Advanced Driver Assistance Systems (ADAS) para Integração em Todo o Tipo de Veículos

Mestrado em Engenharia Automóvel

Celso Augusto Lourenço Luz

Leiria, abril de 2026

Advanced Driver Assistance Systems (ADAS) para Integração em Todo o Tipo de Veículos

Mestrado em Engenharia Automóvel

Celso Augusto Lourenço Luz

Trabalho de Projeto realizado sob a orientação do Professor Doutor Carlos Ferreira e do Professor Doutor Nuno Vieira Lopes

Leiria, abril de 2026

Originalidade e Direitos de Autor

O presente relatório de projeto é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Reproduções parciais deste documento serão autorizadas na condição de que seja mencionado o Autor e feita referência ao ciclo de estudos no âmbito do qual o mesmo foi realizado, a saber, Curso de Mestrado em Engenharia Automóvel, no ano letivo 2024/2025, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, Portugal, e, bem assim, à data das provas públicas que visaram a avaliação destes trabalhos.

Dedicatória

Dedico este projeto a todos os meus entes mais próximos que, ao acreditarem sempre nas minhas capacidades, me deram a força necessária para me sentir capaz de superar este desafio.

Agradecimentos

Agradeço, com a maior sinceridade, a todos os meus familiares e amigos que contribuíram para a realização deste projeto, quer tenha sido através do apoio técnico, quer pelo indispensável incentivo pessoal.

Um agradecimento especial à minha namorada, Ana Duarte, por todo o apoio e compreensão demonstrados ao longo de todo o meu percurso de mestrado. Agradeço igualmente ao meu amigo Tomás Rei, pelo apoio e companheirismo constantes desde o primeiro ano de licenciatura até à conclusão desta etapa.

Finalmente, expresso a minha profunda gratidão aos meus orientadores, os Professores Carlos Ferreira e Nuno Lopes, por todo o conhecimento e apoio disponibilizados durante o desenvolvimento deste projeto. A sua disponibilidade, mesmo fora de horários letivos, para me auxiliarem em todas as questões levantadas, foi absolutamente fundamental para a orientação e sucesso deste trabalho.

A todos os que de alguma forma contribuíram para o meu sucesso académico e pessoal, o meu muito obrigado.

Resumo

O presente projeto descreve o desenvolvimento e a implementação de um sistema avançado de assistência à condução (ADAS — *Advanced Driver Assistance System*), focado na monitorização em tempo real da trajetória do veículo, alerta de perigos e na prevenção de colisões frontais. O sistema utiliza uma unidade de processamento NVIDIA Jetson Xavier AGX, combinando técnicas de processamento computacional de redes neuronais para a deteção de objetos, obstáculos e faixas de rodagem. Para a deteção de objetos, foi implementado o modelo YOLOv8, enquanto para a identificação das faixas de rodagem utilizou-se o modelo UFLDv2, otimizado para alta velocidade de processamento.

A arquitetura do sistema assenta numa estratégia de fusão sensorial, integrando dados de profundidade provenientes de uma câmara Intel RealSense e de um sensor LiDAR, cruzando-os com a telemetria do veículo (velocidade e estado dos pedais) obtida via barramento CAN. No âmbito do *hardware*, o projeto evoluiu de uma fase de prototipagem em *breadboard* para a conceção e fabrico de uma Placa de Circuito Impresso (PCB) dedicada.

Os resultados demonstram que o sistema é capaz de processar cenários dinâmicos com baixa latência, ativando alertas graduais baseados no cálculo do tempo para colisão e no desvio lateral do veículo da faixa de rodagem. A lógica de decisão implementada permite distinguir entre avisos preventivos e alertas de perigo crítico, adaptando a resposta visual e física ao comportamento real do condutor, garantindo assim uma solução de segurança ativa robusta e fiável.

Palavras-chave: ADAS, NVIDIA, YOLO, UFLD, Fusão Sensorial, CAN, PCB.

Abstract

This project describes the development and implementation of an Advanced Driver Assistance System (ADAS), focused on real-time vehicle trajectory monitoring, hazard alerting, and forward collision prevention. The system utilizes an NVIDIA Jetson Xavier AGX processing unit, combining neural network computational techniques for the detection of objects, obstacles, and lane markings. The YOLOv8 model was implemented for object detection, while the UFLDv2 model, optimized for high-speed processing, was used for lane identification.

The system architecture is based on a sensor fusion strategy, integrating depth data from an Intel RealSense camera and a LiDAR sensor, cross-referencing them with vehicle telemetry (speed and pedal status) obtained via the CAN bus. Regarding the *hardware*, the project evolved from a breadboard prototyping phase to the design and manufacture of a dedicated Printed Circuit Board (PCB).

The results demonstrate that the system is capable of processing dynamic scenarios with low latency, activating gradual alerts based on Time-to-Collision (TTC) calculations and the vehicle's lateral deviation from the lane. The implemented decision logic allows for the distinction between preventive warnings and critical danger alerts, adapting the visual and physical response to the driver's actual behavior, thus ensuring a robust and reliable active safety solution.

Keywords: ADAS, NVIDIA, YOLO, UFLD, Sensor Fusion, CAN, PCB.

Índice

Lista de Figuras

Lista de tabelas

Lista de siglas e acrónimos

ACC	Adaptative Cruise Control
ADAS	Advanced Driver Assistance Systems
AEB	Autonomous Emergency Brake
AKC	Acknowlegment
BSM	Blind Spot Monitoring
CAN	Controller Area Network
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
EOF	End of Frame
Euro NCAP	The European New Car Asessment Programme
FCW	Forward Collision Warning
FPS	Frames Per Second
FSIPLeiria	Formula Student Instituto Politécnico de Leiria
FVSA	Forward Vehicle Start Alarm
GPIO	General-Purpose Input/ Output
GPU	Graphics Processing Unit
GPS	Global Positioning System
HDMI	High-definition Multimedia Interface
HLS	Hue, Lightness, Saturation
HMW	Headway Monitoring & Warning
HOG	Histograma de Gradientes Orientados
I2C	Inter-Integrated Circuit
IA	Inteligência Artificial
ID	Identificador
IMU	Inertial Measurement Unit
LDW	Lane Departure Warning

LED	Light-Emitting Diode
LiDAR	Light Detection and Ranging
LKA	Lane Keep Assist
mAP	Mean Average Precision
NMS	Non-Maximum Suppression
OBD	On-Board Diagnosis
OBD-I	On-Board Diagnosis I
OBD-II	On-Board Diagnosis II
PCB	Printed Circuit Board
PCW	Pedestrian & Cyclist Collision Warning
PWM	Pulse Width Modulation
RAM	Random Acess Memory
R-CNN	Regional Convolutional Neural Network
RGB	Red, Green, and Blue
RLVW	Red Light Violation Warning
ROI	Região de Interesse
RPM	Rotações por minuto
SAE	Society of Automotive Engineers
SLI	Speed Limit Indication
SOF	Start of Frame
SPI	Serial Peripheral Interface
TLW	Traffic Light Warning
ToF	Time of Flight
TTC	Time-To-Collision
UE	União Europeia
UFLD	Ultra-Fast Lane Detection
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
YOLO	You Only Look Once

1. Introdução

Os Sistemas Avançados de Assistência ao Condutor, do inglês Advanced Driver Assistance Systems (ADAS) referem-se a todo o tipo de sistemas utilizados em veículos automóveis que providenciam assistência ao condutor durante a condução, alertando para eventuais situações de risco. Esses sistemas utilizam uma combinação de sensores, câmaras, radares e algoritmos avançados para monitorizar o ambiente em redor do veículo e ajudar o condutor a tomar decisões mais seguras por meio de alertas, automatizações e outras funções destinadas a prevenir acidentes.

1.1. Motivação e Objetivos

No ano de 2024, de janeiro a maio, em Portugal, foram registados 14 658 acidentes com vítimas, entre elas, 182 mortais [1]. Relativamente a dados mundiais, cerca de 1.19 milhões de pessoas morrem todos os anos devido a acidentes de tráfego rodoviário, onde entre as causas dos acidentes se destaca o erro humano, nomeadamente, excesso de velocidade, condução sobre o efeito de estupefacientes e distrações (como o uso de dispositivos móveis) [2]. Em contraste, um estudo na Irlanda chegou à conclusão de que o uso de avisos em tempo real de situações de perigo diminui drasticamente o comportamento de risco e distrações durante a condução. O aviso de distração reduziu eventos de saída de faixa e colisão frontal em mais de 50% após seis meses de uso. Para além disso, situações de acelerações e travagens bruscas diminuíram 76 % e 65 %, respetivamente, com o uso de sistemas ADAS [3], demonstrando a eficácia destes sistemas e as suas capacidades.

Segundo a European Union General Safety Regulation [4], a partir de 7 de julho de 2024, todos os veículos ligeiros novos devem sair de produção com as seguintes características de segurança pelo menos: Adaptive Cruise Control (ACC), câmara de marcha-atrás ou sensores, aviso visual ou sonoro em caso de sonolência do condutor, aviso de travagem de emergência e medidas de segurança cibernética. Para carros e carrinhas de mercadorias, os seguintes sistemas são obrigatórios: assistente de centralização de faixa, travagem de emergência com deteção de pedestres, veículos e ciclistas e gravadores de informações de eventos. Para autocarros e pesados, são os

seguintes sistemas: deteção e aviso de eventuais colisões com ciclistas e pedestres, e sistema de monitorização da pressão dos pneus.

Como os sistemas ADAS não são uniformes entre fabricantes de automóveis, e como são tecnologias sujeitas a otimizações constantes, requerem testes de validação, de modo a garantir o seu desempenho correto em qualquer tipo de condição. Para tal, a companhia The European New Car Assessment Programme (Euro NCAP) [5] criou um sistema de avaliação de segurança de cinco estrelas. Este sistema é composto por seis níveis, onde o mais baixo é de 0 estrelas e apresenta o mínimo de sistemas de segurança avançados, e o nível mais alto corresponde ao de 5 estrelas, onde o veículo apresenta um desempenho de excelência em sistemas de segurança [6]. O processo de validação e avaliação destes sistemas de segurança, segue um protocolo rigoroso com um conjunto de testes a realizar, consoante o tipo de sistemas de segurança incorporados no veículo. Estes testes são realizados em ambiente de vida real e não virtual. Alguns dos parâmetros de avaliação do sistema de segurança podem envolver tempo de resposta do sistema e comportamento face a situações de alto risco [7]. Na Figura 1-1 pode-se analisar um dos testes que são realizados para avaliar a performance de um sistema ACC.

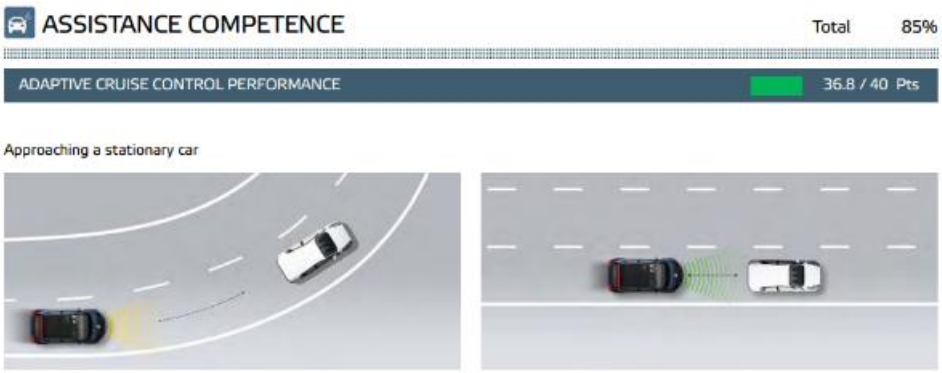


Figura 1-1 - Teste de performance de um sistema ACC de um Mercedes-Benz E400e.

Como objetivo deste trabalho, pretende-se desenvolver um sistema avançado de assistência à condução de baixo custo e que possa ser instalado em qualquer veículo segundo o diagrama da Figura 1-2. Para tal, é necessário, em primeiro lugar realizar uma revisão bibliográfica sobre os projetos já existentes, e que tipo de tecnologias utilizam nomeadamente, que tipo de *hardware* (câmaras de profundidade, Light Detection and Ranging - LiDAR, entre outros). Numa fase mais avançada, pretende-se projetar e implementar sistemas de alerta ao condutor luminosos e/ou sonoros assim como,

implementação de um barramento de comunicação Controller Area Network (CAN), para diagnóstico, configuração e comunicação com os sistemas do veículo.

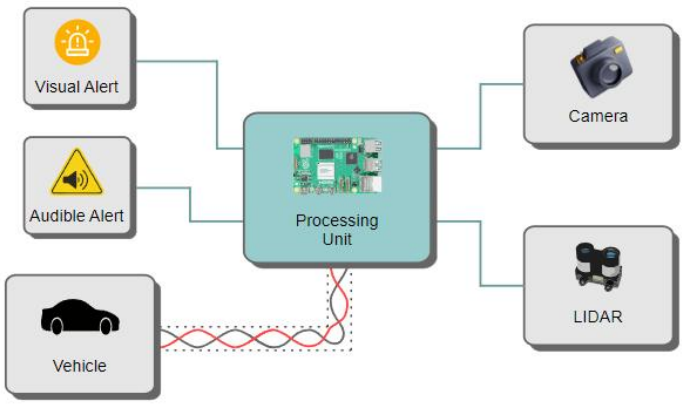


Figura 1-2 - Diagrama de blocos referente ao objetivo do projeto.

1.2. Estrutura do Trabalho

Este documento encontra-se dividido em seis capítulos, onde cada um é organizado em diferentes secções abordando diferentes aspetos do trabalho desenvolvido.

No Capítulo 1 apresenta-se o tema em estudo, explicando o contexto e os objetivos para com o trabalho e a estruturação do mesmo.

No Capítulo 2 é feita uma revisão bibliográfica de trabalhos semelhantes ao do presente estudo, abordando diferentes aspetos que servem como apoio ao desenvolvimento do projeto, tais como as bases tecnológicas, quer sejam a nível de *hardware* como de *software*.

