INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA - CAMPUS CAMPINA GRANDE
COORDENAÇÃO DO CURSO SUPERIOR BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

LUCAS BIVAR FONSÊCA TAVARES
LUÍS HENRIQUE LIMA SANTOS

DESENVOLVIMENTO DE UM SISTEMA EMBARCADO PARA CLASSIFICAÇÃO AUTOMATIZADA DE PRODUTOS USANDO VISÃO COMPUTACIONAL

Campina Grande - PB

2025

Lucas Bivar Fonsêca Tavares
Luís Henrique Lima Santos

Desenvolvimento de um Sistema Embarcado para Classificação Automatizada de Produtos Usando Visão Computacional

Monografia apresentada ao Curso de Bacharelado em Engenharia de Computação, do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – Campus Campina Grande, em cumprimento às exigências parciais para a obtenção do título de Bacharel em Engenharia de Computação

Instituto Federal da Paraíba

Orientador: Prof. Dr. Moacy Pereira da Silva

Campina Grande - PB
2025

Catalogação na fonte:
Ficha catalográfica elaborada por Gustavo César Nogueira da Costa - CRB 15/479

T233d Tavares, Lucas Bivar Fonsêca
Desenvolvimento de um sistema embarcado para classificação automatizada de produtos usando visão computacional / Lucas Bivar Fonsêca Tavares, Luís Henrique Lima Santos. - Campina Grande, 2025.
63f. : il.

Trabalho de Conclusão de Curso (Curso Superior de Engenharia de Computação) - Instituto Federal da Paraíba, 2025.
Orientador: Prof. Dr. Moacy Pereira da Silva

1. Engenharia de computação 2. Indústria 4.0 - visão computacional 3. Desenvolvimento de sistemas - sistemas embarcados 4. Inteligência artificial I. Santos, Luís Henrique Lima II. Silva, Moacy Pereira da III.Título.

CDU 004.8

Lucas Bivar Fonsêca Tavares
Luís Henrique Lima Santos

Desenvolvimento de um Sistema Embarcado para Classificação Automatizada de Produtos Usando Visão Computacional

Monografia apresentada ao Curso de Bacharelado em Engenharia de Computação, do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – Campus Campina Grande, em cumprimento às exigências parciais para a obtenção do título de Bacharel em Engenharia de Computação

Trabalho aprovado. Campina Grande - PB, 26 de Fevereiro de 2025:

Prof. Dr. Moacy Pereira da Silva
Orientador

Prof. Dr. Igor Barbosa da Costa
Membro da Banca

Prof. Dr. Fagner De Araujo Pereira
Membro da Banca

Campina Grande - PB
26 de Fevereiro de 2025

Dedicamos este estudo a Deus, cuja presença nos guiou e fortaleceu em cada etapa da pesquisa, as nossas famílias, professores e amigos, sobretudo pelo apoio incondicional e palavras de encorajamento durante nossa árdua jornada acadêmica, bem como a todos que empreendem esforços para que a tecnologia seja instrumento para o progresso da humanidade e jamais um fim em si mesma.

Agradecimentos

Por Lucas Bivar

Primeiramente, agradeço a Deus, por me conceder saúde, força e sabedoria ao longo desta jornada acadêmica de 5 anos no Bacharelado em Engenharia de Computação. Sem Sua presença em minha vida, a graduação e este trabalho não teria sido possível.

Agradeço profundamente à minha família, Maria de Fátima, Francisco Bivar, Kaio Victor, e aos meus avós, que sempre estiveram ao meu lado, oferecendo apoio e incentivo durante todos esses anos. Vocês são minha base e minha inspiração.

À minha namorada, Maria Fernanda, que esteve presente na etapa final deste percurso, oferecendo motivação nos momentos mais desafiadores. Sua presença foi essencial para que eu pudesse chegar até aqui.

Aos amigos que fiz ao longo desta jornada em outra cidade, Antônio Farias, Arlan Santos, Diogo da Silva Santos, João Victor Negreiros, Jonathan Pontes, Jorge Castro, Luiz Eduardo, Luiz Henrique Lima, Luiz Medeiros Neto, Lucas Alves, Lohan Oliveira, Pedro Macêdo, Rafael Guimarães, e Rennyson Cavalcante, agradeço pela amizade sincera e pelas experiências compartilhadas. Vocês tornaram essa caminhada mais leve e divertida, e cada um de vocês tem um lugar especial em meu coração.

Ao meu orientador, Professor Moacy Pereira, agradeço pela orientação, paciência e apoio durante o desenvolvimento deste trabalho. Sua expertise e conselhos foram fundamentais para a conclusão deste projeto. Além disso, gostaria de agradecer pelas nossas conversas sobre filosofia, religião e literatura que foram essenciais para minha formação, ampliando minha visão de mundo e enriquecendo minha trajetória acadêmica.

À Professora Ianna Sodré, sou imensamente grato pelas inúmeras oportunidades em projetos de pesquisa e extensão. Sua confiança em mim desde o primeiro período foi um grande incentivo e me proporcionou um crescimento acadêmico e pessoal inestimável.

Por fim, agradeço a todos os meus professores do IFPB - Campus Campina Grande, que com dedicação e conhecimento, me guiaram e inspiraram ao longo do curso. Suas lições serão levadas comigo para sempre.

Por Luís Henrique

Agradeço a Deus, por Sua benção durante meus cinco anos de graduação e por ter permitido conhecer pessoas incríveis durante minha trajetória acadêmica.

Aos meus pais Lucirino Fernandes e Veridiana Borges pelo seu apoio, educação e valores. Ao meu irmão Matheus Felipe por seu companheirismo em diversos momentos e a minha irmã Maria Eduarda por suas conversas descontraídas, tornando minha graduação um pouco mais leve e divertida.

À minha avó Lúcia e ao meu avô José Quirino (in memoriam), à minha avó Maria da Paz e ao meu avô Celso Rodrigues, à minha tia Dayse e seu esposo Vandertônio, à Luiz Fernandes e Diana Paula por seu apoio e acolhimento durante minha jornada acadêmica.

Ao meu orientador, Professor Dr. Moacy Pereira, pelo apoio e conhecimento em suas aulas na disciplina de Sinais e Sistemas, Processamento Digital de Sinais, Técnicas de Prototipagem e em outros campos do conhecimento e da vida. Jamais esquecerei nossas conversas acerca dos diálogos entre tecnologia, filosofia e literatura.

Aos professores Emanuel Dantas Filho e Ianna Sodré, pelas oportunidades ofertadas, as quais que me permitiram aplicar, aprofundar e aprender ainda a respeito de diversas áreas do conhecimento e a respeito do Desenvolvimento de *Software* e Inteligência Artificial, área na qual atuo.

A todo o corpo de docentes do Instituto Federal da Paraíba - Campus Campina Grande, por todo aprendizado e incentivo durante a graduação. Por meio deles, pude adquirir o conhecimento que são o alicerce para minha profissão.

Por fim, aos meus colegas Lucas Bivar, Antônio Farias, João Victor Negreiros, Luiz Medeiros, Pedro Macedo e Rennyson Cavalcante e tantos outros, que tive a alegria de compartilhar momentos. Todos vocês foram fonte de inspiração pessoal e profissional.

"A cada aumento de conhecimento,
assim como a cada invenção de ferramenta,
o trabalho humano é diminuído".
(Charles Babbage)

Resumo

No contexto da automação de processos, a verificação manual dos resultados de uma produção apresenta desafios significativos, como a elevada demanda por esforço físico da força de trabalho e o tempo considerável necessário para realizar a classificação e separação de mercadorias. Em um cenário específico da produção, a separação manual de produtos saudáveis e não saudáveis é uma das dificuldades que pode impactar negativamente a qualidade do produto ao destinatário final. Assim, o presente estudo parte da seguinte inquietude: como um sistema baseado em visão computacional pode automatizar e otimizar a classificação de produtos em processos industriais? Compreende-se que a visão computacional pode colaborar em termos de eficiência no retorno do investimento realizado. Desse modo, o presente estudo propõe um sistema automatizado de classificação e separação de produtos utilizando sistemas embarcados, técnicas de visão computacional e inteligência artificial. Para tanto, adotou-se uma pesquisa aplicada, de caráter exploratório e experimental a partir da criação de um protótipo. O estudo visa contribuir com solução tecnológica para otimizar processos, reduzir custos operacionais e aumentar a eficiência na gestão da qualidade de produtos. Dos resultados obtidos, verificou-se que o protótipo se mostrou eficiente em termos de classificação e separação do produto utilizados durante as simulações, com uma taxa de acerto de 82% para produtos saudáveis e de 100% para produtos não saudáveis.

Palavras-chave: Classificação de Produtos; Indústria 4.0; Visão Computacional; YOLO.

Abstract

In the context of process automation, manual verification of production results presents significant challenges, such as the high demand for physical effort from the workforce and the considerable time needed to sort and separate goods. In a specific production scenario, the manual separation of healthy and unhealthy products is one of the difficulties that can negatively impact the quality of the product for the final recipient. Therefore, this study is based on the following concern: how can a system based on computer vision automate and optimize the classification of products in industrial processes? It is understood that computer vision can collaborate in terms of efficiency and return on investment. This study proposes an automated system for sorting and separating products using embedded systems, computer vision techniques and artificial intelligence. To this end, an applied, exploratory and experimental study was adopted, based on the creation of a prototype. The study aims to provide a technological solution to optimize processes, reduce operating costs and increase efficiency in product quality management. From the results obtained, it was found that the prototype proved to be efficient in terms of product classification and separation used during the simulations, with a hit rate of 82% for healthy products and 100% for unhealthy products.

Keywords: Computer Vision; Industry 4.0; Product Classification; YOLO.

Lista de ilustrações

Figura 1 – Pinagem do ESP-32 18
Figura 2 – Funcionamento do HTTP 21
Figura 3 – Funcionamento do MQTT 22
Figura 4 – Funcionamento do *WebSocket* 23
Figura 5 – Arquitetura de uma RNA 29
Figura 6 – Campos de Estudo da Visão Computacional 30
Figura 7 – Exemplo do Uso de Caixas Delimitadoras 34
Figura 8 – *Dashboard* Contabilizador 39
Figura 9 – Visão Superior do Protótipo 44
Figura 10 – Visão Lateral do Protótipo 45
Figura 11 – Visão Frontal do Protótipo 46
Figura 12 – Esquemático Elétrico do Sistema 48
Figura 13 – Arquitetura do Projeto 48
Figura 14 – Matriz de Confusão 52
Figura 15 – Resultados de Perda e Treino 52
Figura 16 – Curva F1-*Score* 53
Figura 17 – Curva de Precisão 53
Figura 18 – Curva de Precisão-Revocação 54
Figura 19 – Curva de Revocação 55

Lista de abreviaturas e siglas

AP	Average Precision
API	Application Programming Interface
APP	Aplicação de Internet
COCO	Common Objects in Context
CSS	Cascading Style Sheets
CUDA	Compute Unified Device Architecture
CNN	Convolutional Neural Networks
CV	Computer Vision
DC	Direct Current
F1	Média harmônica
FN	Falsos Negativos
FP	Falsos Positivos
GPU	General Processing Unit
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IA	Inteligência Artificial
IP	Internet Protocol
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
mAP	Mean Average Precision
MMX	MultiMedia eXtension
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
NoSQL	Not Only SQL

OSI	Open Systems Interconnection
PWM	Pulse-Width Modulation
REST	Representational State Transfer
RNA	Rede Neural Artificial
RestNet	Residual Neural Network
SQL	Structured Query Language
SSE	Server-Sent Events
USB	Universal Serial Bus
URL	Uniform Resource Locator
V	Volts
VN	Verdadeiros Negativos
VP	Verdadeiros Positivos
XML	Extensible Markup Language
YOLO	You Only Look Once
YAML	YAML Ain't Markup Language

Sumário

1 INTRODUÇÃO 14

2 REFERENCIAL TEÓRICO 17
2.1 Sistemas Embarcados 17
2.2 Protocolos de Comunicação 20
2.3 Desenvolvimento de *Software* 22
2.4 Aplicações de Internet 24
2.5 Inteligência Artificial 27
2.5.1 Aprendizado de Máquina 28
2.6 *Overfitting* e *Underfitting* 32
2.7 Processamento de Dados 33
2.8 Avaliação de Desempenho de Modelo 34
2.9 Avaliação do Referencial Teórico 35

3 METODOLOGIA 36
3.1 Revisão bibliográfica 36
3.2 Requisitos do protótipo 37
3.3 Desenvolvimento da Aplicação de Internet 38
3.4 Coleta de Dados e Pré-processamento 39
3.5 Treinamento do Modelo 41
3.6 Teste em ambiente simulado 43
3.7 Elaboração e Construção do Protótipo do Sistema 43
3.8 Arquitetura e Integração do Sistema 47

4 RESULTADOS 50
4.1 Da avaliação de desempenho do modelo 51
4.2 Do protótipo de sistema embarcado 55
4.3 Da classificação e separação 56

5 CONSIDERAÇÕES FINAIS 57
5.1 Avaliação do sistema de automação 57
5.2 Desafios Enfrentados no Desenvolvimento do Protótipo 58
5.3 Lições Aprendidas 59
5.4 Trabalhos Futuros 59

REFERÊNCIAS 60

1 Introdução

No contexto da produção e separação de mercadorias, em que a eficiência e a qualidade são cruciais para o retorno do investimento, os empresários enfrentam o desafio de garantir que os melhores bens cheguem ao consumidor final. A tarefa de identificar e separar produtos adequados dos não adequados, quando realizada de forma manual, está sujeita a erros humanos, método que pode ser ineficiente e nada escalável. Desse modo, o processo pode resultar em perdas significativas, afetando a lucratividade do empresário e a satisfação do cliente.

Por exemplo, em uma fazenda que, durante a colheita precisa processar toneladas de batatas em um curto período, a equipe de trabalhadores, por mais dedicada que seja, não consegue manter um padrão uniforme de qualidade devido à fadiga e à subjetividade na avaliação visual. Como resultado, batatas danificadas ou de qualidade inferior acabam sendo embaladas junto com as saudáveis, gerando reclamações de clientes e aumento nos custos em termos de devolução e substituição (CATTO, 2024).

Assim, a crescente demanda por alimentos de qualidade e a necessidade de otimizar processos industriais têm impulsionado a busca por soluções inovadoras (SOUZA; ADANIYA, 2024). Nesse contexto, o desenvolvimento de ferramentas que automatizam e aprimoram a precisão do processo se torna essencial.

Do exposto, a proposta deste estudo foi explorar a aplicação de tecnologias, como inteligência artificial, sistemas embarcados e visão computacional, para criar um protótipo capaz de identificar e separar produtos em dois grupos, dado um padrão de qualidade. Na investigação, utilizou-se como caso de estudo para simulação imagens de batatas, analisadas pelo protótipo que tem como infraestrutura uma esteira de produção.

O sistema proposto é composto por várias camadas tecnológicas integradas para garantir que o processo de classificação seja automatizado e eficiente. Primeiramente, foi desenvolvido um modelo de visão computacional para identificar características visuais de mercadorias em conformidade e fora de conformidade. Para aplicação do caso de uso, adotou-se como parâmetros cor, textura e sinais de degradação em batatas.

A esteira transportadora, equipada com sensor infravermelho e câmera, foi pensada para detectar a presença de objetos. O controle da esteira, dos sensores e atuadores foi realizado por um microcontrolador. Além disso, foi desenvolvido um *dashboard* dinâmico e responsivo, que exibe os dados acerca da quantidade de itens considerados saudáveis e não saudáveis, bem como a data e hora da última classificação.

Por último, o sistema inclui um mecanismo de separação automatizado, acionado por um servo motor, que direciona os produtos para recipientes específicos com base na

classificação recebida.

Ao integrar as tecnologias supramencionadas, espera-se não apenas aumentar a eficiência do processo de classificação e separação, mas também garantir um padrão de qualidade previamente estabelecido, tecnologia que pode ser utilizada por produtores rurais.

A justificativa acadêmica do estudo reside na possibilidade de colaborar com estudos que promovam modernização, otimização e automação de processos de seleção de produtos.

No campo socioeconômico, a relevância do estudo decorre da capacidade de reduzir perdas, aumentar a eficiência operacional e garantir a entrega de produtos de alta qualidade ao consumidor final e destinação social dos produtos que não atenderam a expectativa de um padrão previamente estabelecido de qualidade. Ainda, no campo social, a automação agrícola viabiliza reestruturação das funções das relações de trabalho, em que o labor humano pode ser aproveitado noutras tarefas, promovendo o uso mais eficiente dos recursos humanos e tecnológicos.

Diante dessa perspectiva, definiu-se o objetivo geral do estudo, o qual buscou-se: desenvolver um sistema baseado em visão computacional para automatizar a classificação e separação de produtos. Para tanto, foram estabelecidos os seguintes objetivos específicos:

a) Contextualizar o conhecimento do tema a partir da exposição do referencial teórico;

b) Estruturar um sistema automatizado de classificação e separação de produtos;

c) Desenvolver um *dashboard* com atualização em tempo real para monitoramento do processo de classificação;

d) Avaliar a eficácia do sistema em um ambiente simulado, medindo a eficiência e a revocação do processo de classificação e separação.

O estudo está dividido em 5 (cinco) capítulos. O primeiro corresponde a presente introdução e aduz aspectos de justificativa e objetivos do estudo.

O segundo capítulo apresenta a contextualização do conhecimento, que compreende estudos relacionados aos temas: sistemas embarcados; protocolos de comunicação; desenvolvimento de *software*; inteligência artificial; visão computacional e processamento de dados.

O terceiro capítulo apresenta o método utilizado para a realização do estudo. Nesse sentido, adotou-se como conjunto metodológico: pesquisa aplicada, de caráter exploratório e experimental. Para tanto, utilizou-se um banco de dados de referência contendo imagens de batatas para simulação de teste, associando-o ao protótipo de classificação e separação baseado em inteligência artificial.

O protótipo é composto por uma esteira automatizada, sensores, uma câmera e um modelo de classificação binária. Os dados são processados em tempo real e integrados a uma interface gráfica amigável que exibe as informações de classificação de forma dinâmica.

