Leonardo Leite Leite

# Aplicação de Machine Learning em sistemas embarcados de borda: um estudo de caso

Uberlândia, MG

2023

Leonardo Leite Leite

# Aplicação de Machine Learning em sistemas embarcados de borda: um estudo de caso

Trabalho de Conclusão de Curso da Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Campus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação.

Universidade Federal de Uberlândia - UFU
Faculdade de Engenharia Elétrica - FEELT

Orientador Prof. Dr. Marcelo Barros de Almeida

Uberlândia, MG
2023

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

L533 Leite, Leonardo Leite, 1993-
2023 Aplicação de Machine Learning em sistemas embarcados de borda [recurso eletrônico] : Um estudo de caso / Leonardo Leite Leite. - 2023.

Orientador: Marcelo Barros de Almeida.
Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Uberlândia, Graduação em Engenharia de Controle e Automação.
Modo de acesso: Internet.
Inclui bibliografia.
Inclui ilustrações.

1. Processos de fabricação - Automação. I. Almeida, Marcelo Barros de,1972-, (Orient.). II. Universidade Federal de Uberlândia. Graduação em Engenharia de Controle e Automação. III. Título.

CDU: 67.02:681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

*Aos meus pais e ao meu irmão que seguiram comigo esta jornada.*

# Agradecimentos

Agradeço a minha família que não negou apoio e suporte para a realização deste projeto. Ao meu pai pelos valores morais e suporte financeiro provido durante todo o percurso acadêmico que culmina neste trabalho. A minha mãe, cujo carinho e cuidado não deixou passar detalhes como indícios da precarização de minha saúde e a falta de descanso causaram. Ao meu irmão, um grande companheiro cuja presença, por vezes eloquente e por vezes silenciosas, mitigou toda a incerteza que tive sobre minhas habilidades.

Agradeço ao Prof. Dr. Marcelo Barros de Almeida, cuja experiência e postura profissional servem de exemplo e admiração. Sua paciência e certeza em habilidades que eu mesmo duvidei possuir serviram de combustível para existência deste trabalho. Não obstante, sua visão de mercado foi imprescindível para que este tema fosse abordado.

Aos professores da do curso de graduação em Controle e Automação meu profundo agradecimento. A despeito do senso comum que permeia cursos de exatas, os professores desta faculdade mostram em seu trabalho a grande empatia e amor pela profissão que acabam se transmitindo aos alunos. Alguns destes tendo a sensibilidade além da relação professor aluno para perceber que não só de avaliações, mas de excelentes conselhos são formados grandes profissionais.

Aos meus amigos e colegas de curso agradeço imensamente. Certos vínculos se formam se se fortalecem dada a situação atípica que os indivíduos se encontram. Estas pessoas, por pura vontade de ajudar, desviam seus caminhos para estender a mão nos momentos difíceis e tornam a jornada da graduação um pouco menos amarga.

*"A ciência é, portanto, uma perversão de si mesma, a menos que tenha como fim último, melhorar a humanidade."*

*(Nicola Tesla)*

# Resumo

A principal premissa deste projeto é realizar um estudo de caso sobre a utilização de tecnologia de aprendizagem de máquina e computação de borda no desenvolvimento de um protótipo como prova de conceito. O foco será na implementação de inteligência artificial em um dispositivo que utiliza visão computacional para identificar em tempo real a presença de obstáculos móveis durante a manobra de ré de um veículo automotivo.

Ao longo do projeto, serão avaliados os métodos e técnicas utilizados para esse tipo de aplicação, incluindo a eficácia e eficiência de diferentes algoritmos de aprendizagem de máquina, técnicas de visão computacional e abordagens de computação de borda. Além disso, serão exploradas tecnologias de ponta relevantes para esse campo.

Por meio desse estudo de caso, o objetivo é demonstrar o potencial da aprendizagem de máquina e da computação de borda na melhoria da segurança e eficiência das operações de veículos automotivos, especialmente durante manobras desafiadoras, como a marcha à ré. Os resultados e percepções obtidos nesse projeto podem contribuir para avanços adicionais no campo de sistemas de transporte inteligentes.

**Palavras-chave**: Computação de borda; Visão computacional; Aprendizagem de máquina;

# Abstract

The main premise of this project is to conduct a case study on the use of machine learning technology and edge computing in the development of a proof-of-concept prototype. The focus will be on implementing artificial intelligence in a device that utilizes computer vision to real-time identify the presence of moving obstacles during the reverse maneuver of a motor vehicle.

Throughout the project, the methods and techniques used for this type of application will be evaluated, including the effectiveness and efficiency of different machine learning algorithms, computer vision techniques, and edge computing approaches. Additionally, cutting-edge technologies relevant to this field will be explored.

Through this case study, the aim is to demonstrate the potential of machine learning and edge computing in enhancing the safety and efficiency of motor vehicle operations, particularly during challenging maneuvers such as reverse parking. The results and insights obtained from this project can contribute to further advancements in the field of intelligent transportation systems.

**Key-words**: Edge computing; Computer vision; Machine learning;

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| APU | Audio Processing Unit |
| CNN | Convolutional Neural Network |
| GPIO | General Purpose Input Output |
| GPU | Graphic Processing Unit |
| IDE | Integrated Development Environment |
| I2C | Inter-Integrated Circuit |
| I2S | Inter-IC Sound |
| IoT | Internet of Things |
| ISA | Instruction Set Architecture |
| KPU | Keras Processing Unit |
| mAP | mean Average Precision |
| ML | Machine Learning |
| NPU | Neural Processing Unity |
| P&D | Pesquisa e Desenvolvimento |
| PASCAL | Pattern Analysis, Statistical Modelling and Computational Learning |
| QVGA | Quarter Video Graphics Array |
| REPL | Read Evaluate Print Loop |
| RISC | Reduced Instruction Set Computer |
| SoC | System on a Chip |
| SPI | Serial Peripheral Interface |
| SSD | Single Shot Detection |
| TPU | Tensor Processing Units |
| UART | Universal Asynchronous Receiver/Transmitter |
| VOC | Visual Object Class Challenge |
| YOLO | You Only Look Once |

# Sumário

# 1 Introdução

Avaliando a quantidade de sensores presentes em veículos e a eficiência dos mesmos em prover segurança aos condutores e aos pedestres, o projeto surgiu. A ideia é utilizar uma já presente em parte da frota automotiva, a câmera de ré, e através da implementação de visão computacional poder trazer para o motorista e para o transeunte mais segurança durante a execução da manobra de ré. O percalço claro é que o sistema embarcado responsável pela captura e exibição da imagem, bem como o computador de bordo presente na maioria dos veículos, não tem capacidade computacional para executar modelos de identificação de imagem por meio de inteligência artificial. Entretanto, o problema pode ser resolvido utilizando as técnicas propostas por um dos mais proeminentes campos em desenvolvimento na atualidade, o Embeded Machine Learnig.

O Doutor Denis Sato (SATO, 2022) em sua tese apresenta uma boa tentativa de identificar animais cruzando o trânsito em rodovias. Este projeto traz luz ao assunto e incita a possibilidade de expansão do tema. Já o doutor Wander Mendes oferece em sua tese (MARTINS, 2018) uma interessante estudo de algorítimos de visão computacional aplicado a desvio de drone quadro rotor. Este projeto faz parte de uma grande corrente de estudos na automação de métodos de segurança e desvio de obstáculos por meio de visão computacional. Ambos estudos citados exemplificam o grande esforço acadêmico em aplicar visão computacional nos campos de prevenção de acidentes em ações de manobra em veículos autônomos. Estes campos, se combinados, convergem ao objetivo geral que é de trazer segurança ao dia a dia das pessoas por meio de inteligência artificial.

Utilizando métodos de modelagem de redes neurais e, lançando mão de ferramentas para miniaturização destes modelos, é possível embarcar algoritmos que fazem a vez de redes neurais complexas, mas com tamanho e recursos coerentes com a implementação das mesmas em ambientes microcontrolados. Estes, por sua vez, realizarão a tarefa de executar o algoritmo e utilizar os resultados obtidos para reduzir ou, de maneira otimista, acabar com os acidentes causados por obstáculos móveis que possam se colocar no percurso dos carros durante a manobra de ré.

## 1.1 Justificativas

Embora hoje os automóveis disponham de sensores de segurança capazes de facilitar a operação veículos, diversos acidentes ainda ocorrem diariamente por motivos tais como os sensores implementados não disporem da sensibilidade necessária para identificar determinados tipos de obstáculos. Outro objeto de observação é a presença destes sensores, em uma avaliação fria, primariamente como dispositivos de preservação patrimonial. A

falta de capacidade de processamento nos computadores embarcados em automóveis para tarefas mais complexas e o erro humano são variáveis importantes nesse panorama.

Considerados casos em que os acidentes ocorrem no âmbito doméstico, onde o motorista se descuida devido a diversos fatores capazes de desviar sua atenção, a abundância de ocorrências em que crianças foram acometidas leva a criação tratados como o publicado em (WAKSMAN; MARIA; BLANK, 2014). Este excerto ressalta os riscos que crianças correm ao transitar em ambientes de circulação e entradas de garagem. Tais casualidades podem ocorrer em frações de segundo e causar danos irreparáveis aos envolvidos.

Um ramo recente da área de Embarcados se mostrou extremamente promissor na solução deste e de muitos outros problemas. A Computação de Borda tem fomentado o desenvolvimento de circuitos micro controlado com robustez e baixo consumo elétrico para realizar tarefas antes relegadas a equipamentos com alto poder computacional. Devido a esse desenvolvimento, uma ramificação da já consolidada Aprendizagem de Máquina pode ser migrada para o ambiente de circuitos embarcados. Estas ferramentas, que já tão são comumente vistas em aplicativos de smartphone e computadores pessoais, serão utilizadas para o desenvolvimento de uma também já conhecida ferramenta de segurança. Através da utilização de Embedded Machine Learning, tornar-se possível a implementação de análise de imagens em tempo real para que por meio de um modelo de inteligência artificial estes obstáculos dinâmicos sejam identificados. Tal identificação, feita localmente pelo hardware embarcado, permitirá a mais diversa sorte de alarmes e decisões tomadas pelo dispositivo para que com o fim principal seja alcançado e a vida de uma criatura inocente, seja poupada em uma fatalidade que deixará de acontecer devido ao uso da tecnologia em prol da segurança.