No Capítulo 3 descrevem-se todos os componentes selecionados para o projeto, justificando-se as escolhas com base nos requisitos de desempenho. São detalhados os componentes sensoriais, as interfaces de comunicação e os periféricos de interface com o utilizador

No Capítulo 4 aborda-se a implementação prática dos algoritmos de processamento de imagem e a lógica de fusão sensorial. Este capítulo descreve ainda a fase de prototipagem e a transição para a conceção da placa de circuito impresso.

No Capítulo 5 são apresentados testes de funcionamento do sistema em diferentes cenários, acompanhados por uma análise crítica do desempenho do sistema, da sua precisão e eficácia.

No Capítulo 6 são apresentadas as considerações finais sobre o trabalho desenvolvido, as principais limitações encontradas e as sugestões de melhoria para futuros trabalhos.

2. Estado da Arte

Os sistemas ADAS têm desempenhado um papel fundamental na evolução da segurança automóvel, ajudando a prevenir acidentes e a melhorar a experiência de condução [8]. Neste contexto, é essencial compreender as diferentes classificações e tipos de tecnologias de sistemas ADAS disponíveis atualmente, que vão desde funções básicas de alerta até níveis avançados de automação. A seguir, serão discutidos os níveis de categorização dos sistemas ADAS e os principais tipos de funcionalidades oferecidas, com foco nos componentes tecnológicos que possibilitam a sua operação.

2.1. Classificação de Sistemas ADAS

Existem diversos tipos de sistemas ADAS com diferentes funcionalidades. Com a evolução destes sistemas surgiu uma categorização dos mesmos por parte da Society of Automotive Engineers (SAE Internacional) como se pode analisar na Figura 2-1. Existem seis níveis, onde cada serve um propósito específico na segurança durante a condução [9]:

- **Nível 0:** O sistema é capaz de fornecer informações ao condutor sem nenhum tipo de controlo sobre o veículo. Alguns desses sistemas podem ser: deteção de ângulo morto, reconhecimento de sinais de trânsito e aviso de saída da faixa de rodagem;
- **Nível 1:** O sistema é capaz de atuar no sistema de direção do veículo ou sobre o sistema de travagem ou aceleração assistindo assim o condutor. Alguns desses sistemas podem ser: ACC ou centralização na faixa de rodagem do veículo;
- **Nível 2:** Os sistemas são semelhantes aos do nível 1, no entanto, no nível 2 o veículo pode usar mais que um desses sistemas em simultâneo;
- **Nível 3:** Estes sistemas conseguem operar o veículo sem assistência do condutor, mas sobre condições específicas. Sempre que o sistema pede, o condutor terá de assumir o comando do veículo. Alguns destes sistemas podem ser: *traffic jam chaffeur* onde o veículo tem capacidade de condução autónoma em situações de congestionamento;
- **Nível 4:** Estes sistemas são semelhantes ao nível 3 em que só operam em situações muito específicas, embora não requeiram que o condutor assuma o comando do veículo.

- **Nível 5:** Semelhante ao nível 4, no entanto, neste caso o veículo é capaz de condução autónoma em qualquer circunstância e em qualquer local.

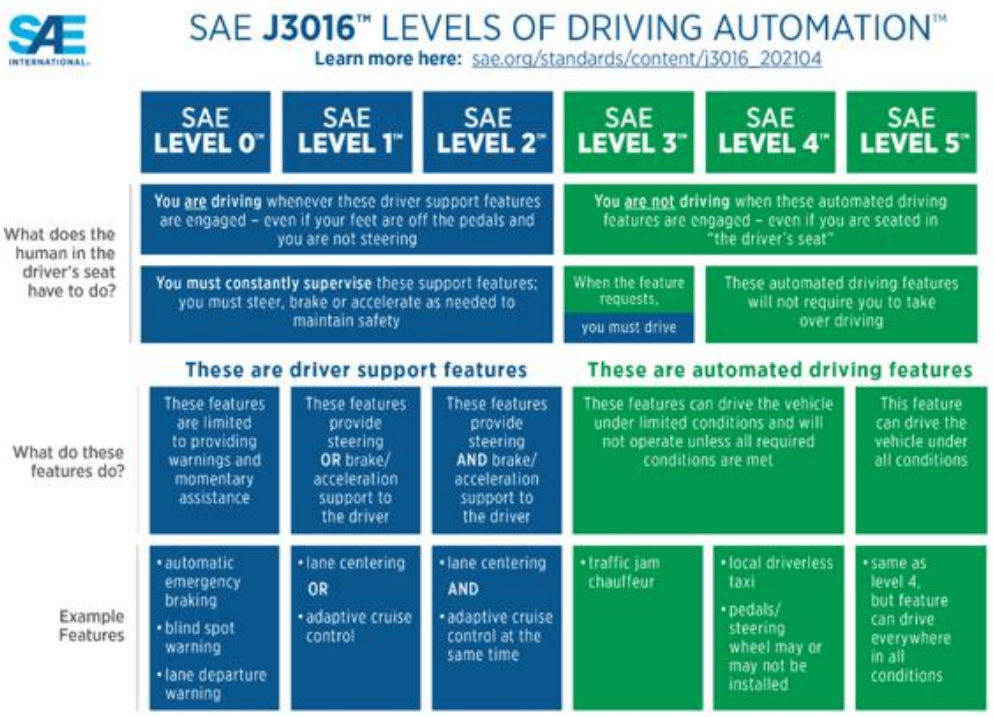


Figura 2-1 - Níveis de condução autónoma segundo SAE International.

É do conhecimento geral que a grande parte dos veículos, atualmente, estão incorporados com sistemas ADAS de pelo menos nível 0, como é o exemplo do *Audi Side Assist,* que se trata de um sistema de ângulo morto que fornece avisos visuais e/ou auditivos ao condutor sem quaisquer tipos de intervenções sobre o veículo [10]. Isto indica que, sistemas mais avançados como os de nível 3, apenas estão disponíveis em veículos de uma gama mais alta como o Assistente de autoestrada da Mercedes-Benz, que permite condução autónoma até uma velocidade de 60 km/h em determinados troços de autoestradas adequados (Alemanha, Nevada e Califórnia) [11]. Estas marcas automóveis notórias pelas suas gamas de veículos luxuosos, são conhecidas também pela segurança dos seus veículos, que em grande parte se deve aos sistemas ADAS que os incorporam, mas que, por serem sistemas caros de desenvolver, apenas estão disponíveis em veículos de gama mais alta.

2.2. Tipos de Sistemas ADAS

Com o passar dos anos, os avanços tecnológicos a nível de sistemas de segurança de veículos têm sido notáveis e de extrema importância para a diminuição do número de acidentes ligeiros e fatais nas estradas. Em 2019, 95% dos acidentes rodoviários na União Europeia (UE) foram causados por erro humano onde, nesse mesmo ano, 22800 pessoas faleceram e cerca de 120000 ficaram gravemente feridas. Estatísticas do Parlamento Europeu mostram também que a utilização de sistemas de condução autónoma, poderiam salvar cerca de 25000 vidas por ano [12].

Atualmente, existem diversos tipos de sistemas ADAS que providenciam assistência ao condutor durante a condução e, que resultam na prevenção de inúmeros acidentes rodoviários. Esses sistemas fazem recurso de vários sensores e câmaras localizados em diferentes localizações no veículo como mostra a Figura 2-2.

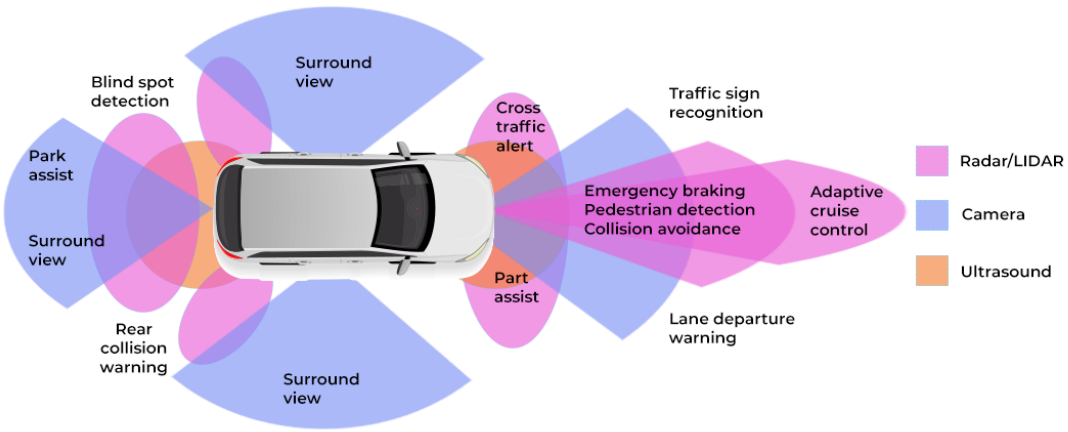


Figura 2-2 - Esquema ilustrativo da localização e alcance de sistemas ADAS no veículo.

Alguns desses sistemas ADAS que podem existir num veículo atualmente são os seguintes [13]:

- **ACC – *Adaptive Cruise Control*:** consiste numa tecnologia semelhante ao *cruise control* comum que estabiliza o veículo numa velocidade pré-definida pelo condutor, no entanto, o *cruise control* adaptativo faz ajustes automaticamente à velocidade do veículo consoante a velocidade dos veículos que circulavam à frente do próprio, ou seja, mantendo uma distância segura dos veículos que o rodeiam. Este sistema tipicamente utiliza radares ou LiDAR que se encontram

situados na frente do veículo para medir distâncias e velocidades relativas dos veículos à frente do próprio.

- **LKA – *Lane Keep Assist*:** tem como função centralizar o veículo na faixa de rodagem em que circula, realizando ajustes à direção do veículo em caso de descentralização do veículo. Este sistema utiliza um sistema de direção elétrica e câmaras situadas no para-brisas frontal do veículo, para detetar as marcações das linhas na faixa de rodagem.
- **LDW – *Lane Departure Warning*:** sistema que emite sinais sonoros e/ou visuais no caso do veículo se desviar da sua faixa de rodagem. Este sistema utiliza câmaras para deteção de linhas das marcações da faixa de rodagem à semelhança do LKA.
- **AEB – *Autonomous Emergency Brake*:** quando o sistema deteta uma colisão iminente, quer seja por um veículo que circule a uma velocidade bastante reduzida, ou pela passagem de pedestres, este aciona o sistema de travagem automaticamente quando o condutor não reage atempadamente. Este sistema também recorre ao uso de câmaras, radares e até dispositivos LiDAR situados na frente do veículo.
- **BSM – *Blind Spot Monitoring*:** é responsável por monitorizar todos os ângulos mortos do veículo e alerta o condutor para a presença de veículos tipicamente através de emissores luminosos nos espelhos retrovisores do mesmo. Este sistema pode utilizar radares, câmaras e sensores ultrassónicos situados nas laterais do veículo.
- **FCW – *Forward Collision Warning*:** sistema responsável pela deteção de possíveis cenários de colisão frontal, avisando o condutor através de sinais sonoros e luminosos. Este sistema utiliza câmaras e radares situados na frente do veículo.

Existem mais sistemas ADAS, para além dos mencionados anteriormente, no entanto, estão bastante relacionados com os acima apresentados. Pode-se verificar que o tipo de equipamento mais utilizado em sistemas ADAS são as câmaras, radares e LiDAR.

2.3. Algoritmos utilizados em ADAS

O *software* utilizado em ADAS é o motor que impulsiona a aquisição, análise e interpretação dos dados provenientes de diferentes dispositivos como câmaras, radares e

LiDAR. Estes *software* permitem que os sistemas tomem decisões em tempo real, tais como detetar obstáculos, manter o veículo na faixa de rodagem ou ajustar a velocidade para evitar colisões. Além disso, integram técnicas avançadas, como *machine learning* ou redes neuronais, para lidar com a complexidade do ambiente rodoviário. Neste capítulo, serão abordados os principais tipos de *software* que sustentam as diversas funcionalidades dos ADAS, destacando as suas características técnicas e capacidades.

As redes neuronais consistem em estruturas artificiais muito semelhantes ao cérebro humano, que é composto por diversos neurónios interligados entre si que emitem respostas mediante entradas recebidas. Estas redes podem ser compostas por diversas camadas, cada uma com um determinado número de neurónios, sendo que cada um desses neurónios está interligado aos neurónios da camada anterior e aos da camada seguinte. Existem diferentes tipos de redes neuronais com propósitos e aplicações específicas. Para o presente trabalho, irão ser abordadas apenas as redes neuronais convolucionais (do inglês Convolutional Neural Network-CNN), pois são as mais usadas para tratamento de imagens nomeadamente para deteção de objetos como se pode analisar na Figura 2-3 [14]. Uma CNN tem a capacidade de identificar características e padrões de imagens automaticamente, independentemente da posição das mesmas. Alguns exemplos destas redes neuronais são a VGG-16, ResNet50 e EfficientNet, que alcançam resultados considerados estado da arte e que podem ser aperfeiçoados facilmente consoante a tarefa a executar.

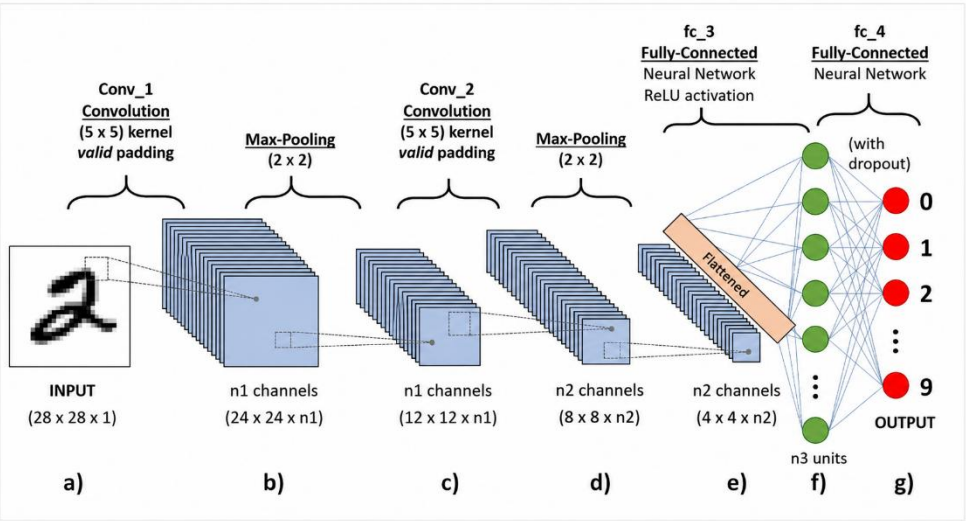


Figura 2-3 - Camadas de uma rede neuronal convolucional (CNN).

