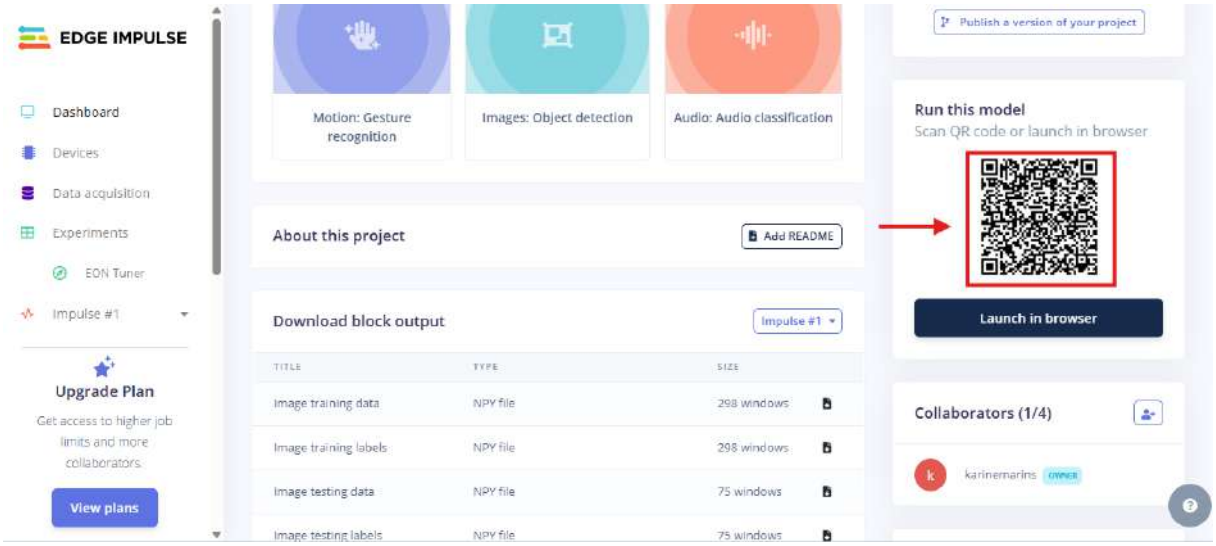


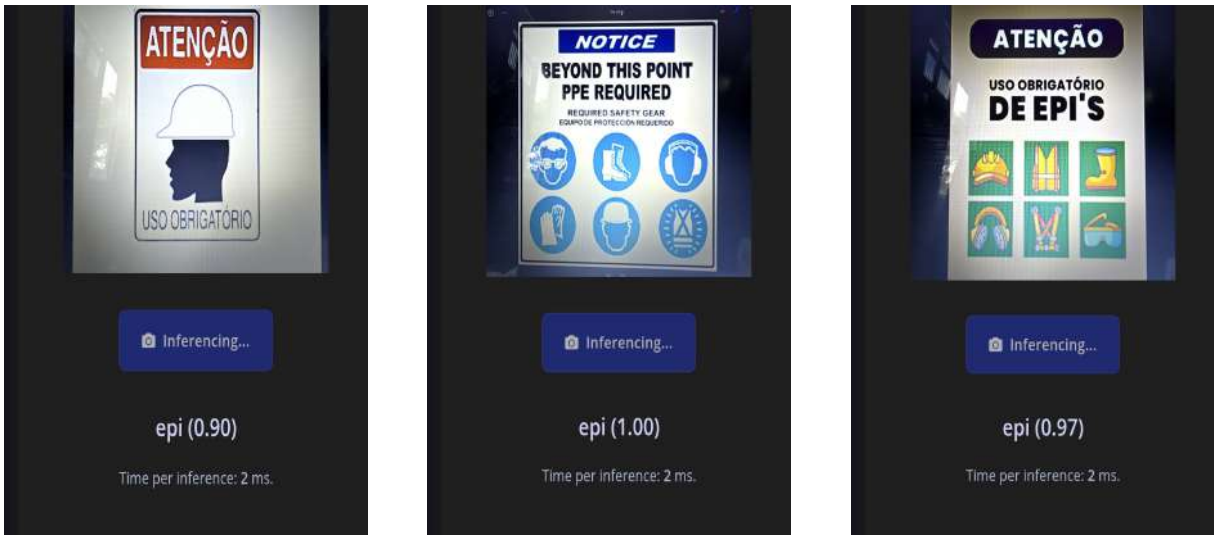
Figura 37 – QR Code do Edge Impulse, utilizado para acessar o modelo de inferência.
Fonte: Própria autora (2025)

Classe EPI

Para a classe *EPI*, os testes simulados apresentaram valores de confiança variando entre 0,69 e 1,00, com média de 0,91, indicando um bom nível de reconhecimento dessa classe pelo modelo. Observa-se, entretanto, a ocorrência de variações pontuais nos níveis de confiança atribuídos, o que sugere maior sensibilidade do classificador a determinadas condições de captura das imagens. Os resultados obtidos nos testes individuais para a classe *EPI* são apresentados nas Figuras 38 a 41.

(a) Teste 1 (b) Teste 2 (c) Teste 3

Figura 38 – Resultados dos testes 1 a 3 da classe **EPI**.
Fonte: Própria autora (2025).



(a) Teste 4 (b) Teste 5 (c) Teste 6

Figura 39 – Resultados dos testes 4 a 6 da classe **EPI**.
Fonte: Própria autora (2025).

(a) Teste 7

(b) Teste 8

(c) Teste 9

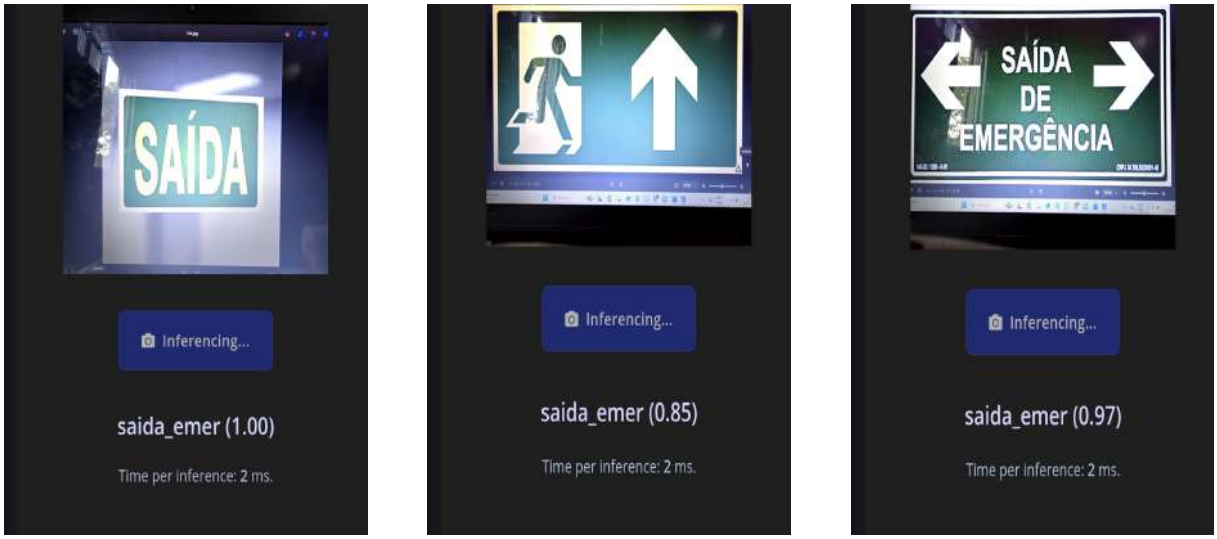
Figura 40 – Resultados dos testes 7 a 9 da classe **EPI**.
Fonte: Própria autora (2025).