Ainda, o terceiro capítulo apresenta informações acerca de requisitos, tratamento de dados, desenvolvimento da aplicação de internet e testes em ambiente simulado como apresentação de informações a partir de um *dashboard*.

Por sua vez, o quarto capítulo apresenta os resultados encontrados após simulações realizadas durante o período de abril de 2024 até dezembro de 2024.

Por fim, em sede de considerações finais, apresentam-se aspectos relacionados à avaliação do sistema desenvolvido, desafios enfrentados, lições aprendidas e perspectivas para trabalhos futuros.

2 Referencial Teórico

Conforme discutido anteriormente, a automação de processos tem se mostrado uma área de grande potencial, oferecendo soluções inovadoras para desafios tradicionais do setor. No contexto da classificação e separação de produtos agrícolas saudáveis e não saudáveis, a aplicação de sistemas de visão computacional, inteligência artificial, sistemas hospedados na internet e sistemas embarcados surgem como uma abordagem promissora. Assim, faz-se necessário compreender conceitos relevantes, relacionados ao objeto desta pesquisa.

2.1 Sistemas Embarcados

Os sistemas embarcados são responsáveis por executar tarefas específicas e são indispensáveis em dispositivos eletrônicos de diversas áreas. Um sistema será considerado quando "[...] este é dedicado a uma única tarefa e interage continuamente com o ambiente a sua volta por meio de sensores e atuadores" (CHASE; ALMEIDA, 2007, p. 3).

Apesar de apresentarem características em comum com os sistemas computacionais, não possuem a mesma uniformidade. Cada aplicação apresenta requisitos diferentes de desempenho, consumo de energia e área ocupada. Isso pode acarretar combinação diferente de módulos de *hardware* e *software* para atender a esses requisitos (WOLF, 2001). Eles são encontrados em uma variedade de dispositivos, desde eletrodomésticos e automóveis até equipamentos médicos e sistemas de controle industrial. A arquitetura de um sistema embarcado geralmente inclui uma memória, interfaces de entrada/saída e *software* dedicado, conhecido como *firmware* (CUNHA, 2007).

Por fim, todos esses componentes são controlados por um microcontrolador. A integração desses componentes permite que o sistema embarcado execute suas funções de maneira autônoma e eficiente.

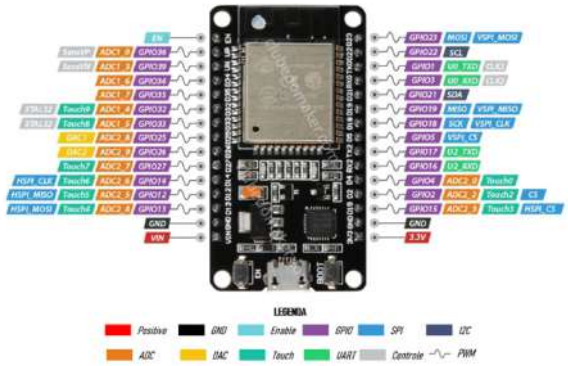
Microcontroladores são circuitos constituídos de um núcleo de processamento, memória voláteis e não voláteis e diversos periféricos de entrada e saída de dados. Um microcontrolador pode ser entendido como um computador compacto. São de extrema importância para a indústria, por serem utilizados nos sistemas embarcados e são acessíveis economicamente (CARDOSO, 2020).

Nesse campo, o microcontrolador ESP-32, desenvolvido pela *Espressif Systems*, é um microcontrolador que se destaca por sua alta performance e eficiência energética. Ele oferece conectividade *WiFi* compatível com o padrão 802.11 b/g/n, além de suporte a *Bluetooth*. O *chip* é equipado com 4MB de memória *flash*, uma Unidade Central de

Processamento com dois núcleos, 448 Kbytes de memória ROM e 520 Kbytes de memória RAM. Possui 36 pinos de entrada/saída, dos quais 18 podem funcionar como conversores analógico-digitais de 12 bits e opera com uma tensão de 3.3V (ESPRESSIF, 2022).

Essa combinação de recursos torna o ESP-32 ideal para uma variedade de aplicações, desde dispositivos de automação residencial até sistemas de monitoramento industrial. Somando-se a isso, há uma ampla variedade de conjunto de código disponibilizado gratuitamente para serem reutilizados por outros programadores (ARDUINO, 2020) e recursos que simplificam o desenvolvimento de projetos complexos. A capacidade de processamento e a conectividade do ESP-32 o tornam uma escolha popular para projetos que exigem comunicação sem fio e controle de *hardware*.

Figura 1 – Pinagem do ESP-32



Fonte: (MAKER, 2024)

O funcionamento eficiente de microcontroladores como o ESP-32 depende diretamente do *firmware*, tipo de *software* que fornece controle de baixo nível para o *hardware* de um dispositivo. Ele é embutido em dispositivos eletrônicos para gerenciar suas funções básicas e é essencial para o funcionamento de qualquer dispositivo que dependa de microcontroladores e/ou processadores (LEVY; REDDY, 1985). Para o desenvolvimento em microcontroladores como o ESP-32, são fornecidas diversas plataformas, por exemplo o Arduino, o ESP-IDF, o MicroPython, entre outras. Para o presente estudo, adotou-se a plataforma Arduino.

O desenvolvimento de *firmware* para o ESP-32 utilizando Arduino simplifica a programação, permitindo que desenvolvedores se concentrem na lógica de aplicação em vez de detalhes de *hardware*. O Arduino utiliza a linguagem de programação C++, que oferece recursos como encapsulamento, herança e polimorfismo, fundamentais para a criação de *software* modular e reutilizável, especialmente em sistemas embarcados (TRAUSS, 2019). Adicionalmente, o Arduino fornece coleção de códigos que facilitam o controle de *hardware* e a comunicação de rede, acelerando o desenvolvimento de protótipos e produtos finais (MARGOLIS, 2020). Sua simplicidade e facilidade de uso tornam a plataforma acessível

tanto para iniciantes quanto para desenvolvedores experientes. O uso do Arduino no ESP-32 permite que desenvolvedores aproveitem a potência e a flexibilidade do microcontrolador, enquanto se beneficiam da simplicidade e da robustez.

Para que os sistemas desenvolvidos com o ESP-32 sejam plenamente funcionais, a integração eficiente com sensores e atuadores torna-se indispensável. Sensores e atuadores são componentes essenciais que permitem a interação com o ambiente externo e a execução de funções designadas de forma eficiente e precisa. Enquanto os sensores são responsáveis por capturar os dados do ambiente, os atuadores realizam um comando recebido de outro dispositivo, com base em uma entrada ou critério a ser seguido (BRUGNARI; MAESTRELLI, 2010). Assim, a escolha e a integração adequadas desses componentes são cruciais para o desempenho e a confiabilidade do sistema.

Dentro desse contexto, um dos atuadores relevantes para o presente estudo é a Ponte H, utilizada para controlar a direção de rotação de motores de corrente contínua (DC). A Ponte H consiste em quatro interruptores dispostos em forma de "H", permitindo que a corrente flua em direções opostas através do motor, possibilitando a inversão de sua rotação (MOHAN, 2003). Dessa forma, esse circuito é essencial para aplicações que requerem controle de movimento, como robôs móveis e sistemas de automação. Além disso, a Ponte H também pode ser utilizada para controlar a velocidade do motor através de modulação por largura de pulso (PWM), oferecendo um controle preciso sobre o desempenho do motor.

Adicionalmente, outro atuador relevante é o Servo Motor, que permite controle preciso de posição angular ou linear, velocidade e aceleração. Composto por um motor acoplado a um sensor de posição e um controlador (BOLTON, 2015), o servo motor recebe um sinal de controle, geralmente um pulso PWM, que determina a posição desejada do eixo. Sua capacidade de manter uma posição estável e de responder rapidamente a comandos de controle torna os servos ideais para aplicações que requerem movimentos precisos, como braços robóticos, sistemas de direção em veículos autônomos e mecanismos de controle de voo em drones.

Além desses atuadores, se destaca o Motor DC devido a sua "[...] velocidade e aplicações de controle de posição por causa da facilidade no controle do torque e excelente desempenho" (KIM, 2017, p. 1). Controlado por tensão contínua, o motor DC geralmente necessita de sensores para controle preciso de posição, sendo comum em sistemas automotivos, impressoras, eletrodomésticos e robôs móveis.

Por fim, complementando o conjunto de sensores e atuadores, o sensor de proximidade infravermelho utiliza a emissão e recepção de luz infravermelha para detectar a presença ou a distância de objetos próximos. Ele funciona emitindo um feixe de luz infravermelha e medindo a quantidade de luz refletida de volta pelo objeto (KEMPER, 2018). A capacidade de ajustar a sensibilidade e a distância de detecção torna o sensor

de proximidade infravermelho versátil para diversas aplicações, como robótica, segurança e automação industrial. Comumente utilizado em robôs móveis para evitar colisões, em sistemas de segurança para detectar intrusões e em linhas de produção para detectar a presença de objetos, esse sensor é uma ferramenta indispensável para a operação precisa e eficiente dos sistemas embarcados.

Concluída a abordagem sobre os sistemas embarcados e suas contribuições para a automação do processo, passa-se agora à análise dos protocolos de comunicação, fundamentais para garantir a integração eficiente entre os diferentes componentes do sistema.

2.2 Protocolos de Comunicação

Os protocolos de comunicação são responsáveis por estabelecer conexões, gerenciar a troca de dados e garantir a integridade e a segurança das informações transmitidas. Eles são o alicerce das redes de computadores modernas, permitindo a interoperabilidade entre dispositivos heterogêneos e garantindo a segurança e integridade dos dados transmitidos (TANENBAUM; BOS, 2019).

Os protocolos podem operar em diferentes camadas, a depender do modelo de interconexão de sistemas abertos, podendo ser o modelo *Open Systems Interconnection* (OSI) ou o *Transmission Control Protocol/Internet Protocol* (TCP/IP).

O modelo OSI conta com sete camadas, a começar pela camada física, enlace de dados, rede, transporte, sessão, apresentação e aplicação. No modelo de referência OSI, os protocolos podem atuar desde a camada física, que trata da transmissão de bits, até a camada de aplicação, que lida com a interação direta com o usuário (FOROUZAN, 2017). Desse modo, a escolha do protocolo adequado depende das necessidades específicas da aplicação, como largura de banda, latência, segurança e confiabilidade.

Por outro lado o modelo TCP/IP conta com quatro camadas, a começar pela camada de host/redes, inter-redes, transporte e aplicação (KESSLER, 2004) e ficou conhecido dessa maneira devido ao seus dois principais protocolos, o TCP e o IP.

Assim sendo, a camada que mantém a rede de computadores conectadas é a camada de inter-redes. Seu principal protocolo é o IP, responsável por manter a interligação entre os computadores conectados na rede. Para a comunicação faz-se necessário identificar na rede qual a origem e o destino de um pacote de dados. Por essa razão, há o endereço IP, que é o identificador único de uma máquina na rede. Sua versão mais atual é a IPv6, o qual é composto por 128 (cento e vinte e oito) campos de endereço.

Por sua vez, a camada de transporte é a responsável por codificar e transportar o dados. Seu principal protocolo, o TCP, responsável por entregar uma sequência de *bytes* fim a fim de maneira confiável, adaptar-se às diferentes propriedades de uma rede e ser

tolerante a falhas (TANENBAUM, 2011, p. 566).

Nesse contexto, a internet se destaca como um exemplo prático de uma rede global que integra diferentes dispositivos e tecnologias por meio de protocolos comuns. Para Andrew S. Tanenbaum (2011, p. 3), a internet é "[...] um vasto conjunto de redes diferentes que utilizam certos protocolos comuns e fornecem determinados serviços comuns.". Ela fornece um esquema de endereçamento que permite o roteamento eficiente de pacotes de dados entre dispositivos em uma rede global (GONT; CHOWN; LINKOVA, 2019).

A internet permite a comunicação entre dispositivos em todo o mundo, suportando uma ampla gama de serviços e aplicações, desde navegação na rede, transmissão de vídeo e comunicação em tempo real. Além disso, sua arquitetura é baseada em um modelo de camadas, onde cada camada desempenha um papel específico no processo de comunicação, promovendo a interoperabilidade entre diferentes tecnologias e plataformas.

Na estrutura supramencionada, o *Hypertext Transfer Protocol* (HTTP) se apresenta como o protocolo de comunicação mais utilizado na Internet, sendo essencial para a transferência de dados entre servidores e clientes, atuando na camada de aplicação (TANENBAUM; BOS, 2019).

Operando em um modelo de requisição/resposta, ele permite que clientes enviem solicitações a servidores e recebam dados em retorno, funcionando como a base da *World Wide Web* [1]. Por meio do HTTP, é possível navegar e interagir com páginas web, consolidando-o como um pilar da comunicação na rede de computadores. Contudo, apesar de sua importância, o HTTP apresenta limitações, especialmente em aplicações que exigem comunicação em tempo real devido à sua natureza síncrona. Como destacado por Fielding e Reschke (2014), esse protocolo é fundamental para a transferência de hipertexto, mas possui restrições que precisam ser consideradas.

Figura 2 – Funcionamento do HTTP

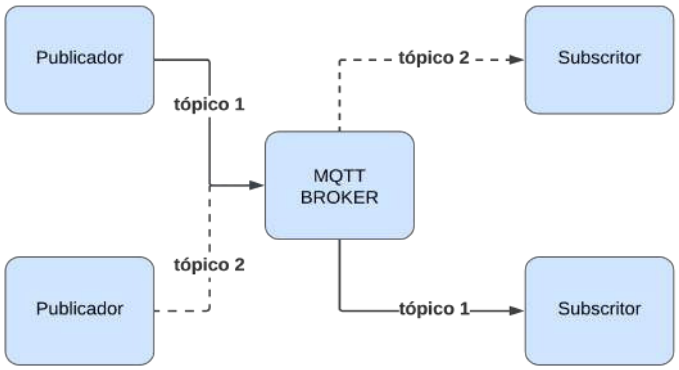
Fonte: (GOURLEY; TOTTY, 2002. 4 p., p. 4).

Embora o HTTP seja amplamente utilizado, suas limitações em aplicações de comunicação em tempo real e dispositivos com restrições de recursos destacam a necessidade de protocolos mais leves e eficientes, como o *Message Queuing Telemetry Transport* (MQTT).

[1] Sistema de páginas web públicas interconectadas e acessíveis pela Internet (MOZILLA, 2024c)

O MQTT é um protocolo de comunicação leve, projetado para dispositivos com recursos limitados e redes de alta latência. Ele opera em um modelo de publicação/assinatura, onde os clientes podem publicar mensagens em tópicos específicos e assinar tópicos para receber mensagens. O MQTT é conhecido pela sua eficiência e baixo consumo de largura de banda, tornando-o ideal para dispositivos que operam em ambientes com conectividade limitada. Dessa forma, esse protocolo se mostra bastante eficaz em ambientes de Internet das Coisas, onde a eficiência e a economia de largura de banda são cruciais (BANKS ED BRIGGS, 2019).

Figura 3 – Funcionamento do MQTT



Fonte: (KHAN et al., 2021, p. 2).

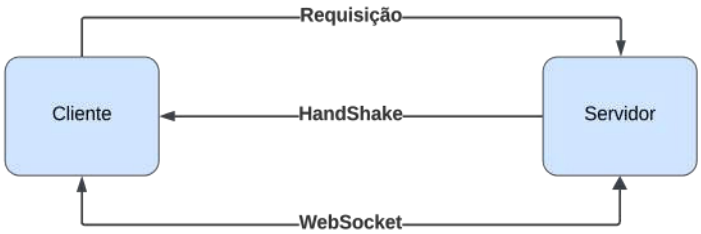
Por fim, diferentemente do HTTP, que segue um modelo de requisição/resposta, e do MQTT, voltado para publicação/assinatura, o *WebSocket* é um protocolo de comunicação que permite a comunicação bidirecional em tempo real entre clientes e servidores (Figura 4). Esse protocolo estabelece uma conexão persistente que permite a troca contínua de dados. Isso o torna ideal para aplicações que requerem atualizações em tempo real, como jogos online, chats e sistemas de monitoramento. O *WebSocket* elimina a sobrecarga de cabeçalhos de requisição/resposta do HTTP, tornando-o mais eficiente em termos de uso de largura de banda e latência (FETT; KÜSTERS; SCHMITZ, 2019).

Analisados aspectos dos sistemas embarcados e protocolos de comunicação, apresentam-se a seguir aspectos relacionados aos desenvolvimento de *software*.

2.3 Desenvolvimento de *Software*

O desenvolvimento de *software* é o processo de concepção, especificação, *design*, programação, documentação, teste e manutenção de aplicativos, *frameworks* ou outros componentes de *software*. Este processo é fundamental para a criação de soluções tecnológicas que atendem às necessidades de usuários e empresas, permitindo a automação de tarefas,

Figura 4 – Funcionamento do *WebSocket*



Fonte: (FETTE; MELNIKOV, 2011).

a análise de dados e a comunicação eficiente. A inovação tecnológica no desenvolvimento de *software* é essencial para a criação de novas ferramentas e plataformas que ampliam as capacidades dos desenvolvedores, permitindo a construção de sistemas mais complexos e eficientes (SMITH; DOE, 2019). O seu desenvolvimento envolve uma combinação de habilidades técnicas e criativas, exigindo conhecimento em linguagens de programação, ferramentas de desenvolvimento e metodologias de gerenciamento de projetos.

Nesse contexto, as linguagens de programação se tornam ferramentas fundamentais para viabilizar esse processo, permitindo que os profissionais da área escrevam instruções que são executadas pelos computadores. Elas variam em termos de sintaxe, paradigmas de programação e áreas de aplicação, e sua escolha depende de fatores como o tipo de aplicação, a plataforma de destino e as preferências do desenvolvedor.

Em relação a etapa de programação no desenvolvimento de *software*, é de fundamental importância a presença de linguagens de programação, as quais viabilizam a solução de problemas por meio de seu uso. Para o presente estudo, empregou-se as linguagens de programação *C++*, *Python*, *JavaScript* e *TypeScript*.