O processo começa com a imagem original (a) a passar por uma primeira camada de filtros que destaca as características mais salientes (b), reduzindo ligeiramente o seu tamanho. De seguida, uma operação de *pooling* encolhe essa imagem para metade do tamanho (c), mantendo apenas a informação mais significativa. Este ciclo repete-se logo a seguir com uma segunda camada de filtros mais complexos (d) e um novo *pooling*, tornando os dados ainda mais compactos (e). Por fim, os dados são transformados numa linha contínua de neurónios (f) que passa por uma rede tradicional, onde o sistema analisa os padrões e ativa o neurónio de saída correspondente ao número correto (g).

Na primeira camada, como o nome indica, é aplicada uma função matemática convolucional que atribui um valor a cada *pixel* da imagem de forma matricial, para posteriormente identificar curvaturas e arestas. Depois disso, são aplicadas funções de ativação que ajudam a relacionar as operações de convolução, e posteriormente identificar padrões. Na camada de *pooling*, são agrupados os aspetos mais evidentes da imagem para reduzir a memória usada pela mesma. Na última camada são organizados os *outputs* da última camada numa nova matriz usando uma função de probabilidade.

2.3.1. Deteção de Objetos

As redes neuronais podem utilizar diferentes métodos para a identificação de múltiplos objetos em imagens, nomeadamente o método de *bounding boxes* e *sliding window.* O método de *bounding boxes* consiste na atribuição de diversas caixas retangulares na imagem com tamanhos pré-definidos, denominadas de *anchor boxes,* e cálculo da probabilidade de corresponder a esse determinado objeto como se pode observar na Figura 2-4. Depois de identificado o objeto, as *bounding boxes* são refinadas para melhor ajuste ao objeto da imagem.

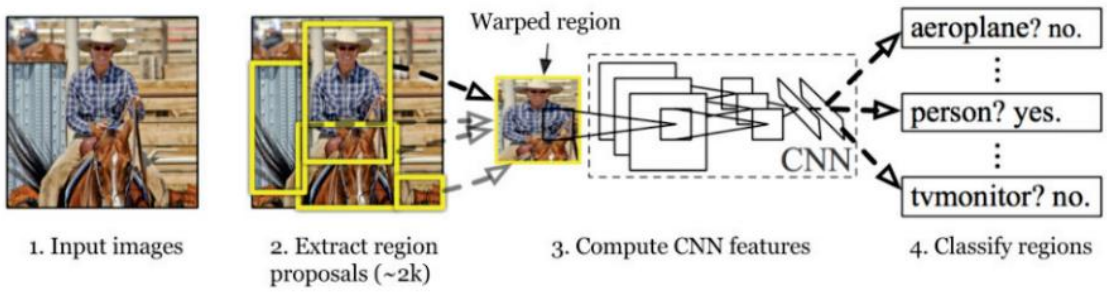


Figura 2-4 - Ilustração do método de *bounding boxes.*

Existem, essencialmente, dois tipos de redes neuronais para deteção de objetos: modelo de deteção de objetos de uma fase e de duas fases. Existem vários tipos de arquiteturas de modelos de deteção de objetos de duas fases, sendo as mais comuns: Regional Convolutional Neuronal Network (R-CNN), Fast R-CNN e Faster R-CNN [15]. Todas estas arquiteturas são muito semelhantes, diferenciando-se apenas da velocidade de análise de imagens em tempo real. Com a evolução dos sistemas ADAS surgiu a necessidade de algoritmos cada vez mais rápidos e eficientes, pelo que surgiram os modelos de deteção de objetos de uma fase, sendo um deles o You Only Look Once (YOLO), que é mais utilizado atualmente. O modelo YOLO apresenta uma melhoria significativa em relação às R-CNN, com melhor precisão e maior capacidade de análise de imagens [16].

2.3.2. Faixa de Rodagem

Relativamente à deteção de faixas de rodagem, existem diversos fatores que podem influenciar a eficiência do sistema na deteção de faixas, nomeadamente condições climatéricas como sombras, nevoeiro ou chuva, a qualidade das faixas (linhas apagadas), e até ambientes complexos como interseções ou pavimentos irregulares. Segundo Kaur e Kumar [17], o método mais típico começa pela conversão da imagem adquirida pela câmara em tons de cinza de modo a minimizar o tempo de processamento, sendo que, de seguida, a imagem deve passar por um conjunto de filtros que eliminam o ruído (como o Gaussiano). Posteriormente, é aplicado um algoritmo para identificação das faixas, em que o mais utilizado é o *Canny Edge*. Este algoritmo produz uma imagem com as arestas das faixas de rodagem. Após isso, é implementada a Transformada de Hough que identifica linhas retas e curvas associadas às faixas de rodagem. Por fim, é feita uma modelação matemática de modo a ajustar a forma das faixas à curvatura real. Esta sequência de processos para identificação de faixas de rodagem é demonstrada na Figura 2-5 e segue a seguinte ordem: imagem original, imagem filtrada, imagem em tons de cinza, imagem binária, imagem suavizada, imagem processada por *Canny Edge Detector*, imagem suavizada e imagem final com linhas marcadas.

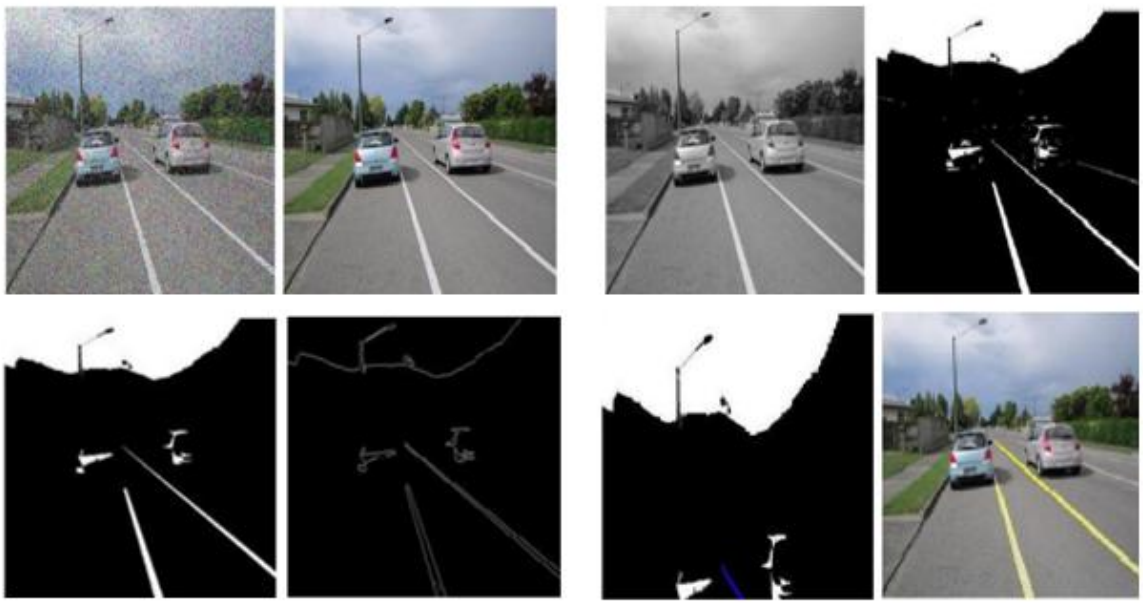


Figura 2-5 - Sequência de processos de deteção de faixas de rodagem.

A Figura 2-6 demonstra o processo de aplicação da Transformada de Hough na imagem, que aplica linhas coloridas na mesma nas faixas de rodagem.

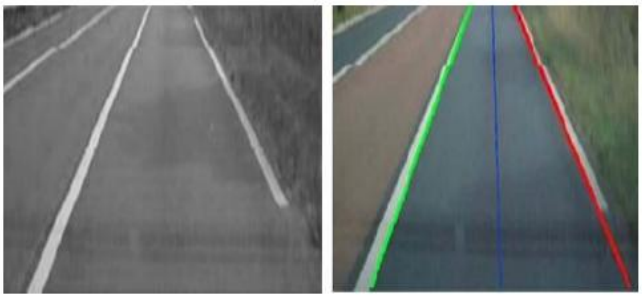


Figura 2-6 - Imagem de estrada processada pela Transformada de Hough.

Atualmente, já existem técnicas mais avançadas para a deteção de faixas de rodagem como o filtro de Kalman. Este filtro é utilizado para fazer o seguimento das faixas de rodagem ao longo de *frame*s consecutivos, de modo a prever a posição futura das faixas consoante a movimentação do veículo.

2.4.Soluções Desenvolvidas

Nesta secção aborda-se os projetos similares com o presente trabalho, analisando os sistemas já existentes no mercado e as diferentes tecnologias utilizadas.

2.4.1. Soluções de Arquiteturas de *Deep Learning*

O desenvolvimento do sistema ADAS baseou-se na integração e otimização de modelos de *Deep Learning* de referência, selecionados pela sua capacidade de resposta em tempo real e precisão em ambientes dinâmicos. Esta secção descreve as arquiteturas que serviram de fundação ao *software* final.

Sistema de Aviso de Saída de Faixa de Rodagem

O sistema desenvolvido por Junsheng Fu [18] baseia-se num *pipeline* sequencial de processamento de imagem que transforma cada *frame* proveniente de uma câmara numa representação matemática das faixas de rodagem. O funcionamento divide-se em cinco etapas principais:

1. Calibração e Correção de Distorção: As lentes das câmaras introduzem distorções óticas (efeito "olho de peixe"). O sistema utiliza uma imagem de um tabuleiro de xadrez para calcular a matriz da câmara e os coeficientes de distorção. Através de uma função em conjunto com os dados de calibração, o algoritmo corrige a curvatura da imagem original, garantindo que as linhas retas no mundo real apareçam como linhas retas na matriz de *pixéis*.

2. Extração de Características: Nesta fase de filtragem, onde o sistema tenta "limpar" tudo o que não é uma faixa de rodagem, o algoritmo utiliza dois métodos combinados: filtros de Sobel e espaço de cor HLS (Hue, Lightness, Saturation). Nesta etapa, é calculado o gradiente de cor (variação de cor) na direção horizontal e, como as faixas de rodagem são maioritariamente verticais em relação à câmara, o Sobel no eixo horizontal, destaca as arestas das linhas. Após isso, o espaço de cor HLS converte a imagem em Red, Green, and Blue (RGB) para HLS, onde depois o canal S (Saturação), é utilizado para isolar as faixas amarelas e brancas, este canal é bastante eficaz mesmo quando há sombras ou variações de luz que confundiriam o canal de cor normal. Como resultado, obtém-se uma imagem binária onde os pontos brancos representam potenciais candidatos a serem faixas.

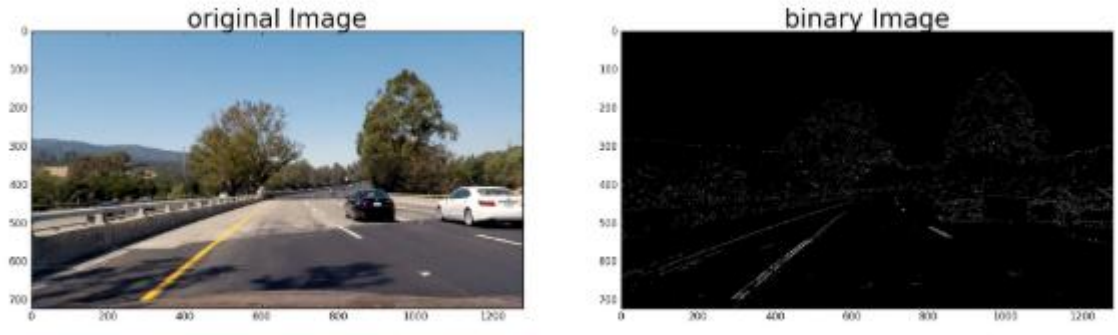


Figura 2-7 - *Frame* de vídeo filtrado por filtros de Sobel e gradientes de cor. [18]

3. Transformação de Perspetiva (*Bird's-Eye View*): Esta etapa serve para medir distâncias e curvaturas, onde sistema seleciona quatro pontos na imagem original que formam um trapézio (a perspetiva da estrada a afunilar no horizonte) e "estica-os" para formar um retângulo. Assim, a imagem passa a parecer uma visão aérea (visto de cima) onde nesta perspetiva, as faixas de rodagem se tornam paralelas, o que facilita o cálculo da curvatura através de polinómios.

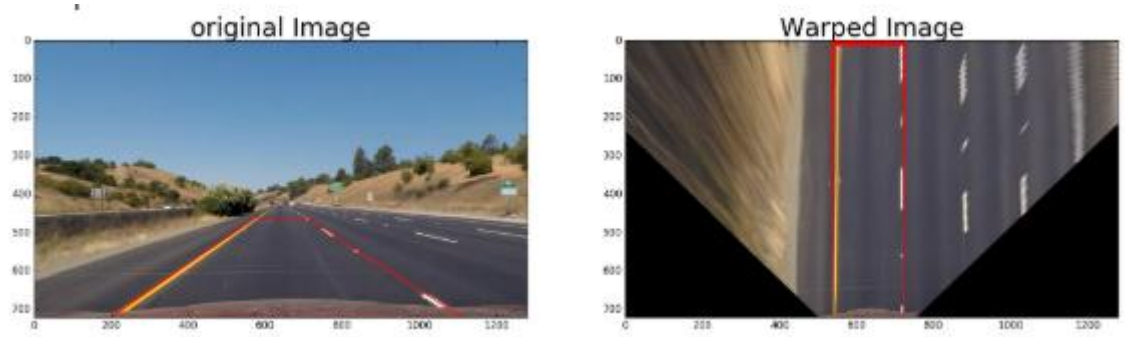


Figura 2-8 - *Frame* de estrada após transformação de perspetiva. [18]

4. Deteção por Janelas Deslizantes: Com a imagem binária e em visão aérea, o sistema precisa de decidir que pixéis pertencem à faixa esquerda e quais pertencem à direita. Para tal, faz-se um histograma da metade inferior da imagem. Os dois picos mais altos de pixéis brancos indicam a base das duas faixas. A partir da base, o algoritmo desenha janelas quadradas e "sobe" a imagem. Se houver *pixéis* brancos dentro da janela, ela centra-se neles e continua a subir. Isto permite seguir a linha mesmo que ela seja curva.