(a) Teste 10

Figura 41 – Resultado do teste 10 da classe **EPI**.
Fonte: Própria autora (2025).

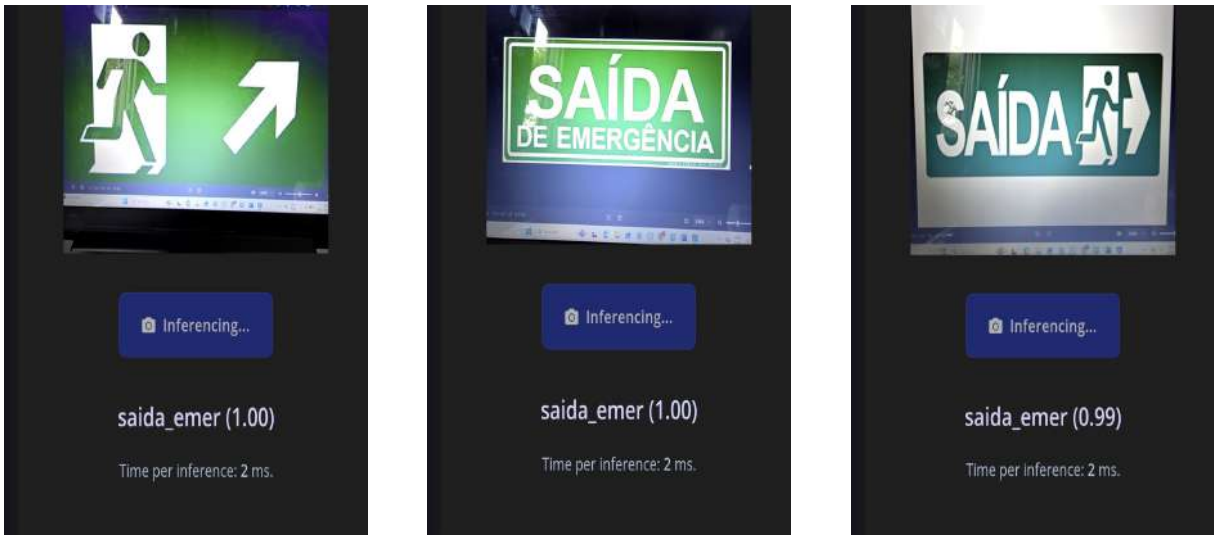
Classe Saída de Emergência

Para a classe *Saída de Emergência*, os testes simulados apresentaram valores de confiança variando entre 0,85 e 1,00, com média de 0,98, evidenciando elevada consistência do modelo na identificação dessa classe. Observa-se que a maioria das inferências resultou em valores próximos de 1,00, indicando alta confiabilidade nas classificações realizadas. Os resultados obtidos nos testes individuais para a classe *Saída de Emergência* são apresentados nas Figuras 62 a 65.



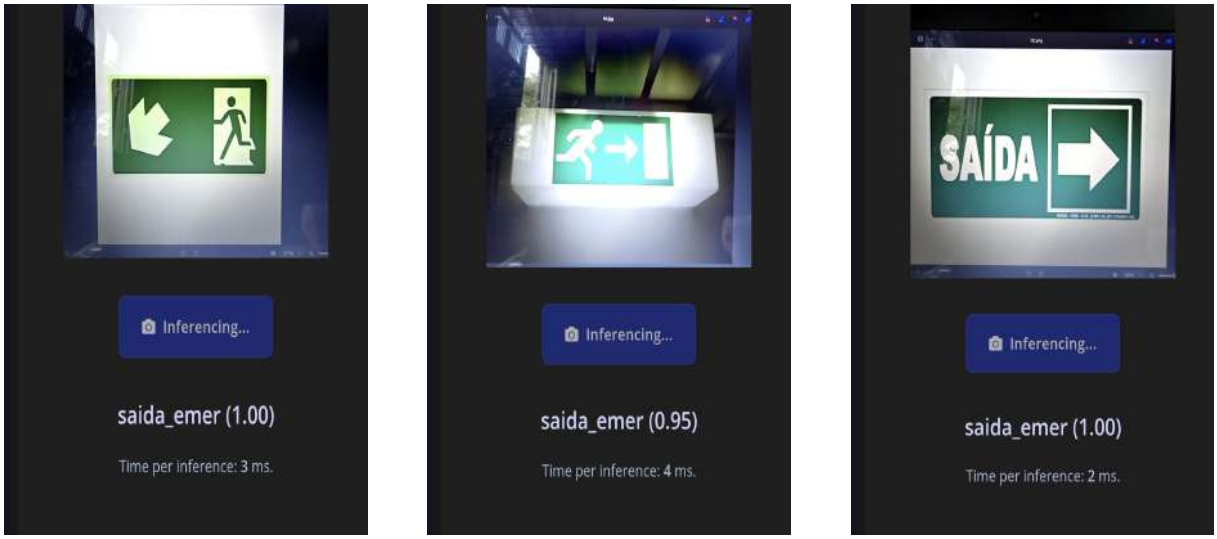
(a) Teste 1 (b) Teste 2 (c) Teste 3

Figura 42 – Resultados dos testes 1 a 3 da classe **Saída de Emergência**.
Fonte: Própria autora (2025).



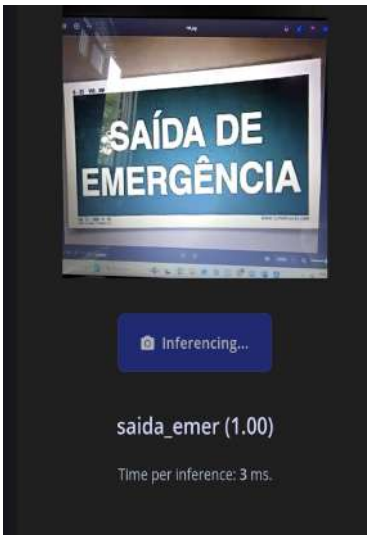
(a) Teste 4 (b) Teste 5 (c) Teste 6

Figura 43 – Resultados dos testes 4 a 6 da classe **Saída de Emergência**.
Fonte: Própria autora (2025).



(a) Teste 7 (b) Teste 8 (c) Teste 9

Figura 44 – Resultados dos testes 7 a 9 da classe **Saída de Emergência**.
Fonte: Própria autora (2025).