A linguagem *C++*, por exemplo, é uma linguagem de programação de alto nível, de propósito geral, que estende a linguagem *C*, oferecendo suporte a abstração de dados e orientação a objetos (STROUSTRUP, 1986).

Já o *Python* é conhecido por sua simplicidade e legibilidade, sendo amplamente utilizado em áreas como ciência de dados, inteligência artificial e automação de tarefas. Sua vasta biblioteca padrão e comunidade ativa tornam a linguagem uma escolha popular para desenvolvedores de todos os níveis de experiência (PYTHON, 2023).

Por outro lado, o *JavaScript* é essencial no desenvolvimento aplicações de internet, sendo usado tanto no lado do cliente quanto do servidor, principalmente quando associado ao ambiente *Node.js*. Sua flexibilidade e a enorme variedade de bibliotecas e *frameworks* a tornam uma escolha robusta para criar páginas dinâmicas e interativas (NETWORK, 2023). No contexto de desenvolvimento de aplicações escaláveis e de alto desempenho de

aplicativos na rede, destaca-se o *Node.js*, por permitir a execução de código *JavaScript* no servidor e lidar com operações de entrada e saída de forma assíncrona (OPENJS, 2023).

Finalmente, o *TypeScript*, uma extensão do *JavaScript*, oferece tipagem estática opcional e é particularmente útil para o desenvolvimento de aplicativos de grande escala, melhorando a manutenção e a escalabilidade do código em projetos tanto de *frontend* quanto de *backend* (MICROSOFT, 2023).

Além das linguagens de programação, há ferramentas e plataformas que desempenham um papel crucial no processo de desenvolvimento, oferecendo recursos que auxiliam na criação, teste e implantação de programas de computador, de rede ou de sistemas embarcados. Elas oferecem funcionalidades que simplificam o gerenciamento de código, a colaboração em equipe e a automação de tarefas, aumentando a eficiência e a qualidade do desenvolvimento.

Entre as ferramentas essenciais utilizadas neste trabalho, destaca-se o *Docker*, uma plataforma de virtualização que permite a criação, implantação e execução de aplicativos em contêineres. Os contêineres são leves e portáteis, permitindo que os desenvolvedores isolem suas aplicações e dependências em um ambiente consistente, o que facilita o desenvolvimento, teste e implantação de *software*, garantindo que os aplicativos funcionem de forma consistente em diferentes ambientes (DOCKER, 2023).

Em adição ao *Docker*, destaca-se o *Git* é um sistema de controle de versão distribuído que permite aos desenvolvedores rastrear mudanças no código-fonte e colaborar em projetos de *software*. Ele é projetado para lidar com tudo, desde pequenos a grandes projetos, com velocidade e eficiência, sendo uma ferramenta essencial para o desenvolvimento colaborativo e a gestão de código (PROJECT, 2023).

Para complementar, o *GitHub* é uma plataforma baseada na rede, que utiliza *Git* para hospedar repositórios de código, facilitando a colaboração e o compartilhamento de projetos. Ela oferece funcionalidades adicionais, como gerenciamento de projetos, integração contínua e revisão de código, tornando-se indispensável para equipes de desenvolvimento (GITHUB, 2023).

Concluída a análise dos principais aspectos relacionados ao desenvolvimento de software, apresentam-se, a seguir, os elementos associados às Aplicações de Internet.

2.4 Aplicações de Internet

De acordo com o Marco Civil da Internet, defini-se como aplicações de internet "o conjunto de funcionalidades que podem ser acessadas por meio de um terminal conectado à internet;"(BRASIL, 2014). Uma das forma de elaborar uma aplicação de internet é por meio do desenvolvimento do *frontend* e do *backend*.

O *frontend* é a parte do desenvolvimento de *software* que lida com a interface do usuário e a experiência do usuário. Ele envolve a criação de elementos visuais e interativos que permitem aos usuários interagir com um aplicativo ou *site* (SHETTY et al., 2020). O desenvolvimento *frontend* envolve habilidades em *design*, usabilidade e programação. Nesse contexto, diversas bibliotecas e *frameworks* são empregados para facilitar o desenvolvimento de interfaces dinâmicas e interativas. Para o presente estudo, adotou-se como tecnologias para implementação da interface de usuário o *framework JavaScript React*, a biblioteca *Socket.IO Client* e o *Tailwind* CSS.

Um exemplo é o *React*, uma biblioteca escrita na linguagem *JavaScript* para a construção de interfaces de usuário, desenvolvida pela META. Ela permite a criação de componentes reutilizáveis que gerenciam seu próprio estado, facilitando a construção de interfaces *dinâmicas* e interativas. O *React* adota um modelo de programação declarativo, onde os desenvolvedores descrevem como a interface deve parecer em diferentes estados, e a biblioteca cuida das atualizações necessárias. Esse *framework* é amplamente utilizado em aplicações de rede modernas devido à sua eficiência e flexibilidade (META, 2023).

Outro recurso importante no desenvolvimento *frontend* é o *Socket.IO Client*, uma biblioteca para *JavaScript* que cuida da comunicação do lado do cliente. Ela estabelece a comunicação bidirecional com o servidor, permitindo o envio e recebimento de mensagens por meio de *WebSockets*. A biblioteca oferece uma *Application Programming Interface* (API) simples e robusta, facilitando a gestão da conexão com o servidor, o que é essencial para manter a interatividade e o dinamismo em aplicações em tempo real (SOCKET.IO, 2023).

Além disso, o *Tailwind* é um *framework* de CSS [2] que permite a criação rápida de interfaces de usuário personalizadas. Ele fornece classes utilitárias que podem ser combinadas para construir *designs* responsivos e modernos sem sair do HTML [3] (TAILWIND, 2023).

Enquanto o *frontend* lida com a interface e a experiência do usuário, o *backend* é a etapa de programação do desenvolvimento que lida com a lógica do servidor, o gerenciamento de dados e a comunicação entre diferentes partes de um sistema. Ele é responsável por processar as requisições enviadas por clientes, executar operações de negócios e interagir com bancos de dados e outros serviços. Para garantir a comunicação eficaz entre o *frontend* e o *backend*, o uso de API e *frameworks* específicos torna-se essencial. A seguir, são descritas as tecnologias e arquiteturas de maior relevância para a presente pesquisa.

[2] "CSS, ou *Cascading Style Sheets*, é uma linguagem de estilo que define a apresentação de documentos escritos em HTML e especifica como os elementos devem ser exibidos em diversas mídias, como tela, papel ou dispositivos de voz."(MOZILLA, 2024a).

[3] "HTML, ou *Hypertext Markup Language*, é o bloco de construção mais básico da web. Define o significado e a estrutura do conteúdo da web."(MOZILLA, 2024b)

A API *Representational State Transfer* (REST) é um estilo de arquitetura para a construção de serviços web. Ela utiliza os verbos do HTTP padrão: *GET*, *POST*, *PUT* e *DELETE* para realizar operações em recursos identificados por URL. As API REST são projetadas para serem escaláveis e independentes da plataforma, ao permitir que diferentes sistemas se comuniquem de forma eficiente. Elas são amplamente utilizadas para conectar aplicativos *frontend* e *backend*, facilitando a troca de dados e a integração de serviços (FIELDING, 2000).

Além disso, o uso de *frameworks* como o *Express* tem se tornado comum no desenvolvimento de *backend*. O *Express* é um *framework* para *Node.js*, projetado para construir aplicações de internet e API de forma rápida e fácil. Ele fornece um conjunto robusto de funcionalidades para o desenvolvimento de servidores HTTP, incluindo roteamento, *middleware*[4] e suporte a modelos (EXPRESS.JS, 2023).

Para a modelagem de dados em aplicativos *backend*, o Mongoose é uma biblioteca de modelagem de dados para *MongoDB* e *Node.js*. Ele oferece uma solução baseada em esquemas para modelar dados, incluindo validação e construção de consultas (MONGOOSE, 2023). Além disso, no lado do servidor, a comunicação bidirecional é facilitada pela biblioteca *Socket.IO Server*, que escuta novas conexões de clientes e coordena a comunicação entre eles. Essa biblioteca oferece uma API robusta para o gerenciamento de conexões e troca de dados em tempo real, permitindo que servidores respondam e interajam com os clientes de maneira eficiente (SOCKET.IO, 2023).

Enquanto o gerenciamento de conexões e comunicação em tempo real são essenciais para o funcionamento de aplicativos dinâmicos, é igualmente importante garantir que os dados gerados e manipulados durante essas interações sejam armazenados de maneira eficiente e segura. É nesse contexto que os bancos de dados desempenham um papel fundamental no desenvolvimento de *backend*, proporcionando a estrutura necessária para armazenar, organizar e acessar dados de forma rápida e confiável.

Os bancos de dados são sistemas que armazenam, organizam e gerenciam grandes volumes de dados. Eles são essenciais para o funcionamento de aplicativos que requerem armazenamento persistente de informações, permitindo consultas eficientes e operações de manipulação de dados (ELMASRI; NAVATHE, 2016). Existem dois principais tipos de bancos de dados: os relacionais e os não relacionais.

Bancos de dados relacionais organizam dados em tabelas com linhas e colunas, utilizando uma linguagem de consulta estruturada (SQL) para gerenciar e acessar os dados. Eles são ideais para aplicações que requerem transações complexas e integridade referencial, como sistemas de gestão empresarial e bancos de dados financeiros (STONEBRAKER et al., 2018).

[4] Camada de *software* que fornece uma API, conectando o sistema operacional, dados e usuários (HAT, 2022)

Bancos de dados não relacionais (*Not Only SQL*, NoSQL), são projetados para armazenar dados de forma mais flexível, sem a rigidez das tabelas relacionais. Eles são adequados para aplicações que lidam com grandes volumes de dados não estruturados ou semi-estruturados, como redes sociais, sistemas de gerenciamento de conteúdo e plataformas de análise de dados (POKORNY, 2018). Para o presente estudo, adotou-se um banco de dados não relacional, utilizado na aplicação: o *MongoDB*.

O *MongoDB* é um banco de dados NoSQL orientado a documentos, conhecido por sua flexibilidade e escalabilidade. Ele armazena dados em documentos estilo *JavaScript Object Notation* (JSON), permitindo que os desenvolvedores trabalhem com dados de forma mais natural e intuitiva. Ele é ideal para aplicações que lidam com grandes volumes de dados não estruturados ou semi-estruturados, como aplicativos de redes sociais, sistemas de gerenciamento de conteúdo e plataformas de análise de dados. Sua capacidade de escalar horizontalmente e suportar consultas complexas o torna uma escolha popular para desenvolvedores que buscam flexibilidade e desempenho (MONGODB, 2023).

Encerrada a discussão sobre o desenvolvimento de software, o próximo capítulo aborda os fundamentos e a aplicação de técnicas de inteligência artificial no contexto deste trabalho.

2.5 Inteligência Artificial

A Inteligência Artificial (IA) é um campo do estudo de extrema importância na construção de modelos para tomada de decisão. A IA está relacionada ao uso de computadores para compreender a inteligência humana, apesar de não precisar se limitar a métodos observáveis biologicamente (MCCARTHY et al., 2007). Ela é um campo vasto e dinâmico que visa criar sistemas capazes de simular habilidades humanas, como raciocínio, percepção e tomada de decisões.

Por outra perspectiva, Elaine Rich apresenta a IA sob uma outra ótica, ao definir IA como "[...] o estudo de como fazer com que os computadores façam coisas nas quais, no momento, as pessoas são melhores"(RICH, 1985). A IA fundamenta-se na integração de outras disciplinas do conhecimento, entre elas a filosofia, matemática, economia, neurociência, psicologia, engenharia de computação, teoria de controle e cibernética, linguística (RUSSELL; NORVIG, 2010). O uso da IA torna-se mais comum a cada dia, oferecendo soluções que vão desde assistentes pessoais virtuais até sistemas de diagnóstico médico avançados.

Com o avanço exponencial das tecnologias e o aumento na capacidade computacional, a IA tem sido usada para resolver problemas complexos e em áreas cada vez mais especializadas. A seguir, serão apresentados tópicos importantes para entender e implementar soluções com IA: aprendizado de máquina, aprendizado profundo, redes neurais, visão

computacional, plataformas e ferramentas de desenvolvimento para visão computacional, sobreajuste (*overfitting*) e subajuste (*underfitting*), técnicas de processamento e aumento de dados, algoritmos de treinamento e avaliação de desempenho dos modelos.

2.5.1 Aprendizado de Máquina

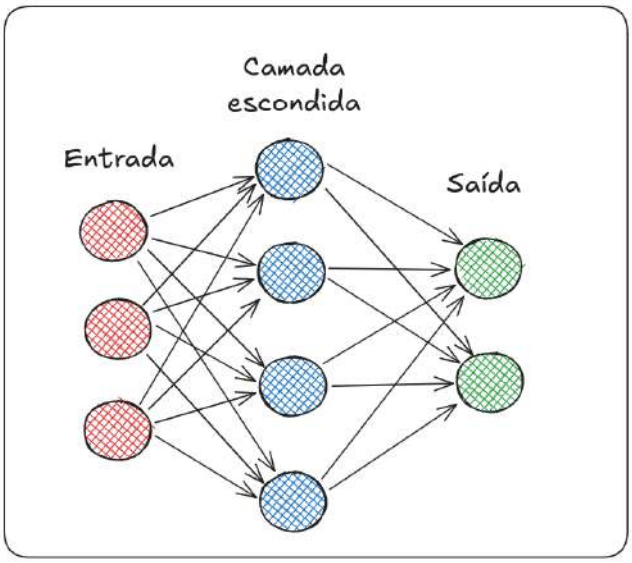
Aprendizado de máquina, ou *Machine Learning* (ML), é o "Campo de estudo que dá aos computadores a capacidade de aprender sem serem explicitamente programados"(SAMUEL, 1959). ML é subárea da IA que envolve a criação de algoritmos capazes de aprender a partir de dados e fazer previsões ou tomar decisões com base em padrões observados (MONARD; BARANAUSKAS, 2003). Diferente de uma programação tradicional, onde o código é escrito para realizar uma tarefa específica, o aprendizado de máquina permite que sistemas ajustem automaticamente suas operações para melhorar seu desempenho em tarefas específicas (GULLI; PAL, 2017). Por meio dessa técnica, é possível fazer com que máquinas aprendam diferentes padrões e formulem soluções que vão desde encontrar a função de uma reta até o reconhecimento de padrões em imagem e voz.

Para Russel e Norvig (2010, p. 694-695), a maneira pela qual uma máquina pode aprender pode ser dividido dividido de quatro maneiras: Aprendizado Não-Supervisionado, que aquele em que os padrões são aprendidos pela máquina sem que haja nenhum tipo de retorno, ou seja, a máquina não tem noção rótulos em seus dados; Aprendizado Supervisionado, o qual trata-se daquele em que a máquina observa uma quantidade de dados observará o par entrada-saída. Em seguida, aprende a função que mapeia uma determinada entrada a uma determinada saída; Aprendizado por Reforço, que é guiado por recompensas e punições, onde uma vez que o modelo aja conforme o desejado ele é recompensado, caso contrário, ele sofrerá uma punição. Esse processo é repetido diversas vezes até o momento em que a máquina alcance em um nível satisfatório de conhecimento; e por fim, Aprendizado Semi-Supervisionado, o qual o modelo receba uma quantidade mínima de amostras comparada ao todo e estas amostra não está acurada, ou seja, há ruídos nos dados.

No processo de aprendizado de máquina, além de selecionar o tipo de aprendizado mais adequado para uma determinada tarefa, é essencial ajustar os parâmetros que governam o comportamento dos algoritmos. Esses parâmetros, conhecidos como hiperparâmetros, são valores definidos antes do treinamento do modelo e têm um impacto significativo no desempenho e na capacidade de generalização da solução (TENSORFLOW, 2022b). Diferentemente dos parâmetros aprendidos durante o treinamento, como pesos em redes neurais, os hiperparâmetros controlam aspectos como a taxa de aprendizado (*learning rate*), o número de camadas em uma rede neural e a profundidade máxima de uma árvore de decisão, sendo frequentemente ajustados por meio de métodos como busca em grade (*grid search*) ou busca aleatória (*random search*).

No contexto da IA, destaca-se a Rede Neural Artificial (RNA), que é um de seus método. Uma RNA é definida como um conjunto de unidades de processamento que estão interligadas (YEGNANARAYANA, 2009). Ela é composta por: uma camada de entrada, uma camada escondida e uma camada de saída (Figura 5).

Figura 5 – Arquitetura de uma RNA



Fonte: (SUZUKI, 2011, p. 4)

Cada uma dessas unidades é um algoritmo que toma como entrada um vetor x de m valores (x_1, x_2, ..., x_n) chamado de *features* e tem como saída 0 (não, falso) ou 1 (sim, verdadeiro) e são chamados de *perceptron* (GULLI; PAL, 2017)

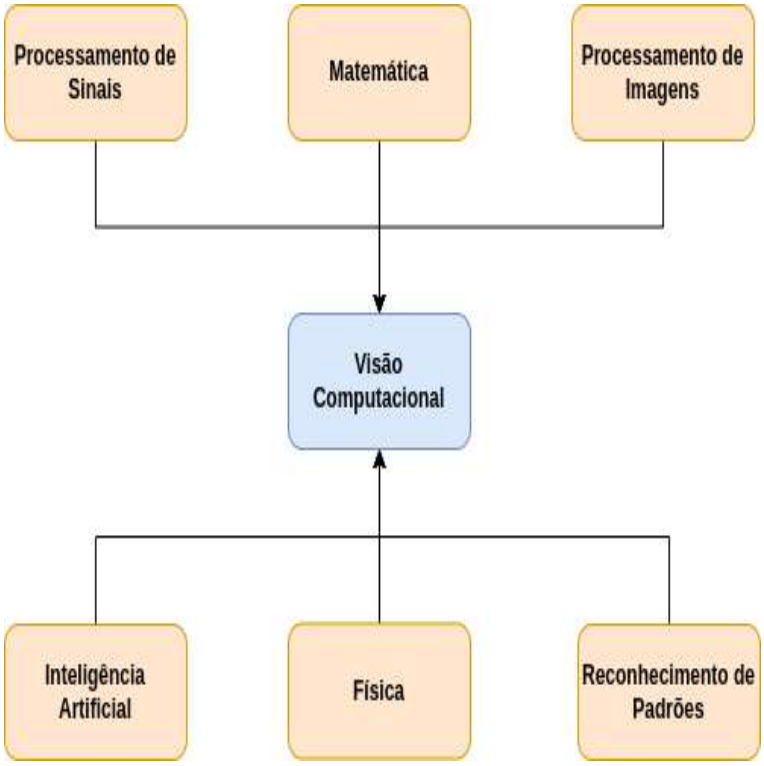
Com a evolução das RNA, surgiram arquiteturas mais complexas que expandiram significativamente as capacidades desses modelos, resultando no desenvolvimento da Aprendizagem Profunda, ou *Deep Learning*. Essa subárea do ML utiliza redes neurais com múltiplas camadas para processar grandes volumes de dados e identificar padrões complexos. Essa abordagem tem se mostrado especialmente eficaz em tarefas como reconhecimento de imagem, processamento de linguagem natural e jogos, entre outras, devido à sua capacidade de modelar relações complexas entre variáveis de entrada e saída (GULLI; PAL, 2017).