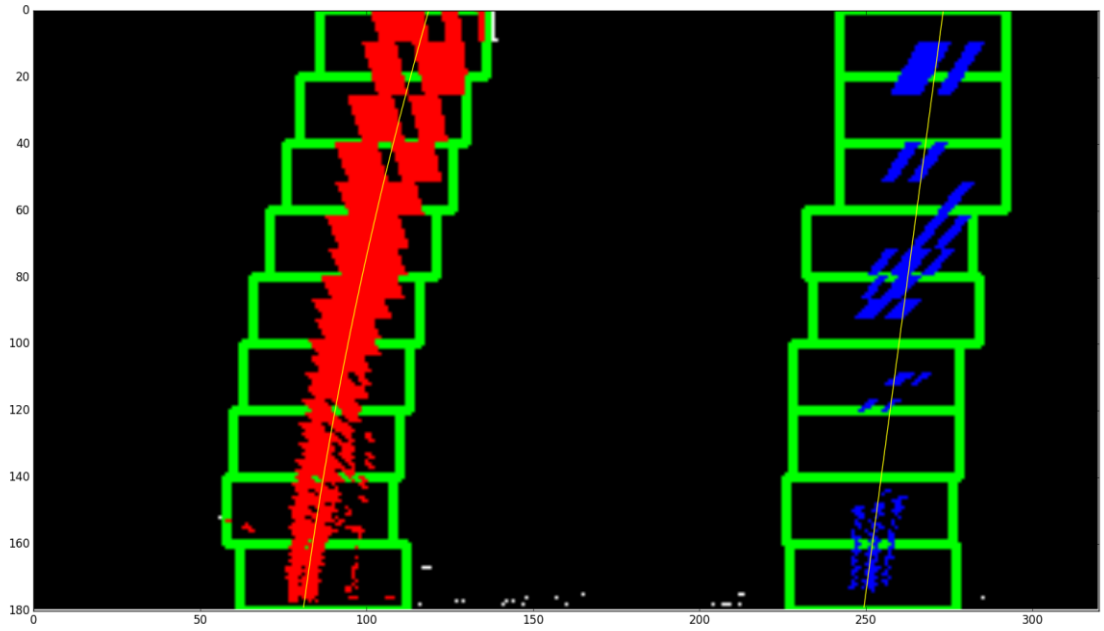


Figura 2-9 - *Frame* de imagem após utilização do método das janelas deslizantes. [18]

5. Ajuste Polinomial e Cálculo de *Off-Center*: Por fim, nesta fase, o sistema aplica uma regressão matemática aos pontos encontrados pelas janelas deslizantes. Utiliza-se um polinómio de 2º grau para desenhar uma linha curva suave sobre os *pixéis* detetados. O cálculo do desvio (*off-center*) é feito assumindo que a câmara está no centro do carro. Assim, o algoritmo calcula a distância entre o centro da imagem e o ponto médio entre as duas faixas detetadas. Se este desvio ultrapassar um determinado limite, o sistema interpreta que o pneu está a pisar a linha e ativa o alerta visual (mudando a cor da área projetada de verde para vermelho).

Deteção de Sinais de Trânsito e Objetos nas Estradas (YOLO) [19]

O sistema implementado a partir dos desenvolvimentos de Mikołaj Kołek baseia-se numa arquitetura de rede neuronal convolucional (CNN) de passagem única, otimizada para identificar e classificar múltiplos elementos rodoviários em milissegundos. Para garantir o bom funcionamento e, portanto, a deteção de objetos com eficácia, o sistema deve passar por duas etapas essenciais:

1. **Pré-treino e *Transfer Learning*:** O modelo não inicia o reconhecimento do zero. Utiliza-se a técnica de *Transfer Learning*, onde pesos pré-treinados do *dataset* Common Objects in Context (COCO) são ajustados com o conjunto de dados BDD100K (Berkeley *DeepDrive*). Esta etapa permite que a rede adquira o

conhecimento de milhões de imagens de estrada, aprendendo a distinguir as formas básicas de veículos, peões e infraestrutura urbana antes mesmo da primeira utilização real.

2. ***Fine-Tuning* e Especialização:** Para garantir a precisão em sinais de trânsito específicos (como os europeus), o modelo é submetido a uma otimização (*Fine-Tuning*). Durante o treino, o sistema utiliza o conceito de *Patience* (Paciência), monitorizando a perda de validação. Se o modelo parar de aprender após um determinado número de fases, o treino é interrompido para evitar o *Overfitting*, garantindo que a rede neuronal consiga identificar objetos novos e não apenas "decorar" as imagens usadas no treino.

Com o sistema treinado para o ambiente em que vai atuar, este funciona da seguinte forma:

1. **Extração de Tensores:** No momento da execução, o sistema recebe o *frame* bruto da câmara e converte-o num tensor. A rede YOLOv8 processa a imagem numa única passagem, dividindo-a numa grelha e prevendo, simultaneamente, a probabilidade de existência de objetos e as suas coordenadas espaciais.
2. **Filtragem por Confiança e *Non-Maximum Suppression* (NMS):** O sistema gera centenas de potenciais deteções e, para limpar o resultado, aplica-se um filtro de confiança normalmente superior a 0.45, descartando "ruído". Seguidamente, o algoritmo NMS elimina caixas sobrepostas para o mesmo objeto, garantindo que cada carro ou peão na estrada seja identificado por uma única moldura precisa, evitando duplicação de alertas.

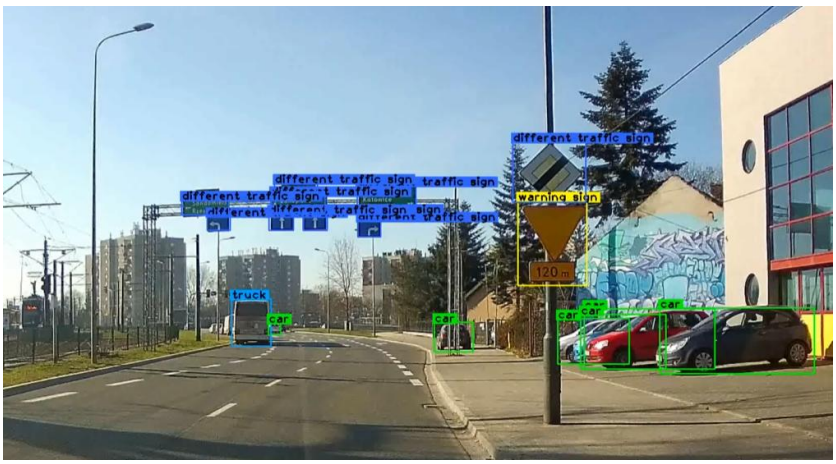


Figura 2-10 - *Frame* de imagem após deteção de objetos por YOLOv8.

Deteção de Faixas de Rodagem Através de Rede Neuronal (YOLOPv2) [20]

O sistema baseado na arquitetura YOLOPv2 (You Only Look Once Panoptic Driving Perception) representa uma evolução para a perceção multitarefa, onde uma única rede neuronal profunda executa simultaneamente a deteção de objetos e a segmentação da estrada. O seu funcionamento detalhado estrutura-se em cinco etapas:

1. **Extração de Características Partilhadas:** O sistema utiliza um codificador comum (baseado em convoluções de alto desempenho) para extrair os mapas de características da imagem. Esta abordagem elimina a necessidade de correr redes separadas para cada tarefa, permitindo que a informação visual básica seja partilhada pelas três tarefas de deteção, otimizando o uso da memória da Graphics Processing Unit (GPU).
2. **Deteção de Objetos e Obstáculos:** Através de uma das ramificações da rede, o sistema identifica veículos, peões e sinais de trânsito. Utilizando uma abordagem *anchor-free*, a rede prevê as coordenadas das caixas delimitadoras (*bounding boxes*) e a classe do objeto. O algoritmo aplica ainda o método NMS para eliminar deteções sobrepostas, garantindo uma identificação limpa dos obstáculos.
3. **Segmentação da Área de Condução:** Ao contrário dos métodos clássicos, o YOLOPv2 realiza uma segmentação semântica *píxel* a *píxel* para identificar toda a superfície asfáltica onde o veículo pode circular com segurança. Esta máscara binária permite ao sistema compreender os limites da estrada mesmo na ausência de marcações laterais, identificando passeios e bermas como áreas interditas.
4. **Segmentação de Faixas de Rodagem:** Uma ramificação específica foca-se exclusivamente na extração das marcações horizontais. O sistema gera uma máscara de alta resolução que identifica as faixas de rodagem como entidades semânticas. Esta técnica é superior aos filtros de gradiente clássicos, pois a rede foi treinada para ignorar sombras, reflexos e variações de iluminação que habitualmente confundem os algoritmos de visão clássica.

A análise da implementação do YOLOPv2 face ao sistema clássico revela as seguintes vantagens: robustez visual por ser baseado em *Deep Learning*, o sistema é extremamente resiliente a condições climatéricas adversas, sombras de árvores e variações de luz que inutilizam filtros de Sobel ou HLS; a capacidade de detetar obstáculos, a área de condução e as faixas de rodagem num único modelo o que reduz a complexidade da arquitetura de

software; identifica facilmente faixas curvas, descontínuas ou em cruzamentos onde o ajuste polinomial clássico falha por falta de dados.

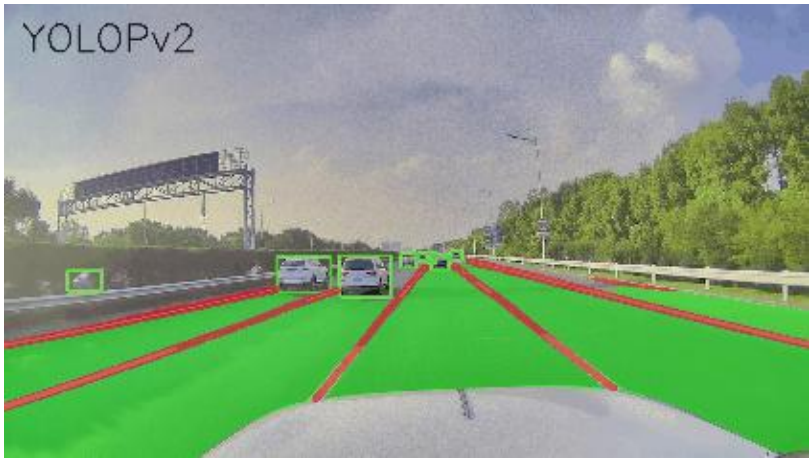


Figura 2-11 - *Frame* de imagem processada pelo modelo YOLOPV2.

Apesar disto, o sistema apresenta algumas desvantagens tal como o elevado custo computacional devido à necessidade de processar segmentações *píxel* a *píxel* para a estrada o que exige uma capacidade de processamento intensa o que tem como sequência a latência de resposta.

2.4.2. Soluções Comerciais

Mobileye ADAS

A Mobileye é uma empresa que disponibiliza uma tecnologia ADAS capaz de se integrar em praticamente todos os veículos. A empresa tem 3 produtos disponíveis para o mercado, sendo eles o Mobileye 6 Series, Mobileye 8 Connect e o Mobileye Shield+, com características distintas [21].

O Mobileye 6 Series é um sistema ADAS de nível 0, ou seja, com a capacidade apenas de alertar o condutor para vários cenários de perigo iminente. É composto pelas seguintes tecnologias: Aviso de Colisão Frontal (FCW), Aviso de Centralização de Faixa (LDW), Aviso de Colisão com Pedestres e Ciclistas (*Pedestrian and Cyclist Collision Warning* - PCW), Aviso de Distância de Segurança (*Headway Monitoring and Warning* - HMW) e Indicador de Limite de Velocidade (*Speed Limit Indication* - SLI). Este produto é composto essencialmente por 2 componentes: a unidade com câmara chip *EyeQ* e coluna de som que estão instalados no para-brisas em baixo do espelho retrovisor que está

identificado como 1 na Figura 2-12; e o *display Eye Watch* para alertas visuais instalado na parte superior do tablier do veículo que está identificado como 2 na Figura 2-12. Este produto utiliza uma câmara Aptina MT9V024 de 752 por 480 *pixéis* e um processador de dois Central Processing Unit (CPU) MIPS24KF de 32 bit.

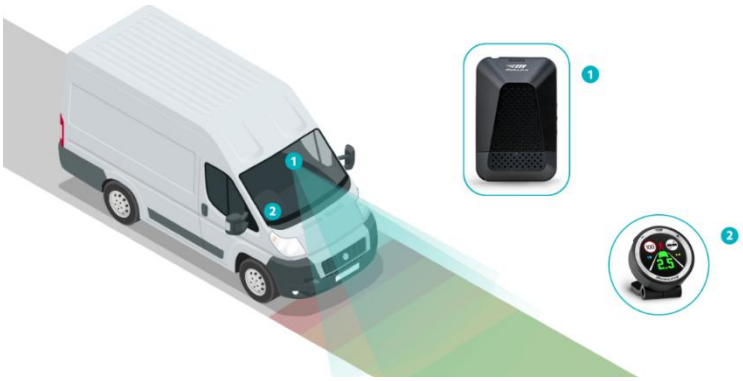


Figura 2-12 - Local de instalação do Mobileye 6 Series.

O Mobileye 8 Connect assemelha-se muito ao 6 Series sendo que possui as mesmas capacidades, para além da capacidade de conectividade entre dispositivos Mobileye fornecendo informações sobre eventuais situações de perigo e capacidade de deteção de pedestres e ciclistas durante a noite. Este produto é composto ainda por um dispositivo Global Positioning System (GPS), o que permite partilhar as informações dos dispositivos com outros dentro da mesma rede. Este produto é então bastante utilizado para frotas de veículos comerciais pois permite ao gestor de tráfego dos veículos conhecer a localização dos mesmos em tempo real, assim como hábitos inseguros do condutor como travagens bruscas, colisões iminentes e excesso de velocidade. Este produto é composto uma câmara OV10642 RCCC CMOS 1.3MP de 1280 por 1080 pixéis e um processador com um CPU *Hyper-Thread* 64 bit RISC interAptiv MIPS.