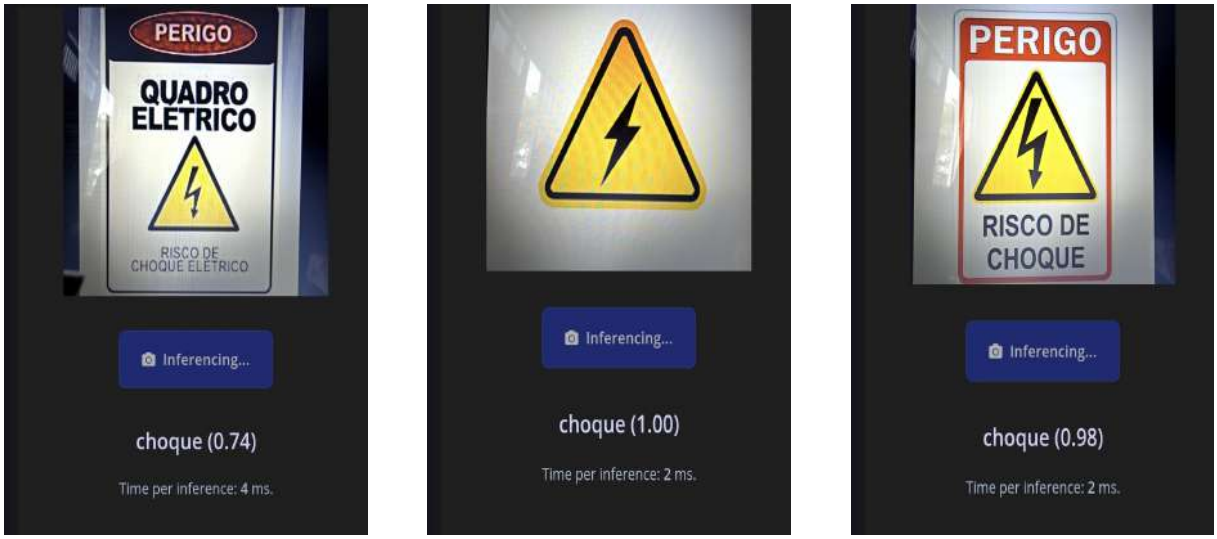


(a) Teste 10

Figura 45 – Resultado do teste 10 da classe **Saída de Emergência**.
Fonte: Própria autora (2025).

Classe Choque

Para a classe *Choque*, os testes simulados apresentaram valores de confiança variando entre 0,67 e 1,00, com média de 0,89, indicando bom desempenho do modelo, porém com maior variabilidade em comparação às demais classes. Essa dispersão nos valores de confiança sugere maior sensibilidade do classificador a características visuais dessa classe. Os resultados obtidos nos testes individuais para a classe *Choque* são apresentados nas Figuras 46 a 49.



(a) Teste 1 (b) Teste 2 (c) Teste 3

Figura 46 – Resultados dos testes 1 a 3 da classe **Choque**.
Fonte: Própria autora (2025).

(a) Teste 4 (b) Teste 5 (c) Teste 6

Figura 47 – Resultados dos testes 4 a 6 da classe **Choque**.
Fonte: Própria autora (2025).

(a) Teste 7 (b) Teste 8 (c) Teste 9

Figura 48 – Resultados dos testes 7 a 9 da classe **Choque**.
Fonte: Própria autora (2025).

(a) Teste 10

Figura 49 – Resultado do teste 10 da classe **Choque**.
Fonte: Própria autora (2025).

A Tabela 2 apresenta os resultados obtidos nos 30 testes de inferência realizados, sendo 10 para cada classe avaliada. Observa-se que a classe *Saída de Emergência* apresentou os maiores níveis de confiança ao longo dos testes, com valores predominantemente iguais ou próximos de 1,00, evidenciando elevada consistência do modelo para essa classe. A classe *EPI* apresentou resultados satisfatórios, embora com maior variação entre os testes, incluindo valores mais baixos em situações específicas. Já a classe *Choque* apresentou a maior variabilidade nos resultados, com alguns testes apresentando valores inferiores às demais classes, indicando maior dificuldade do modelo na identificação desse padrão visual.

Ainda assim, os resultados demonstram desempenho global adequado para todas as classes analisadas.

Tabela 2 – Valores de confiança nos testes de inferência para as classes EPI, Saída de Emergência e Choque

Teste	EPI	Saída de Emergência	Choque
1	0,99	1,00	0,74
2	0,98	0,85	1,00
3	0,69	0,97	0,98
4	0,90	1,00	0,84
5	1,00	1,00	0,93
6	0,97	0,99	1,00
7	0,92	1,00	0,88
8	0,92	0,95	0,67
9	1,00	1,00	0,86
10	0,77	1,00	1,00
Média	**0,91**	**0,98**	**0,89**

Fonte: Própria autora (2025).

Etapa 2: Testes Dinâmicos com o Robô

Após a validação do modelo na Etapa 1, foram realizados testes dinâmicos utilizando o VISION, com o objetivo de avaliar o desempenho do sistema em condições reais de operação. Os ensaios consistiram na avaliação do sistema de visão embarcado no robô em um ambiente controlado, no qual o robô foi posicionado em frente às placas de sinalização, devendo identificar e reagir corretamente às classes **EPI**, **Saída de Emergência** e **Choque**.

Para o controle do robô durante os testes dinâmicos, foi empregada comunicação via Bluetooth entre o módulo embarcado e um dispositivo móvel. O envio dos comandos foi realizado por meio do aplicativo *Arduino Bluetooth Controller*, disponível na plataforma Google Play[1]. O aplicativo possibilitou o acionamento manual dos movimentos do robô (avanço, ré, giro à esquerda e à direita), bem como o acionamento do buzzer, permitindo o correto posicionamento do sistema diante das placas durante os experimentos.

A interface do aplicativo, apresentada na Figura 50, possui botões para envio de comandos seriais via módulo HC-05, facilitando a integração com o Arduino e garantindo praticidade e confiabilidade na execução dos testes.

[1] Disponível em: https://play.google.com/store/apps/details?id=com.giristudio.hc05.bluetooth.arduino.control&hl=pt_BR. Acesso em: 11 fev. 2026.

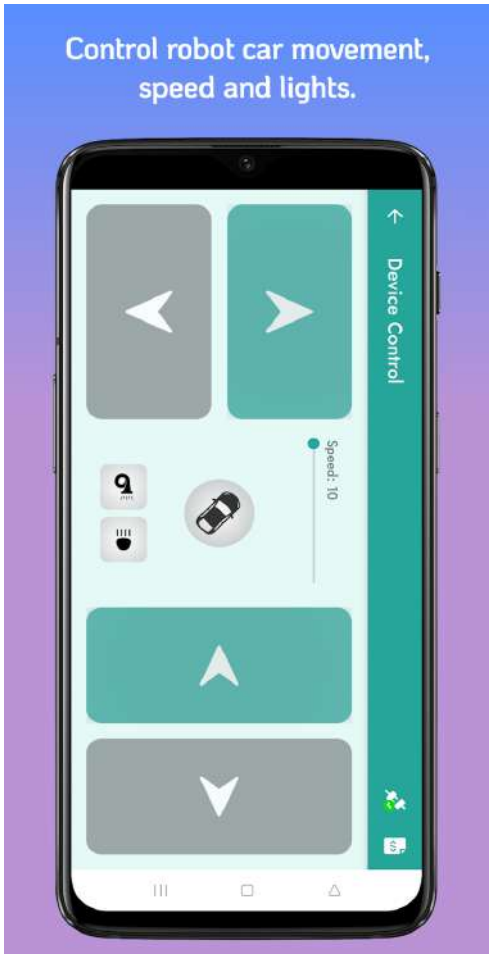


Figura 50 – Tela principal do aplicativo Arduino Bluetooth Controller utilizado para envio de comandos ao VISION.
Fonte: Adaptado de (GIRI STUDIO, 2026)

Para essa etapa, definiu-se as placas a serem utilizadas, conforme ilustrado nas Figuras 51, 52 e 53.