No contexto da visão computacional, um dos campos beneficiados pelos avanços da Aprendizagem Profunda, busca-se permitir que sistemas de software e hardware sejam capazes de captar, analisar e compreender objetos, imagens e/ou *frames* de vídeo, com o objetivo de tomar decisões baseadas na interpretação visual. Assim como os humanos utilizam a visão para interagir com o ambiente, a visão computacional tem como objetivo fazer com que os sistemas de *software* e *hardware* sejam capazes de captar, analisar e compreender objetos e imagens e/ou *frames* de vídeo.

A visão computacional é uma subárea da IA que foca na "coleta, análise e síntese de dados visuais através de computadores, com objetivos diversos como a identificação de

rostos e biometria, a análise de representações de objetos, entidades, conceitos e contextos em imagens, entre outros"(WANG; ZHANG; MARTIN, 2015, p. 1). Além do ponto de vista biológico, a visão computacional pode também ser entendida como a composição outras áreas (Figura 6).

Figura 6 – Campos de Estudo da Visão Computacional



Fonte: (BARELLI, 2018)

A fim de realizar as tarefas citadas acima, a visão computacional utiliza algoritmos complexos de reconhecimento de padrões. Uma técnica essencial é a de detecção de objetos, que identifica e classifica objetos em uma imagem. Segundo Barelli (BARELLI, 2018), essa tarefa pode ser dividida em 5 etapas, na ordem descrita abaixo:

a) **Aquisição da imagem**: Obter as imagens desejadas por meio de um *hardware* próprio (câmera digital, por exemplo) e guardar esse conteúdo de maneira digital;

b) **Pré-processamento**: Definir objetos de interesse na imagem e aplicar técnicas para destacar essas partes de interesse das outras partes da imagem;

c) **Segmentação**: Técnica que separa os objetos desejados do restante da imagem, recebendo mais destaque;

d) **Extração de características**: Com o objeto de interesse em destaque, são extraídas características que do objeto em questão;

e) **Reconhecimento de padrões**: Etapa onde há mais um processamento, buscando reconhecer o padrão do objeto de interesse.

Em conjunto com as RNA, há os modelos de Redes Neurais Convolucionais (*Convolutional Neural Network*, CNN) que são frequentemente utilizados nessa área devida a sua capacidade em lidar com aprendizagens complexas, combinando unidades de processamento linear e não-linear várias camadas (KHAN et al., 2020). Os parágrafos a seguir apresentam as principais ferramentas no emprego da solução de visão computacional para a presente pesquisa.

Ademais, das CNN, destaca-se o *You Only Look Once* (YOLO), um sistema de detecção no estado da arte conhecido por sua habilidade de identificar múltiplos objetos em uma única passada pela imagem. Essa característica proporciona eficiência e precisão para detecção em tempo real, além de garantir modelos menores e mais rápidos (JIANG et al., 2022).

Nesse contexto, a utilização de bibliotecas como o *OpenCV* potencializa ainda mais o uso de técnicas avançadas de visão computacional, como o YOLO. *OpenCV* é uma biblioteca de *software* de visão computacional e aprendizado de máquina de código aberto. Ela oferece interfaces para linguagens de programação como *C++*, *Python*, *Java* e *MATLAB*, suportando múltiplos sistemas operacionais, incluindo *Windows*, *Linux*, *Android* e *MacOS*. Focada em aplicações de visão computacional em tempo real, a biblioteca aproveita instruções MMX e SSE, além de contar com interfaces para CUDA e *OpenCL*, que estão em desenvolvimento (OPENCV, 2024). Com um acervo de mais de 500 algoritmos e milhares de funções, o *OpenCV* viabiliza a criação de diversas aplicações de visão computacional, incluindo manipulação de imagens e controle de dispositivos com câmeras.

Como mencionado, YOLO é uma técnica revolucionária para detecção de objetos, considerada pelos seus criadores como o estado da arte em detecção de objetos em tempo real. Essa abordagem utiliza modelos pré-treinados juntamente com um conjunto de dados em larga escala conhecido como *Common Objects in Context* (COCO) (REDMON, 2024).

Diferentemente de outros métodos, que analisam as imagens múltiplas vezes, o YOLO processa a imagem apenas uma vez. Ele divide a imagem em uma grade e tenta identificar objetos em cada célula dessa grade. Cada célula estima a classe e a posição do objeto, resultando em um modelo rápido e preciso. Versões mais avançadas, como YOLOv3 e YOLOv4, oferecem melhorias significativas em termos de precisão e capacidade de detectar objetos menores (FARHADI; REDMON, 2018).

Concluída a abordagem sobre inteligência artificial e suas aplicações, inicia-se a discussão acerca dos conceitos de *underfitting* e *overfitting*, fundamentais para compreender o desempenho e a generalização de modelos de aprendizado de máquina.

2.6 *Overfitting* e *Underfitting*

No contexto de aprendizado de máquina, *overfitting* e *underfitting* estão relacionados ao desempenho de modelos durante o treinamento e à sua capacidade de generalização para novos dados. Ambos os fenômenos representam problemas que podem comprometer a eficácia de um modelo.

O *overfitting* ocorre quando o modelo está tão bem ajustado aos dados de treino, que o mesmo não é capaz de realizar previsões baseadas em novos dados (DIETTERICH, 1995). Como resultado, o modelo apresenta um excelente desempenho no conjunto de treinamento, mas falha em generalizar para novos dados, apresentando alto erro no conjunto de teste (YING, 2019). Esse problema é comum quando o modelo é excessivamente complexo em relação à quantidade ou diversidade dos dados disponíveis, fazendo-o não generalizar o suficiente.

Por outro lado, o *underfitting* acontece quando o modelo é incapaz de capturar os padrões subjacentes dos dados, apresentando baixo desempenho tanto no conjunto de treinamento quanto no conjunto de teste. Isso geralmente ocorre quando o modelo é muito simples, tem pouca capacidade de aprendizado ou quando não foi treinado por tempo suficiente, fazendo com que o modelo generalize demais, ao ponto de um pequeno ruído nos dados ser o suficiente para entender como um padrão a ser aprendido (AALST et al., 2010).

Ambos os problemas podem ser mitigados por meio de técnicas como a validação cruzada, ajuste de hiperparâmetros, aumento do conjunto de dados e uso de regularização para equilibrar a complexidade do modelo.

No contexto da visão computacional, os conceitos de *overfitting* e *underfitting* têm implicações significativas, pois influenciam diretamente o desempenho de modelos utilizados em tarefas como classificação de imagens e detecção de objetos.

Em um modelo de visão computacional aplicável ao presente estudo, por exemplo, o *overfitting* pode levar o modelo a memorizar detalhes irrelevantes, como o fundo da imagem ou condições específicas de iluminação, em vez de aprender características. Como consequência, o modelo apresenta alto desempenho no conjunto de treinamento, mas falha em generalizar para novas imagens capturadas em diferentes ambientes ou condições.

Por outro lado, um modelo com *underfitting* não será capaz de capturar os padrões visuais fundamentais necessários para distinguir entre as classes, levando-o a não identificar corretamente características importantes, como a textura ou cor de algum produto, resultando em baixo desempenho tanto no conjunto de treinamento quanto no conjunto de teste.

Afim de atenuar esses problemas, é essencial adotar estratégias como aumento de dados, que visa gerar variações artificiais das imagens para aumentar a diversidade do

conjunto de treinamento. Além disso, o uso de técnicas de regularização, como a remoção de neurônios da rede neural de forma aleatória ou normalização de peso, pode ajudar a reduzir o *overfitting*, enquanto arquiteturas de redes mais profundas e sofisticadas, combinadas com conjuntos de dados mais robustos, podem evitar o *underfitting*.

Portanto, a aplicação dessas estratégias não apenas contribui para melhorar a generalização e a robustez do modelo, mas também garante que a etapa de treinamento seja mais eficiente e confiável, mesmo diante de conjuntos de dados limitados ou heterogêneos. Contudo, antes de realizar o treinamento propriamente dito, é indispensável assegurar que os dados estejam devidamente preparados, organizados e estruturados, uma vez que o processamento de dados desempenha um papel crucial na qualidade do modelo final e nos resultados alcançados. A seguir, serão abordados os principais métodos e técnicas envolvidos no processamento de dados do presente estudo.

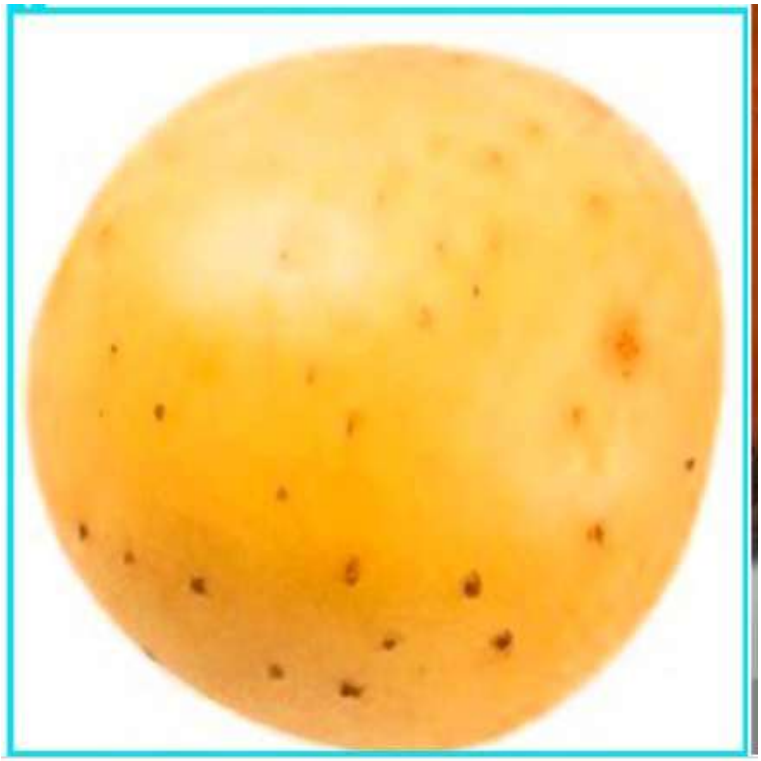
2.7 Processamento de Dados

Um dos passos críticos no desenvolvimento de modelos de IA é o processamento de dados. Em visão computacional, as etapas mais comuns de processamento são anotações e aumento de dados.

No que tange a visão computacional, o treinamento de modelos requer dados anotados, isto é, imagens que foram marcadas para indicar a localização e a classe dos objetos. Por exemplo, um modelo que deve detectar veículos em uma imagem precisa de dados rotulados onde as áreas contendo carros, caminhões, motos, entre outros, estejam claramente identificadas. Esse processo manual é essencial para que o modelo aprenda a reconhecer os diferentes objetos presentes nas imagens, pois nessa etapa são fornecidos os rótulos do objeto o qual deseja-se detectar. Dentre os tipos de anotação de dados, um dos tipos mais conhecidos são as caixas delimitadoras, que consiste em destacar o objeto em questão utilizando formas geométricas (ALVI, 2024) (Figura 7).

Como muitos modelos precisam de grandes quantidades de dados para alcançar bons resultados, o aumento dos dados é uma técnica valiosa para ampliar o conjunto de dados de forma artificial. Ela consiste em aplicar transformações nas imagens existentes, como rotação, inversão, ajuste de brilho e contraste, para gerar novas variáveis de entrada sem a necessidade de coletar mais imagens (TENSORFLOW, 2022a). Isso ajuda a reduzir o *overfitting* e melhora a robustez do modelo, pois ele será exposto a uma maior variedade de cenários.

Figura 7 – Exemplo do Uso de Caixas Delimitadoras



Fonte: De autoria própria.

2.8 Avaliação de Desempenho de Modelo

Após treinar um modelo, é essencial avaliar sua performance para garantir que ele cumprirá seu propósito de maneira confiável. A avaliação de modelos pode ser obtida por meio de métodos matemáticos que serão descritos abaixo. Para cada um dos métodos avaliativos, considere as siglas abaixo:

$$\begin{cases} VP = \text{Verdadeiros Positivos} \\ VN = \text{Verdadeiros Negativos} \\ FP = \text{Falsos Positivos} \\ FN = \text{Falsos Negativos} \end{cases}$$

Na IA, alguns dos principais métodos de avaliação incluem: a acurácia é uma medida simples que indica a proporção de previsões corretas em relação ao total de previsões. Funciona bem em conjuntos de dados balanceados, mas pode ser enganosa se houver classes desbalanceadas. Pode ser definida matematicamente como:

$$\text{Acurácia} = \frac{\text{Classificações Corretas}}{\text{Total de Classificações}} = \frac{VP + VN}{VP + VN + FP + FN}$$

A Revocação é uma das métricas mais avançadas para problemas de classificação. A precisão indica a proporção de previsões corretas dentre todas as previsões de uma

classe específica, enquanto a revocação mostra a proporção de previsões corretas de uma classe em relação ao total real dessa classe. Em conjunto, ajudam a equilibrar previsões corretas e incorretas.

$$\text{Revocação} = \frac{\text{Classificado corretamente verdadeiros positivos}}{\text{todos os verdadeiros positivos}} = \frac{VP}{VP + FN}$$

A Média Harmônica (Métrica F1) é uma métrica que combina precisão e revocação, útil em casos onde é necessário balancear a presença de falsos positivos e falsos negativos. Isso é especialmente relevante em tarefas de detecção de objetos, onde é importante identificar corretamente todos os objetos sem incluir objetos inexistentes.

$$F1 = 2 \times \frac{\text{Precisão} \times \text{Revocação}}{\text{Precisão} + \text{Revocação}}$$

A Média de Precisão Média (mAP), no contexto de detecção de objetos, atua como uma métrica popular, calculada com base na precisão de cada classe em várias faixas de revocação. Ele oferece uma visão geral da performance do modelo em identificar corretamente as classes e localizações dos objetos.

$$MAP = \frac{1}{n} \sum_{k=1}^{n} AP_K, \text{ onde } \begin{cases} n = \text{Número de classes} \\ AP_k = \text{Precisão média da classe } k \end{cases}$$

2.9 Avaliação do Referencial Teórico

O referencial teórico apresentado fornece uma base sólida para a compreensão das tecnologias e conceitos fundamentais que sustentam o desenvolvimento do sistema proposto para a classificação automatizada. A integração de sistemas embarcados, protocolos de comunicação, desenvolvimento de *software* e inteligência artificial representa uma abordagem inovadora e multidisciplinar, essencial para enfrentar os desafios da automação.

Do exposto, o referencial teórico não apenas fundamenta o desenvolvimento do presente estudo, mas também mostra como diferentes áreas do conhecimento se conectam e reforça a necessidade de uma abordagem integrada para a inovação tecnológica no campo da automação.

3 Metodologia

O presente trabalho baseia-se em uma pesquisa aplicada e visa a construção de um protótipo para a separação automatizada de produtos saudáveis e não saudáveis. Com um caráter exploratório e experimental, o estudo adota uma abordagem quali-quantitativa, buscando tanto a medição da precisão do modelo de classificação e eficiência da esteira, quanto a análise qualitativa de aspectos visuais do caso de uso, como cor e textura para identificar sinais de não conformidade.

Para a construção do protótipo, foram realizadas etapas que incluíram uma revisão bibliográfica sobre modelos de classificação binária, especificação de *hardware* para automação da esteira, implementação de um sistema de visão computacional para detecção e classificação de produtos, além da integração e dos testes em um ambiente controlado. Utilizou-se o método experimental, com a execução e avaliação dos testes no sistema automatizado, observando os resultados para validar a eficácia da solução proposta.

3.1 Revisão bibliográfica

Foram pesquisados trabalhos que tratam da classificação de imagens, com o objetivo de respaldar a escolha e o aprimoramento dos algoritmos de visão computacional aplicados. Além disso, foram levantadas referências sobre *hardwares* destinados à separação de objetos, fundamentais para a definição dos componentes físicos do protótipo, e também sobre protótipos que integrem processos de classificação e separação automática, visando obter uma visão abrangente de soluções já implementadas e identificar tecnologias e abordagens relevantes para a construção do sistema proposto.

Em relação ao uso de modelos de classificação, foram encontrados trabalhos relevantes na área de classificação de imagens que utilizavam Redes Neurais Convolucionais, como por exemplos a ResNet-50, MobileNet e GoogleNet, que são diferentes tipo de arquiteturas de Redes Neurais, além do próprio YOLO. O principal trabalho, e que serviu de inspiração para elaboração do estudo, foi o uso de uma técnica híbrida de *Deep Learning*, que combinou a arquitetura MobileNet V2 com *Long short-term memory* (LSTM) (FARIA et al., 2023).

Em relação a projetos de *hardware*, foram encontrados trabalhos diversas construções de protótipos de *hardware*, dentre os quais nos inspiraram uma esteira controlada por um motor de máquina de costura, sensores para identificação de materiais e mecanismos de separação baseados em servo motores, gerenciado por um Arduino MEGA2560 R3 (BRANDT; RENKEN; LEITE, 2019).