## 1.2 Objetivos

Este projeto tem por objetivo desenvolver uma implementação do campo de Aprendizagem de Máquina aplicada em computação de borda e conhecimentos em projetos de circuitos para produzir um equipamento capaz de identificar obstáculos em tempo real em um sistema auxiliar de ré veicular assistido por câmeras. A proposta é um dispositivo completamente modular e independente, em termos mecânicos, do automóvel no qual for implementado. O motorista terá, além das já existentes imagens de câmera de ré, informações sobre a presença de obstáculos moveis no curso da manobra. Com identificação autônoma e em alta velocidade, vidas poderão ser salvas fornecendo ainda mais segurança para pessoas e animais, bem como para o motorista, posto que uma vez identificado o obstáculo alarmes podem ser acionados para garantir que a atenção do condutor se volte a este perigo e medidas possam ser tomadas em tempo hábil.

# 2 Referenciais Teóricos

Os presentes avanços da tecnologia IoT tem permitido a criação de aplicações microcontroladas hábeis a coletar, manipular e transmitir dados, inclusive para a nuvem. Estas aplicações, no entanto, não tem o intuito de agir como nós da rede inteligente, relegando o processamento dos dados coletados a algorítimos complexos hospedados em sistemas distribuídos em nuvem, lançando mão do conceito de Big Data Analytics.

Dada uma recente mudança de paradigma, denominada Embeded Machine Learning, é possível habilitar ao dispositivo microcontrolado executar modelos que foram preparados por equipamentos com mais capacidade computacional ou em nuvem. Dessa forma, a aplicação é capaz de responder ao ambiente de maneira mais direta e livre de latência, bem como mitiga a necessidade de transmissão de dados e a vulnerabilização da informação que é inerente do processo.

Posto que a demanda de recursos em termo de processamento para o treinamento da inteligência artificial é alta, bem como a de volume de dados, esta tarefa é relegada a servidores com potência para tal. Este dispõe de hardware dedicado como unidades de processamento e renderização de elementos gráficos (GPUs) e unidades de processamento projetadas especificamente para acelerar operações em redes neurais artificiais (TPUs) específicas para o trabalho. Estas máquinas podem ser construídas para o fim de realizar o treinamento, ou pode ser contratado um serviço distribuído, como o serviço Google colab ou o AWS Sagemaker.

Opções mais dedicadas são possíveis também, como o caso da Sipeed com o serviço MAIXhub. Um certo padrão entre as empresas que estão produzindo dispositivos e microcontroladores preparados para estes equipamentos com inteligência de borda é disponibilizar a própria nuvem e data sets para serem usados. Esta prática tem como intuito sedimentar uma base de usuários acostumados com os produtos e ferramentas da companhia.

O volume de dados da saída de um modelo treinado não é hábil a ser implementado diretamente em sistemas embarcados, por limitações de memória e de processamento. Em modelos baseados em deep learning, a saber, diversas técnicas de quantização de valores são aplicadas nos pesos do modelo, de forma que, o dito volume cai consideravelmente posto que o trabalho passa a se basear em ponto fixo em detrimento de ponto flutuante. Ligações que não afetam de maneira contundente o resultado da análise são descartados da rede, técnica chamada pruning (BANDARU, 2020). Essas avaliações e técnicas pesam a conta partida entre o tamanho da rede e a sua capacidade de generalização. Assim, a execução no dispositivo de ponta é feita a partir de um algorítimo simplificado em comparação ao

mesmo implementado em um equipamento com processamento abundante disponível para a tarefa.

A redução destes algorítimos para serem embarcados já conta com o auxílio de frameworks. Um bom exemplo é a máquina de inferência para dispositivos embarcados, a Tensor Flow Lite, baseada no framework robusto de mesmo nome. As bibliotecas do framework reduzido permitem a redução de modelos gerados com bibliotecas padrão da ferramenta original e a geração da máquina de execução do modelo.

Também é passível de menção a ferramenta chamada Embeded Leaning Library, de autoria da Microsoft. Esta permite a geração de artefatos em C++ e python para processadores da série ARM Cortex, nas famílias A ou M. A ARM por sua vez apresentou um processador específico para tarefas relacionadas a redes neurais, como multiplicação de matrizes, um NPU chamado Ethos, bem como o ARM-NN, o framework de suporte a inferência de redes neurais em dispositivos baseados na arquitetura ARM cortex A, além do ARM Mali GPU. Na documentação (CMSIS, 2019) o CMSIS-NN implementa algorítimos de aprendizagem de máquina para Microcontroladores da série Cortex M. Este suporta modelos desenvolvidos em Tensor Flow bem como aplicações com peso operando em ponto fixo. Não obstante, a MicroML, uTensor, Sklearnporter, EmLearn e M2cgen são iniciativas promissoras para geração de código com foco em plataformas com recurso escasso. Estas são baseadas na conversão de modelos criados no framework Scikit-learn.

É natural antever cenários muito auspiciosos no campo da robótica e da bioengenharia com esta nova modalidade de aprendizagem embarcada em ambientes com pouco recurso computacional, mesmo sendo esta técnica tão recente e com desenvolvimento ainda em estágios iniciais. Reconhecimento de objetos e processamento de sinais biométricos, tal qual redução de custo em produção de próteses de alta tecnologia.

Para que todo esse potencial seja alcançado o modelo deve ser preparado e, neste estágio, as tecnologias de inteligência embarcada e IoT são empregadas. Um dispositivo capaz de coletar dados de imagem, pode atuar como o transmissor desses dados para um serviço computacional distribuído. Tanto o coletor quanto o transmissor, ambos num único equipamento quanto separados em hardwares distintos, podem, por bem, estar embarcados em uma câmera de ré, alimentando a base de dados de uma rede em treinamento contínuo. Esta base de dados pode, por sua vez, ser alimentada por câmeras em uma rede de esforço conjunto capaz de lapidar a operação de todas essas câmeras na detecção e tomada de ação mediante a um obstáculo. A partir daí este treinamento pode resultar em um modelo mais eficiente que será carregado nas câmeras através por meio de atualização do modelo TinyML presente na câmera e retroalimentando a cadeia.

## 2.1 Metodo de detecção de imagem

Dentre as diversas implementações possíveis para identificação de imagem, este projeto lança mão do YOLOv2. Segundo Joseph Redmon e Ali Farhadi (REDMON; FARHADI, 2016), YOLOv2 trata de um padrão de detecção de imagem em tempo real que teve desempenho de 78,6 mAP. A mAP é termo é comumente utilizado em avaliações de desempenho de algoritmos de detecção e classificação de objetos, como definido em (LAPIX, 2019). Este desempenho foi amostrado quando aplicada a rede ao banco PASCAL VOC 2007 (PASCAL, 2023), um data-set público utilizado como base de referência para algorítimos de detecção de imagem. Tal desempenho se mostra melhor que redes concorrentes como Faster R-CNN (REN et al., 2017), rede que utiliza arquitetura de rede neural convolucional e uma rede de regiões para detectar e localizar objetos em uma imagem, e SSD (LIU et al., 2016), rede de detecção de imagens que usa apenas uma imagem e inteligência artificial para inferir objetos e aumentar a velocidade de resposta.

YOLOv2 lança mão de ferramentas complexas para acelerar o tempo de resposta da rede. Classificadores de alta resolução tornam a rede capaz de treinar com imagens de resolução de 448x448px em data-sets complexos em 10 epochs, termo usado para definir quantas vezes o mesmo dado passa pela rede antes que a mesma consiga prover um resultado. Usa-se de Batch Normalization, método em que as informações são normalizadas entre camadas da rede neural, ao invés de normalizar o data-set inteiro. O algorítimo aplica técnica de Anchor boxes que, como define (MATHWORKS, 2023), são regiões pré-determinadas que serão refinadas de maneira que ao fim do processo de identificação estas regiões delimitem exclusivamente os objetos identificados na imagem dada. Melhorias nos filtros aplicados a imagem entre as camadas da rede de identificação, bem como um melhor cálculo de coordenadas do objeto identificado na imagem. A rede é capaz de Multi-Scaling training para manipular imagens. Dessa forma ela aplica de up-scaling, aumentando o tamanho das imagens, e down-scaling, reduzindo o tamanho, entre etapas de treinamento sobre o mesmo data-set, de maneira que o resultado tem a acurácia maior por ser revisitado em diferentes condições.

Para o algorítimo de inferência YOLOv2 usa Darknet19, uma rede neural convolucional de 19 camadas, esta rede consegue, com poucas operações, obter acurácia alta em data-sets como ImageNet. Embora seja natural configurar a CNN para apenas uma tarefa, seja Classificação ou detecção. YOLOv2 possui mecanismos para contrabalancear ambas as tarefas e conseguir fazer ambas. O resultado é que a rede é capaz de detectar, encontrar objetos na imagem, e classificar, definir um objeto, com precisão e velocidade.

Dentre as diversas implementações possíveis para identificação de imagem, este projeto lança mão do YOLOv2. Segundo Joseph Redmon e Ali Farhadi (REDMON; FARHADI, 2016), YOLOv2 trata de um padrão de detecção de imagem em tempo real que teve desempenho de 78,6 mAP quando aplicado ao PASCAL VOC 2007, um data-set

publico utilizado como base de referência para algorítimos de detecção de imagem. O um desempenho melhor que redes concorrentes como Faster R-CNN, uma rede convolucional para detecção de objetos, e SSD, uma rede de detecção de imagens que usa inteligência artificial para inferir objetos e aumentar a velocidade de resposta.

YOLOv2 lança mão de ferramentas como complexas para acelerar o tempo de resposta da rede. Classificadores de alta resolução tornam a rede capaz de treinar com imagens de resolução de 448x448px em data-sets complexos em 10 epochs, termo usado para definir quantas vezes o mesmo dado passa pela rede antes que a mesma consiga prover um resultado. Batch Normalization, método em que as informações são normalizadas entre camadas da rede neural, ao invés de normalizar o data-set inteiro. Anchor boxes pre determinam regiões que serão refinadas a fim de que no fim do processo de identificação estas regiões delimitem exclusivamente os objetos identificados na imagem dada. Melhorias nos filtros aplicados a imagem entre as camadas da rede de identificação, bem como um melhor cálculo de coordenadas do objeto identificado na imagem. A rede e capaz de Multi-Scaling training para manipular imagens. Dessa forma ela aplica de up-scaling, aumentando o tamanho das imagens, e down-scaling, reduzindo o tamanho, entre etapas de treinamento sobre o mesmo data-set, de maneira que o resultado tem a acurácia maior por ser re-visitado em diferentes condições.