Figura 51 – Placas EPI para experimentos
Fonte: Própria autora (2025)

Figura 52 – Placas Saida de Emergência para experimentos
Fonte: Própria autora (2025)

Figura 53 – Placas Choque para experimentos
Fonte: Própria autora (2025)

As placas foram dispostas de modo a permanecer dentro do campo de visão da câmera embarcada no robô. Da mesma forma, o VISION foi orientado de maneira a garantir a correta captura das imagens das sinalizações. As Figuras 54 a 57 apresentam exemplos da configuração utilizada durante a realização dos testes experimentais.

Figura 54 – VISION em testes dinâmicos - Exemplo 1
Fonte: Própria autora (2026)

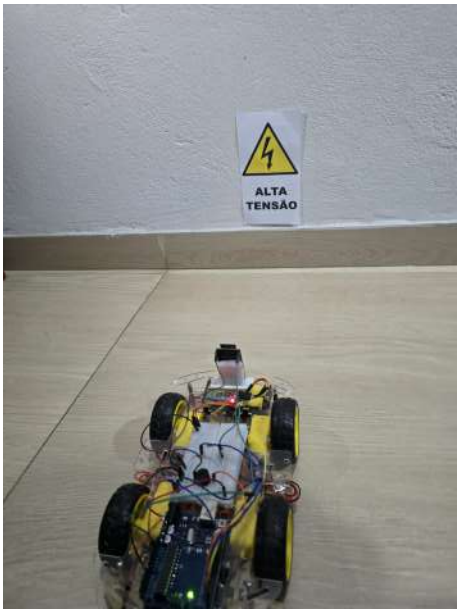


Figura 55 – VISION em testes dinâmicos - Exemplo 2
Fonte: Própria autora (2026)

Figura 56 – VISION em testes dinâmicos - Exemplo 3
Fonte: Própria autora (2026)

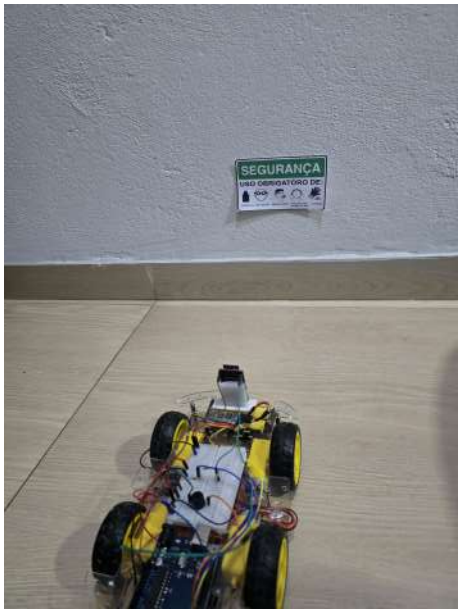


Figura 57 – VISION em testes dinâmicos - Exemplo 4
Fonte: Própria autora (2026)

Com o robô VISION devidamente posicionado, a inferência do modelo de visão computacional foi realizada no celular, por meio do acesso via QR Code (Figura 37), a partir da imagem recebida do sistema embarcado. Para isso, as imagens capturadas pela câmera do ESP32 foram transmitidas por meio de um link HTTP gerado pelo servidor embarcado no próprio módulo do ESP32. O celular foi então segurado com as mãos de frente para o computador de forma a capturar essa imagem transmitida, permitindo a classificação das placas no campo de visão do robô. Esse procedimento possibilitou a visualização em

tempo real das imagens adquiridas em um computador, permitindo o monitoramento do cenário observado pelo robô durante a execução dos testes experimentais.

Os resultados obtidos nos testes foram organizados por classes, com o objetivo de facilitar a análise qualitativa do desempenho do sistema.

Classe EPI

Para a classe **EPI**, os valores de confiança obtidos variaram entre 0,75 e 0,98, apresentando média aproximada de 0,93. As Figuras 58 a 61 apresentam os resultados obtidos durante os testes experimentais para essa classe, evidenciando boa capacidade de detecção mesmo sob variações de posicionamento e iluminação.

(a) Teste 1 (b) Teste 2 (c) Teste 3

Figura 58 – Testes Dinâmicos 1 a 3: detecção da classe **EPI**.
Fonte: Própria autora (2026).

(a) Teste 4 (b) Teste 5 (c) Teste 6

Figura 59 – Testes Dinâmicos 4 a 6: detecção da classe **EPI**.
Fonte: Própria autora (2026).

(a) Teste 7 (b) Teste 8 (c) Teste 9

Figura 60 – Testes Dinâmicos 7 a 9: detecção da classe **EPI**.
Fonte: Própria autora (2026).

(a) Teste 10

Figura 61 – Teste Dinâmico 10: detecção da classe **EPI**.
Fonte: Própria autora (2026).

Classe Saída de Emergência

Para a classe **Saída de Emergência**, os resultados visuais são apresentados nas Figuras 62 a 65. Neste caso, os valores de confiança obtidos para essa classe variaram entre 0,88 e 1,00, apresentando média aproximada de 0,96. De modo geral, os resultados demonstram desempenho consistente do modelo na detecção das sinalizações de saída de emergência, mantendo estabilidade nas respostas mesmo sob variações de iluminação e posicionamento da câmera ao longo dos testes realizados.

(a) Teste 1 (b) Teste 2 (c) Teste 3

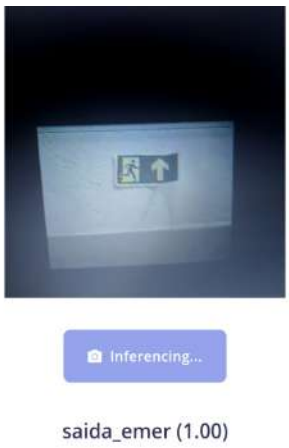
Figura 62 – Testes Dinâmicos 1 a 3: detecção da classe **Saída de Emergência**.
Fonte: Própria autora (2026).

(a) Teste 4 (b) Teste 5 (c) Teste 6

Figura 63 – Testes Dinâmicos 4 a 6: detecção da classe **Saída de Emergência**.
Fonte: Própria autora (2026).

(a) Teste 7 (b) Teste 8 (c) Teste 9

Figura 64 – Testes Dinâmicos 7 a 9: detecção da classe **Saída de Emergência**.
Fonte: Própria autora (2026).



(a) Teste 10

Figura 65 – Teste Dinâmico 10: detecção da classe **Saída de Emergência**.
Fonte: Própria autora (2026).

Classe Choque

Para a classe **Choque**, as Figuras 66 a 68 ilustram os resultados obtidos nos testes dinâmicos. Em suma, os valores de confiança obtidos para essa classe variaram entre 0,75 e 1,00, apresentando média aproximada de 0,93. Observou-se ainda a ocorrência de um resultado classificado como incerto em um dos testes realizados, o que pode estar associado a variações nas condições de aquisição da imagem, como iluminação, posicionamento ou ângulo de captura. De modo geral, a análise dos resultados indica que o sistema apresentou bom desempenho na identificação desse tipo de sinalização, mesmo diante de variações no ângulo de visão e nas condições do ambiente durante a realização dos testes.