Esses elementos serviram como base para a concepção do protótipo separador e elaboração do modelo classificador, que apresenta funcionalidades adaptadas ao contexto de classificação de batatas próprias e impróprias para consumo.

A esteira é um elemento central e é controlada por um ESP-32, que atua como o microcontrolador principal.

Um sensor infravermelho é responsável por identificar a presença de um objeto e parar a esteira em frente a câmera. A partir disso, é feita a captura da imagem do objeto em questão e realiza-se a classificação de acordo com o modelo de classificação construído com o algoritmo YOLO.

Essa abordagem substitui os sensores específicos e a análise acústica do projeto anterior, priorizando uma solução baseada em aprendizado de máquina para maior precisão.

3.2 Requisitos do protótipo

A seguir, serão apresentados os requisitos funcionais e não funcionais levantados para o desenvolvimento do protótipo. Esses requisitos abrangem tanto a parte de *software* quanto de *hardware*, visando a criação de um sistema automatizado para a separação de produtos saudáveis e não saudáveis.

Os requisitos funcionais do projeto foram definidos com base nas ações as quais o protótipo deve desempenhar. Abaixo são descritos os requisitos funcionais.

a) Algoritmo de visão computacional, para a classificação de batatas em saudáveis e não saudáveis, de acordo com os critérios de cor e textura;

b) Protótipo com esteira, sensores e atuadores, visando separar automaticamente as batatas de acordo com a classificação da etapa anterior;

c) *Dashboard* web para apresentação da classificação em tempo real, informando: a quantidade de classificações saudáveis e não saudáveis, data e a hora do última objeto classificado.

Os requisitos não funcionais do sistema foram definidos com base nas necessidades de desempenho, confiabilidade e usabilidade conforme descrito abaixo:

a) O sistema deve realizar a classificação e separação em tempo real, com tempos mínimos de parada da esteira para garantir eficiência operacional;

b) O modelo YOLO deve operar com precisão acima de 80%;

c) O *hardware* (sensores, câmeras e servo motor) deve operar de forma consistente, suportando ciclos contínuos de operação;

d) O servidor deve manter estabilidade e disponibilidade contínua para registrar os dados de classificação em tempo real;

e) O painel de controle deve apresentar as informações de classificação de forma clara, intuitiva e atualizada em tempo real para os operadores;

f) O sistema deve ser simples de configurar e operar por usuários com conhecimentos básicos em tecnologia.

3.3 Desenvolvimento da Aplicação de Internet

A etapa de desenvolvimento do App dividiu-se em desenvolvimento do *backend* e *frontend*, nessa ordem.

Para o *backend* do sistema, o banco de dados não relacional MongoDB foi escolhido devido à sua flexibilidade e escalabilidade. Foi utilizada uma imagem do Docker para facilitar a implantação e gerenciamento do banco de dados.

O *backend* foi desenvolvido em *TypeScript*, por meio do *framework Express.js*, que permitiu manter o servidor no *online* de forma eficiente e segura. Foi implementada uma configuração de *WebSocket* utilizando o *Socket.io* para possibilitar o envio de dados em tempo real para uma plataforma *frontend*, garantindo que as informações sobre a classificação sejam atualizadas instantaneamente para os usuários a medida que novas telemetrias sejam salvas e enviando a quantidade de cada tipo de classificação.

Para conectar o *backend* ao banco de dados, faz-se uso do Mongoose, uma biblioteca que facilita a modelagem de dados e as interações com o banco de dados. O modelo dos dados de telemetria no Mongoose possui a seguinte estrutura:

```
{
  isHealthy: {
    type: Boolean, required: true,
  },
  timestamp: {
    type: Date, required: true,
  }
}
```

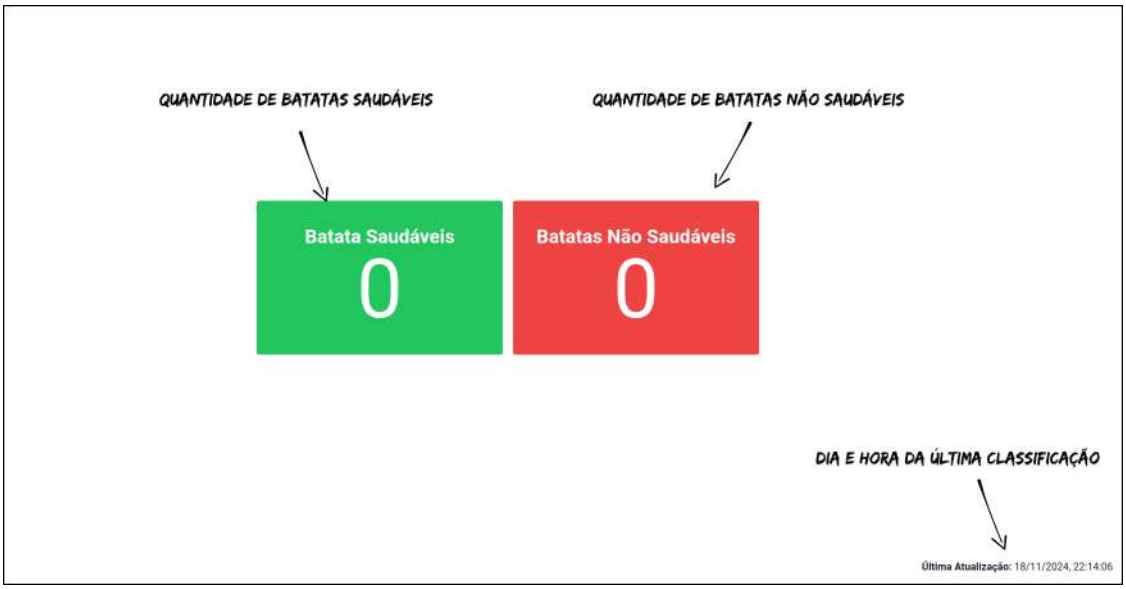
Além disso, criou-se uma rota *POST* seguindo o padrão RESTful API, que permite adicionar uma nova telemetria ao *MongoDB*. Esta rota não só armazena os dados de classificação, mas também envia, via *WebSocket*, a quantidade atualizada de mercadorias saudáveis e não saudáveis no banco de dados juntamente com a data e hora da última leitura, após a adição da nova telemetria. Isso permite que o *frontend* exiba essas informações

em tempo real, proporcionando uma visão clara e imediata do desempenho do sistema de classificação.

Para o desenvolvimento do *frontend*, optou-se por utilizar o *framework React* com *TypeScript*. Ele proporcionou uma base sólida por ser uma linguagem de programação fortemente tipada, voltada para a construção de interfaces de usuário dinâmicas e escaláveis. Para a estilização, empregou-se o *Tailwind* CSS, que colaborou para a elaboração um design moderno e responsivo, utilizando classes utilitárias que facilitam a personalização visual.

A comunicação em tempo real com o *backend* foi implementada usando o *Socket.IO*, que estabelece uma conexão *WebSocket* robusta. Assim, garantiu-se que a interface do usuário receba atualizações instantâneas sobre a quantidade de objetos classificadas como adequados e não adequados, além de exibir a data e hora da última telemetria salva. Essa abordagem assegura que os usuários tenham acesso às informações mais recentes e precisas, melhorando a experiência geral de monitoramento e interação com o sistema, ao passo que novas classificações sejam realizadas.

Figura 8 – *Dashboard* Contabilizador



Fonte: De autoria própria.

3.4 Coleta de Dados e Pré-processamento

Como citado na Introdução (Seção 1), para fins do presente estudo, utilizou-se como base de dados para treino do modelo de visão computacional (Seção 3.5) um conjunto de imagens com fotos de batatas saudáveis e não saudáveis. As razões pelas quais escolheu-se esse conjunto de dados foram: dados previamente catalogados, maior facilidade na aquisição das imagens e a presença da cultura do plantio de batatas na região da Paraíba.

O processo de coleta de dados para o treinamento do modelo baseou-se em múltiplas fontes, visando construir um conjunto abrangente e representativo para a classificação de batatas saudáveis e não saudáveis. Ao todo reuniu-se um acervo com 800 imagens de batatas, dentre os quais 300 imagens são de batatas saudáveis e 500 imagens de batatas não saudáveis.

A base de dados foi composta por imagens extraídas de três principais fontes: provenientes da plataforma Kaggle, fotografias próprias produzidas para o estudo e imagens obtidas da internet com licenças *Creative Commons*. Esse conjunto incluiu imagens que representavam cinco doenças diferentes em batatas, utilizadas para o treinamento do modelo de detecção. Por se tratar de um modelo binário, toda doença ou sinal que indicasse que uma batata estava fora dos padrões de qualidade, faria com que a mesma fosse considerada doente.

Antes de passar as imagens brutas para o YOLO, realizou-se manualmente o processo de anotação dos dados, sendo essencial para preparar as imagens das batatas para a classificação. A anotação consistiu em delimitar, por meio de caixas delimitadoras, as áreas relevantes das imagens que correspondiam a batatas saudáveis ou doentes, além de associá-las às suas respectivas classes. Ao final dessa etapa, é gerado um arquivo texto contendo as informações da posição dos 4 vértices da caixa delimitadora e o rótulo (batata saudável ou batata doente). Esse procedimento foi fundamental para fornecer ao modelo YOLO as informações estruturadas e categorizadas necessárias para identificar e classificar corretamente os objetos durante o treinamento.

Além disso, foi empregada a técnica de aumento de dados para ampliar a diversidade do conjunto de dados, aumentando a robustez do modelo frente a variações nas condições das imagens. Foram aplicadas transformações como diferentes graus de rotação, adição de ruídos e alterações nas escalas das imagens. Essas técnicas permitiram que o modelo fosse treinado com um conjunto de dados mais amplo e representativo, melhorando sua capacidade de generalização ao lidar com dados reais em diferentes condições. O aumento de dados também foi fundamental para mitigar problemas de *overfitting* e *underfitting*, uma vez que as classes estavam desbalanceadas, o que poderia comprometer a capacidade do modelo de aprender padrões adequados para ambas as categorias. Ao final dessa etapa, o conjunto de dados sofreu um aumento de 800 para 2000 imagens, com cerca de 850 imagens de batatas saudáveis e 1150 imagens de batatas não saudáveis.

Ao fim dessas duas etapas, foi utilizada a proporção 80:20, sendo 80% do conjunto de dados usados para o treinamento do modelo e os outros 20% para testes, validação e avaliação.

3.5 Treinamento do Modelo

Inicialmente, desenvolveu-se uma rede neural própria para a classificação de batatas. Contudo, percebeu-se que essa abordagem seria desvantajosa, uma vez que exigiria um tempo considerável para ajustes nas quantidades de *perceptron*, em seus pesos, treinamento e otimização do modelo, o que não se alinhava com o cronograma disponível.

Posteriormente, tentou-se utilizar a arquitetura *ResNet-50*, amplamente reconhecida por sua eficiência em tarefas de classificação. Apesar de seu desempenho satisfatório nessa tarefa específica, a rede *ResNet-50* apresentou limitações, pois não atendia à necessidade de realizar tanto a classificação quanto a detecção de objetos em tempo real, funcionalidades essenciais para o projeto.

Por fim, para construção do modelo classificador de batatas, escolheu-se o algoritmo YOLO, que atendeu as expectativas de criar um modelo para classificação binária e detecção de objetos. As razões pelas quais escolheu-se YOLO ao invés de uma arquitetura de rede neural, foram (JIANG et al., 2022):

a) Processo de treinamento é rápido;

b) Conta com modelos pré-treinados, os quais podem ser carregados para serem utilizados na melhoria do modelo que seria gerado utilizando apenas o conjunto de treino;

c) Classificador de alta resolução, capaz de adaptar-se e mudar a quantidade de *pixels* utilizados quando percebe que trata-se de um modelo de classificação;

d) Além de classificar, também pode ser treinado para detecção de objetos em tempo real;

e) Criação de métricas de avaliação de desempenho de maneira automática: Com YOLO, para cada treino são geradas métricas e gráficos do desempenho do modelo.

Para a criação do modelo de classificação, foi estabelecida uma regra binária: as batatas que apresentassem características visuais de uma batata não saudável, baseada em sua cor, textura e formato, seriam classificadas como impróprias para consumo, enquanto as demais seriam consideradas próprias. O modelo de detecção e classificação foi desenvolvido com o algoritmo YOLO, utilizando a biblioteca *ultralytics* e a linguagem de programação *Python* na versão 3.10. A configuração de treinamento foi ajustada para 50 épocas e incluía dois rótulos principais: "*healthy-potatoes*" para as batatas saudáveis e "*unhealthy-potatoes*" para as doentes.

Após a etapa de coleta e pré-processamento dos dados, definiu-se um arquivo de configuração em formato YAML [5], nesse caso chamado de config.yaml. Esse arquivo determina: o caminho para o diretório raiz do conjunto de dados, qual a pasta com os dados de treino, qual a pasta com os dados de validação; e especifica as classes de saída.

O código para o treinamento inicial configura o modelo para utilizar a GPU [6], caso disponível. Caso contrário, será utilizada a CPU[7]. Em seguida, é instanciado o modelo e em qual unidade de processamento será treinado o modelo.

Durante o desenvolvimento do modelo, ajustou-se a quantidade de épocas com o objetivo de encontrar um equilíbrio entre o tempo de treinamento e o consumo de recursos computacionais. Para isso, foram realizados testes variando o número de épocas e analisando o impacto no desempenho da classificação. A escolha do valor final levou em consideração a taxa de revocação obtida em diferentes treinamentos, garantindo que o modelo alcançasse um bom desempenho sem comprometer a eficiência computacional. Após essa análise comparativa, definiu-se que 50 épocas representavam o melhor compromisso entre tempo de processamento e qualidade dos resultados.

Por fim, é chamado o método de treinamento, que recebe como parâmetros o caminho até o arquivo config.yaml e a quantidade de épocas [8] do treino, conforme o código abaixo:

```
from ultralytics import YOLO
import torch
device: str = 'cuda' if torch.cuda.is_available() else 'cpu'
model = YOLO("yolov8n.yaml").to(device)
results = model.train(
    data="/path/to/config.yaml",
    epochs=50
)
```

Ao final do treinamento é gerado um arquivo binário chamado best.pt representando o modelo otimizado.

[5] Linguagem de serialização de dados compatível com todas as linguagens de programação (YAML, 2024)

[6] GPU é a sigla para *Graphics Processing Unit*, utilizada em tarefas como renderização de vídeo, por exemplo (INTEL, 2024).

[7] A *Central Processing Unit*, ou CPU, é a unidade que executa os comandos necessários para o computador e sistema operacional (INTEL, 2024)

[8] Valor que indica a quantidade de vezes que o algoritmo de aprendizado de máquina percorrerá o conjunto de dados de treinamento (BROWNLEE, 2018)

3.6 Teste em ambiente simulado

Para a fase de testes, foi utilizou-se a biblioteca *OpenCV*, com um *script* escrito na linguagem *Python* 3.10, que realiza a captura em tempo real das imagens, segmentação e classificação das batatas. Esse processo inclui as seguintes etapas:

a) O modelo best.pt é carregado utilizando a biblioteca *ultralytics*;

b) A câmera captura quadros em tempo real, que são processados para a detecção de batatas;

c) Para cada batata detectada, a caixa delimitadora é desenhada no quadro, indicando a classificação (saudável ou não saudável) caso o nível de confiança ultrapasse 80%;

d) Um contador rastreia três detecções consecutivas de uma mesma batata com um nível de confiança superior ao limiar. Quando isso ocorre, uma solicitação HTTP é enviada para o *backend*, enviando a classificação ("verdadeiro" para batatas saudáveis e "falso" para batatas não saudáveis).

O código implementa essa lógica com o uso de funções da *OpenCV* para desenhar as caixas delimitadoras e exibir os resultados, além de permitir a interrupção do teste ao pressionar uma tecla específica. Com essa abordagem, o modelo foi validado em condições simuladas, garantindo uma análise detalhada da precisão de detecção e classificação das batatas em tempo real.

3.7 Elaboração e Construção do Protótipo do Sistema

Após a etapa de coleta e treino do modelo, iniciou-se a elaboração e construção do protótipo do sistema embarcado.

O ponto de partida foi a aquisição da esteira transportadora, local que servirá como base para todo o processo de identificação, classificação e separação das batatas. As dimensões da esteira são 57x9x9 centímetros. A velocidade da esteira varia enter 8 a 18 metros por minuto. O elemento transportador possui as dimensões 52x8 centímetros.

A esteira transportadora conta com uma correia com base aderente afim de transportar objetos de maneira eficiente sem escorregar ou cair. A correia é montada sobre dois roletes, um em cada ponta da esteira com uma fina plataforma entre eles (Figura 9), tudo para facilitar o movimento da correia.

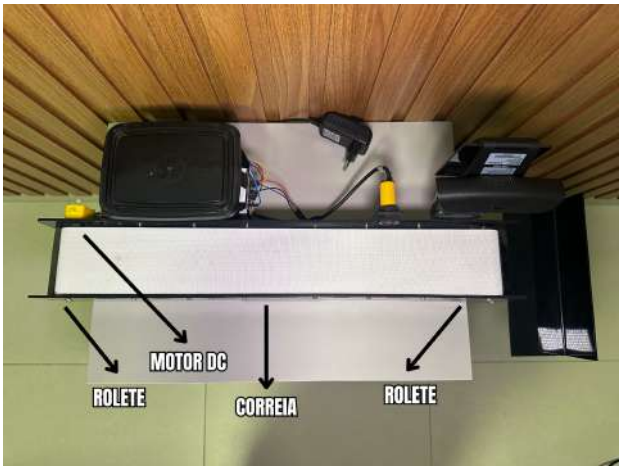
Para movimentar a esteira, fez-se o uso de um motor DC 3V-6V, conhecido por sua simplicidade de controle e capacidade de fornecer torque suficiente para mover a correia carregada. O motor DC foi acoplado à estrutura da esteira de forma a garantir

um alinhamento preciso (Figura 9), com o objetivo de evitar possíveis desalinhamentos e comprometeria o funcionamento do sistema.