Para o algorítimo de inferência YOLOv2 usa Darknet19, uma rede neural convolucional de 19 camadas, esta rede consegue, com poucas operações, obter acurácia alta em data-sets como ImageNet. Embora seja natural configurar a CNN para apenas uma tarefa, seja Classificação ou detecção. YOLOv2 possui mecanismos para contrabalancear ambas as tarefas e conseguir fazer ambas. O resultado é que a rede é capaz de detectar, encontrar objetos na imagem, e classificar, definir um objeto, com precisão e velocidade.

## 2.2 RiscV

A arquitetura RISCV implementa tarefas complexas em processadores de maneira rápida e com baixo consumo. Trata-se, como visto em (RISCV, 2015), de uma ISA, a saber, a especificação que define o conjunto de instruções que um processador pode executar, criada inicialmente para proporcionar processadores capazes de realizar processamento paralelo, bem como a capacidade de produzir os para projetos acadêmicos que não dependessem de patentes, Open Hardware. A arquitetura tem metodologia modular, tendo uma base de instruções definida, seja a RV32I ou RV64I, comum a todos os processadores desta ISA. Possui também instruções extras que podem ser adicionadas para atender a aplicações específicas.

Naturalmente, processamento de algoritmos como redes neurais, inteligência de máquina consomem muitos recursos do processador, aumentando a potência demandada

e diminuindo vida útil de dispositivos embarcados devido a aquecimento. Ao nível de arquitetura, a flexibilidade de um processador RISCV permite que instruções complexas sejam realizadas de forma rápida nativamente. A licença aberta da ISA fomenta o desenvolvimento da arquitetura e a quantidade de pesquisas na mesma permite inovações no campo das instruções extra que o processador dispõe. Essas características permitem que o processador possa seja personalizado para as mais diversas aplicações e otimizado para realizá-las com consumo elétrico baixo característico do set de instruções. Estes predicados tornam o processador RISCV a base para hardwares de computação de borda extremamente potentes e a gama constantemente aumentando diariamente. Lançando mão das possibilidades inúmeras que a RISCV ISA habilita, algumas empresas têm criado SoC com a arquitetura. Este movimento possibilita criar hardwares extremamente potentes e prontos para utilização em campo. Nomes como Microchip, NXP e Nvidia possuem SoC tem aderido rápido a ISA. Estranhamente a STMicroelectronics, embora tenha doado o primeiro protótipo da arquitetura, não possui nenhum produto voltado a este mercado ainda.

## 2.3 K210 e Maixduino.

Uma implementação do RISCV ISA interessante é a produzida pela Canaan AI. A empresa tem foco em mineração de criptomoedas e criou o Núcleo K210, projetado para aplicações com inteligência artificial. Baseado neste Núcleo, a empresa Sipeed desenvolveu a série de plataformas de desenvolvimento e prototipagem, notoriamente a placa MaixDuino.

Segundo a SipeedWiki (SIPEED, 2022d) a placa Sipeed MaixDuino, ilustrado na Figura 2, é compatível com ambiente de desenvolvimento e shields do Arduino. A seguir a Tabela 1 com as características do módulo M1 da Sipeed, ilustrado pela Figura 1.

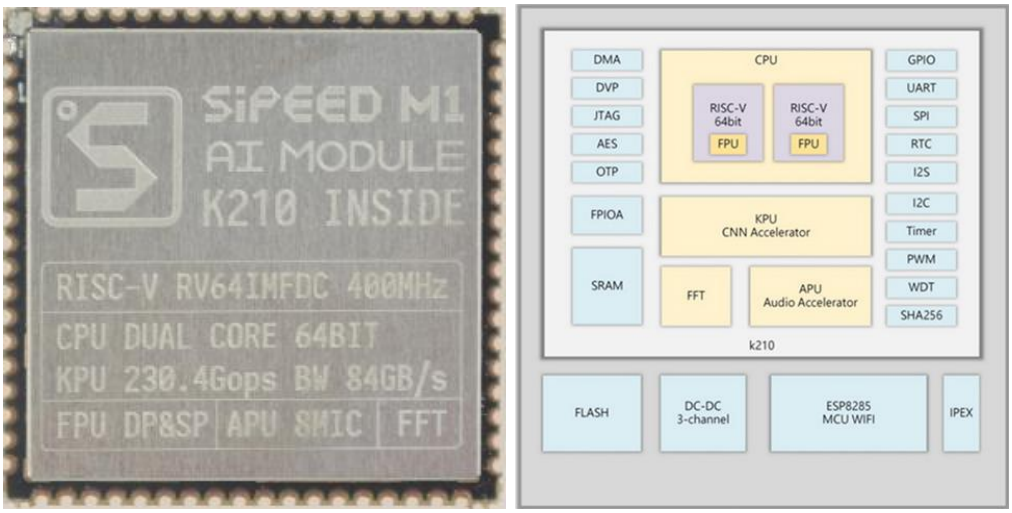
Tabela 1 – Parâmetros básicos do módulo K210

| | |
|---|---|
| Kernel | RISC-V Dual Core 64bit, with FPU |
| Main frequency | 400MHz (overclockable to 600MHz) |
| SRAM | Built-in 8M Byte |
| Image Identification | QVGA@60fps/VGA@30fps |
| Speech Recognition | Microphone array (8mics) |
| Neural network model | Support YOLOv3<br>Mobile netv2<br>TinyYOLOv2 |
| Deep Learning Framework | Support mainstream frameworks such as<br>TensorFlow \ Keras \ Darknet \ Caffe |
| Peripherals | FPIOA, UAR, GPIO, SPI, I2C, I2S, TIMER |
| Video processing | Neural Network Processor (KPU)<br>FPU meets IEEE754-2008 standard<br>Audio Processor (APU)<br>Fast Fourier Transform Accelerator (FFT) |

**Fonte:** Adaptado de Sipeed (2019b).

Além destas características é importante ressaltar a presença de FreeRtOS com suporte a versão proprietária de Micropython da Sipeed, o MaixPy. O kit conta também com suporte para Redes Neurais Convolucionais mediante um KPU. Audição de máquina é possível com a um processador de áudio de alto desempenho. A conectividade fica a cargo de um módulo ESP-32 que se comunica com o núcleo por comando AT através da comunicação SPI. Esta ESP-32 pode ser programada de forma independente, o que é desencorajado pelo fabricante, no entanto, ela está preparada para fornecer conexão WiFi e Bluethooth.

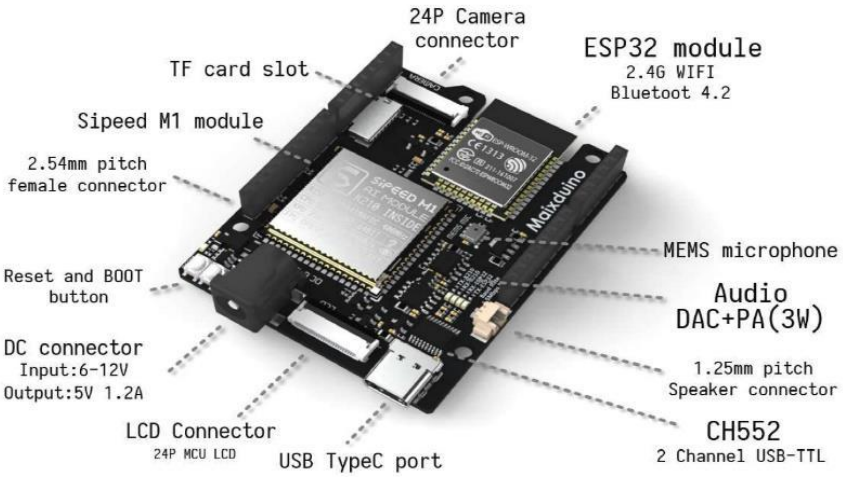
Figura 1 – Núcleo M1 presente no kit Maixduino




**Fonte:** Adaptado de Technology (2021).

A tensão de operação do módulo M1 é 5V com corrente de no mínimo 300mA. Entretanto, o kit MaixDuino pede uma fonte de 6 a 12V. Esta diferença se deve a presença de conectores para um alto-falante de 1.2mm. O kit conta com 2 conectores de 24 vias para conectar uma câmera e um LCD.

Figura 2 – Seeed Studio Sipeed Maixduino




**Fonte:** Adaptado de Sipeed (2019b).

# 3 Metodologia

Os presentes avanços da tecnologia IoT tem permitido a criação de aplicações microcontroladas hábeis a coletar, manipular e transmitir dados, inclusive para a nuvem. Estas aplicações, no entanto, não tem o intuito de agir como nós da rede inteligente, relegando o processamento dos dados coletados a algorítimos complexos hospedados em sistemas distribuídos em nuvem, lançando mão do conceito de Big Data Analytics.

O processo adotado para a implementação da ideia proposta está descrito a seguir. A supracitada fundamentação teórica aborda apenas os temas relacionados ao produto final. O que será apresentado é o melhor resultado obtido mediante as ferramentas disponíveis para a implementação.

## 3.1 Pesquisa e desenvolvimento de hardware

Ao pesquisar hardwares capazes de executar o projeto e a fim de gerar um MVP viável. Para tal intuito este projeto buscou por uma solução já consagrada no mercado que fosse capaz de realizar as tarefas propostas, visão computacional na borda. O Hardware foi sugerido e cedido pelo orientador deste projeto, e depois de alguma pesquisa, mostrou-se uma excelente opção.

Sendo assim, o Kit Maixduino, ilustrado pela Figura 3, foi adotado para a parte de implementação do projeto. A aplicação inicialmente visa utilizar visão computacional embarcada apara produzir um espelho de ré inteligente e independente de qualquer hardware presente no veículo. O kit é projetado ao redor do núcleo K210 que possui aceleração de hardware e capacidades de inteligência artificial embarcada por definição.