Relativamente ao Mobileye Shield+, este produto é direcionado para veículos comerciais e é bastante semelhante ao Mobileye 8 Connect com a diferença de adicionar uma funcionalidade que é a Deteção de Ângulo Morto. Para além disso, este produto adicionalmente utiliza um conjunto de câmaras externas para deteção de ângulo morto, e um maior número de alertas visuais.

2.4.3. Soluções Académicas

ADAS Baseado em Jetson Nano

Neste projeto, Viet Anh [22] desenvolveu um sistema ADAS capaz de integrar veículos automóveis de baixa gama e antigos que não possuam esta tecnologia. O produto final possui 3 funções base, nomeadamente, alerta de colisão frontal com veículos e pedestres, aviso de centralização de faixa e, deteção e interpretação de sinais de trânsito.

Em relação ao *hardware* utilizado, o autor faz recurso de um NVIDIA Jetson Nano, um pequeno computador com elevado poder de processamento capaz de executar diversas redes neuronais em simultâneo, para a interface com o utilizador utiliza um ecrã tátil de 5 polegadas e dois altifalantes de 5 W. A aquisição de dados não é feita pelo utilizador, no entanto o *software* é preparado para receber dados de uma câmara, e para comunicação CAN com o veículo de modo a adquirir dados do veículo como a velocidade do mesmo e sinais de mudança de direção, pelo que o design do *hardware* é o demonstrado na Figura 2-13.

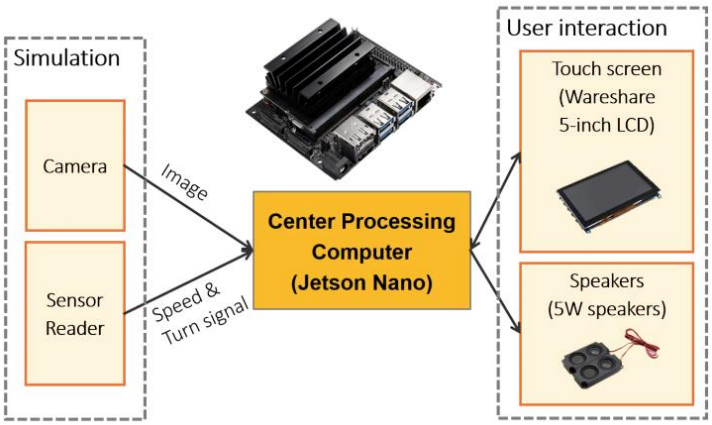


Figura 2-13 - Design de *hardware* do sistema ADAS desenvolvido por Viet Ahn.

O *software* do trabalho utiliza três redes neuronais como base onde para o subsistema de alerta de colisão frontal e reconhecimento de sinais de trânsito o autor utiliza uma rede neuronal com base no *software* da Center-Net. Relativamente ao subsistema de alerta de ultrapassagem de faixa de rodagem o *software* utiliza uma rede neuronal baseada na U-Net.

ADAS Baseado em Raspberry Pi

Neste trabalho foi desenvolvido um sistema ADAS com as seguintes capacidades: sistema de deteção e assistência de faixa, sistema de deteção e alerta de ângulo morto, sistema de alerta de colisão frontal e sistema de deteção de pedestres. O sistema utiliza um Raspberry Pi 3 como unidade de processamento, um *buzzer* para alerta sonoro, um visor para alerta visual, uma câmara Pi-Camera, e sensores ultrassónicos HC-SR04 e JSN-SR04T, e segue a arquitetura da Figura 2-14.

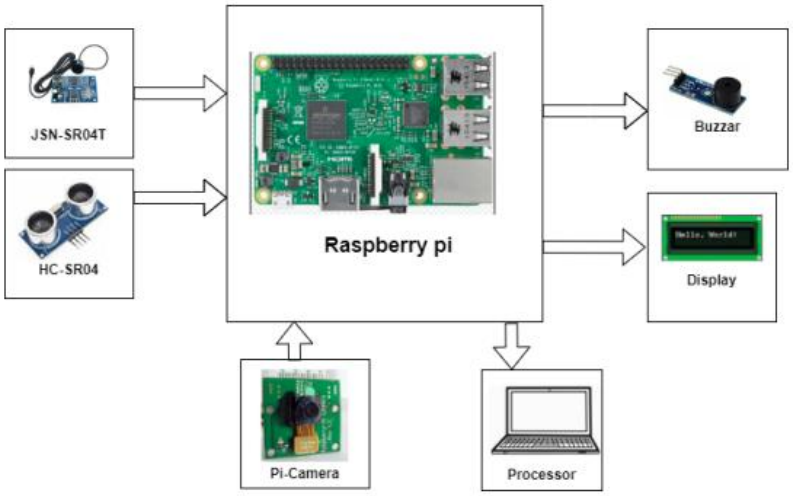


Figura 2-14 - Arquitetura de *hardware* do projeto de sistema ADAS utilizando Raspberry Pi 3.

Apesar do sistema desenvolvido ter sido utilizado num pequeno veículo protótipo, este sistema demonstra a capacidade do Raspberry Pi de processamento de dados provenientes de uma câmara e de outros sensores para implementação de sistemas de segurança e de alerta ao condutor durante a condução.

Relativamente ao algoritmo utilizado, o autor utilizou um sistema do tipo *Canny Edge* para identificação de faixas de rodagem e um algoritmo de histograma de gradientes orientados (HOG) para deteção de pedestres baseados em *OpenCV* e *NumPy*. Este sistema demonstrou um bom desempenho em diversas condições apesar da ocorrência de algumas situações de falsos positivos.

ADAS Baseado em Raspberry Pi 4 [23]

Neste trabalho o autor desenvolve um sistema capaz de integrar veículos mais antiquados que não possuam sistemas ADAS integrados. Este projeto tem a capacidade dos seguintes sistemas ADAS: alerta de mudança de faixa (LDW), alerta de colisão

frontal (FCW), alerta de movimento do veículo à frente (Forward Vehicle Start Alarm - FVSA), e alerta para violações de semáforos (Traffic Light Warning - TLW). Relativamente a *hardware*, este projeto utiliza um Raspberry Pi 4 como computador, como câmara utiliza uma Pi-Camera e/ou uma câmara *Logitech Webcam* C920 USB, utiliza ainda um adaptador ELM327 *Bluetooth* para comunicação com o veículo, um sensor de efeito de *Hall* para reconhecimento do sinal de mudança de direção, um *buzzer* para alerta sonoro e uma barra de Light-Emitting Diode (LED) para alertas visuais, sendo que o custo total do sistema ronda os 200 €. Para teste de funcionamento do sistema desenvolvido, aplicou-se o mesmo num Volkswagen Touran 1.4 TSI de 2007 que não possui quaisquer tipos de sistema ADAS, como se pode analisar na Figura 2-15.

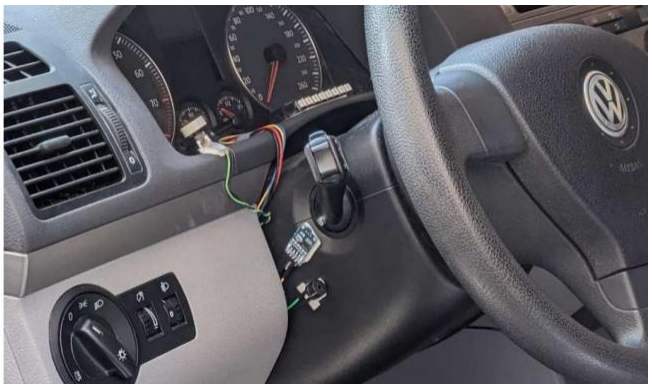


Figura 2-15 - Sistema ADAS desenvolvido em Raspberry PI 4 aplicado num Volkswagen Touran 1.4 TSI.

Como algoritmos utilizados para processamento das informações e aplicação do sistema ADAS, o autor utilizou redes neuronais convolucionais baseadas no modelo MobileNet para reconhecimento de objetos, veículos e semáforos, um algoritmo do tipo *Canny Edge* para identificação de faixas de rodagem, e um algoritmo de segmentação de cor para identificação de sinais luminosos como semáforos e luzes de travagem.

Apesar das limitações dos sensores e sistemas de processamento, o sistema mostrou-se viável na medida em que cumpre os objetivos definidos inicialmente de alerta e prevenção de acidentes. Os subsistemas ADAS como alerta de mudança de faixa (Lane Departure Warning - LDW), alerta de colisão frontal (Frontal Collision Warning - FCW), alerta de início de movimento do veículo da frente (Front Vehicle Start Alarm - FVSA) e alerta de violação de semáforo vermelho (Red Light Violation Warning - RLVW) demonstraram um nível de precisão alta como indica na Tabela 2-1, onde cada valor em

percentagem indica a precisão de cada subsistema e o valor entre parêntesis indica o número de amostras utilizado.

Tabela 2-1 - Tabela de precisão de cada subsistema ADAS baseado em *Rapsbery Pi 4*.

	LDW		FCW	FVSA	RLVW
Dia	**Auto-estrada**	95,0% (18)	82,8% (35)	93,0% (14)	64,0% (14)
	Urbano	68,2% (22)			
	Rural	66,0% (12)			
Noite	**Auto-estrada**	73,0% (15)	72,0% (11)	90,5% (21)	44,4% (9)
	Urbano	66,0% (15)			
	Rural	72,0% (4)			

2.5. Componentes Principais

Os sistemas ADAS exigem um conjunto de componentes robustos e precisos de modo a constituir um sistema de segurança altamente fiável. Alguns desses componentes podem ser câmaras de alta qualidade, radares e LiDAR que produzem um grande conjunto de informações visuais e sensórias que precisam, posteriormente, de um processamento requintado.

Assim, para o desenvolvimento deste sistema ADAS, a seleção de *hardware* foi feita com base em critérios como compatibilidade entre dispositivos, capacidade de processamento, alcance, fiabilidade e custo-benefício. Os componentes mais fundamentais a selecionar para este projeto são os seguintes: câmara, LiDAR e sistema de processamento.

2.5.1. Câmara

A câmara é o componente responsável pela captação de imagens de alta-definição que posteriormente servirão para a identificação de objetos, veículos, pedestres e sinais de trânsito. Para o presente trabalho surgiram três opções de câmaras para implementação no sistema.

A Pi-Camera [24], como mostra a Figura 2-16, é uma das opções com uma resolução de 12.3 megapixéis e um campo de visão de cerca de 66 graus, baseado na Sony IMX500 *Intelligent Vision Sensor.* Tem a vantagem de ser facilmente implementada num Raspberry Pi e ser compatível com diversos modelos de redes neuronais, para além de ter um preço relativamente acessível de cerca de 80 euros.

Figura 2-16 - Imagem ilustrativa de uma câmara Pi-Camera AI.

Outra opção seria uma câmara mais comum como uma Logitech HD PRO C920, como mostra a Figura 2-17. Com uma resolução de 3.0 megapixéis e um ângulo de visão de cerca de 78 graus, esta câmara tem a vantagem de ter uma ligação simples de cabo Universal Serial Bus (USB) e um custo de cerca de 80 euros. Apesar disso é uma câmara desenvolvida como *webcam* e por isso não possui características específicas que a permitam ter um bom desempenho para obtenção de imagens durante as mais diversas condições adversas de condução.

Figura 2-17 - Imagem ilustrativa de uma câmara Logitech HD PRO C920 [25].

Outra opção seria uma câmara Intel RealSense Depth Camera D435i [26] como mostra a Figura 2-18. A câmara de profundidade inclui um sensor de inércia de seis eixos (Bosch BMI055) capaz de detetar e medir acelerações lineares e velocidades angulares. Esta câmara têm uma resolução de imagem de 1920 por 1080 pixéis a uma velocidade de 30 *frames per second* (FPS), um grande campo de visão de profundidade de 87º por 58º com uma resolução de 1280 por 720 pixéis a uma velocidade de 90 FPS, e tem um alcance recomendado de 10 metros. Todas estas características são favoráveis à utilização da câmara para implementação num sistema ADAS, para além da vantagem que a Intel disponibiliza uma biblioteca de *software open-source* que facilita o desenvolvimento de projetos utilizando este tipo de câmaras. A única desvantagem desta câmara é efetivamente o preço, que tem um custo de cerca de 320 euros.

Figura 2-18 - Ilustração da câmara de profundidade Intel RealSense Depth Camera D435i.

2.5.2. LiDAR

Segundo Tejal [27], um sensor LiDAR consiste num dispositivo capaz de fazer um mapeamento do ambiente que o rodeia tal como um sonar ou radar, com a diferença que usa luz e não ondas sonoras ou ondas de rádio. O LiDAR emite um laser que percorre uma certa distância até alcançar o alvo, esse alvo reflete esse laser até à sua fonte, como a velocidade da luz é contante, então consegue-se determinar a distância do objeto ao

LiDAR. Sabendo a posição do sensor, consegue-se então determinar a posição por coordenadas de cada objeto refletido. Tal como um dispositivo radar, os LiDAR podem ser categorizados por dimensão, ou seja, existem LiDAR de uma, duas e três dimensões que apresentam as seguintes características:

- **1 Dimensão:** dispositivo que emite um laser apenas numa direção e calcula a distância do objeto que alcança nessa dimensão.
- **2 Dimensões:** dispositivo emite lasers à medida que o dispositivo roda, o que permite alcançar alvos em dois eixos de coordenadas.
- **3 Dimensões:** dispositivo que emite diversos lasers em direções diferentes permitindo um mapeamento do ambiente que rodeia em três eixos de coordenadas.

Para o presente trabalho decidiu-se analisar três opções de tecnologias LiDAR comparando as suas características técnicas e o seu desempenho em sistemas ADAS. Como LiDAR de uma dimensão, surgiu o Garmin LIDAR-Lite v3 [28] como mostra a Figura 2-19. Este dispositivo tem um alcance de até 40 metros, uma precisão de mais ou menos 2,5 cm e permite comunicação Inter-Integrated Circuit (I2C) ou Pulse Width Modulation (PWM), para além disso tem um custo de cerca de 143 euros.

Figura 2-19 - Ilustração do sensor LiDAR Garmin LIDAR-LITE V3HP.