O controle do motor DC é realizado por meio de uma ponte H modelo L298N, no protótipo foi instalada dentro da caixa de componentes (Figura 9). A ponte H é um circuito eletrônico que permite controlar a direção e a velocidade do motor, operação crucial para ajustar a velocidade da esteira de acordo com as necessidades do processo de classificação.

A integração do motor DC com a ponte H foi cuidadosamente projetada para garantir uma operação estável e confiável. Desse modo, foram realizados testes para calibrar a resposta do motor em relação aos tempo de resposta do sensores e tempo de resposta dos atuadores que estavam acoplados na esteira. O microcontrolador, o qual foi conectado a ponte H, é responsável por gerenciar este controle de velocidade, parada e sentido de rolamento da esteira.

Figura 9 – Visão Superior do Protótipo



Fonte: De autoria própria.

Após a aquisição da esteira, acoplou-se o sensor infravermelho modelo E18-D80NK, posicionado estrategicamente um pouco antes do final da esteira (Figura 10). A escolha deste ponto específico para o sensor foi de extrema importância afim de garantir que ele detectasse a presença de um objeto à medida em que aproximava-se do final do percurso.

O sensor infravermelho E18-D80NK é conhecido por sua precisão e confiabilidade na detecção de objetos, tornando-o ideal para este tipo de aplicação. Ele emite um feixe de luz infravermelha que, ao ser refletido por um objeto, é detectado pelo próprio sensor, indicando a presença do item. Esta tecnologia permite que o sensor identifique rapidamente quando uma batata está prestes a chegar ao final da esteira.

Assim que o sensor detecta a presença de um item, ele envia um sinal para o microcontrolador do sistema, que imediatamente para o movimento da esteira. Esta parada

precisa é essencial para que o sistema de classificação possa entrar em ação.

Com a esteira parada, é importante falar sobre o próximo passo na elaboração e construção do sistema embarcado. Para possibilitar a classificação por meio da visão computacional, conectou-se após sensor e no fim da esteira, uma câmera com um suporte posicionada acima para capturar imagens claras e estáveis do produto parado em frente ao sensor infravermelho (Figura 10), possibilitando que o sistema de visão computacional analisasse suas características e retorne a classificação saudável ou não saudável.

Apesar de estar fisicamente unida a esteira, a câmera está conectada via USB a uma máquina local que executa o programa de captura de imagem, desenvolvido com a linguagem *Python*. Este *script* utiliza a biblioteca *OpenCV* para capturar as imagens da câmera em tempo real.

Uma vez capturada a imagem, o modelo gerado no treinamento é instanciado para realizar a classificação do objeto detectado, atribuindo-o uma classe com base em suas características visuais. Após a classificação, o resultado juntamente com a data e hora da classificação são enviado ao *backend*. Esse processo é essencial para armazenar os dados no banco de dados, formando um histórico de telemetrias que pode ser utilizado para análises futuras e otimização do sistema.

Além disso, o mesmo algoritmo é responsável por enviar o resultado da classificação ao microcontrolador via protocolo MQTT, por meio da biblioteca *Paho*[9]. Essa comunicação é vital para que o microcontrolador possa acionar o servo motor e separar o produto de acordo com sua classe, direcionando-a para o recipiente de coleta apropriado.

Figura 10 – Visão Lateral do Protótipo

Fonte: De autoria própria.

Após o microcontrolador receber o resultado da classificação, um componente crítico

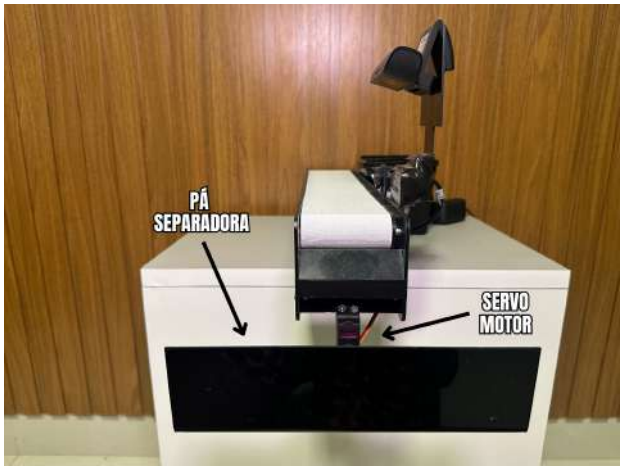
[9] A biblioteca *Paho* é uma implementação do cliente MQTT para *Python*. Ela fornece uma interface simples e eficiente para conectar, publicar e assinar tópicos em um *broker* MQTT, facilitando a comunicação em redes de Internet das Coisas (FOUNDATION, 2023).

do sistema embarcado entra em ação: a pá separadora. A pá (Figura 11) foi cuidadosamente projetada para ser acoplada ao final da esteira, permitindo a separação eficiente.

Para controlar a pá separadora, fez-se uso de um servo motor modelo MG995 (Figura 11), conhecido por sua precisão e torque adequados para aplicações que exigem movimentos controlados e repetitivos. O servo motor é programado para responder à classificação recebida pelo microcontrolador.

Quando aplicado ao caso de uso para separação de batatas, se a classe recebida indicar que a ela é saudável, o servo motor MG995 rotaciona a pá para a esquerda, direcionando a batata para o recipiente designado para batatas saudáveis. Por outro lado, se a classe recebida for não saudável, o servo motor rotaciona a pá para a direita, enviando a batata para o recipiente de batatas não saudáveis.

Figura 11 – Visão Frontal do Protótipo



Fonte: De autoria própria.

Para controlar e receber o sinal de todos esses sensores e atuadores, foi utilizado o microcontrolador ESP-32, localizado dentro da caixa com componentes (Figura 10). Este microcontrolador foi escolhido principalmente por sua capacidade de se conectar à internet, o que é essencial para a comunicação via protocolo MQTT e para a integração com o sistema de *backend*.

O ESP-32 gerencia eficientemente o controle da velocidade, parada e direção da esteira, além de processar os sinais dos sensores e acionar os atuadores, como o servo motor da pá separadora. Sua conectividade à internet permite uma comunicação fluida e em tempo real com o sistema de visão computacional e o *backend*, garantindo que o processo de classificação e separação das batatas seja realizado de forma precisa e eficiente. A escolha do ESP-32 como núcleo do sistema embarcado assegura uma operação robusta e adaptável, capaz de atender às demandas do processo automatizado de classificação.

O esquemático elétrico (Figura 12) ilustra a integração de componentes do sistema.

Esse esquema integra a parte elétrica do sistema necessária para automatizar o processo de classificação, viabilizando o controle preciso e comunicação eficiente entre *hardware* e *software*, proporcionando a todos os elementos do sistema operem de maneira sincronizada e funcional.

Nesse contexto, o ESP-32 desempenha o papel de gerenciar a lógica de controle do sistema e coordena a comunicação com os demais componentes.

Para controlar o motor DC da esteira, utilizou-se a Ponte H modelo L298N por fornecer controle bidirecional do motor, ajuste de velocidade e direção de acordo com os sinais enviados pelo ESP-32. Esse motor, por sua vez, é responsável por movimentar a esteira e garantir o fluxo contínuo na esteira transportadora.

Adicionalmente, o Servo Motor MG995 é utilizado para acionar a pá separadora, que direciona os produtos para os recipientes apropriados com base na classificação realizada. O servo motor é controlado por sinais PWM enviados pelo ESP-32, permitindo ajustes precisos em sua posição. O sistema também conta com o Sensor Infravermelho E18-D80NK, que detecta a presença de objetos na esteira transportadora. Ao identificar uma batata, o sensor envia um sinal para o ESP-32, que para a esteira e inicia o processo de classificação.

Para garantir o funcionamento adequado de todos os componentes, o circuito deve ser alimentado por fontes entre 5V e 12V, fornecendo as tensões necessárias para a operação eficiente e estável do sistema.

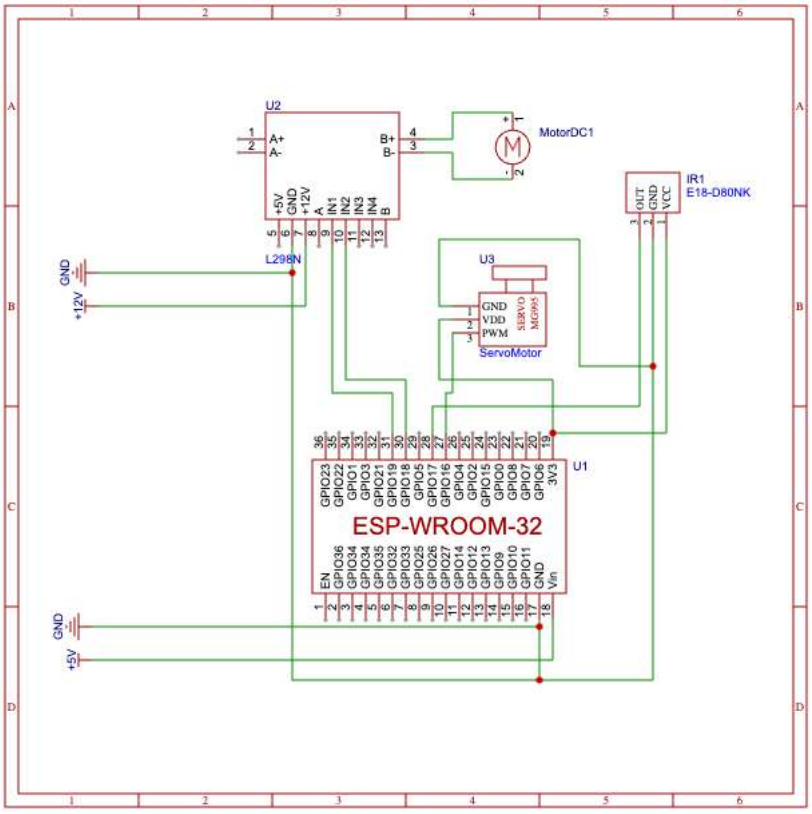
3.8 Arquitetura e Integração do Sistema

A arquitetura (Figura 13) detalha o sistema desenvolvido nesta pesquisa e a integração das diversas tecnologias e componentes para alcançar uma solução eficiente e robusta. Este sistema foi projetado para otimizar o processo de classificação e separação de forma automatizada, utilizando uma combinação de *hardware* e *software*.

Nesse sentido, o *MongoDB* é utilizado para armazenar informações acerca dos resultados da classificação e permite as operações de leitura e escrita para manter um histórico detalhado das classificações realizadas. O *backend* gerencia toda a lógica do servidor, conectando-se ao *MongoDB* para manipular os dados e utilizando *WebSocket*, por meio da biblioteca *Socket.io*, para enviar atualizações em tempo real para o *frontend*. Este, por sua vez, fornece um painel de controle que exibe a quantidade de batatas saudáveis e não saudáveis, bem como o registro de data e hora da última classificação, recebendo dados diretamente do *backend*.

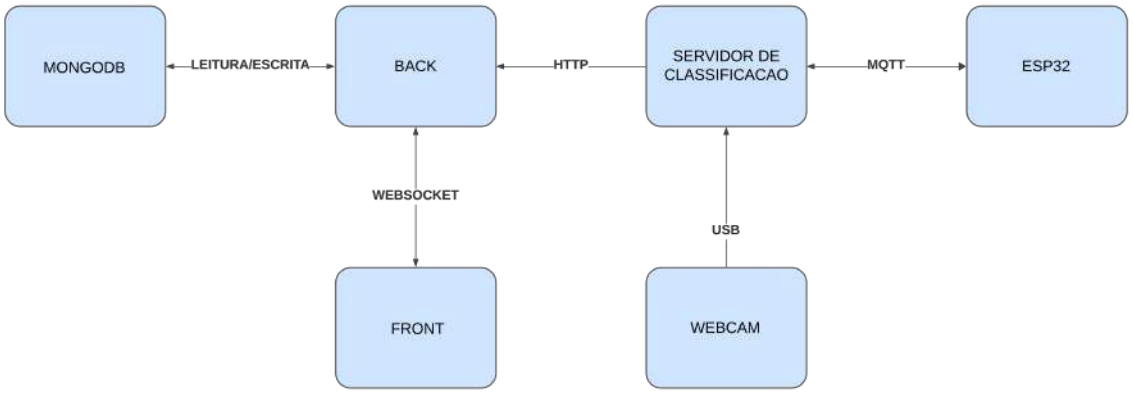
O microcontrolador ESP-32 desempenha um papel essencial no gerenciamento da esteira transportadora, dos sensores e dos atuadores, identificando a presença de elementos ao final da esteira e solicitando o respectivo rótulo ao servidor de classificação através do

Figura 12 – Esquemático Elétrico do Sistema



Fonte: De autoria própria.

Figura 13 – Arquitetura do Projeto



Fonte: De autoria própria.

protocolo MQTT. Esse servidor processa as imagens capturadas pela câmera utilizando o modelo de visão computacional gerado com YOLO, determinando a classe do objeto classificado e enviando os resultados ao ESP-32 por intermédio do protocolo MQTT. Ademais, o servidor de classificação comunica-se com o *backend* via HTTP, transmitindo dados de telemetria, como a classe resultante, data e hora da classificação da última classificação. A câmera também é responsável pela captura das imagens dos elementos

enquanto estes percorrem a esteira, enviando-os diretamente ao servidor de classificação via conexão USB.

A comunicação entre os diferentes componentes é realizada de forma eficiente, utilizando HTTP para a integração entre o *backend*, o servidor de classificação, *WebSocket* para a troca de informações em tempo real entre o *backend* e o *frontend* e MQTT para a interação leve e rápida entre o ESP-32 e o servidor de classificação. Essa arquitetura foi projetada para ser escalável e eficiente ao propiciar a classificação automatizada em tempo real, com atualizações instantâneas no painel de controle.

Com base nas etapas descritas, a próxima seção deste trabalho apresentará os resultados obtidos a partir dos testes realizados, os quais visam avaliar a precisão do modelo de classificação e a eficiência do sistema proposto.

4 Resultados

Este capítulo apresenta os resultados obtidos com o desenvolvimento do protótipo de classificação e separação automatizada. O foco será na análise do desempenho do sistema em termos de precisão e eficiência, evidenciando como a integração de tecnologias de visão computacional e automação contribuiu para o sucesso do projeto. Serão discutidos os desafios enfrentados durante o desenvolvimento, proporcionando uma visão detalhada das melhorias alcançadas. A seção também abordará a eficácia do sistema em um ambiente controlado, destacando sua capacidade de operar de forma autônoma e precisa, sem a necessidade de intervenção manual.

Realizou-se testes manuais sob duas condições de iluminação: ambiente natural e iluminação forçada por LED. Como caso de uso, analisou-se 1000 amostras de batatas, divididas em 500 saudáveis e 500 não saudáveis. Essas amostras foram avaliadas por meio de testes unitários realizados exclusivamente via software, para verificar a precisão do algoritmo de classificação, e também por meio de testes de integração, nos quais o algoritmo de visão computacional foi combinado com a esteira automatizada, garantindo a sincronização entre a detecção das batatas e o mecanismo de separação.

No cenário com iluminação ambiente, o sistema apresentou uma taxa de acerto de 74% para batatas saudáveis, abaixo do que foi proposto nos requisitos funcionais. Para batatas não saudáveis a taxa de acerto foi de 100%.

Com a implementação da iluminação LED, observou-se uma melhoria nos resultados, alcançando 82% na taxa de acerto para batatas saudáveis, enquanto manteve os 100% de taxa de acerto para batatas não saudáveis.

Apesar da melhoria nos resultados, com a implementação da iluminação LED, a precisão da classificação ainda depende do posicionamento adequado das batatas na esteira. A identificação correta de batatas não saudáveis ocorre apenas quando a região comprometida está visível para a câmera no momento da captura da imagem. Caso contrário, o sistema pode classificar erroneamente uma batata não saudável como saudável, visto que a câmera não detectará os sinais de degradação. Esse fator evidencia a influência da disposição do objeto no campo de visão da câmera na qualidade da classificação.

Estes resultados indicam que, embora o sistema aparente ser confiável para identificar produtos inadequados, ainda há espaço para aprimoramento na redução de falsos negativos na classificação.

Na seção a seguir, são apresentados e analisados os resultados obtidos do modelo de classificação desenvolvido para o protótipo de separação automática. Conforme citado na Seção 3.4, os dados de treinamento são referentes a um conjunto de imagens de batatas,

utilizado para construção de um modelo de classificação de batatas saudáveis e não saudáveis.

4.1 Da avaliação de desempenho do modelo

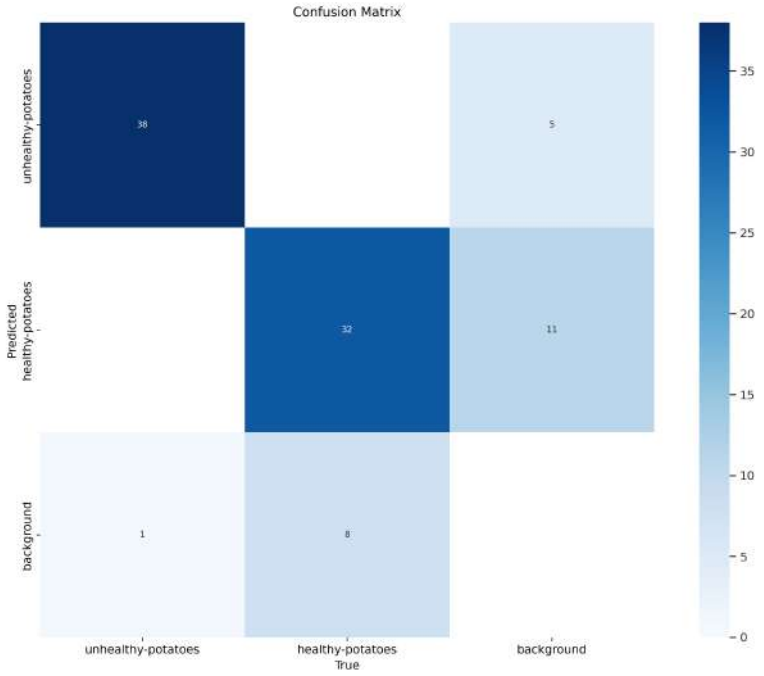
A avaliação inclui métricas fundamentais como precisão, revocação, curva de precisão-revocação, matriz de confusão e média harmônica. Essas métricas permitem verificar a eficiência do modelo na identificação e classificação do caso de uso com batatas, classificando-as como próprias ou impróprias para consumo. Além disso, os gráficos gerados auxiliam na interpretação do desempenho do modelo sob diferentes limiares de decisão, fornecendo uma visão detalhada sobre sua adequação ao contexto do projeto e sua viabilidade prática.