(a) Teste 1 (b) Teste 2 (c) Teste 3

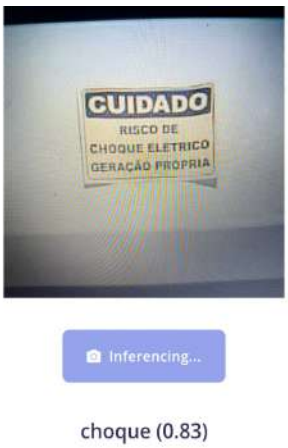
Figura 66 – Testes Dinâmicos 1 a 3: detecção da classe **Choque**.
Fonte: Própria autora (2026).

(a) Teste 4 (b) Teste 5 (c) Teste 6

Figura 67 – Testes Dinâmicos 4 a 6: detecção da classe **Choque**.
Fonte: Própria autora (2026).

(a) Teste 7 (b) Teste 8 (c) Teste 9

Figura 68 – Testes Dinâmicos 7 a 9: detecção da classe **Choque**.
Fonte: Própria autora (2026).



(a) Teste 10

Figura 69 – Teste Dinâmico 10: detecção da classe **Choque**.
Fonte: Própria autora (2026).

A Tabela 3 apresenta os resultados obtidos nos testes dinâmicos realizados. Em síntese, a classe **EPI** apresentou elevados índices de confiança ao longo dos testes, com valores predominantemente superiores a 0,90, indicando boa capacidade de detecção mesmo sob variações de iluminação e ângulo de captura. De forma semelhante, a classe **Saída de Emergência** apresentou resultados consistentes, incluindo valores máximos de confiança iguais a 1, evidenciando elevada robustez do modelo para essa categoria. Já a classe **Choque** também apresentou bom desempenho geral, embora tenha sido observada maior variação nos resultados, incluindo a ocorrência de um valor classificado como incerto, o que sugere maior sensibilidade dessa classe às condições de aquisição das imagens. De modo geral, os resultados obtidos demonstram a eficácia do sistema proposto para a identificação das classes avaliadas em condições experimentais.

Tabela 3 – Valores de confiança nos testes dinâmicos para as classes EPI, Saída de Emergência e Choque

Teste	**EPI**	**Saída de Emergência**	**Choque**
1	0,96	0,96	0,98
2	0,89	0,97	0,98
3	0,92	0,92	0,99
4	0,95	1,00	0,75
5	0,75	0,91	1,00
6	0,97	0,98	0,97
7	0,97	0,88	0,90
8	0,98	1,00	0,99
9	0,96	0,98	Incerto
10	0,92	1,00	0,83
Média	**0,93**	**0,96**	**0,93**

Fonte: Própria autora (2026).

5 Conclusões

O presente trabalho abordou o desenvolvimento do VISION, um robô móvel equipado com visão computacional para a detecção de placas de segurança, como saída de emergência, risco de choque elétrico e uso obrigatório de equipamentos de proteção individual (EPI). A proposta foi motivada pelos avanços tecnológicos nas áreas de robótica e inteligência artificial, com destaque para a aplicação de Redes Neurais Convolucionais (CNNs) em sistemas autônomos embarcados de baixo custo.

A partir da metodologia adotada, que envolveu revisão bibliográfica, desenvolvimento do hardware e do software, além da coleta e preparação de dados para o treinamento do modelo, foi possível estruturar um protótipo funcional utilizando um kit chassi 4WD, motores DC, módulo Bluetooth, câmera e microcontrolador ESP32. A utilização da plataforma Edge Impulse mostrou-se adequada para o treinamento e implementação da CNN, facilitando a integração entre o modelo de aprendizado de máquina e o dispositivo embarcado.

Os resultados experimentais demonstraram o bom desempenho do sistema proposto. Nos **testes simulados**, o modelo apresentou acurácia de 97,33%, com *F1 Score* de aproximadamente 99% para as classes avaliadas, indicando elevada capacidade de generalização. Nos **testes de inferência realizados com o celular**, observou-se desempenho consistente entre as classes. A classe *Saída de Emergência* apresentou confiança média do modelo predominantemente próxima de 1,00, enquanto a classe *EPI* apresentou confiança média aproximada de 0,91, e a classe *Choque* apresentou confiança média aproximada de 0,93, embora com maior variabilidade entre os testes. Nos **testes dinâmicos com o robô**, a classe *EPI* apresentou confiança média do modelo predominantemente superior a 0,90, a classe *Saída de Emergência* manteve elevada estabilidade, com confiança média próxima de 1,00, e a classe *Choque* apresentou bom desempenho geral, com confiança média de aproximadamente 0,93, evidenciando maior sensibilidade às condições de aquisição das imagens.

Por fim, conclui-se que este trabalho contribuiu para a aplicação prática de conceitos relacionados à robótica móvel, visão computacional e CNN, evidenciando o potencial dessas tecnologias para aplicações em ambientes industriais e sistemas autônomos inteligentes.

Como perspectivas de trabalhos futuros, sugere-se a ampliação da base de dados de treinamento, a inclusão de novos tipos de placas de segurança, a otimização do modelo visando melhorar a precisão e o tempo de resposta, a integração do sistema de detecção com estratégias de navegação autônoma do robô, bem como a utilização de sensores adicionais e técnicas mais avançadas de aprendizado profundo, visando aumentar a robustez do sistema

em aplicações reais.

Referências

ALMEIDA, Andressa Ruviaro; SANTOS CARRARA, Alcy Rodolfo dos; YUAN, Waldomiro Soares. As Vantagens de Robôs de Combate na Área de Pesquisa. *Universidade Federal do Paraná, Departamento de Engenharia Elétrica e Mecânica*, Curitiba, 2013. Citado 1 vez na página 16.

AZEVEDO, Eduardo; CONCI, Aura; LETA, Fabiana. *Computação gráfica: teoria e prática: geração de imagens*. E-book. Rio de Janeiro: Editora Alta Books, 2022. v. 2, p. 19. Acesso em: 25 mar. 2025. ISBN 9786555209860. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9786555209860/. Citado 2 vezes na página 18.

BERARDINUCCI, Francesco; URGO, Marcello. Advanced Computer Vision for Industrial Safety: Indoor Human Worker Localization Using Deep Learning. *In:* ALEXOPOULOS, Kosmas; MAKRIS, Sotiris; STAVROPOULOS, Panagiotis (ed.). *Advances in Artificial Intelligence in Manufacturing II*. Cham: Springer Nature Switzerland, 2025. p. 134–143. DOI: 10.1007/978-3-031-86489-6_15. Disponível em: https://link.springer.com/chapter/10.1007/978-3-031-86489-6_15. Citado 2 vezes nas páginas 11, 12.

CASTRO SOUZA, Renan Rubim de; PEDRON, Cristiane Drebes; SILVA, Luciano Ferreira da. Barreiras e Desafios na Implementação de Projetos de Robótica: Uma Revisão Sistemática de Literatura. *In:* UNINOVE – UNIVERSIDADE NOVE DE JULHO. 9º Encontro dos Programas de Pós-Graduação Profissionais em Administração (EMPRAD). 2023. Citado 1 vez na página 16.