Uma opção viável para um LiDAR de duas dimensões seria o SLAMTEC RPLIDAR A2 [29] como mostra a Figura 2-20. Este dispositivo tem um alcance de até 12 metros, um alcance angular de 360º, uma precisão de cerca de 2,5% do alcance e tem um custo de cerca de 230 euros.

Figura 2-20 - Imagem ilustrativa do LiDAR de duas dimensões SLAMTEC RPLIDAR A2M12.

Finalmente, uma opção viável para LiDAR de três dimensões seria o Livox MID-40 [30] como mostra a Figura 2-21. Este dispositivo tem um alcance de até 130 metros, um alcance angular de 38,4° circular, uma precisão de cerca de 2 cm, uma precisão angular de cerca de 0,1º, uma taxa de pontos de cerca de 100 000 pontos por segundo e um custo de cerca de 600 euros.

Figura 2-21 - Imagem ilustrativa do LiDAR de três dimensões LIVOX MID-40.

2.5.3. Sistema de Processamento

O sistema de processamento é um dos elementos-chave deste projeto uma vez que será ele a processar todas as informações provenientes dos diversos sensores e componentes de aquisição de dados do sistema. Este componente terá de identificar objetos em imagens e executar algoritmos e cálculos complexos em tempo real de modo a obter respostas rápidas e fiáveis para o condutor. Assim, através da revisão bibliográfica realizada surgiram diversas opções viáveis com características técnicas diferentes nomeadamente, NVIDIA Jetson Nano, Raspberry Pi 5 e Minisforum EM680.

O NVIDIA Jetson Nano, como mostra a Figura 2-22 é um pequeno computador embebido com alta capacidade de processamento, que lhe permite processar diversas redes neuronais em paralelo. Este dispositivo utiliza uma arquitetura de Graphics Processing Unit (GPU) de 128 núcleos NVIDIA CUDA, uma arquitetura de CPU de 4 núcleos, 4 gigabytes de memória Random Access Memory (RAM) a uma frequência de 1600 megahertz. Para além disso, possui quatro entradas USB 3.0, duas entradas USB 2.0 Micro-B e possui outras formas de barramento como Inter-Integrated Circuit (I2C), Serial Peripheral Interface (SPI) ou Universal Asynchronous Receiver-*Transmitter* (UART). Este componente encontra-se disponível à venda por um valor de cerca de 250 euros.

Figura 2-22 - Imagem ilustrativa de um NVIDIA Jetson Nano.

O Raspberry Pi 5, como mostra a Figura 2-23, é um computador embebido mais direcionado para o desenvolvimento de pequenos projetos. Apesar disso, tem um elevado poder de processamento com um CPU de quatro núcleos que opera a uma frequência de 2.4 GHz e com uma memória RAM de até 8 GB. Para além disso possui saídas de vídeo HDMI, duas entradas USB 3.0, duas entradas USB 2.0, entrada para microSD, Wi-Fi integrado e *Bluetooth* integrado. A versão de 8 Gb de memória RAM deste dispositivo é vendida em kit composto por: Raspberry Pi 5 8 Gb, cartão microSD de 128 Gb de memória, fonte de alimentação de 27 W, encapsulamento com ventoinha, cabo High-definition Multimedia Interface (HDMI) de 1 metro e um dissipador de calor em alumínio, por um preço de cerca de 140 euros.

Figura 2-23 - Imagem ilustrativa de um Raspberry Pi 5.

A última opção a analisar será então o Minisforum EM 680 [31], como pode-se observar na Figura 2-24, que se trata de um pequeno computador com grande poder de processamento. Este dispositivo tem uma CPU AMD Ryzen de 8 núcleos que opera a uma frequência de 4.7 MHz, uma memória RAM de até 32 GB a uma frequência de 6400 MHz, capacidade de armazenamento de até 2 TB, Wi-Fi e *Bluetooth* incorporados, saída de vídeo HDMI, três portas USB 3.2, duas portas USB 4 e o sistema operativo é o *Windows* 11. Este computador é o que apresenta maior poder de computação relativamente aos outros dois dispositivos já mencionados, mas tem um custo elevado de cerca de 600 euros.

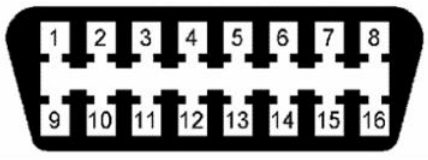
Figura 2-24 - Imagem ilustrativa do Minisforum EM680.

2.5.4. Interface com Veículo

Cada vez mais tem havido uma tendência crescente de incorporação de sensores por todo o veículo automóvel. Esta tendência deve-se a diversos fatores como aumentar a segurança durante a condução, otimização de eficiência de combustível do motor e até assistência à condução. Atualmente, o automóvel admite um enorme volume de informação que é processado e utilizado para o controlo de sistemas complexos como

controlo de estabilidade e diagnóstico de avarias. Os sistemas ADAS mais complexos para além de adquirirem informações de sensores como câmaras, LiDAR, ou sensores ultrassónicos, utilizam dados provenientes diretamente do veículo. Para subsistemas ADAS como alerta de colisão frontal, alguns dados provenientes do veículo otimizariam a precisão deste sistema, como a velocidade efetiva do veículo e a posição do pedal de acelerador.

A forma mais direta de aceder a estes dados é através da CAN, esta rede consiste numa linha de informação que circula pelo veículo adquirindo e transmitindo dados entre os diversos sistemas de veículos. Esta linha passa por pontos essenciais dos veículos pelos quais o utilizador do veículo pode aceder, que são o quadrante do veículo (mesmo que forneça um conjunto limitado de informações), eventuais ecrãs dentro do veículo e pela porta On-Board Diagnosis *(*OBD). A porta OBD consiste numa ficha onde se podem ligar dispositivos de diagnóstico, para efetuar a leitura de eventuais códigos de erro que podem ter sido gerados durante a condução no caso de uma avaria, leitura de dados e variáveis em tempo real, ou até teste de atuadores no veículo [32]. Existiram duas fichas OBD: OBD-I e OBD-II, onde a ficha OBD-II é a utilizada na maior parte dos veículos produzidos atualmente. De forma a uniformizar a ficha OBD-II surgiu a norma SAE J1979 que regulariza cada pino utilizado na porta OBD como mostra a Figura 2-25 - Ficha OBD-II e os respetivos pinos segundo a norma SAE J1979.



Pino	Descrição
1	Fabricante
2	Bus positive Line of SAE-J1850
3	Fabricante
4	Chassis ground
5	Signal ground
6	CAN high (ISO 15765-4 and SAE-J2234)
7	K line of ISO 9141-2 and ISO 14230-4
8	Fabricante
9	Fabricante
10	Bus negative Line of SAE-J1850
11	Fabricante
12	Fabricante
13	Fabricante
14	CAN low (ISO 15765-4 and SAE-J2234)
15	L line of ISO 9141-2 and ISO 14230-4
16	Battery voltage

Figura 2-25 - Ficha OBD-II e os respetivos pinos segundo a norma SAE J1979.

Através da porta OBD é então possível aceder a diversos parâmetros do veículo que são transmitidos pela linha CAN. Existem essencialmente dois tipos de mensagens CAN: *Standard CAN* e *Extended CAN*, onde a *Standadt CAN* consiste numa mensagem com um identificador de 11 bits e a *Extended CAN* consiste numa mensagem com identificador de 29 bits [33]. Uma mensagem CAN (Figura 2-26 - Imagem ilustrativa de uma transmissão CAN) é composta por sete campos distintos:

- ***Start of Frame* (SOF)** – 1 *bit* que indica o início da transmissão de uma mensagem CAN.
- **Identificador (ID) –** este campo pode ser de 11 ou 29 *bits* e contém uma mensagem que identifica os dispositivos, as mensagens, informação e define prioridade de acesso ao barramento.
- **Campo de controlo –** neste campo composto por 6 *bits*, existe informação sobre se é uma mensagem de envio ou receção de informação, se é uma mensagem *Standard CAN* ou *Extended CAN* e o número de bytes de informação pelo que a mensagem é composta.
- **Campo de informação –** é o principal campo da mensagem e pode conter até 64 *bits* de informação (8 *bytes*).

- **Cyclic Redundancy Check (CRC) –** campo de 15 bits que garante a integridade da informação transmitida detetando erros ou corrupção da mesma.
- **Acknowledgment (ACK) –** constituído por 2 bits, este campo assegura que a mensagem foi corretamente recebida.
- **End of Frame (EOF) –** constituído por 7 bits, é o último campo da mensagem CAN que tem como propósito definir o fim da mensagem e a disponibilidade para a próxima transmissão.

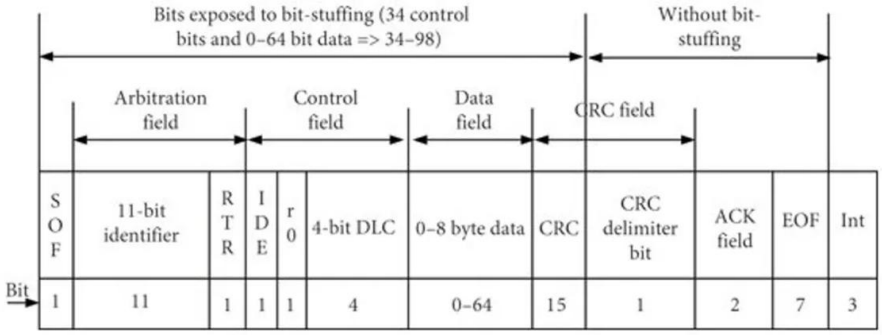


Figura 2-26 - Imagem ilustrativa de uma transmissão CAN.

Através da linha CAN é possível então aceder aos parâmetros fundamentais para os diferentes sistemas ADAS, parâmetros esses que são os seguintes:

- **Sinal de mudança de direção –** indispensável para o LDW uma vez que indica a necessidade de mudança de faixa de rodagem por parte do condutor, dispensado o alerta de mudança de faixa.
- **Velocidade do veículo –** requisitado pelo FCW permite perceber se o condutor está ciente da velocidade do veículo da frente, e se está a reduzir a velocidade do próprio veículo. Também é requisitado pelo sistema de reconhecimento de sinais de trânsito para perceber se o condutor está a circular dentro dos limites de velocidade.
- **Interruptor de travão –** serve de auxílio ao FCW uma vez que permite perceber se o condutor está em situação de travagem.
- **Posição do pedal de acelerador –** serve de auxílio ao FCW uma vez que permite entender a perceção do condutor da velocidade do veículo à frente.

Para aquisição destes dados da linha CAN existem diversas soluções. Uma delas, e talvez a mais simples, seria o uso de um dispositivo de descodificação de mensagens na

porta OBD como o ELM 327 como ilustrado na Figura 2-27. Este dispositivo já foi utilizado por Katsanos [34], e permite comunicação através de *Bluetooth* o que facilita a comunicação com o sistema de processamento. Para além disso, e uma vez que o sistema é suposto poder ser implementado em quaisquer tipos de veículos, este dispositivo pode ser usado em veículos com porta OBD e, para veículos que não tenham, basta dispensar o dispositivo e focar-se apenas nos sensores do sistema ADAS não recorrendo a informações adquiridas pelo próprio veículo. As vantagens deste método é a facilidade de implementação no sistema e o custo reduzido de cerca de 15 euros. Apesar disso tem como desvantagens o facto de ocupar a porta OBD constantemente, a hipótese de problemas de comunicação com o sistema de processamento e as limitações de comunicação com a linha CAN.

Figura 2-27 - Imagem ilustrativa de um ELM 327.

Outra opção seria a utilização de um microcontrolador CAN que comunica com o veículo através de uma ligação física direta à linha CAN, e comunica com o sistema de processamento através de porta série ou comunicação SPI. Um exemplo deste tipo de microcontroladores é o Microchip MCP2515-I/SO como mostra a Figura 2-28 [35]. A implementação deste dispositivo tem como vantagens trazer robustez ao sistema uma vez que se trata de uma ligação física e não *wireless*, um preço baixo de cerca de 2 euros e o facto de não ocupar a porta OBD.

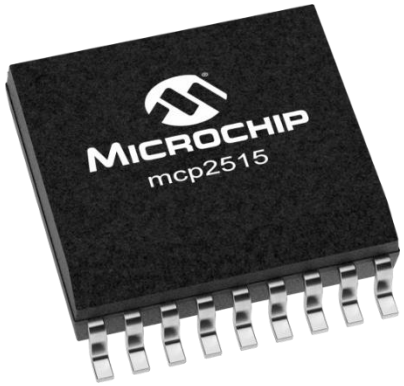


Figura 2-28 - Imagem ilustrativa de um Microchip MCP2515.

3. Arquitetura de *Hardware* e Interfaces

Após a revisão teórica e análise das soluções existentes, iniciou-se a fase prática do projeto, centrada na implementação e teste de diversos tipos de sistemas ADAS com funcionalidades de deteção e alerta em tempo real. Esta fase teve como ponto de partida a exploração de diferentes tecnologias de processamento de imagem, sensores e dispositivos de processamento, com o objetivo de construir um sistema funcional, fiável e adaptável a diferentes ambientes e veículos.

O desenvolvimento seguiu uma abordagem iterativa, em que cada etapa contribuiu para a consolidação da solução final, permitindo uma evolução gradual das funcionalidades e da complexidade do sistema. Nesta fase, foram realizados testes com diferentes modelos, câmaras e dispositivos, de forma a avaliar o desempenho e a compatibilidade de cada componente.

3.1. Seleção e Arquitetura de *Hardware*

A escolha dos componentes para o sistema ADAS foi precedida por um estudo comparativo entre diferentes plataformas de computação, de modo a garantir que a solução final cumprisse os requisitos de tempo real e portabilidade.

3.1.1. Plataformas de Computação em Análise

Antes da definição do protótipo final, foram realizados testes de desempenho em quatro plataformas distintas, permitindo analisar o comportamento dos algoritmos em arquiteturas CPU e GPU em termos de Frames Per Second (FPS) processadas e custo-benefício. As características principais de cada unidade são detalhadas em seguida:

1. **ASUS TUF Gaming A15 (FA507NV):** Equipado com um processador AMD Ryzen 7 7735HS (8 núcleos/16 *threads*) e uma placa gráfica dedicada NVIDIA RTX 4060 Laptop (8 GB GDDR6) [36].