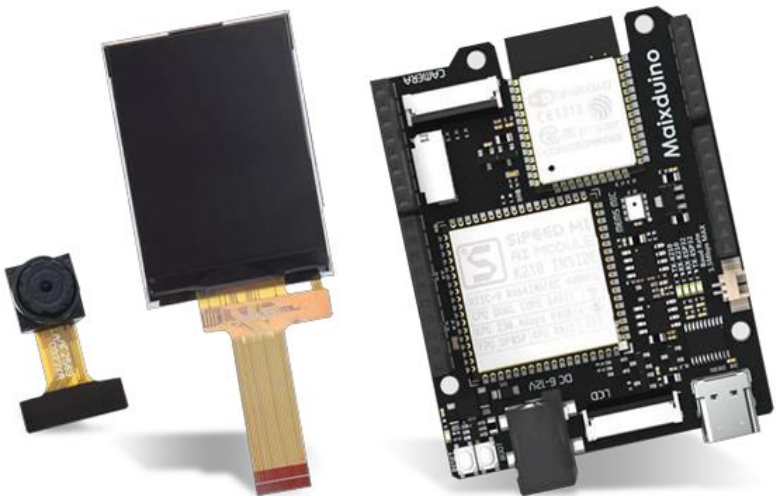
O feed de imagens do produto proposto provem de uma câmera posicionada na parte traseira do veículo, conforme usual no mercado para câmeras de ré. No tocante ao kit Maixduino, este dispõe de um conector de 24pinos para conexão de câmera, esta comunica-se diretamente com o núcleo sem nenhuma interface intermediaria, o que garante que as limitações de tempo de resposta estarão relacionadas as limitações do K210.

No objeto de estudo, a câmera de ré possui uma interface visual para o usuário, seja integrada ao sistema de bordo do carro ou integrada ao retrovisor do para-brisa do automóvel. A solução da Sipeed apresenta um conector de 24pinos para utilização de uma tela de LCD, bem como capacidades nativas de interface com telas sensíveis ao toque.

Juntamente a interface visual é interessante a possibilidade de mais opções de atuação a fim de garantir a agência do MVP em garantir a segurança do condutor. Usualmente carros possuem buzzers que são interfaceados pelo sistema de sensoriamento

ultra sônico de ré, este é um feedback audível bastante eficiente e é uma grande adição ao produto desenvolvido. O kit dispõe diversas portas GPIO, além de periféricos responsáveis pelos protocolos UART, SPI, I2S, I2C entre outros. Estes periféricos fornecem uma gama quase ilimitada de possibilidades de interfaces extra no produto. Para fins de prova de conceito, um buzzer foi adicionado para sinal sonoro tal qual o modelo no qual o projeto se inspira. No entanto, este sinal sonoro não está vinculado a sensores de distância, ele é utilizado pelo fimware diretamente para indicar perigo.

Figura 3 – Seeed Studio Sipeed Maixduino Kit




**Fonte:** Adaptado de Mouser (2022).

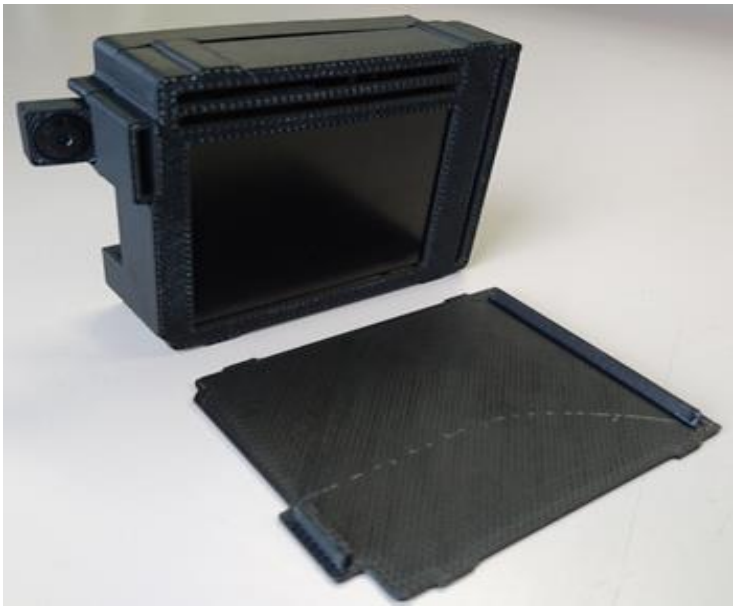
## 3.2 Testes e validação de hardware

De posse do kit, alguns testes foram feitos com o intuito de verificar os módulos e as funcionalidades descritas. Nesta etapa observou-se uma fragilidade devida a contatos expostos inerentes a kits de prototipagem. A solução encontrada foi, mediante impressão 3D, produzir um invólucro que protegesse o kit durante o manuseio.

O dispositivo proposto neste projeto utiliza uma case impressa em 3D como prova da flexibilidade do design. A fabricação aditiva por meio da impressão 3D permite a criação de estruturas personalizadas e adaptáveis conforme as necessidades específicas do projeto. A utilização de uma case impressa em 3D demonstra a capacidade de produzir um invólucro flexível, resistente e sob medida para abrigar os componentes do dispositivo, reforçando ainda mais a proposta de prova de conceito.

O estojo escolhido foi o proposto pelo usuário komix na plataforma Thingiverse (THINGIVERSE.COM, 2023), um agregador de designs open source voltados a impressão 3D mantido pela UltiMaker. O estojo escolhido está ilustrado pela Figura 4, que segue:

Figura 4 – Case utilizada para o kit Maixduino



**Fonte:** Adaptado de Thingiverse.com (2023).

## 3.3 Pesquisa e desenvolvimento de Software

Conhecimentos de inteligência artificial, em geral, se mostraram indispensáveis no curso de desenvolvimento da aplicação. Primeiramente foi necessário definir uma estratégia de projeto que permitisse implementar partes do projeto a medida que a teoria necessária para tal fosse sedimentada. Em suma, foi implementado o algoritmo visão computacional no kit e em seguida as demais características do projeto foram abordadas. As buscas por fontes me levaram a um projeto chamado aXeleRate (AIWINTERMUTEAI, 2021). Este projeto é voltado para usuários que pretendem executar visão computacional em dispositivos de borda. Trata de uma série de roteiros que facilitam o processo de implementação da rede.

O percurso para implementar a visão computacional no kit é, em suma, implementar um modelo de rede neural capaz de realizar tarefas sobre as imagens. Esta rede tem a habilidade de através de seu algoritmo, dada uma imagem como entrada, uma Rede Neural Convolucional tratará este dado a fim de delimitar regiões, estas então são submetidas a avaliações a fim de refinar o que eventualmente se tornará uma secção.

Deste ponto a rede pode seguir para duas opções não mutuamente exclusivas, a rede pode seguir avaliando as secções a fim de encontrar os limites de todos os componentes pertencentes a mesma secção e então definir uma área que contenha esta secção por completo, este processo é chamado identificação. O que é identificado no fim do processo é o que nos humanos conhecemos como um objeto, este processo é de fato genérico e abrangente dado que o algoritmo pode ser especializado em encontrar objetos específicos, bicicletas, por exemplo, ou isolar objetos, independente do que ele seja. Outra capacidade é,

após delimitar objetos, buscar em bancos de dados por definições desses objetos de forma hierárquica em termos de nomenclatura. Este processo de classificação é o responsável pela capacidade da rede de inferir o nome do objeto observado, como o caso do YOLOv2 (REDMON; FARHADI, 2016).

Para a rede ter acurácia e precisão é necessário que ela seja submetida a diversas variações nas entradas a fim de que reconheça o mesmo objeto nas mais diversas posições e formas que ele pode se apresentar. Cada rede tem seus próprios métodos para realizar esta tarefa. E como dito anteriormente não são mutuamente exclusivas, então de fato existem redes capazes de identificar e classificar objetos.

Selecionado o algorítimo capaz de realizar classificação e identificação, deve-se submetê-lo a uma base de dados diversa a fim de que o mesmo esteja treinado para os casos específicos de uso. É possível criar esta base de dados, conhecida como data-set, mas ela precisa ser vasta e diversa, com muitas imagens e variações da mesma imagem a fim de que a rede se torne especializada. Perceba que se a rede for especializada em diversos objetos este data-set tende a expandir exponencialmente. Sendo assim, o mais usual é utilizar data-sets públicos como o PASCAL VOC, o qual é atualizado de tempos em tempos. A Google disponibiliza uma ferramenta de busca de data-sets (GOOGLE, 2013), o Google data-set search, onde é possível inserir o tipo de dados e os conteúdos do data-sets e encontrar um pacote disponível com o data-set desejado.

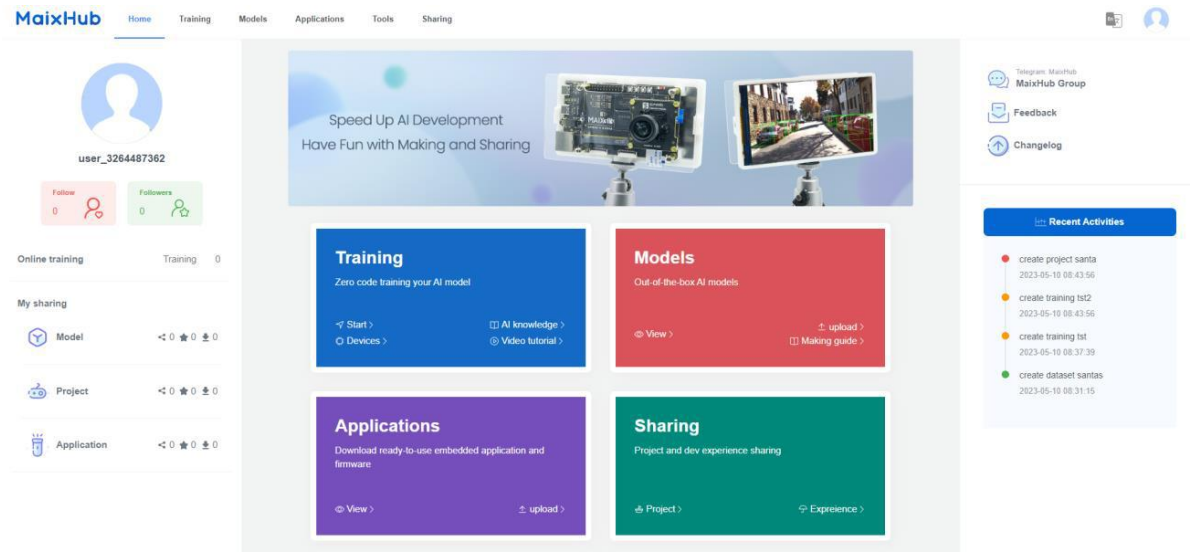
Definido o data-set adequado a sua aplicação, configura-se a rede para treinar encima desse. O tempo tem influência nesta etapa, pois a intenção é obter acurácia e precisão na rede. No entanto, a correlação tempo e acurácia é mais bem definida como logarítmica. Estes conceitos em engenharia são relativos ao nível de precisão buscado. Então prepara-se a rede para que ela treine em etapas, as epochs, e define-se a pontuação que cada epoch deve atingir para que se passe para seguir adiante. Basicamente, cada epoch pode ou não ser sobre o data-set em sua forma original. Algumas redes aplicam filtros próprios, impurezas e manipulações a fim de deformar os dados entre cada epoch e melhorar a rede.