CAVALCANTE, Douglas *et al.* VEÍCULOS AUTÔNOMOS -IMPACTOS PARA SOCIEDADE, ago. 2023. Citado 1 vez na página 11.

COPEL ELETRÔNICA. *Resistor 1 k 5% 1/4W*. Acesso em: 20 jan. 2026. 2026. Disponível em: https://www.copeleletronica.com.br/p-resistor-1k-5-1-4w-2873. Acesso em: 20 jan. 2026. Citado 1 vez na página 44.

EDGE IMPULSE. *Edge Impulse Documentation*. 2024. https://docs.edgeimpulse.com. Acesso em: 02 mar. 2026. Citado 2 vezes nas páginas 26, 39.

ELETROGATE. *Buzzer Passivo 5V*. 2026. Acesso em: 20 jan. 2026. Disponível em: https://www.eletrogate.com/buzzer-passivo-5v. Citado 1 vez na página 44.

ELETROGATE. *Kit Chassi 4WD Robô para Arduino*. 2026. Acesso em: 20 jan. 2026. Disponível em: https://www.eletrogate.com/kit-chassi-4wd-robo-para-arduino. Citado 1 vez nas páginas 40, 41.

ELGIN. *Pilhas Alcalinas Palito AAA*. 2026. Acesso em: 20 jan. 2026. Disponível em: https://www.amazon.com.br/Pilhas-Alcalinas-Palito-Elgin-Baterias/dp/B0754J12RW/. Citado 1 vez na página 43.

GÉRON, Aurélien. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 2. ed. Sebastopol, CA: O'Reilly Media, 2019. Citado 6 vezes nas páginas 24, 25.

GIRI STUDIO. *Arduino Bluetooth Controller (HC-05)*. Aplicativo disponível na plataforma Google Play. 2026. Disponível em: https://play.google.com/store/apps/details?id=com.giristudio.hc05.bluetooth.arduino.control&hl=pt_BR. Acesso em: 11 fev. 2026. Citado 0 vez na página 66.

GOODFELLOW, Ian *et al. Deep learning*. MIT press Cambridge, 2016. v. 1. Citado 6 vezes nas páginas 20–22, 26.

GOOGLE. *Google Imagens*. 2025. Acesso em: jul. 2025. Disponível em: https://images.google.com/. Citado 2 vezes nas páginas 19, 20, 22, 23, 29, 31–33, 43.

HORA, Breno Aires da. *Sistema de reconhecimento automático de placas veiculares utilizando visão computacional*. 2024. f. 60. Trabalho de Conclusão de Curso (Bacharelado em Engenharia da Computação) – Universidade Federal do Pará, Faculdade de Engenharia da Computação, Campus Universitário de Tucuruí, Tucuruí. Orientador: Daniel da Conceição Pinheiro. Disponível em: https://bdm.ufpa.br/jspui/handle/prefix/7573. Citado 1 vez na página 11.

ISMAEL, Khansaa Dheyaa; SAEED, Elaf A. Controlling Mobile Robot Navigation Equipped with Computer Vision Perception for Text using OCR Technique. *In:* 2024 21st International Multi-Conference on Systems, Signals Devices (SSD). 2024. p. 1–7. DOI: 10.1109/SSD61670.2024.10549002. Citado 1 vez na página 26.

KRICHEN, Moez. Convolutional Neural Networks: A Survey. *Computers*, v. 12, n. 8, 2023. ISSN 2073-431X. DOI: 10.3390/computers12080151. Disponível em: https://www.mdpi.com/2073-431X/12/8/151. Citado 1 vez na página 12.

LAN, Roy; AWOLUSI, Ibukun; CAI, Jiannan. Computer Vision for Safety Management in the Steel Industry. *AI*, v. 5, n. 3, p. 1192–1215, 2024. ISSN 2673-2688. DOI: 10.3390/ai5030058. Disponível em: https://www.mdpi.com/2673-2688/5/3/58. Citado 4 vezes nas páginas 11, 12.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. *nature*, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015. Citado 1 vez na página 22.

LI, Hanju. Computer network connection enhancement optimization algorithm based on convolutional neural network. *In:* 2021 International Conference on Networking,

Communications and Information Technology (NetCIT). 2021. p. 281–284. DOI: 10.1109/NetCIT54147.2021.00063. Citado 2 vezes na página 12.

LOU, Guangxin; SHI, Hongzhen. Face image recognition based on convolutional neural network. *China Communications*, v. 17, n. 2, p. 117–124, 2020. DOI: 10.23919/JCC.2020.02.010. Citado 1 vez na página 12.

MAKERHERO. *Placa Uno R3 + Cabo USB para Arduino*. 2026. Acesso em: 20 jan. 2026. Disponível em: https://www.makerhero.com/produto/placa-uno-r3-cabo-usb-para-arduino/. Citado 1 vez na página 41.

OLIVEIRA, P. V. S.; SANTOS, L. de F.; FERREIRA, M. P. Inteligência artificial na automação de processos industriais e seus impactos. *Revista de Economia Mackenzie*, v. 21, n. 1, p. 162–182, 2024. DOI: 10.5935/1808-2785/rem.v21n1p.162-182. Citado 1 vez na página 11.

PEDRINI, Hélio; SCHWARTZ, William R. *Análise de imagens digitais: princípios, algoritmos e aplicações*. E-book. Porto Alegre: +A Educação - Cengage Learning Brasil, 2007. p. 44. Acesso em: 25 mar. 2025. ISBN 9788522128365. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788522128365/. Citado 3 vezes nas páginas 18, 19.

RAGAB, Heba *et al.* Real-Time Object Detection and Grasping with an Autonomous Mobile Robot: A Case Study. *In:* 2024 9th International Conference on Robotics and Automation Engineering (ICRAE). 2024. p. 40–44. DOI: 10.1109/ICRAE64368.2024.10851490. Citado 1 vez na página 26.

REDAÇÃO SANAR. *Anatomia e Fisiologia Ocular: entenda tudo!* Sanarmed. 21 jul. 2019. Disponível em: https://sanarmed.com/anatomia-e-fisiologia-ocular/. Citado 0 vez na página 19.

REDDY, C. *et al.* Object detection and tracking for Personal Protective Equipment. *In:* p. 1–6. DOI: 10.1109/ICCIRT59484.2024.10921826. Citado 1 vez na página 12.

ROBOCORE. *Protoboard 400 Pontos*. 2026. Acesso em: 20 jan. 2026. Disponível em: https://www.robocore.net/protoboard/protoboard-400-pontos. Citado 1 vez nas páginas 43, 44.

ROBÓTICA EDUCACIONAL. *Módulo Bluetooth HC-05 RS232 Master Slave para Arduino*. 2026. Acesso em: 20 jan. 2026. Disponível em: https://www.roboticaeducacional.art.br/modulo-bluetooth-hc-05-rs232-master-slave-para-arduino. Citado 1 vez na página 42.

ROCHA, Pedro Alexandre Moura Cirilo. *Implicações do veículo autónomo no futuro das cidades*. 2023. Dissertação (Mestrado em Planeamento e Projecto Urbano) – Universidade do Porto. Citado 1 vez na página 11.