Figura 3-1 - Computador portátil ASUS TUF Gaming A15 (2023) FA507NV

2. **ASUS Vivobook (X512JP):** Um sistema portátil com processador Intel Core i7-1065G7 e GPU dedicada NVIDIA MX330 (2 GB) [37].

Figura 3-2 - Imagem ilustrativa de um ASUS Vivobook X512JP.

3. **NVIDIA Jetson AGX Xavier (Developer Kit)** [38]**:** Destaca-se pela sua GPU de 512 núcleos com arquitetura Volta e 64 núcleos Tensor dedicados a Inteligência Artificial (IA), com 16 GB de memória LPDDR4x [39].

Figura 3-3 - Imagem ilustrativa de uma NVIDIA Jetson AGX Xavier.

4. **Raspberry Pi 5** [40]**:** A mais recente iteração da plataforma ARM, equipada com um CPU *quad-core* Cortex-A76 a 2.4 GHz. Apesar de ser um salto qualitativo face às versões anteriores, a ausência de uma GPU compatível com aceleração de redes neuronais *(VideoCore VII* é limitada para este fim) limita a sua capacidade para o sistema [41].

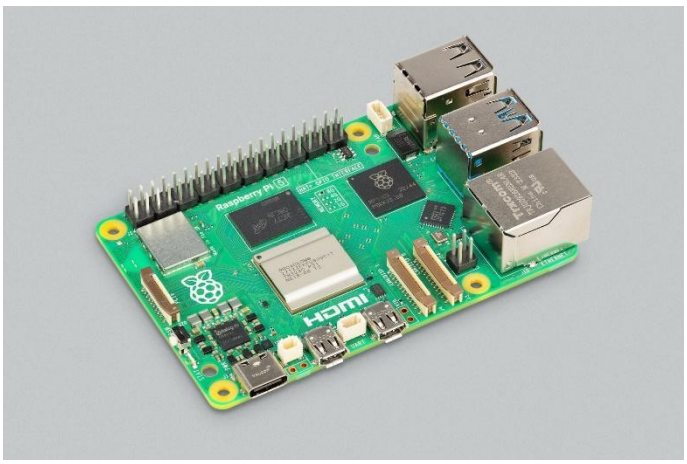


Figura 3-4 - Imagem ilustrativa de um *Raspberry Pi 5*.

Com base nestes sistemas de computação e nas suas respetivas arquiteturas, o sistema ADAS será submetido a testes de desempenho em cada plataforma, com o intuito de validar a viabilidade de cada componente face aos requisitos críticos de segurança. Esta análise comparativa permitirá determinar não só a capacidade de processamento bruto, mas também a eficiência económica (relação Preço/FPS) e a segurança ativa, medida através da distância percorrida pelo veículo entre cada ciclo de processamento.

A diversidade do *hardware* testado, desde GPU de alto desempenho (RTX 4060) a arquiteturas baseadas em CPU (Raspberry Pi 5), serve para demonstrar que a otimização do *software* é crucial para garantir que o tempo de resposta do sistema seja compatível com a condução em ambiente real.

3.1.2. Sensores de Perceção e Aquisição de Dados

Para garantir que o sistema ADAS opera com segurança em diversos cenários rodoviários, foi implementado uma configuração de fusão sensorial composta por um sensor de visão estereoscópica e um sensor de telemetria *laser*.

Câmara de Profundidade Intel RealSense D435i

A Intel RealSense D435i foi selecionada como o principal sensor de visão. Ao contrário das câmaras monoculares convencionais, esta unidade integra:

- **Tecnologia Estéreo Ativa:** Utiliza dois sensores de infravermelhos e um projetor para calcular a disparidade entre pixéis, gerando um mapa de profundidade em tempo real.
- **Sensor IMU Integrado:** A presença de uma Unidade de Medição Inercial (Inertial Measurement Unit - IMU) permite ao sistema compensar as vibrações do veículo e as variações de inclinação da estrada, melhorando a estabilidade da imagem.
- **Resolução e Campo de Visão:** Com um campo de visão de aproximadamente 87° por 58°, a câmara permite monitorizar não só a faixa de rodagem atual, mas também as faixas adjacentes e obstáculos laterais até uma distância recomendada de 10 metros.

LiDAR Garmin v3HP (*High Performance*)

Como sistema de redundância e precisão para o alerta de colisão frontal (FCW), utilizou-se o LiDAR Garmin v3HP. Este sensor laser atua de forma independente das condições de iluminação externa, colmatando as falhas da câmara em cenários de encadeamento solar ou escuridão total.

- **Alcance e Precisão:** Apresenta um alcance de até 40 metros com uma precisão de cerca de 2.5 centímetros, permitindo medições de distância muito mais rigorosas para o cálculo do tempo para colisão (Time-to-Collision - TTC).
- **Frequência de Atualização:** Com uma taxa de leitura configurável até 250 hertz, o LiDAR fornece dados de telemetria com uma latência quase nula, permitindo que o sistema detete uma redução súbita na distância para o veículo da frente antes mesmo do algoritmo de visão processar o *frame*.

3.2. Desenvolvimento e Teste de Redes Neuronais

O desenvolvimento do *software* foi estruturado segundo uma metodologia iterativa e incremental. Esta abordagem permitiu não só validar cada solução tecnológica face aos requisitos rigorosos de precisão e tempo de resposta, mas também identificar precocemente as limitações físicas do *hardware* testado. O ciclo de desenvolvimento focou-se na transição de algoritmos baseados em regras para modelos de aprendizagem profunda (*Deep Learning*), sendo cada fase avaliada segundo duas métricas fundamentais:

1. **Taxa de Atualização:** A frequência com que o sistema processa a imagem, determinante para a fluidez e continuidade da monitorização.
2. **Robustez Ambiental:** A capacidade de o algoritmo manter a deteção fidedigna em cenários de iluminação variável, sombras e oclusões parciais das marcações rodoviárias.

Esta evolução permitiu filtrar as soluções que, embora precisas no papel, se revelaram inviáveis em termos computacionais para as plataformas embebidas (como a Raspberry Pi 5 ou a NVIDIA Jetson), garantindo que o produto final oferecesse o melhor compromisso entre custo, desempenho e, acima de tudo, segurança ativa.

3.2.1. Lane Departure Warning System for Autonomous Driving

A fase inicial do desenvolvimento de *software* consistiu na implementação e teste do sistema intitulado *"Lane Departure Warning System for Autonomous Driving"*, desenvolvido originalmente por Junsheng [18]. Esta escolha serviu o propósito de

estabelecer uma linha de base tecnológica utilizando visão computacional clássica para a deteção de faixas e cálculo de desvio lateral.

Pipeline de Processamento de Imagem

O algoritmo segue um fluxo sequencial de processamento baseado em *OpenCV*, que foi adaptado para a câmara Intel RealSense D435i. O fluxo baseia-se em quatro partes distintas:

1. **Calibração da Câmara:** O processo inicia-se com a preparação de "pontos de objeto" (coordenadas 3D de cantos de um tabuleiro de xadrez) e "pontos de imagem" (posições de *píxeis* 2D) para calcular a matriz da câmara e os coeficientes de distorção. A correção de distorção é então aplicada às imagens brutas.

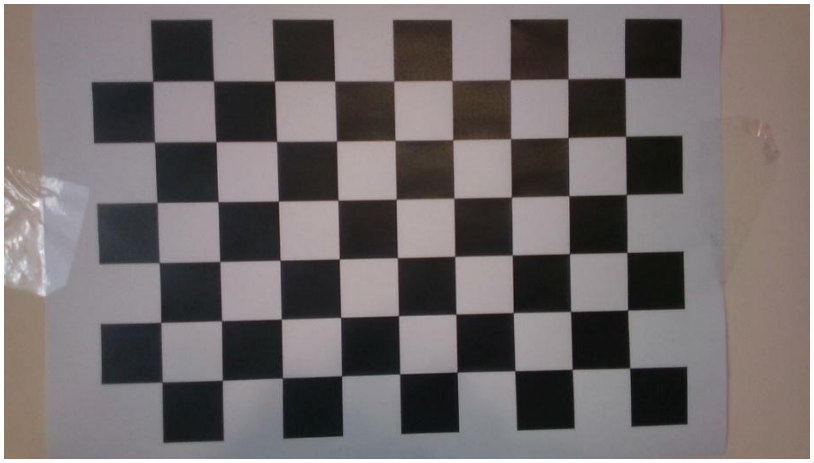


Figura 3-5 - Fotografia de calibração da câmara *Intel*.

2. **Deteção de Arestas e Filtragem de Cores:** É criada uma imagem binária utilizando transformações de cor e gradientes. O sistema converte a imagem para o espaço HLS, utilizando o canal S (Saturação) por ser mais robusto na localização de faixas amarelas e brancas sob diferentes condições de iluminação. Adicionalmente, aplica-se um filtro de Sobel (direção x) e um filtro de direção de gradiente para eliminar linhas horizontais.

3. **Transformação de Perspetiva (*Bird's-Eye View*):** O sistema aplica uma transformação de perspetiva para retificar a imagem binária, simulando uma visão aérea onde as linhas da faixa aparecem paralelas.

4. **Identificação e Ajuste de Faixas:**

    - **Para imagens isoladas:** Utiliza-se um histograma dos *píxeis* na metade inferior da imagem para identificar os picos que indicam a base das linhas. A partir daí, aplica-se o método de janelas deslizantes para seguir as linhas até ao topo.

    - **Para vídeo:** Implementa-se um sistema de seguimento que utiliza as posições da *frame* anterior como ponto de partida. Para suavizar a deteção, utiliza-se um *buffer* de memória para guardar as posições das últimas N *frame*s.

As linhas são posteriormente modeladas matematicamente através de um polinómio de 2.ª ordem.

Relativamente à análise de saída de faixa de rodagem e estado do veículo, o sistema calcula o raio de curvatura da faixa e a posição do veículo em relação ao centro. Assume-se que a câmara está montada no centro do para-brisas, comparando a base das faixas detetadas com o meio da *frame*. Se o veículo circular demasiado próximo da borda da faixa, o sistema alerta o condutor, alterando a cor da região da faixa projetada de verde para vermelho na visualização final.

Os testes realizados com o código original em condições reais de condução demonstraram um desempenho satisfatório no que respeita à fluidez, atingindo aproximadamente 14 FPS (Figura 3-6). No entanto, este valor é enganador para o objetivo final do projeto, pelas seguintes razões:

- **Processamento Limitado:** O processamento restringe-se exclusivamente à deteção de faixas de rodagem, sem a carga computacional necessária para a deteção de obstáculos ou peões.

- **Ausência de Alertas:** O sistema não possui qualquer lógica de aviso ao condutor ou processamento de distâncias, funcionando apenas como uma sobreposição visual na imagem.

- **Vulnerabilidade:** A dependência de técnicas de processamento de imagem clássica (filtros de cor e gradientes) revelou-se insuficiente para lidar com a variabilidade de cenários do mundo real, onde a iluminação e o estado do pavimento oscilam.

Estas limitações ditaram a necessidade de evoluir para uma arquitetura baseada em *Deep Learning*, capaz de realizar múltiplas tarefas simultaneamente com maior robustez.

Figura 3-6 - *Frame* de teste de processamento de faixas de rodagem segundo Junsheng Fu.

3.2.2. Integração de Deteção de Objetos em Tempo Real (YOLOv8)

Nesta segunda iteração, o sistema foi expandido para além da deteção de faixas, integrando a deteção de obstáculos e utilizadores vulneráveis da estrada (peões, veículos, sinais de trânsito). Esta fase fundamentou-se no trabalho de Duy Anh Tran et al. (2020) e na investigação de Alen Smajic, focada no desempenho de redes neuronais aplicadas ao conjunto de dados *Berkeley DeepDrive* (BDD100K).

Este trabalho disponibilizava dois modelos distintos de deteção de objetos: Faster R-CNN e YOLO. Para a seleção do modelo de deteção, analisaram-se as duas arquiteturas de referência:

- **Faster R-CNN:** Conhecido pela sua elevada precisão (Mean Average Precision - mAP - de 41.8% no estudo de Smajic), mas com uma taxa de *frame*s limitada (aproximadamente 17 FPS em *hardware* de servidor), o que inviabiliza o seu uso em sistemas embebidos de baixa latência.

- **YOLO *(You Only Look Once)*:** Conhecido pela sua abordagem de "passagem única", que prioriza a velocidade de processamento. No estudo de referência, o YOLO atingiu taxas superiores a 200 FPS em GPU de alta gama, permitindo prever que, mesmo em *hardware* limitado como a Jetson ou Raspberry Pi, manteria uma cadência utilizável para o ADAS.

Assim, nesta fase, utilizou-se o YOLOv8 pela sua capacidade de detetar múltiplas classes em simultâneo com um peso computacional reduzido.

A aplicação da rede neuronal foi estruturada para processar em simultâneo o código de deteção de faixas de rodagem e deteção de objetos do YOLOv8. Os testes realizados revelaram resultados cruciais para a análise de desempenho. Quando operado apenas com o código clássico de deteção de faixas, o sistema atingiu uma velocidade de processamento de cerca de 14 FPS. Este valor demonstra que o sistema é eficiente em termos de processamento, mas carece da capacidade necessária para identificar obstáculos.

Quando os sistemas foram fundidos e testados em condições reais, verificou-se uma baixa taxa de processamento de cerca de 7 FPS (Figura 3-7), tal como previsto. Para além disso, a deteção de faixas de rodagem revelou pior comportamento aquando da fusão.