As configurações, tais como quantas epochs, qual o score mínimo ao qual a rede deve convergir treinando sobre o data-set, o tamanho do data-set, quantos elementos do data-set serão destinados a treino e quantas serão destinadas à validação, são de suma importância. Definir corretamente esses parâmetros têm influência sobre tempo que levará para a rede neural se especializar e gerar o modelo treinado para o seu data-set, e o quão rápido e preciso este modelo será.

Este modelo, embora especializado, não está necessariamente otimizado para ser implementado no hardware. Estas configurações devem ser feitas em sequência a geração do modelo para miniaturizá-lo. Frameworks como TinyML, Pytorch, TensorFlow, entre outros, possuem seus próprios métodos para tal miniaturização. Existem também projetos como o

aXeleRate possuem scripts para gerar modelos otimizados e prontos para implementação. A própria fabricante do kit, a Sipeed, fornece um ambiente extremamente interessante com estes passos condensados em uma interface de usuário amigável, tal qual ilustra a Figura 5.

Figura 5 – Interface de usuario MaixHub




**Fonte:**Adaptado de Sipeed (2019a).

Estes ecossistemas fornecem acesso máquinas em nuvem com grande capacidade de processamento para treinar redes complexas sobre grandes data-sets de forma remota e rápida. O modelo produzido tem a vantagem de já ser otimizado para implementar no kit.

O projeto aXeleRate é abordado completamente em um artigo hospedado no site Instructables (INSTRUCTABLES, 2019). Através dele os conhecimentos obtidos nas pesquisas sobre os temas isoladamente, foram consolidados. Entretanto, o método descrito, assim como o projeto aXeleRate estavam desatualizados em suas dependências, tornando os scripts obsoletos. Os outros frameworks citados se atualizaram e observa-se uma grande tendência da comunidade a utilização do Pytorch. Tais problemas foram responsáveis por o processo de aprendizagem e desenvolvimento do objetivo deste documento não estar dando frutos adequados.

A solução que encontrada para prosseguir no objetivo deste projeto foi utilizar o MaixHUB, onde existem diversos projetos compartilhados pelos usuários da plataforma, bem como projetos oficiais. Este repositório fornece o modelo treinado para a tarefa definida no projeto e são dependentes de hardware. Assim, após avaliar as diferentes propostas do repositório, o modelo do usuário Neucrack se mostra o mais adequado a tarefa proposta neste documento. Este modelo implementa a rede YOLOv2 em núcleos K210. O sistema de detecção é resultado de um trabalho feito para otimizar e acelerar o tempo de resposta da

iteração anterior, o YOLO, com a utilização de uma nova rede neural chamada Darknet19. Na proposta de Neucrack (NEUCRACK, 2023), YOLOv2 foi treinado sobre o data-set PASCAL VOC 2007 (PASCAL, 2023) cujo banco agrega 20 classes distintas de objetos, a saber:

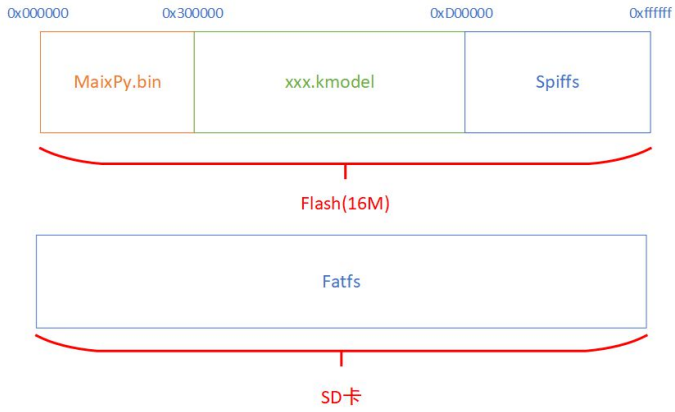
- Pesoa: Pessoa
- Animal: Passaro, Gato, Vaca, Cão, Cavalo, Ovelha
- Veiculos: Moto, Carro, Bicicleta, Onibus, Trem, Avião
- Interiores: Garrafa, Planta envasada, Cadeira, Mesa de jantar, Tv/Monitor, Sofá

Atente-se ao fato que o data-set abranger alguns objetos além do escopo do protótipo proposto neste estudo, que idealmente precisa diferenciar pessoas, animais domésticos e alguns veículos. No entanto, jugando a velocidade de resposta do modelo (entre 50ms e 70ms) e a presença de categorias importantes para a aplicação pretendida, optou-se por utilizar este modelo como base da implementação para prova de conceito e produção do dispositivo. Com relação às classes não coerentes com o escopo, estas foram suprimidas no script da aplicação.

Ter acesso ao kmodel treinado na plataforma exige um certo trato além do comum, pois o repositório é dedicado a usuários dos produtos da Sipeed. Para ter acesso ao modelo é preciso fornecer um machine code. Esta é uma chave criptográfica gravada na memória do SoC que é única. O processo de aquisição dessa chave é abordado nos forums de atualizados usuários e mantido pela Sipeed. Sua aquisição e detalhada no compilado da documentação publicada na sua wiki (SIPEED, 2022e) e descreve um firmware que deve ser gravado no início da memória flash do SoC e quando acessado via comunicação serial imprime a chave a ser usada.

Após baixar o modelo deve-se implementar o código que o acessará. Este modelo estará disponível para o KPU presente no núcleo e poderá ser acessado por meio de métodos disponibilizados ao usuário. O kit dispõe de cartão de memória, que pode ser usado para guardar modelos e mídias que serão usadas nas aplicações. Entretanto, considerando padrões industriais de microcontroladores, pode-se inferir que, se configurado corretamente, endereços presentes na memória flash destes podem ser facilmente acessados e utilizados de forma rápida e prática. Este método inclusive é encorajado pela Sipeed na documentação do microcontrolador como visto em Sipeed (2022b). A representação visual da distribuição de memória fornecida pelo fabricante, aqui ilustrada pela Figura 6 mostra que existe uma região dedicada aos kmodels. Estando no flash é plausível inferir que módulos como DMA ou mesmo a própria KPU, entre outros, poderão acessar essa região de memória sem depender de instruções do processador, tornando assim modelo mais acessível e responsivo a aplicação.

Figura 6 – Ilustração do sistema de armazenamento MaixPy




**Fonte:** Adaptado de Sipeed (2022b).

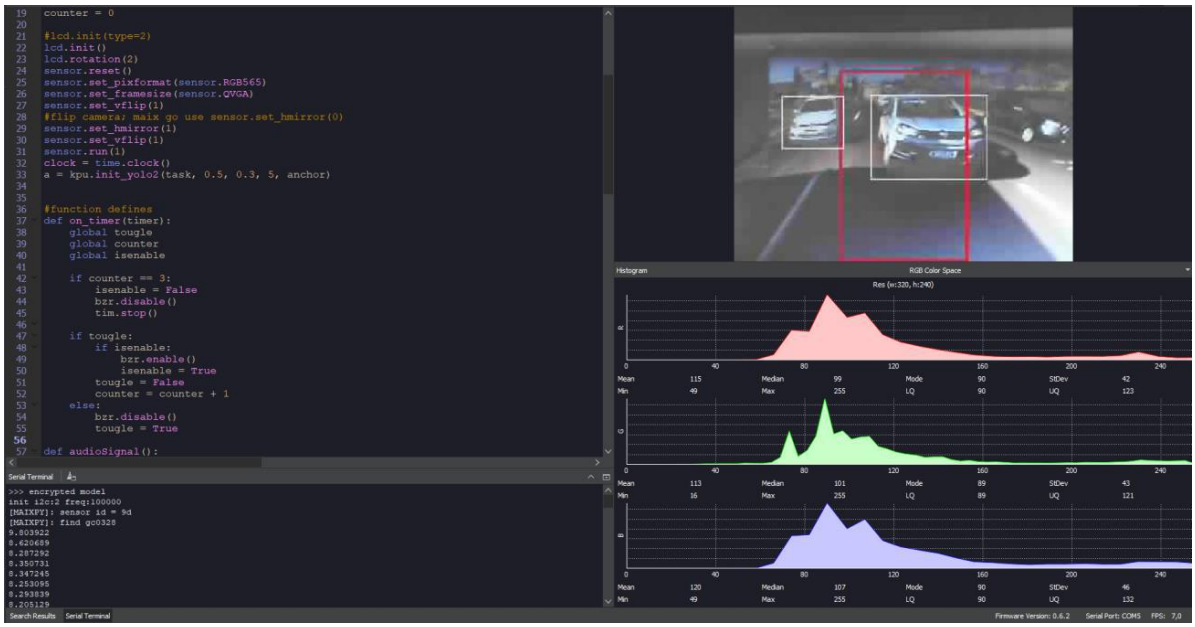
Na mesma fonte pode-se observar a recomendação do fabricante de não utilizar modelos no cartão de memória por motivos de, por se tratar de uma unidade flash removível, está sujeito aos problemas comuns desse meio. O sistema operacional pode não reconhecer a unidade, ela pode ser desmontada abruptamente por mal contato. Sendo assim, recomenda-se usar o cartão apenas para modelos que não caibam na região determinada para tal. Importante notar que a gravação do modelo é diretamente em uma região e tem definidos seu tamanho e o off-set onde se deve iniciar a gravação, sendo possível gravar quantos modelos a região comportar.

Quanto ao firmware, se faz relevante salientar que este termo está sendo usado alegórica neste documento, uma vez que a aplicação de software produzido de fato é na forma de script. A placa de prototipagem Maixduino possui por nativamente um RTOS sobre o qual roda uma versão modificada da implementação embarcada do Python, o Micropython, com a roupagem MaixPy. Este RTOS gerencia boa parte das interações de baixo nível, sendo o MaixPy uma linguagem interpretada de alto nível que segue o padrão REPL. Este padrão adotado por algumas linguagens de alto nível define processo em laço de leitura, avaliação e impressão, como definido em (HORCASITAS, 2021), de forma que cada linha do script é executada individualmente pelo interpretador da linguagem. Desta maneira criar aplicações para a placa é basicamente criar um código no formato MaixPy e definir no script de inicialização da placa que a sua aplicação é o script que deve ser iniciado após o boot.