A análise do desempenho do modelo evidencia um progresso consistente no que se refere à precisão e à revocação ao longo do treinamento. A matriz de confusão (Figura 14) indica que o modelo foi capaz de classificar corretamente 38 batatas identificadas como não saudáveis, enquanto apenas 5 foram incorretamente classificadas como saudáveis. No entanto, constatou-se maior dificuldade na identificação de batatas saudáveis, uma vez que 32 foram classificadas de forma equivocada. Esse desequilíbrio pode estar relacionado a limitações na base de dados utilizada para o treinamento ou a características específicas das imagens que dificultaram o aprendizado do modelo.

Além disso, ao analisar os erros relacionados ao *background* na matriz de confusão, observa-se que houve um caso de *Background* FP, em que um objeto do fundo foi erroneamente detectado como uma batata não saudável. Esse tipo de erro pode ocorrer quando elementos externos à classe de interesse apresentam padrões visuais semelhantes aos aprendidos pelo modelo, levando a falsas detecções. Já no caso de *Background* FN, nota-se que 8 objetos que pertenciam à classe de fundo foram classificados incorretamente como batatas saudáveis, o que indica que o modelo pode estar confundindo elementos do ambiente com os produtos analisados. Esse fenômeno pode comprometer a robustez do sistema em um cenário real, onde variações no ambiente podem gerar interferências na classificação.

Os gráficos de perda e métricas (Figura 15) reforçam que o modelo obteve uma redução consistente nas perdas durante o treinamento e a validação, indicando estabilidade no aprendizado e uma generalização moderada. Os valores de precisão média (mAP@50) e de revocação demonstraram evolução ao longo das iterações, embora a oscilação inicial sugira desafios na estabilização do modelo durante as etapas iniciais do processo de treinamento. A curva de confiança F1 (Figura 16), reforça que o modelo alcança seu melhor desempenho em níveis médios de confiança (0,382), com pontuação geral de 0,84. Isso demonstra uma capacidade equilibrada entre precisão e revocação, especialmente para a

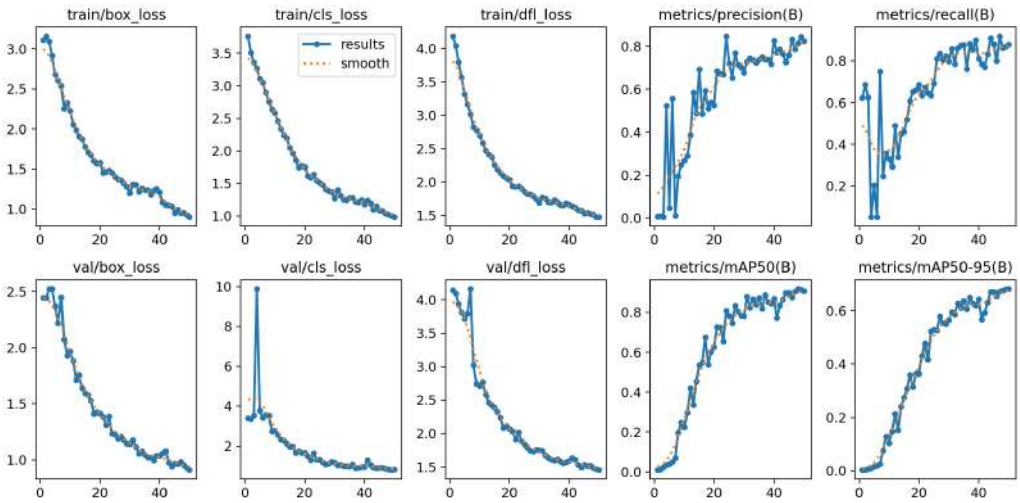
Figura 14 – Matriz de Confusão



Fonte: Dados da pesquisa.

classe "não saudável". Ainda assim, o modelo poderia ser ajustado para reduzir os falsos positivos na classe "saudável" e melhorar a separação entre as classes. No entanto, isso não foi possível devido ao tempo e custo na aquisição de imagens sob outras perspectivas, tamanhos, luz, cor, entre outros aspectos.

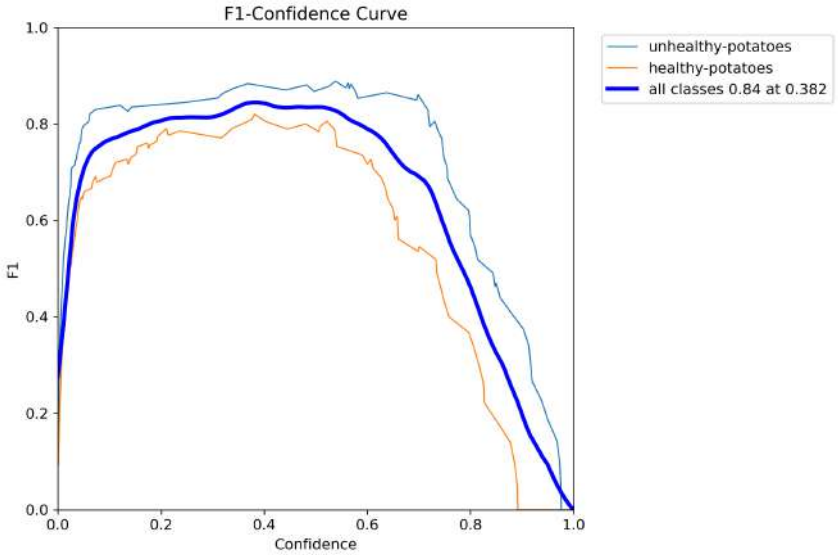
Figura 15 – Resultados de Perda e Treino



Fonte: Dados da pesquisa.

A análise da curva de precisão (Figura 17) demonstra um comportamento consistente em limiares mais altos, indicando que o modelo é capaz de realizar previsões positivas

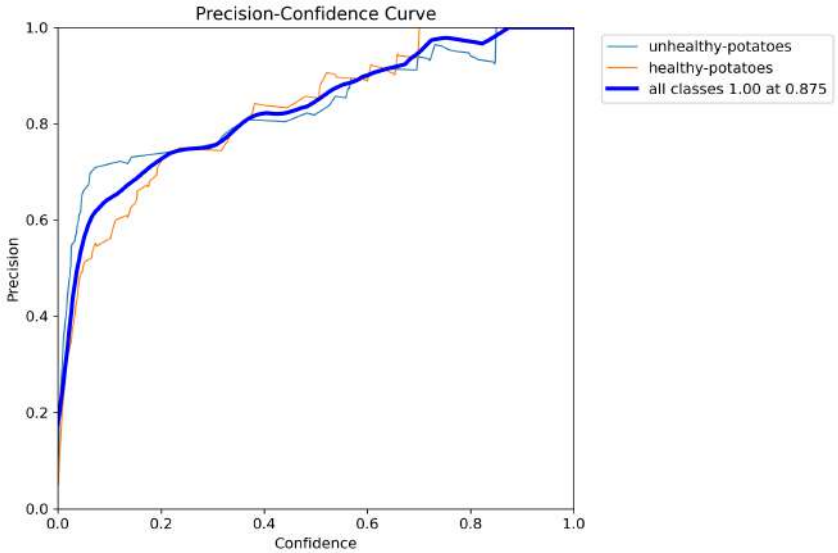
Figura 16 – Curva F1-*Score*



Fonte: Dados da pesquisa.

com elevado grau de confiança. Essa característica é essencial no contexto do protótipo desenvolvido, pois a classificação incorreta de batatas saudáveis como impróprias (falsos positivos) pode reduzir a eficiência do sistema e aumentar o desperdício. No entanto, é perceptível que, em limiares mais baixos, a precisão do modelo apresenta uma queda, sugerindo uma maior ocorrência de falsos positivos nessas condições. Apesar disso, o desempenho geral do modelo na precisão se mostra adequado para a aplicação proposta, sobretudo em limiares intermediários e altos, onde o equilíbrio entre precisão e revocação é mais evidente.

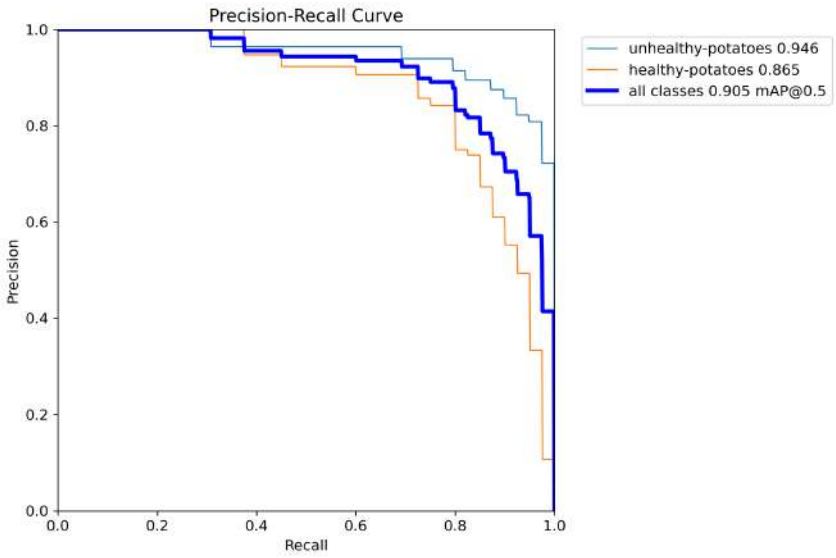
Figura 17 – Curva de Precisão



Fonte: Dados da pesquisa.

A curva de precisão-revocação gerada (Figura 18), revela uma performance sólida, com uma área sob a curva (AUC-PR) significativa. O modelo conseguiu manter um equilíbrio satisfatório entre precisão e revocação, mesmo diante de um conjunto de dados potencialmente desbalanceado, onde a classe de batatas impróprias para consumo pode ser menos representada. Esse resultado indica que o modelo é eficiente em identificar batatas defeituosas sem comprometer excessivamente a precisão. No entanto, observa-se que, em alguns pontos da curva, há uma leve queda na precisão quando a revocação aumenta, algo esperado devido à natureza conflitante dessas métricas. De forma geral, o desempenho reforça a viabilidade do modelo para o cenário proposto, especialmente considerando a importância de equilibrar a identificação de batatas impróprias sem gerar excessos de falsos positivos.

Figura 18 – Curva de Precisão-Revocação

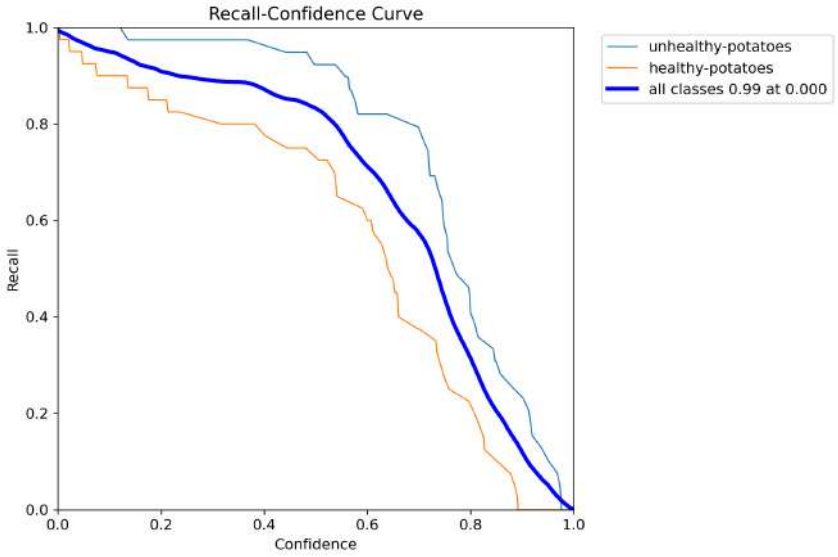


Fonte: Dados da pesquisa.

Por fim, a análise da curva de revocação (Figura 19) destaca que o modelo é robusto na identificação de batatas impróprias para consumo, apresentando altos valores de revocação para a maior parte dos limiares testados. Apesar de uma ligeira redução na revocação em limiares mais rigorosos, o equilíbrio observado na curva de precisão-revocação assegura que essa redução não comprometa drasticamente o desempenho geral. Essa consistência e confiabilidade demonstram que o modelo está bem adaptado para o propósito de classificação e separação automatizada no protótipo desenvolvido.

Resumidamente o modelo de classificação de batatas, desenvolvido para o protótipo de separação automática, apresentou resultados promissores, demonstrando equilíbrio entre precisão e revocação, estabilidade no aprendizado e uma capacidade robusta de generalização. Embora desafios relacionados à classificação de batatas saudáveis tenham sido identificados, os dados indicam que o modelo é eficiente para o cenário proposto, com

Figura 19 – Curva de Revocação



Fonte: Dados da pesquisa.

potencial para otimização futura.

O desempenho geral reforça sua viabilidade prática, contribuindo significativamente para o objetivo de automatizar o processo de classificação e separação com precisão e eficiência.

Encerrada a análise sobre a avaliação do modelo, direciona-se agora o foco para o protótipo de sistema embarcado, abordando sua arquitetura, funcionamento e integração com os demais componentes desenvolvidos.

4.2 Do protótipo de sistema embarcado

A arquitetura do sistema, que combina *hardware* e *software* de forma integrada, foi fundamental para o sucesso do projeto. A escolha do microcontrolador ESP-32, juntamente com a implementação de um protocolo de comunicação MQTT, permitiu uma interação rápida e confiável entre os componentes do sistema. O uso de uma câmera para capturar imagens de objetos e a subsequente análise por meio do modelo de visão computacional garantiram que o caso de uso mencionado fossem classificados de forma precisa antes de serem separadas pela pá separadora.

Além disso, o *design* da pá para separação mostrou-se eficiente e resistente, lidando com diferentes pesos de batatas. Junto a isso, soma-se o sensor de luz infra-vermelho, que de maneira quase imediata, identifica a presença de objetos na esteira.

Essa abordagem não só melhorou a precisão da classificação, mas também reduziu a necessidade de intervenção manual, aumentando a confiabilidade do sistema.

4.3 Da classificação e separação

O sistema embarcado desenvolvido para a classificação e separação de produtos demonstrou robustez e precisão em sua implementação. Utilizando a metodologia descrita anteriormente, o sistema foi capaz de identificar e classificar corretamente o caso de uso de batatas saudáveis e não saudáveis com uma precisão esperada, conforme os requisitos estabelecidos. A integração do algoritmo YOLO para visão computacional permitiu uma análise detalhada das características visuais das batatas, como cor e textura, garantindo que o processo de classificação fosse realizado em tempo real, com tempos mínimos de parada da esteira.

Por fim, a implementação de um painel de controle em tempo real, desenvolvido com React e Socket.IO, proporcionou uma interface de usuário intuitiva e informativa. Os operadores puderam monitorar o desempenho do sistema e visualizar a quantidade de batatas classificadas como saudáveis ou não saudáveis, além de receber atualizações instantâneas sobre a última classificação realizada. Essa capacidade de monitoramento em tempo real melhorou bastante o controle e a usabilidade do sistema.

5 Considerações Finais

Este trabalho teve como objetivo o desenvolvimento de um protótipo voltado à automação de processos, utilizando visão computacional e inteligência artificial para a classificação binária de produtos e sua separação. A proposta busca atender às necessidades do setor de automação, ao oferecer uma solução eficiente e escalável que contribua para a redução de esforços manuais, a otimização de processos e a diminuição de perdas.

O desenvolvimento do trabalho foi realizado no contexto do uso de visão computacional na detecção e classificação de coisas, práticas de aprendizado de máquina e integração entre diferentes sensores e atuadores para a elaboração de um sistema embarcado. Além disso, foram conduzidos testes experimentais em ambiente controlado, visando medir a eficiência do modelo, bem como das capacidades físicas dos controladores.

5.1 Avaliação do sistema de automação

O sistema integra componentes de *hardware* e *software* de forma eficiente, utilizando arquitetura baseada no microcontrolador ESP-32, sensores de movimento, câmeras e um servidor *backend* em *Node.js*, que gerencia os dados em tempo real e os armazena em um banco de dados *MongoDB*. A interface *frontend* foi desenvolvida para apresentar uma visualização clara e dinâmica dos dados, permitindo que o usuário acompanhe a operação do sistema em tempo real.

A integração entre as diferentes áreas da tecnologia, a visão computacional e sistemas embarcados, permitiu a criação de um protótipo que não apenas melhora a eficiência operacional, mas também garante a entrega de produtos de alta qualidade ao consumidor final.

O uso do algoritmo YOLO para a classificação binária, aliado ao desempenho da esteira automatizada e dos sensores no transporte e detecção de objetos, juntamente com microcontrolador ESP-32 e à comunicação via MQTT, resultou em um sistema robusto e confiável para a separação física dos objetos dado sua classificação, capaz de operar em tempo real com alta precisão.

A implementação de um painel de controle em tempo real, utilizando *React* e *Socket.IO*, proporcionou uma interface de usuário intuitiva, permitindo que os operadores monitorassem o desempenho do sistema de forma eficaz.

5.2 Desafios Enfrentados no Desenvolvimento do Protótipo

O desenvolvimento do sistema automatizado de classificação e separação apresentou diversos desafios ao longo de sua execução, tanto no âmbito da coleta de dados quanto no aspecto técnico. Esses desafios demandaram soluções criativas e adaptações ao planejamento inicial, contribuindo para o aprimoramento do projeto.