ROJAS-ROMERO, Mariana; SMITH CAICEDO SAENZ, Joseph Brayan. Deep learning model based on MobileNetV2 for mask detection with Raspberry Pi. *In:* 2022 IEEE Colombian Conference on Applications of Computational Intelligence (ColCACI). 2022. p. 1–6. DOI: 10.1109/ColCACI56938.2022.9905388. Citado 1 vez na página 38.

RS ROBÓTICA. *Módulo ESP32-WROVER Dev com Câmera OV2640 40 pinos*. 2026. Acesso em: 20 jan. 2026. Disponível em: https://www.rsrobotica.com.br/modulo-esp32-wrover-dev-com-camera-ov2640-40-pinos. Citado 1 vez nas páginas 41, 42.

RUSSELL, Stuart; NORVIG, Peter. *Inteligência Artificial: Uma Abordagem Moderna*. 4th. Rio de Janeiro: GEN LTC, 2022. Citado 5 vezes nas páginas 16, 17, 21, 22.

SANDLER, Mark *et al.* MobileNetV2: Inverted Residuals and Linear Bottlenecks. *In:* 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Los Alamitos, CA, USA: IEEE Computer Society, jun. 2018. p. 4510–4520. DOI: 10.1109/CVPR.2018.00474. Disponível em: https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00474. Citado 2 vezes na página 38.

SANTOS, Caio Farias Felipe dos; FIGUEIRA, Lucas Baggio. O uso de visão computacional para detecção de condução sonolenta. *In:* FACULDADE de Tecnologia de FATEC Ribeirão Preto (FATEC). Ribeirão Preto, SP - Brasil, 2024. Citado 1 vez na página 12.

SECCHI, Humberto. *Uma Introdução aos Robôs Móveis*. Edição: Cynthia Netto de Almeida e Felipe Nascimento Martins. 2012. Tradução: Cynthia Netto de Almeida e Felipe Nascimento Martins, Revisão técnica: Felipe Nascimento Martins, NER@ – Núcleo de Estudos em Robótica e Automação, IFES – Instituto Federal de Educação, Ciência e Tecnologia do Espírito Santo, Edição: Abril de 2012. Citado 1 vez na página 16.

SEPEEH, Muhamad Syazmie *et al.* Development of Autonomous Mobile Robot Based IoTs Integration for Surveillance Guard. *In:* 2024 IEEE 22nd Student Conference on Research and Development (SCOReD). 2024. p. 323–327. DOI: 10.1109/SCOReD64708.2024.10872765. Citado 1 vez na página 27.

SILVA, Alexandre; FIGUEIREDO, Douglas; DUTRA, Frederico. Aplicação da inteligência artificial na segurança do trabalho: uma revisão sistemática de literatura. *Revista Científica Multidisciplinar Núcleo do Conhecimento*, p. 116–129, jun. 2023. DOI: 10.32749/nucleodoconhecimento.com.br/tecnologia/aplicacao-da-inteligencia. Citado 1 vez na página 11.

SILVA, Alexandre Pinto da; DUTRA, Frederico Giffoni de Carvalho *et al.* Aplicação da inteligência artificial na prevenção de acidentes de trabalho: uma revisão sistemática de literatura. *Revista de Gestão e Secretariado*, Revista de Gestão e Secretariado (GeSec), v. 14, n. 8, p. 12934–12960, ago. 2023. DOI: 10.7769/gesec.v14i8.2585. Disponível em: https://ojs.revistagesec.org.br/secretariado/article/view/2585. Citado 1 vez na página 11.

SNEHALAKSHMI, S *et al.* Smart Safety Surveillance System: Personal Protective Equipment and Drowsiness Detection in Industrial Environments. *In:* p. 1–6. DOI: 10.1109/ICCCNT61001.2024.10724032. Citado 2 vezes na página 12.

SUPRIYADI, Ujang; BACHTIAR, Mochamad Mobed; WIBOWO, Iwan Kurnianto. Goalpost Detection and Recognition on Robot Soccer ERSOW Using SSD MobileNet-V2 FPNlite. *In:* 2023 International Electronics Symposium (IES). 2023. p. 342–347. DOI: 10.1109/IES59143.2023.10242560. Citado 1 vez na página 38.

TOPOLSKY, D.V. *et al.* Optimization of Functional Tasks for a Vision System of a Mobile Robot. *In:* 2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). 2019. p. 0566–0570. DOI: 10.1109/SIBIRCON48586.2019.8958356. Citado 1 vez na página 14.

TORRES, Mário César Delunardo. *Desenvolvimento e implementação da eletrônica embarcada em um dispositivo robótico móvel para inspeção: espeleorobô*. 2022. f. 68. Monografia (Graduação em Engenharia de Controle e Automação) – Universidade Federal de Ouro Preto, Escola de Minas, Ouro Preto. Citado 2 vezes na página 11.

UNSPLASH. *Unsplash*. 2025. Acesso em: jul. 2025. Disponível em: https://unsplash.com/. Citado 1 vez nas páginas 30–33.

VLADCONTROL. *Ponte H L298N com Arduino*. 2026. Acesso em: 03 fev. 2026. Disponível em: http://www.vladcontrol.com.br/arduino-basico/ponte-h-l298n/. Citado 1 vez na página 43.

WIDAYAKA, Parama Diptya *et al.* Implementation of Computer Vision for Object Alignment Mechanism in RHI-MOLA Mobile Robot as Robotic Learning Platform. *In:* 2023 International Conference on Technology, Engineering, and Computing Applications (ICTECA). 2023. p. 1–6. DOI: 10.1109/ICTECA60133.2023.10490858. Citado 1 vez na página 27.

XD4SOLUTIONS. *New Creature of Embodied AI Unitree Go2 Infinite Revolution*. 2025. Disponível em: https://xd4solutions.com.br/go2/. Acesso em: 10 fev. 2026. Citado 2 vezes nas páginas 12, 14.

XD4SOLUTIONS. *Unitree and Artificial Intelligence: How Machine Learning Makes Robots Smarter*. 2025. Disponível em: https://xd4solutions.com.br/unitree-ai-machine-learning-robots/. Acesso em: 10 fev. 2026. Citado 2 vezes nas páginas 12, 17.

YANG, Libo. Design of Robot Obstacle Recognition Location Ranging Algorithm Based on Computer Vision and Artificial Intelligence. *In:* 2023 5th International Conference on Applied Machine Learning (ICAML). 2023. p. 346–350. DOI: 10.1109/ICAML60083.2023.00071. Citado 1 vez na página 27.

ZHANG, Yongyue; ZHANG, Junhao; ZHOU, Wenhao. Research on Image Classification Improvement Based on Convolutional Neural Networks with Mixed Training. *In:* 2022 IEEE 4th International Conference on Civil Aviation Safety and Information Technology (ICCASIT). 2022. p. 7–10. DOI: 10.1109/ICCASIT55263.2022.9986643. Citado 1 vez na página 12.

Apêndices

APÊNDICE A – Firmware do ESP32 para Streaming de Vídeo

O Código A.1 apresenta o firmware completo implementado no módulo.