Para depurar a placa é possível se conectar com a mesma mediante uma interface USB-serial e acessar o terminal gerado pelo RTOS. Qualquer ferramenta de comunicação serial pode fazê-lo e com conhecimento deste método de interface de hardware é possível programar aplicações e enviar arquivos para a memória do kit. Outra opção e a do fabricante que disponibiliza uma IDE capaz de executar scripts na placa direto do computador host utilizando da comunicação supracitada, a MaixPy IDE, como ilustra a Figura 7. Nesta

ilustração se pode observar a presença de ferramentas de depuração como o campo linter, ferramenta de análise sintática e estética de código, e um terminal de monitoramento de comunicação serial. Além destes, a figura mostra a presença de um campo de exibição do framebuffer do dispositivo, bem como campos com histogramas de incidência de cores nos espectros mais comuns a análise técnica (RGB, YUV e Gray Scale, a saber).

Figura 7 – MaixPy IDE




**Fonte:** Autoria Própria

O código da aplicação tem a mesma estrutura de script no padrão Micropython, com o adicional das bibliotecas fornecidas pelo fabricante especificamente para os módulos presentes no K210. Inicialmente importam-se as bibliotecas ou módulos necessários. Definem-se as variáveis globais e as funções criadas pelo usuário e então o LCD e a câmera devem ser inicializados.

Vale notar a importação do módulo board_info presente na biblioteca board. O motvio pelo qual esta chamada é importante é que esta biblioteca é inserida pelo usuário través da execução de um código elencado no github (SIPEED, 2023). Esta tarefa é importante, pois esta biblioteca é referênciada em outras partes da documentação, no entanto, a sua instalação e abordada apenas na referência encontrada em (SIPEED, 2022a). O que é feito é inserir os nomes dos pinos em um arquivo presente na memória flash da placa a fim de facilitar a referênciação dos mesmos nos códigos. O motivo é que a placa tem diferentes referências para os pinos externos, sendo possível acessar um pino através de seu índice GPIO indexado no K210 ou através do silk presente, ou que deveria estar, na placa Maixduino.

A importação da biblioteca KPU permite manipular diretamente este modulo

presente no SoC (SIPEED, 2022c). Ele pode processar cálculos de uma rede neural convolucional acelerado por hardware e com baixo consumo de energia. Este KPU possui as limitações descritas na fonte citada, no entanto, consideradas essas limitações, o modelo pode ser treinado por quaisquer frameworks de uso consagrado.

O método kpu.load() recebe como parâmetro o caminho para o modelo. Como já foi citado anteriormente, este pode estar na unidade removível ou na memória flash do K210. Gravar este modelo na memória é um procedimento realizado usando a ferramenta chamada Kflash. Este utilitário é fornecido pelo fabricante (SIPEED, 2022f) do kit em seu github. Esta é uma ferramenta feita para copiar arquivos binários para a memória do K210, a esta ferramenta foi adicionada uma interface gráfica e é com ela que se gravam os binários tanto dos firmwares oficiais, quanto qualquer outro dado que se deseje colocar no flash do núcleo.

Deve-se atentar a ser necessário adequar o firmware presente na placa o projeto a ser implementado. O modelo selecionado para este projeto demanda que o firmware presente na placa consiga executar kmodels, modelo baseado na biblioteca de rede neural de alto nível conhecida como keras (KERAS, 2018), em sua quarta iteração. Esta e outras informações são fornecidas pelo desenvolvedor do modelo em sua página do projeto (NEUCRACK, 2023).

Deve se atentar que, ao utilizar o ambiente de desenvolvimento MaixPy, certos métodos devem ser obedecidos. O firmware cujo suporte é solicitado para a execução da rede específica para este projeto pode ser encontrado no repositório de downloads da Sipeed (SIPEED, 2020). Neste estão todos os firmwares oficiais com e sem suporte a ide. A Figura 8 ilustra um excerto da lista de firmwares disponíveis, de forma que se possa observar a nomenclatura de cada entrada. No documento (SIPEED, 2022g) o fabricante deixa expressa como a empresa nomeia e indica a versão de seus binários e, atente-se que existem versões com e sem suporte a MaixPy IDE.

Figura 8 – Opções de firmware adequados


**Fonte:** Autoria Própria

Com o firmware correto e o kmodel salvo na memória flash é possível usar o modelo nos scripts e aplicações do usuário. Basta, como dito anteriormente, carregar o modelo no KPU, inicializá-lo com o vetor de âncoras definidas durante o treinamento da rede. Lembrando que o modelo escolhido é baseado em YOLOv2, existem certas configurações inerentes a este algorítimo que são definidas durante o processo de treinamento e que

não podem ser alteradas. Estas informações foram fornecidas pelo criador do modelo na página do projeto, mais precisamente no código exemplo, onde é possível conferir o vetor de âncoras.

A partir deste ponto é implementada a lógica usando como entrada as imagens capturadas pela câmera. Esta imagem é passada para o modelo que como resposta retorna uma lista de objetos classificados, bem como suas respectivas coordenadas na imagem e o tamanho da região delimitante que a rede pôde identificar. Esta região delimitante é referida como bounding-box e com essas informações torna-se possível usar as funções presentes na biblioteca do LCD para desenhar na tela as regiões e tratá-las como for necessário.

Para implementar proposta deste documento foi feita uma análise de como funcionam as câmeras de ré e os sistemas de assistência de manobra. O feed da câmera posicionada na traseira do veículo é mostrado para o motorista em uma tela com um overlay projetado para servir como referência de distância. Além deste existe o sistema auxiliar de sinais sonoros ativado pela interpretação dos dados dos sensores ultrassônicos posicionados ao redor do carro.

Buscando manter um padrão visual, a solução apresentada é semelhante nos aspectos já citados. Como pode se verificar na Figura 9, a região vermelha representa a área segura para manobra de ré, foi definida de forma empírica. Observe as bounding-boxes dos carros identificados, fruto da interpretação dos dados fornecidos pela aplicação do kmodel sobre a imagem. Note ainda que a qualidade da imagem é baixa devido à resolução utilizada como entrada da rede, que está limitada ao padrão QVGA (320x240). Quanto maior a resolução da imagem de entrada, mais pesada seria o modelo capaz de interpretá-la, esta é outra contrapartida a ser pesada durante a avaliação de viabilidade. Neste caso não há comprometimento do resultado efetivo do dispositivo, o modelo utilizado já foi treinado considerando que as entradas teriam esta resolução.

Figura 9 – HUD proposta

**Fonte:** Autoria Própria

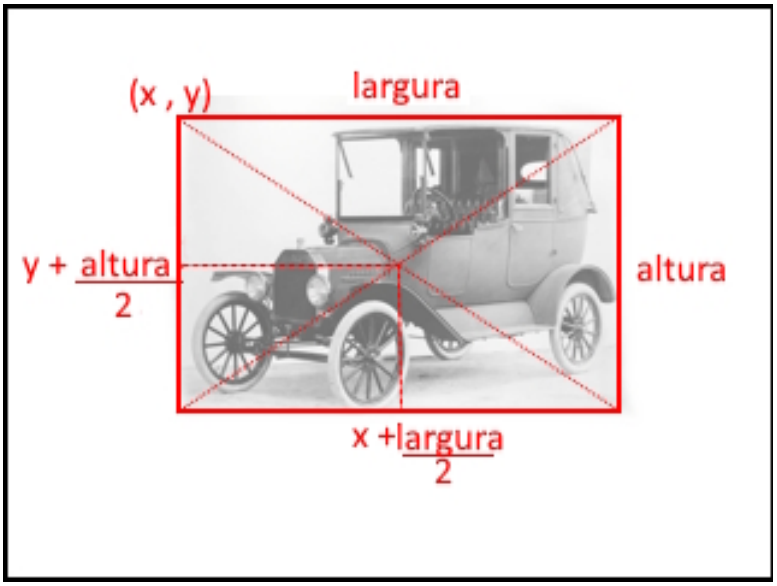
Para determinar se o objeto encontrado oferece perigo ao carro que está realizando a manobra, este documento propõe a lógica a seguir:

- Define-se as coordenadas do centro da região delimitante do objeto

  O retorno da rede após tratada a imagem é um vetor que contém as coordenadas da origem da região que contém o objeto, bem como sua largura e altura respectivas.

  Somando a coordenada X com metade da largura fornecida obtém-se a coordenada X do centro. Analogamente somando a coordenada Y com metade da altura fornecida, obtêm-se o Y do centro. Trecho ilustrado pela Figura 10

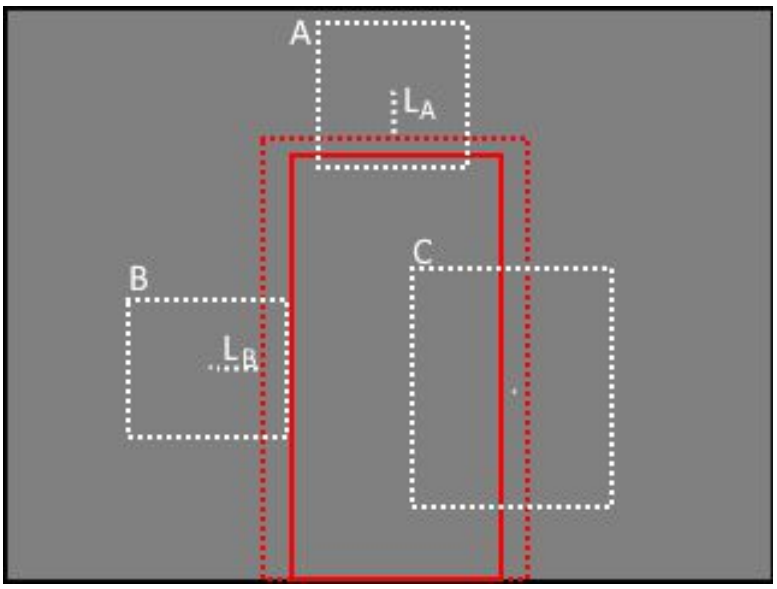
Figura 10 – Cálculo do centro do bounding-box do objeto




**Fonte:** Autoria Própria

- Compara-se as coordenadas calculadas com as da região segura pré-definida anteriormente.