Um dos principais obstáculos enfrentados foi a dificuldade em encontrar largas bases de dados já catalogadas que atendessem aos requisitos específicos do projeto. Tal lacuna obrigou a reduzir o escopo de possíveis construção de modelos para caso de uso. Por essa razão, buscou-se dados mais simples de serem encontrados. Falta de dados já anotados, o que demandou anotações manuais em todo o conjunto de dados, empregando-se tempo e esforço significativo. Para contornar a limitação da quantidade de imagens disponíveis, foi necessária a aplicação de técnicas de aumento de dado. Essas técnicas, como rotação, espelhamento e alterações de brilho, permitiram ampliar a diversidade do conjunto de dados e melhorar a capacidade de generalização do modelo de classificação, no entanto, a quantidade de imagens gerados por meio dessa técnica não foi suficientes para que o modelo alcançasse sua melhor performance.

No aspecto computacional, um desafio inicial foi a ausência de um computador adequado para o treinamento dos modelos de aprendizado profundo. Modelos de visão computacional, como o utilizado, exigem alto poder computacional para processar as imagens e ajustar os parâmetros das redes neurais. Para contornar esse problema, fez-se uso de serviços em nuvem que disponibilizavam GPUs, como o ambiente do Kaggle e GoogleColab, o que possibilitou o treinamento dos primeiros modelos sem custos adicionais de infraestrutura. Posteriormente, com a aquisição de uma computador com uma placa de vídeo dedicada, o processo de treinamento tornou-se mais eficiente e independente.

Outro desafio técnico relevante foi a escolha do servo motor responsável por rotacionar a pá de separação das batatas. Durante os testes iniciais, verificou-se que o uso de um servo motor comum não era suficiente para realizar a tarefa com eficiência, especialmente devido ao peso e à força necessários para movimentar o mecanismo. Por isso, foi necessário adquirir o modelo MG995, que apresenta maior torque e potência, garantindo o funcionamento adequado do sistema.

Esses desafios ilustram a complexidade envolvida no desenvolvimento de um sistema automatizado e integrado. Apesar das dificuldades, as soluções implementadas contribuíram para a robustez do protótipo, garantindo sua funcionalidade e alinhando-se aos objetivos propostos.

5.3 Lições Aprendidas

A elaboração do protótipo proporcionou um aprofundamento significativo nos conhecimentos em sistemas embarcados, envolvendo o desenvolvimento de dispositivos eletrônicos por meio do uso de microcontroladores, sensores e atuadores, bem como na área de inteligência artificial, com o desenvolvimento de diferentes modelos, exploração de arquiteturas variadas e estudos sobre métricas de avaliação.

Durante o processo, destacou-se a importância da integração de tecnologias, como visão computacional, sistemas embarcados e comunicação em tempo real, como fator crucial para o sucesso de sistemas automatizados complexos. Além disso, o treinamento de modelos de visão computacional revelou-se um desafio significativo, exigindo conjuntos de dados diversificados e bem anotados para alcançar alta precisão e generalização.

Por fim, evidenciou-se a necessidade de projetar sistemas automatizados com flexibilidade, permitindo sua adaptação a diferentes condições operacionais e requisitos de mercado.

5.4 Trabalhos Futuros

Para aprimorar este trabalho, considerando as lições aprendidas e as limitações identificadas, algumas direções promissoras incluem a ampliação da base de dados de modo a melhorar a precisão e a generalização do modelo de classificação. Além disso, é relevante explorar arquiteturas mais avançadas de redes neurais ou técnicas alternativas de aprendizado de máquina, com o objetivo de aumentar a eficiência e a acurácia do sistema.

Outra vertente relevante deste trabalho é a capacidade de generalização do protótipo, de modo a possibilitar a seleção de diferentes modelos de classificação para distintos casos de uso. Dessa forma, o protótipo da esteira não ficaria restrito a um único modelo, permitindo ao usuário a escolha de um modelo adequado por meio de uma interface interativa na aplicação de rede.

Por fim, importante é o desenvolvimento de soluções que possibilitem a escalabilidade do sistema para operações em larga escala. Isso inclui a otimização do hardware para tornar mais eficiente o transporte, a detecção e a separação, bem como a implementação de soluções em nuvem para processamento de dados, permitindo a computação distribuída e maior capacidade de processamento.

Referências

AALST, W. M. Van der et al. **Process mining: a two-step approach to balance between underfitting and overfitting**. *Software & Systems Modeling*, Springer, v. 9, p. 87–111, 2010.

ALVI, F. ***Data Annotation – A Beginner's Guide***. 2024. Disponível em: https://opencv.org/blog/data-annotation/. Acesso em: 29 out. 2024.

ARDUINO. ***Arduino Libraries***. 2020. Disponível em: https://docs.arduino.cc/libraries. Acesso em: 21 jan. 2025.

BANKS ED BRIGGS, K. B. R. G. A. **MQTT Version 5.0**. *OASIS Standard*, 2019.

BARELLI, F. **Introdução à visão computacional: Uma abordagem prática com Python e OpenCV**. Editora Casa do Código, 2018.

BOLTON, W. ***Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering***. 6. ed. Harlow: Pearson, 2015.

BRANDT, M. L. C.; RENKEN, A. L. F.; LEITE, M. **LIXEIRA INTELIGENTE: Desenvolvimento de um dispositivo separador de materiais recicláveis**. *Anais da Mostra Nacional de Iniciação Científica e Tecnológica Interdisciplinar (MICTI)-e-ISSN 2316-7165*, v. 1, n. 12, 2019.

BRASIL. 2014. Lei n° 12.965, de 23 de abril de 2014. Estabelece princípios, garantias, direitos e deveres para o uso da Internet no Brasil. **Lei Geral de Proteção de Dados Pessoais (LGPD)**, 2014. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2011-2014/2014/lei/l12965.htm. Acesso em: 21 jan. 2025.

BROWNLEE, J. **What is the Difference Between a Batch and an Epoch in a Neural Network**. *Machine learning mastery*, v. 20, p. 1–5, 2018.

BRUGNARI, A.; MAESTRELLI, L. H. M. ***Automação Residencial via Web***. 2010. 36f. Trabalho de conclusão de curso – Curso de Graduação em Engenharia de Computação – PUC-PR, Curitiba, 2010.

CARDOSO, M. ***O Que É Um Microcontrolador?*** 2020. Disponível em: https://edu.ieee.org/br-ufcgras/o-que-e-um-microcontrolador/. Acesso em: 21 jan. 2025.

CATTO, M. G. A. R. S. d. B. ***Garantia de qualidade em produtos: estudo de caso sobre a gestão de reclamações de qualidade para sucos de laranja de origem brasileira***. 90f p. Dissertação (Dissertação (Mestrado em Administração)) — Universidade Estadual Paulista "Julio de Mesquita Filho", Jaboticabal, 2024.

CHASE, O.; ALMEIDA, F. **Sistemas embarcados**. *Mídia Eletrônica*, v. 10, n. 11, p. 3, 2007.

CUNHA, A. F. **O que são sistemas embarcados**. *Saber Eletrônica*, v. 43, n. 414, p. 1–6, 2007.

DIETTERICH, T. **Overfitting and undercomputing in machine learning**. *ACM computing surveys (CSUR)*, ACM New York, NY, USA, v. 27, n. 3, p. 326–327, 1995.

DOCKER, I. ***What is a Container?*** 2023. Disponível em: https://www.docker.com/resources/what-container. Acesso em: 10 out. 2024.

ELMASRI, R.; NAVATHE, S. B. ***Fundamentals of Database Systems***. 7. ed. [S.l.]: Pearson, 2016.

ESPRESSIF. ***ESP32 Series: Datasheet Version 4.8***. 2022. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Acesso em: 26 jan. 2025.

EXPRESS.JS. ***Express - Node.js web application framework***. 2023. Disponível em: https://expressjs.com/. Acesso em: 10 out. 2024.

FARHADI, A.; REDMON, J. Yolov3: An incremental improvement. In: SPRINGER BERLIN/HEIDELBERG, GERMANY. ***Computer vision and pattern recognition***. [S.l.], 2018. v. 1804, p. 1–6.

FARIA, F. T. J. et al. **Classification of potato disease with digital image processing technique: a hybrid deep learning framework**. In: IEEE. *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*. [S.l.], 2023. p. 0820–0826.

FETT, D.; KÜSTERS, R.; SCHMITZ, G. **The WebSocket Protocol**. *RFC 6455*, 2019.

FETTE, I.; MELNIKOV, A. ***The websocket protocol***. 2011. Disponível em: https://www.rfc-editor.org/rfc/rfc6455.html. Acesso em: 26 jan. 2025.

FIELDING, R. T. ***Architectural Styles and the Design of Network-based Software Architectures***. Tese (Doutorado) — University of California, Irvine, 2000. Accessed: 2023-10-10. Disponível em: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm.

FOROUZAN, B. A. ***Data Communications and Networking***. 5. ed. New York: McGraw-Hill Education, 2017.

FOUNDATION, E. ***Eclipse Paho - MQTT and MQTT-SN software***. 2023. Disponível em: https://www.eclipse.org/paho/. Acesso em: 10 de outubro de 2023.

GITHUB, I. ***GitHub***. 2023. Disponível em: https://github.com. Acesso em: 10 de out. de 2024.

GONT, F.; CHOWN, T.; LINKOVA, J. **Internet Protocol, Version 6 (IPv6) Specification**. *RFC 8200*, 2019.

GOURLEY, D.; TOTTY, B. **HTTP: the definitive guide**. 1. ed. califórnia. O'Reilly Media, Inc., 2002. 4 p.

GULLI, A.; PAL, S. ***Deep learning with Keras***. [S.l.]: Packt Publishing Ltd, 2017.

HAT, R. ***What is middleware?*** 2022. Disponível em: https://www.redhat.com/en/topics/middleware/what-is-middleware. Acesso em: 30 out. 2024.

INTEL. ***CPU versus GPU:*** *opções poderosas para suas necessidades de computação*. 2024. Disponível em: https://www.intel.com.br/content/www/br/pt/products/docs/processors/cpu-vs-gpu.html. Acesso em: 27 jan. 2025.

INTEL. ***What Is a GPU?*** 2024. Disponível em: https://www.intel.com/content/www/us/en/products/docs/processors/what-is-a-gpu.html. Acesso em: 29 out. 2024.

JIANG, P. et al. **A Review of Yolo algorithm developments**. *Procedia computer science*, Elsevier, v. 199, p. 1066–1073, 2022.

KEMPER, M. S. L. A. K. R. ***Sensors and Transducers***. 3. ed. Boca Raton: CRC Press, 2018.

KESSLER, G. C. **An overview of TCP/IP protocols and the internet**. *InterNIC Document, Dec*, v. 29, p. 42, 2004.

KHAN, A. et al. **A survey of the recent architectures of deep convolutional neural networks**. *Artificial intelligence review*, Springer, v. 53, p. 5455–5516, 2020.

KHAN, M. A. et al. **A deep learning-based intrusion detection system for MQTT enabled IoT**. *Sensors*, MDPI, v. 21, n. 21, p. 2, 2021.

KIM, S.-H. *Electric motor control: DC, AC, and BLDC motors*. [S.l.]: Elsevier, 2017.

LEVY, H. M.; REDDY, R. **The Design and Implementation of a Distributed Operating System**. *IEEE Transactions on Computers*, v. 34, n. 12, p. 1178–1193, 1985.

MAKER, C. do. ***Pinagem ESP32: Detalhes e Conexões***. 2024. Disponível em: https://clubedomaker.com/esp32-pinout. Acesso em: 10 out. 2024.

MARGOLIS, M. ***Arduino Cookbook***. 3. ed. Sebastopol: O'Reilly Media, 2020.

MCCARTHY, J. et al. **What is artificial intelligence**. Stanford University, 2007.

META, I. ***React – A JavaScript library for building user interfaces***. 2023. Disponível em: https://react.dev/. Acesso em: 10 out. 2024.

MICROSOFT. ***TypeScript: JavaScript With Syntax For Types***. 2023. Disponível em: https://www.typescriptlang.org/. Acesso em: 10 out. 2024.

MOHAN, T. M. U. W. P. R. N. ***Power Electronics: Converters, Applications, and Design***. 3. ed. Nova York: John Wiley & Sons, 2003.

MONARD, M. C.; BARANAUSKAS, J. A. **Conceitos sobre aprendizado de máquina**. *Sistemas inteligentes-Fundamentos e aplicações*, v. 1, n. 1, p. 32, 2003.

MONGODB, I. ***MongoDB: The most popular database for modern apps***. 2023. Disponível em: https://www.mongodb.com/. Acesso em: 10 out. 2024.

MONGOOSE. ***Mongoose - elegant mongodb object modeling for node.js***. 2023. Disponível em: https://mongoosejs.com/. Acesso em: 10 out. 2024.

MOZILLA. ***CSS: Cascading Style Sheets***. 2024. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/CSS. Acesso em: 29 out. 2024.

MOZILLA. ***HTML: Linguagem de Marcação de Hipertexto***. 2024. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTML. Acesso em: 29 out. 2024.

MOZILLA. ***World Wide Web***. 2024. Disponível em: https://developer.mozilla.org/en-US/docs/Glossary/World_Wide_Web. Acesso em: 21 de jan. de 2025.

NETWORK, M. D. ***JavaScript***. 2023. Disponível em: https://developer.mozilla.org/en-US/docs/Web/JavaScript. Acesso em: 10 out. 2024.

OPENCV. ***About OpenCV***. 2024. Disponível em: https://opencv.org/about/. Acesso em: 11 nov. 2024.

OPENJS. ***Node.js***. 2023. Disponível em: https://nodejs.org/. Acesso em: 10 out. 2024.

POKORNY, J. **NoSQL Databases: a Step to Database Scalability in Web Environment**. *International Journal of Web Information Systems*, v. 14, n. 1, p. 1–15, 2018.

PROJECT, T. G. ***Git - Distributed Version Control System***. 2023. Disponível em: https://git-scm.com/. Acesso em: 10 out. 2024.

PYTHON. ***Welcome to Python.org***. 2023. Disponível em: https://www.python.org/. Acesso em: 10 out. 2024.

REDMON, A. F. J. ***YOLO***. 2024. Disponível em: https://pjreddie.com/. Acesso em: 29 out. 2024.

RICH, E. **Artificial intelligence and the humanities**. *Computers and the Humanities*, JSTOR, p. 1, 1985.

RUSSELL, S.; NORVIG, P. ***Artificial Intelligence: A Modern Approach***. 3. ed. Upper Saddle River, NJ: Pearson, 2010.

SAMUEL, A. L. **Machine learning**. *The Technology Review*, v. 62, n. 1, p. 42–45, 1959.

SHETTY, J. et al. **Review paper on web frameworks, databases and web stacks**. *Intern Res J Eng Technol (IRJET)*, v. 7, n. 5, 2020.

SMITH, J.; DOE, J. **Emerging Technologies in Software Development**. *Journal of Software Engineering*, v. 25, n. 3, p. 123–134, 2019.

SOCKET.IO. ***Socket.IO: Bidirectional and low-latency communication for every platform***. 2023. Disponível em: https://socket.io/. Acesso em: 10 out. 2024.

SOUZA, L. N. R. de; ADANIYA, M. H. A. **O uso da tecnologia no aumento de produtividade no agronegócio**. *Revista Terra Cultura: Cadernos de Ensino e Pesquisa*, v. 40, n. especial, p. 466–485, 2024.

STONEBRAKER, M. et al. **The End of an Architectural Era (It's Time for a Complete Rewrite)**. *Proceedings of the VLDB Endowment*, v. 11, n. 12, p. 1841–1848, 2018.

STROUSTRUP, B. **An overview of C++**. In: *Proceedings of the 1986 SIGPLAN workshop on Object-oriented programming*. [S.l.: s.n.], 1986. p. 7–18.

SUZUKI, K. ***Artificial neural networks: methodological advances and biomedical applications***. [S.l.]: BoD–Books on Demand, 2011.

TAILWIND. ***Tailwind CSS: Rapidly build modern websites without ever leaving your HTML***. 2023. Disponível em: https://tailwindcss.com. Acesso em: 10 out. 2024.

TANENBAUM, A. S. ***Redes de Computadores***. [S.l.]: Campus, 2011.

TANENBAUM, A. S.; BOS, H. ***Modern Operating Systems***. 4. ed. Hoboken: Pearson, 2019.

TENSORFLOW. ***Aumento de dados***. 2022. Disponível em: https://www.tensorflow.org/tutorials/images/data_augmentation?hl=pt-br. Acesso em: 29 out. 2024.

TENSORFLOW. ***Introdução ao sintonizador Keras***. 2022. Disponível em: https://www.tensorflow.org/tutorials/keras/keras_tuner?hl=pt-br. Acesso em: 21 jan. 2025.

TRAUSS, P. ***C++ for Embedded System***. Londres: Packt Publishing, 2019.

WANG, J.; ZHANG, S.; MARTIN, R. R. **New advances in visual computing for intelligent processing of visual media and augmented reality**. *Science China Technological Sciences*, Springer, v. 58, p. 2210–2211, 2015.

WOLF, W. ***Computer as Components: principles of embedded computing system design***. New York: McGraw-Hill, 2001.

YAML. ***What It Is:***. 2024. Disponível em: https://yaml.org/. Acesso em: 25 nov. 2024.

YEGNANARAYANA, B. ***Artificial neural networks***. [S.l.]: PHI Learning Pvt. Ltd., 2009.

YING, X. **An overview of overfitting and its solutions**. In: IOP PUBLISHING. *Journal of physics: Conference series*. [S.l.], 2019. v. 1168, p. 022022.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Campina Grande - Código INEP: 25137409
R. Tranqüilino Coelho Lemos, 671, Dinamérica, CEP 58432-300, Campina Grande (PB)
CNPJ: 10.783.898/0003-37 - Telefone: (83) 2102.6200

Documento Digitalizado Ostensivo (Público)

Trabalho de Conclusão de Curso

Assunto:	Trabalho de Conclusão de Curso
Assinado por:	Luis Henrique
Tipo do Documento:	Dissertação
Situação:	Finalizado
Nível de Acesso:	Ostensivo (Público)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Luís Henrique Lima Santos**, **ALUNO (202011250011) DE BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO - CAMPINA GRANDE**, em 17/03/2025 10:49:59.

Este documento foi armazenado no SUAP em 17/03/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1420973
Código de Autenticação: 15200559e6