Listing A.1 – Firmware do ESP32 para streaming de vídeo

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

const char* ssid = "XXXXXXX";
const char* password = "YYYYYYYYY";

#define CAM_PIN_PWDN  -1
#define CAM_PIN_RESET -1
#define CAM_PIN_XCLK  21
#define CAM_PIN_SIOD  26
#define CAM_PIN_SIOC  27
#define CAM_PIN_D7    35
#define CAM_PIN_D6    34
#define CAM_PIN_D5    39
#define CAM_PIN_D4    36
#define CAM_PIN_D3    19
#define CAM_PIN_D2    18
#define CAM_PIN_D1    5
#define CAM_PIN_D0    4
#define CAM_PIN_VSYNC 25
#define CAM_PIN_HREF  23
#define CAM_PIN_PCLK  22

#include "esp_http_server.h"
static esp_err_t stream_handler(httpd_req_t *req);

httpd_handle_t startCameraServer() {
  httpd_config_t config = HTTPD_DEFAULT_CONFIG();
  httpd_handle_t server = NULL;

  if (httpd_start(&server, &config) == ESP_OK) {
    httpd_uri_t uri = {
      .uri = "/",
      .method = HTTP_GET,
      .handler = stream_handler,
      .user_ctx = NULL
    };
    httpd_register_uri_handler(server, &uri);
  }
  return server;
}

static const char* STREAM_CONTENT_TYPE =
  "multipart/x-mixed-replace;boundary=frame";
```

```
45        static const char* STREAM_BOUNDARY = "\r\n--frame\r\n";
46        static const char* STREAM_PART =
47          "Content-Type: image/jpeg\r\nContent-Length: %u\r\n\r\n";
48
49        static esp_err_t stream_handler(httpd_req_t *req) {
50          camera_fb_t *fb = NULL;
51          char buf[64];
52
53          httpd_resp_set_type(req, STREAM_CONTENT_TYPE);
54
55          while (true) {
56            fb = esp_camera_fb_get();
57            if (!fb) return ESP_FAIL;
58
59            httpd_resp_send_chunk(req, STREAM_BOUNDARY,
60                                  strlen(STREAM_BOUNDARY));
61            sprintf(buf, STREAM_PART, fb->len);
62            httpd_resp_send_chunk(req, buf, strlen(buf));
63            httpd_resp_send_chunk(req, (const char *)fb->buf, fb->len);
64
65            esp_camera_fb_return(fb);
66          }
67          return ESP_OK;
68        }
69
70        void setup() {
71          Serial.begin(115200);
72
73          camera_config_t config;
74          config.ledc_channel = LEDC_CHANNEL_0;
75          config.ledc_timer = LEDC_TIMER_0;
76          config.pin_d0 = CAM_PIN_D0;
77          config.pin_d1 = CAM_PIN_D1;
78          config.pin_d2 = CAM_PIN_D2;
79          config.pin_d3 = CAM_PIN_D3;
80          config.pin_d4 = CAM_PIN_D4;
81          config.pin_d5 = CAM_PIN_D5;
82          config.pin_d6 = CAM_PIN_D6;
83          config.pin_d7 = CAM_PIN_D7;
84          config.pin_xclk = CAM_PIN_XCLK;
85          config.pin_pclk = CAM_PIN_PCLK;
86          config.pin_vsync = CAM_PIN_VSYNC;
87          config.pin_href = CAM_PIN_HREF;
88          config.pin_sccb_sda = CAM_PIN_SIOD;
89          config.pin_sccb_scl = CAM_PIN_SIOC;
90          config.pin_pwdn = CAM_PIN_PWDN;
91          config.pin_reset = CAM_PIN_RESET;
92          config.xclk_freq_hz = 20000000;
93          config.pixel_format = PIXFORMAT_JPEG;
94          config.frame_size = FRAMESIZE_QVGA;
95          config.jpeg_quality = 12;
96          config.fb_count = 2;
97
98          esp_camera_init(&config);
99
100         WiFi.begin(ssid, password);
101         while (WiFi.status() != WL_CONNECTED) {
```

```
102             delay(500);
103           }
104
105           startCameraServer();
106         }
107
108         void loop() {}
109
```

APÊNDICE B – Firmware do Arduino para Controle do Robô via Bluetooth

O código-fonte completo do firmware responsável pelo controle do robô via comunicação Bluetooth encontra-se apresentado neste apêndice.

Listing B.1 – Firmware do Arduino para controle do robô via Bluetooth

```
#include <SoftwareSerial.h>

// Bluetooth
SoftwareSerial bt(3, 4); // RX, TX

// Ponte H - motores
#define IN1 10   // esquerda frente
#define IN2 11   // esquerda r
#define IN3 12   // direita frente
#define IN4 13   // direita r

// Buzzer
#define BUZZER 2

void setup() {
  Serial.begin(9600);
  bt.begin(9600);

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(BUZZER, OUTPUT);

  parar();
}

void loop() {
  if (bt.available()) {
    char cmd = bt.read();
    Serial.println(cmd);

    switch (cmd) {
      case 'F': frente(); break;
      case 'B': re(); break;
      case 'L': esquerda(); break;
      case 'R': direita(); break;
      case 'S': parar(); break;
      case 'Y': buzzer(); break;
    }
  }
}

```

```
44      void frente() {
45        digitalWrite(IN1, HIGH);
46        digitalWrite(IN2, LOW);
47        digitalWrite(IN3, HIGH);
48        digitalWrite(IN4, LOW);
49      }
50
51      void re() {
52        digitalWrite(IN1, LOW);
53        digitalWrite(IN2, HIGH);
54        digitalWrite(IN3, LOW);
55        digitalWrite(IN4, HIGH);
56      }
57
58      void esquerda() {
59        digitalWrite(IN1, LOW);
60        digitalWrite(IN2, HIGH);
61        digitalWrite(IN3, HIGH);
62        digitalWrite(IN4, LOW);
63      }
64
65      void direita() {
66        digitalWrite(IN1, HIGH);
67        digitalWrite(IN2, LOW);
68        digitalWrite(IN3, LOW);
69        digitalWrite(IN4, HIGH);
70      }
71
72      void parar() {
73        digitalWrite(IN1, LOW);
74        digitalWrite(IN2, LOW);
75        digitalWrite(IN3, LOW);
76        digitalWrite(IN4, LOW);
77      }
78
79      void buzzer() {
80        digitalWrite(BUZZER, HIGH);
81        delay(200);
82        digitalWrite(BUZZER, LOW);
83      }
84
```

APÊNDICE C – Uso de Ferramentas de Inteligência Artificial

Tipo de solicitação: Revisão de texto, tradução e auxílio na formatação em LaTeX.

Plataforma utilizada: ChatGPT (OpenAI).

Data e horário aproximados da utilização: Julho de 2025 a Março de 2026, em horários variados.

Comandos (prompts) utilizados: Foram solicitadas revisões dos textos apresentados, com o objetivo de identificar e corrigir eventuais erros de ortografia, gramática e pontuação. Também foi solicitada a tradução do resumo para o inglês em formato LaTeX, bem como orientações para ajustes na formatação e organização das figuras no documento.

Declaração de responsabilidade: Os autores são os únicos responsáveis pelo conteúdo final deste trabalho. Todo o conteúdo gerado com o auxílio de Inteligência Artificial foi criticamente revisado, validado e adaptado pelos autores antes de sua inclusão neste documento.