Figura 11 – Representação visual da lógica de identificação de risco

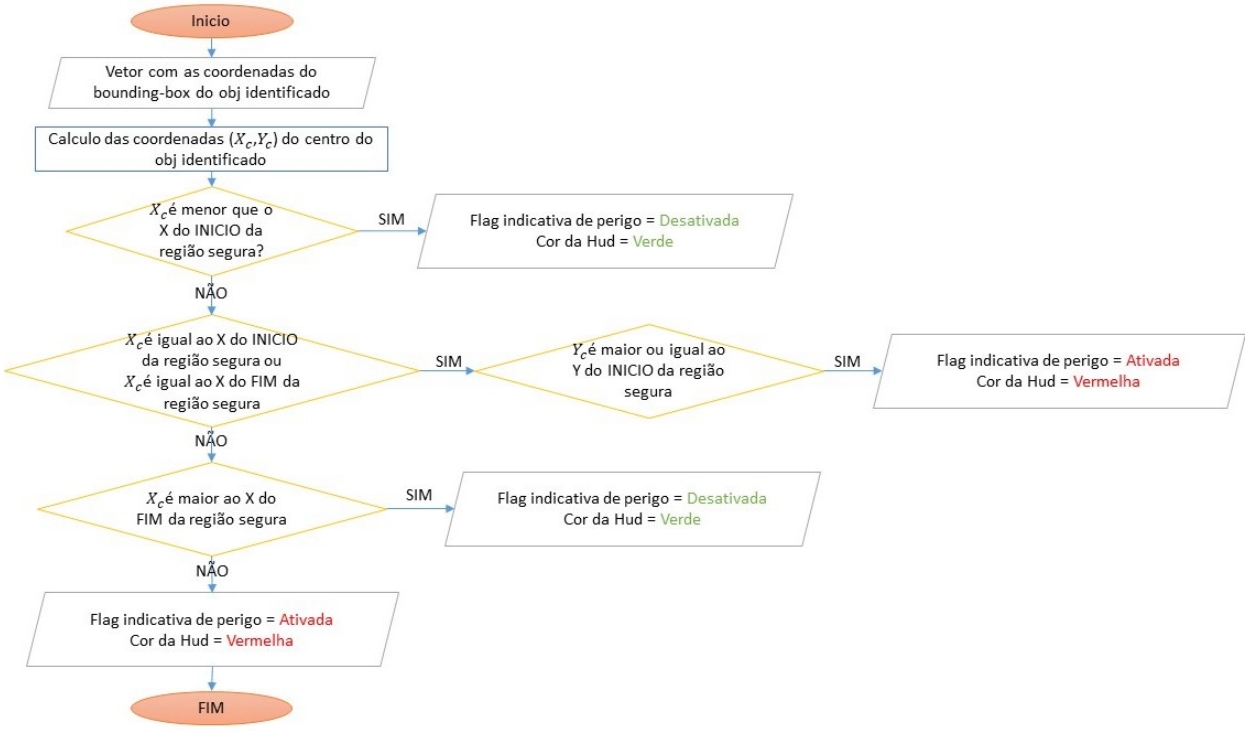



**Fonte:** Autoria Própria

A Figura 11 apresenta uma situação hipotética em que A, B e C representam bounding-boxes de três objetos identificados pela rede. A zona de segurança é delimitada pelo retângulo vermelho, enquanto o retângulo vermelho pontilhado representa uma região extra ao redor da zona de risco que da margem de segurança ao cálculo. O objeto A embora

esteja na direção do carro não apresenta risco a manobra por não estar na região de risco definida, esta inferência é avaliada através da distância $L_A$. O retângulo B também não apresenta risco imediato por, mesmo tendo parte do seu bounding-box na região de risco, o objeto não está próximo o suficiente da região, pode ser inferido pela distância $L_B$. Da mesma forma, o objeto C oferece risco a manobra, segundo a lógica proposta, por estar no caminho do veículo, pode se inferir a proximidade pela posição do centro da coordenada do respectivo bounding-box, presente na região de perigo. Embora rudimentar, esta lógica tem o poder de identificar a proximidade dos objetos com a região de risco para tomar as medidas necessárias para evitar acidentes.

Figura 12 – Implementação da lógica de detecção de riscos




**Fonte:** Autoria Própria

A Figura 12 é um exemplo da implementação descrita anteriormente. Como argumento, essa função recebe ao vetor contendo as coordenadas X e Y da origem do bounding-box do objeto, sua largura e sua altura. Em seguida, o código verifica a posição do X do centro em relação ao X do início e fim da região de perigo. Se por acaso o X do centro do bounding-box esteja entre os X de início e fim da região, então verifica-se o Y do centro do bounding-box está acima do Y do início da região de perigo.

Se for detectado perigo, as flags de perigo e de cor são alteradas. Através do estado dessas flags, o usuário é avisado da situação contundentemente em duas vias. O retângulo

que delimita a zona de segurança é exibido na cor verde, quando a flag de perigo se torna verdadeira o retângulo muda a cor para vermelho. Para um alarme sonoro, um buzzer é acionado com frequência de 800hz de alarme, como visto em (EATON, 2016), é facilmente distinguível da poluição sonora do ambiente.

## 3.4 Testes de implementação

Dada a natureza do projeto como prova de conceito, o ambiente de testes foi controlado e manipulado para verificar possíveis falhas logicas. O dispositivo sugerido neste trabalho tem aplicação definida, mas conceitualmente não existe a necessidade de replicar rigorosamente o ambiente de testes específico do caso de uso abordado. Sendo assim, embora o dispositivo seja projetado para funcionar como câmera de ré, devido ao seu modelo conseguir identificar as classes já mencionadas em qualquer imagem, optou-se por flexibilizar o feed de imagens a fim de enxergar inclusive novas aplicações não antes aventadas.

Para validar a implementação, o protótipo foi posicionado de maneira que sua câmera enquadrasse uma tela de LCD de forma tal que não fosse possível distinguir que as imagens não eram capturadas de uma cena real ou de um video público. Na tela foram reproduzidos vídeos diversos contendo câmeras de dashboard de carros, câmeras de capacete de motociclistas e câmeras de policiais em patrulhas rodoviárias, nenhuma com intenção específica, além de testar o algoritmo de identificação em condições de trânsito verossímeis.

Figura 13 – Frame de Exemplo para teste da rede


**Fonte:** Adaptado de Brasil (2020).

A Figura 13 anterior é um frame de um dos vídeos usados como feed para teste. A seguir a imagem extraída da ferramenta de depuração já com os overlays apresentados na tela do dispositivo. Além dessas informações, na tela do dispositivo ainda se pode se ver as tags com os nomes dos objetos identificados. Estas labels não estão presentes na Figura 14 por elas não serem aplicadas sobre o framebuffer da câmera, e sim desenhados direto na tela de LCD do dispositivo por meio dos métodos nativos da biblioteca de interface do kit com a tela. Na dita Figura pode-se observar os veículos identificados pela rede com suas respectivas bounding-boxes.

Figura 14 – Framebuffer do dispositivo com a câmera apontada para a Figura 14

**Fonte:** Autoria Própria

O código de exemplo para implementação da solução de software proposta pode ser encontrado no repositório github (LEITE, 2023).

# 4 Resultados e Discussões

O objetivo de produzir um estudo de caso acerca da área de visão computacional embarcada foi alcançado. O resultado do projeto é um protótipo funcional que demonstra a possibilidade de um produto aplicados os devidos refinamentos.

A necessidade de testes em situações reais é notória e deve ser considerada quando o projeto evoluir. Falhas intrínsecas do quotidiano não podem ser previstas em ambiente controlado e, por se tratar de uma proposta de dispositivo de segurança, não podem ser tratadas levianamente em um produto disponível ao público.

A lógica de identificação de perigo carece de lapidação nas etapas mais avançadas de prototipagem. Esta é a função cerne do equipamento e não pode haver nenhuma possibilidade de falso negativo. Uma lógica mais abrangente ou mais casos de teste podem ser implementados para melhorar a confiabilidade desta função.

A discussão sobre velocidade de resposta entre linguagem interpretada e compilada não é nova e ao nível de microcontroladores é ainda mais claro que implementações compiladas são mais responsivas. É preciso avaliar se o ganho de velocidade valerá a pena em troca das funcionalidades que o MaixPy favorece. Trocar apenas o script executado pela placa na inicialização, podendo ser feito inclusive através da internet, dada a possibilidade de conexão que o kit fornece, é uma vantagem que não pode ser descartada.

A escassez de documentação é um entrave a ser suplantado. Observa-se que a página de referência do fabricante carece de informações mais aprofundadas. A lista de comandos mostrados na página de referência da linguagem tem explicações rasas sobre a sintaxe e aplicações. A estrutura de memória do núcleo, por exemplo, é abordada junto a documentação da linguagem do kit. Faltam informações sobre prioridades de interrupção. Embora seja possível configurar interrupções de hardware e software as mesmas ao nível de usuário, não há garantia alguma de que tais interrupções interferirão no funcionamento da aplicação. Sobrescrever prioridades de timers e outros módulos internos do sistema é ma questão delicada neste campo, supõe-se por boa prática que não, mas sem definição documental relega-se esta inferência apenas a esperança, o que é inaceitável numa aplicação que demande confiabilidade.

Estas falhas são corrigidas nas comunidades de usurários do kit, entretanto as mesmas estão escritas no idioma da região com usuários mais ativos, mandarim, a saber. O próprio site da documentação é escrito neste idioma, embora possua a versão em inglês, esta e feita de forma automática, o que já é sabido conter alguns erros de concordância que prejudicam o entendimento. Além disso, a tradução dos conteúdos só torna clara a vaga documentação que a pagina oferece. Uma possibilidade de explicação para esta falha é que

ao lançar o produto o fabricante criou a documentação para suprir informações iniciais aos usuários e num futuro próximo às lacunas seriam preenchidas. Este futuro pode não ter se concretizado por mudanças de foco da empresa ou baixa aceitação do kit como produto no mercado, especulações plausíveis para um campo em que tecnologias evoluem rápido e produtos se tornam obsoletos antes mesmo de chegar ao consumidor.

# 5 Conclusões e Trabalhos Futuros

A proposta deste projeto foi apresentar um estudo de caso, cujo objeto de avaliação seria a produção de um protótipo implementando visão computacional em um dispositivo embarcado por meio dos avanços da computação de borda. Pôde-se, com o resultado apresentado, avaliar que de fato esta aplicação é totalmente viável e numa visão de escopo mais amplo apresenta oportunidades de mercados ainda não explorados. Pouco foi encontrado além de trabalhos acadêmicos sobre o assunto e o protótipo produzido tem grande potencial de evoluir para um hardware proprietário com valor agregado considerável. Como proposta implícita, a modularidade do dispositivo também pode ser ressaltada. O equipamento de inteligência independe do veículo no qual está implantado, de maneira que até mesmo sua alimentação pode ser fornecida por meios independentes e versáteis. A fonte de imagens não tem conexão limitante ao módulo também, de maneira que o feed de imagens de um sistema de captura já presente no veículo pode ser utilizado sem muito esforço.

A disposição do equipamento também é altamente customizável, posto que a identificação depende exclusivamente de uma entrada de imagem. Essa característica não limita o projeto ao padrão de câmeras de ré utilizadas no mercado atualmente, podendo inclusive ser reimaginada a fim de atender os mais diversos veículos e suas dimensões intrínsecas. Considere a presença deste equipamento, por exemplo, em veículos longos como caminhões, cuja visão da parte traseira é prejudicada pela natureza do veículo. Outro exemplo podem ser os ciclomotores, bicicletas e motocicletas, onde o posicionamento do condutor para visualizar a parte traseira pode colocá-lo em situação de risco. A modularidade do protótipo aqui proposto permite que estes veículos se beneficiem da segurança proveniente da utilização da implementação e permite inclusive novas possibilidades de lógicas embarcadas.

Alguns automóveis mais modernos contam com câmeras auxiliares posicionadas nas laterais para atuação em manobras de conversão. A implementação do projeto aqui proposto para uma ferramenta que mitigaria os pontos mortos na visão do condutor é extremamente plausível.

A tecnologia aqui comprovada tem um potencial imenso e a cada instante aumenta mais. As pesquisas em torno de inteligência artificial estão em voga no momento desta publicação. Aplicações como o ChatGPT da OpenAI, o DLSS da Nvidia, o Copilot do Github e o Adobe Firefly são provas contundentes de como a inteligência artificial avança a passos largos e deixa o âmbito acadêmico e militar para exercer um papel fundamental na sociedade. Simplificar tarefas do dia a dia das pessoas e tornar a interação da humanidade

com o mundo a sua volta de maneira sustentável e segura deve ser a premissa mais fundamental do engenheiro e este documento atesta que as possibilidades são imensas e estão aumentando cada vez mais.

# Referências Bibliográficas

AIWINTERMUTEAI. *GitHub - AIWintermuteAI/aXeleRate: Keras-based framework for AI on the Edge*. 2021. Disponível em: <https://github.com/AIWintermuteAI/aXeleRate>. Citado na página 23.

BANDARU, R. *Pruning Neural Networks - Towards Data Science*. Towards Data Science, 2020. Disponível em: <https://towardsdatascience.com/pruning-neural-networks-1bb3ab5791f9>. Citado na página 14.

BRASIL, E. *ACIDENTE ENTRE MOTOS DURANTE A ESCOLTA*. 2020. Disponível em: <https://www.youtube.com/watch?v=kAlUEhRSzi0>. Citado na página 34.

CMSIS. *CMSIS NN Software Library*. 2019. Disponível em: <https://www.keil.com/pack/doc/CMSIS/NN/html/index.html>. Citado na página 15.

EATON. *Asserta A1 Eaton Fire Tone*. 2016. Disponível em: <https://videos.eaton.com/detail/video/6060800815001/asserta-a1-eaton-fire-tone?autoStart=true&q=Asserta%20A1>. Citado na página 34.

GOOGLE. *Dataset Search*. Google Dataset Search, 2013. Disponível em: <https://datasetsearch.research.google.com/help>. Citado na página 24.

HORCASITAS, J. *What Is REPL?* DigitalOcean, 2021. Disponível em: <https://www.digitalocean.com/community/tutorials/what-is-repl>. Citado na página 27.

INSTRUCTABLES. *Image Recognition With K210 Boards and Arduino IDE/Micropython*. Instructables, 2019. Disponível em: <https://www.instructables.com/Transfer-Learning-With-Sipeed-MaiX-and-Arduino-IDE/>. Citado na página 25.

KERAS. *Keras: Deep Learning for humans*. 2018. Disponível em: <https://keras.io/>. Citado na página 29.

LAPIX. *Mean Average Precision*. UFSC, 2019. Disponível em: <https://lapix.ufsc.br/ensino/visao/visao-computacionaldeep-learning/visao-computacionalmetricasmean-average-precision/>. Citado na página 16.

LEITE, L. L. *leonardowolf/img-detection-app*. 2023. Disponível em: <https://github.com/leonardowolf/img-detection-app>. Citado na página 35.

LIU, W. et al. *SSD: Single Shot MultiBox Detector*. 2016. Disponível em: <https://arxiv.org/pdf/1512.02325.pdf>. Citado na página 16.

MARTINS, W. M. *ESTUDO DE ALGORITMOS DE VISÃO COMPUTACIONAL PARA IDENTIFICAÇÃO E DESVIO DE OBJETOS EM TEMPO REAL: UMA APLICAÇÃO PARA QUADROTORES*. 2018. Disponível em: <https://repositorio.unifei.edu.br/jspui/bitstream/123456789/1874/1/dissertacao_2019009.pdf>. Citado na página 12.

MATHWORKS. *Anchor Boxes for Object Detection - MATLAB & Simulink*. 2023. Disponível em: <https://www.mathworks.com/help/vision/ug/anchor-boxes-for-object-detection.html>. Citado na página 16.

MOUSER. *Sipeed Maixduino Kit for RISC-V AI + IoT*. 2022. Disponível em: <https://br.mouser.com/pdfDocs/Product_Overview-SeeedStudioSipeedMaixduinoKit.pdf>. Citado na página 22.

NEUCRACK. *MaixHub*. 2023. Disponível em: <https://maixhub.com/model/zoo/63>. Citado 2 vezes nas páginas 26 e 29.

PASCAL. *The PASCAL Visual Object Classes Challenge 2007 (VOC2007)*. PASCAL, 2023. Disponível em: <http://host.robots.ox.ac.uk/pascal/VOC/voc2007/>. Citado 2 vezes nas páginas 16 e 26.

REDMON, J.; FARHADI, A. *YOLO9000: Better, Faster, Stronger*. 2016. Disponível em: <https://arxiv.org/pdf/1612.08242.pdf>. Citado 2 vezes nas páginas 16 e 24.

REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 39, p. 1137–1149, 06 2017. Disponível em: <https://ieeexplore.ieee.org/document/7485869>. Citado na página 16.

RISCV. *History – RISC-V International*. 2015. Disponível em: <https://riscv.org/about/history/>. Citado na página 17.

SATO, D. *Sistema De Detecção De Animais Nas Rodovias Com Aprendizado De Máquina E Visão Computacional*. 2022. Disponível em: <https://www.teses.usp.br/teses/disponiveis/10/10134/tde-01122022-160523/publico/Denis_Sato_corrigida.pdf>. Citado na página 12.

SIPEED. *MaixHub*. 2019. Disponível em: <https://maixhub.com/home>. Citado na página 25.

SIPEED. *Sipeed Maixduino Datasheet v1.0 Key Feature*. 2019. 4 p. Disponível em: <https://cxem.net/ckfinder/userfiles/a879191a668fd1d6c525814f0fccb5e3/files/MAIXDUINO/Sipeed_Maixduino_Datasheet_V1_0.pdf>. Citado 2 vezes nas páginas 19 e 20.

SIPEED. *Releases MaixpySipeed*. 2020. Disponível em: <https://dl.sipeed.com/shareURL/MAIX/MaixPy/release/master/maixpy_v0.6.2_85_g23d09fbcc>. Citado na página 29.

SIPEED. *Board - Sipeed Wiki*. 2022. Disponível em: <https://wiki.sipeed.com/soft/maixpy/en/api_reference/builtin_py/board_info.html>. Citado na página 28.

SIPEED. *Introduction to Storage System - Sipeed Wiki*. 2022. Disponível em: <https://wiki.sipeed.com/soft/maixpy/en/get_started/get_started_fs.html>. Citado 2 vezes nas páginas 26 e 27.

SIPEED. *KPU - Sipeed Wiki*. 2022. Disponível em: <https://wiki.sipeed.com/soft/maixpy/en/api_reference/Maix/kpu.html#Module-method>. Citado na página 29.

SIPEED. *MaixDuino Development Board - Sipeed Wiki*. 2022. Disponível em: <https://wiki.sipeed.com/hardware/en/maix/maixpy_develop_kit_board/maix_duino.html>. Citado na página 18.

SIPEED. *MF Firmware related upgrade instructions - Sipeed Wiki*. Sipeed, 2022. Disponível em: <https://wiki.sipeed.com/hardware/maixface/en/mf_ml_module/mf_update_firmwave.html?highlight=mchine%20key>. Citado na página 26.

SIPEED. *sipeedkflash_gui: Cross Platform GUI Wrapper for kflash.py (download(/burn) Tool for k210)*. 2022. Disponível em: <https://github.com/sipeed/kflash_gui>. Citado na página 29.

SIPEED. *Update MaixPy Firmware - Sipeed Wiki*. 2022. Disponível em: <https://wiki.sipeed.com/soft/maixpy/en/get_started/upgrade_maixpy_firmware.html#Get-the-firmware>. Citado na página 29.

SIPEED. *sipeed/MaixPy_scripts: micropython scripts for MaixPy*. 2023. Disponível em: <https://github.com/sipeed/MaixPy_scripts/tree/master>. Citado na página 28.

TECHNOLOGY, S. *Sipeed M1 Datasheet v1.12 Key Features*. 2021. Disponível em: <https://dl.sipeed.com/fileList/MAIX/HDK/Sipeed-M1&M1W/Specifications/Sipeed%20M1%20Datasheet%20EN%20V1.12.pdf>. Citado na página 20.

THINGIVERSE.COM. *A case for Sipeed Maixduino by komix*. 2023. Disponível em: <https://www.thingiverse.com/thing:3768964>. Citado 2 vezes nas páginas 22 e 23.

WAKSMAN, R. D.; MARIA, R.; BLANK, D. *Acidentes de transporte*. [S.l.]: Ufrgs, 2014. Citado na página 13.