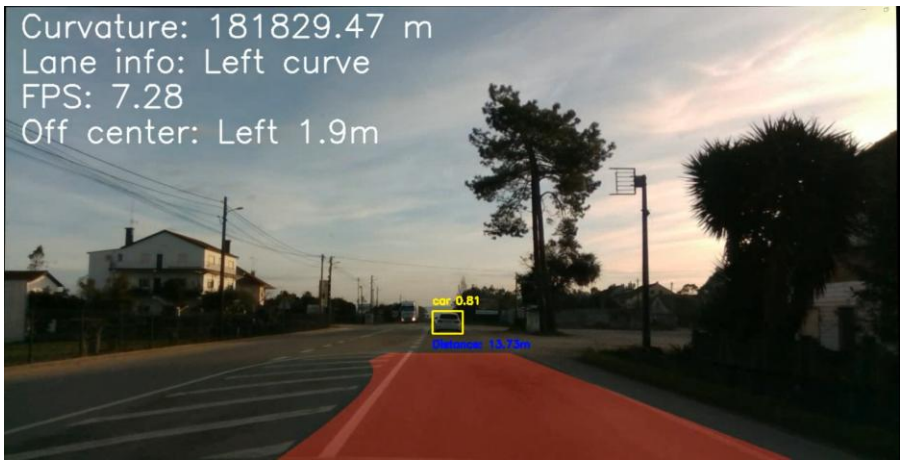


Figura 3-7 - *Frame* de teste de processamento da fusão entre deteção de faixas de rodagem e de objetos.

A integração destes dois sistemas permitiu concluir que, embora o YOLOv8 seja extremamente rápido, a sua integração com algoritmos de visão clássica pesados (como o de Junsheng Fu) cria um constrangimento de processamento.

Para que o sistema seja verdadeiramente viável em termos de segurança ativa, tornou-se evidente a necessidade de uma otimização do sistema, focada na substituição do processamento clássico de faixas de rodagem por modelos de *Deep Learning* mais eficientes e leves.

3.2.3. Implementação de *Deep Learning* (YOLOPv2 + YOLOv8)

Nesta fase, o sistema foi atualizado para uma arquitetura de *Deep Learning* multitarefa. Enquanto na fase anterior se utilizava uma combinação de IA para deteção de objetos e métodos manuais para faixas, aqui introduziu-se o YOLOPv2 (You Only Look Once Panoptic Driving Perception) [20].

A transição da visão clássica (Junsheng Fu) para o YOLOPv2 alterou radicalmente a forma como o sistema deteta a estrada. O processamento de imagem clássico depende de *thresholds* fixos de cor e gradientes (Figura 3-8), falhando facilmente em cenários com sombras por exemplo. O YOLOPv2, sendo uma rede neuronal convolucional (CNN), aprendeu a identificar faixas através de milhares de exemplos, tornando-se muito mais resiliente a ruído visual e condições climatéricas adversas.

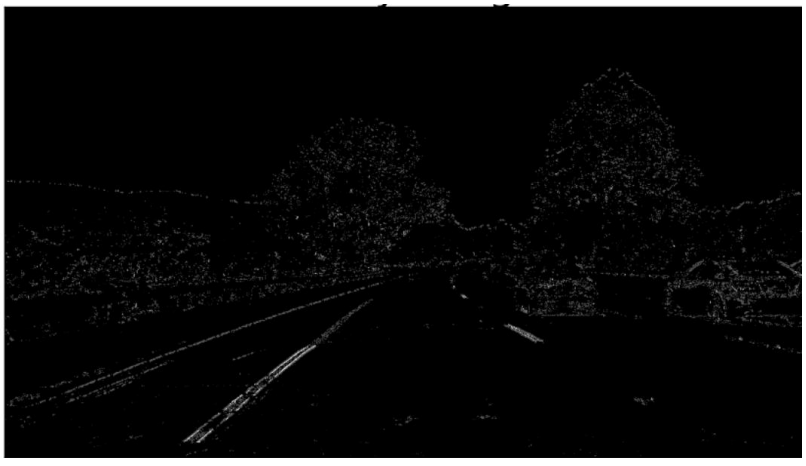


Figura 3-8 - *Frame* de imagem filtrada por gradientes. [18]

Ao contrário do método de Junsheng Fu, que tenta encontrar linhas através de *píxeis* isolados e janelas deslizantes, o YOLOPv2 realiza uma segmentação semântica *píxel* a

píxel. Isto permite identificar não só as linhas, mas também a área de condução livre (*Driving Area*), essencial para perceber onde o veículo pode circular com segurança. No processamento de imagem clássico, o programador tem de definir manualmente cada passo. No YOLOPv2, a rede recebe a imagem bruta e entrega diretamente a máscara sobre a estrada e faixas, otimizando a deteção de formas complexas.

O fluxo de execução neste sistema, segue então um *pipeline* rigoroso para garantir a sincronização entre os sensores e a lógica ADAS. A rede neuronal recebe a imagem e gera duas máscaras de saída: área de condução e a linhas de faixa. Este modelo utiliza uma arquitetura baseada em *kernels* convolucionais que atribuem uma classe a cada *píxel*, permitindo ao sistema "compreender" onde termina o asfalto e onde começam as marcações. Em paralelo, o YOLOv8 analisa a imagem em busca de classes específicas como carros, camiões, pedestres e semáforos (Figura 3-9). O modelo YOLOv8 foi escolhido para minimizar o tempo de processamento, permitindo ainda a extração de *bounding boxes* com alta confiança.

Figura 3-9 - Excerto de *frame* de deteção de objetos (YOLOv8).

Uma inovação crucial nesta fase foi a integração de dados de profundidade extraídos da câmara. Assim, para cada objeto detetado pelo YOLOv8, o sistema calcula o centro geométrico da *bounding box*. Após isso, o algoritmo acede então à matriz de profundidade no ponto (x, y) correspondente e calcula a média de uma pequena janela no centro (5 por 5 *píxeis*) para evitar erros de leitura pontuais.

Finalmente, o valor resultante é convertido para metros, fornecendo ao utilizador a distância exata até ao obstáculo.

A deteção de faixas do YOLOPv2 é convertida em alertas através de um cálculo geométrico. O sistema analisa a máscara de faixas de rodagem numa linha horizontal específica (a cerca de 90% da altura da imagem) para encontrar os *píxeis* ativos das faixas esquerda e direita. É ainda determinado o centro da estrada através da média das posições das duas faixas, onde o desvio lateral é consequentemente a diferença entre o centro da estrada e o centro ótico da câmara. Relativamente à interface visual, o algoritmo desenha um ícone dinâmico no canto superior direito da imagem de saída (Figura 3-10). Se o desvio for excessivo, o sistema alerta com a mensagem "*WARNING LANE DEPARTURE*" em vermelho, alertando o condutor para o risco iminente.

Figura 3-10 - *Frame* de saída de sistema de deteção baseado em YOLOPv2 e YOLOv8.

Esta configuração revelou-se a mais exigente em termos de *hardware* devido à densidade das redes neuronais utilizadas. Os testes realizados demonstram que a precisão acrescida teve um custo elevado na fluidez do sistema. Para determinar a capacidade máxima do *hardware*, os testes abaixo referem-se ao cenário de máximo desempenho, ou seja, sem visualização do resultado em tempo real e sem gravação do resultado, permitindo que todos os recursos computacionais sejam dedicados à inferência das redes.

Assim, realizou-se uma tabela (Tabela 3-1) onde compara-se os resultados relativos ao *frame rate* de output do sistema e distância de resposta correspondente a cada *hardware*.

Tabela 3-1 - Tabela de resultados de processamento relativos a YOLOPv2+YOLOv8

***Hardware* de Processamento**	**FPS**	**Distância por *frame* (60km/h) [m]**	**Distância de Resposta (60km/h) [m]**
ASUS A15	14.82	1.12	7.87
NVIDIA AGX Xavier	4.40	3.79	26.52
ASUS VIVOBOOK	4.19	3.98	27.85
Raspberry Pi 5	2.49	6.69	46.86

A análise destes resultados à luz das normas de segurança para sistemas ADAS revela a importância crítica da taxa de *frames*. O *frame rate* determina diretamente a distância de paragem necessária para evitar uma colisão. Por exemplo, a 80 km/h, se se aumentar a taxa de 10 FPS para 30 FPS pode reduzir a distância de paragem em cerca de 10 metros [42]. Isto porque uma vez que 80 km/h corresponde a 22.22 m/s, e sabendo que o tempo entre *frame* para 10 FPS é 0.1 s e para 30 FPS é 0.033 s, isto significa que entre cada *frame* para 10 FPS o veículo circula cerca de 2.22 m e a 30 FPS circula cerca de 0.74 m, diminuindo em cerca de 67% a distância entre *frame*.

No caso da Jetson, a 4.40 FPS, isto traduz-se numa distância entre *frame* de cerca de 3.79 metros para uma velocidade de 60km/h, ou seja, a cada 3.79 metros, o sistema processa um *frame*, isto corresponde a uma latência de perceção de cerca de 1.6 segundos. A uma velocidade de 60 km/h, o veículo percorreria aproximadamente 26.5 metros antes de os travões serem sequer acionados, o que é inaceitável para segurança ativa, sendo que uma distância de travagem segura seria de no máximo cerca de 19 m.

O artigo desmitifica que o *frame rate* não serve apenas para "fluidez visual", mas para reduzir a distância percorrida pelo veículo durante o tempo de cálculo do processador. Com uma distância por *frame* de 3.79 metros na Jetson, 3.98 metros no

Vivobook e 6.69 metros no Raspberry Pi, o sistema apresenta janelas de incerteza demasiado grandes para prevenir acidentes em tempo útil.

Os testes provaram que, embora o YOLOPv2 seja tecnologicamente superior ao processamento de imagem clássico, a sua carga computacional resulta em latências que comprometem a segurança rodoviária em *hardware* embebido. Esta fundamentação, justifica a transição para outra iteração do sistema.

3.2.4. Implementação de *Deep Learning* Otimizada (UFLDv2 + YOLOv8)

Nesta fase de desenvolvimento, o sistema evoluiu para uma arquitetura de *Deep Learning* focada na máxima eficiência computacional. Mantendo a necessidade de uma solução multitarefa, substituiu-se o modelo de segmentação panótica YOLOPv2 pelo Ultra Fast Lane Detection v2 (UFLDv2) para a deteção de faixas, mantendo-se o YOLOv8 para a deteção de objetos.

A transição para o UFLDv2 [43] representou uma mudança de paradigma na forma como o sistema processa a geometria da estrada. Enquanto o YOLOPv2 realiza uma segmentação semântica *píxel* a *píxel*, o que é computacionalmente dispendioso, o UFLDv2 trata a deteção de faixas como um problema de classificação baseada em linhas e grelhas (*row-based classification*). Esta abordagem permite identificar as faixas através de pontos de intersecção numa grelha pré-definida, reduzindo drasticamente o número de operações necessárias.

Ao contrário da abordagem panótica anterior, o UFLDv2 foi desenvolvido especificamente para atingir velocidades de deteção ultraelevadas. Esta leveza estrutural permite libertar recursos críticos da GPU para que o YOLOv8 possa operar com maior fluidez, otimizando o processamento total do ADAS.

O fluxo de execução foi otimizado para garantir que a inferência ocorra com a menor latência possível, utilizando a aceleração CUDA de forma mais equilibrada entre as duas redes. A rede neuronal *Ultra-Fast Lane Detection* (UFLD) processa a imagem e, em vez de gerar uma máscara densa, devolve coordenadas de *row anchors*. Este método é robusto mesmo em situações de falta de marcações visíveis, pois a rede efetua uma "previsão" da continuidade da faixa de rodagem com base na geometria global do *frame*.

Relativamente ao sistema de deteção de objetos, o YOLOv8 opera em paralelo, identificando obstáculos críticos tal como na iteração anterior. A lógica de processamento de profundidade permanece integrada, cruzando as *bounding boxes* com a matriz de profundidade da câmara. A interface visual e a lógica de alerta foram mantidas (Figura 3-11), mas beneficiam agora da maior taxa de atualização. Com o UFLD, a deteção de faixas torna-se mais estável a altas velocidades. O cálculo de desvio lateral mantém-se como a diferença entre o centro da estrada (extraído da grelha do UFLD) e o centro ótico da câmara.

Figura 3-11 - *Frame* de saída de sistema de deteção baseado em UFLDv2 e YOLOv8.

Esta configuração provou ser a mais eficaz para operação em tempo real. Tal como na fase anterior, os testes foram realizados no cenário de máximo desempenho para aferir os limites do *hardware*.

Tabela 3-2 - Tabela de resultados de processamento relativos a UFLDv2+YOLOv8

***Hardware* de Processamento**	**FPS**	**Distância por *frame* (60km/h) [m]**	**Distância de Resposta (60km/h) [m]**
ASUS A15	38.99	0.43	3.00
NVIDIA AGX Xavier	10.10	1.48	10.38
ASUS VIVOBOOK	11.23	1.65	11.55
Raspberry Pi 5	3.04	5.48	38.37

A análise destes resultados demonstra o sucesso desta iteração. Ao elevar a performance na Jetson para 10.10 FPS, o tempo de resposta total do sistema desceu para aproximadamente 0.62 segundos. Comparando com a fase anterior, a distância de resposta a 60 km/h baixou de 26.52 metros para apenas 10.38 metros. Este valor coloca o sistema ADAS num patamar de segurança ativa real, reagindo significativamente mais rápido do que um condutor humano médio. O *frame rate* elevado não providencia apenas fluidez, mas sim a capacidade técnica de reduzir a "zona morta" de cálculo, permitindo que a travagem de emergência ou o alerta de saída de faixa sejam acionados em tempo útil para prevenir o acidente.

3.2.5. Conclusão e Análise Comparativa

A evolução iterativa do *software* permitiu transformar um sistema inicialmente pesado e ineficiente numa solução capaz de operar em tempo real. A Tabela 3-3 compara os resultados obtidos de desempenho de cada iteração para cada *hardware* utilizado, servindo de base para a tomada de decisão sobre a viabilidade do *hardware*. Como demonstrado, a transição para a arquitetura UFLDv2 + YOLOv8 representou um salto qualitativo em todos os dispositivos testados.

Tabela 3-3 - Tabela final comparativa entre resultados das diferentes abordagens de sistema em cada *Hardware.*

	Tipo de Sistema de Deteção	*Frame Rate* [FPS]	Distância/ *Frame Rate* (60km/h) [m]	Distância de Resposta (60km/h) [m]	Preço/ *Frame Rate* (euros)	Aumento percentual de desempenho
ASUS TUF GAMING A15	YOLOPv2+YOLOv8	14,82	1,12	7,87	54,80 €	163%
	UFLDv2+YOLOv8	38,99	0,43	3,00	20,83 €	
Raspberry Pi 5	YOLOPv2+YOLOv8	2,49	6,69	47,86	35,74 €	22%
	UFLDv2+YOLOv8	3,04	5,48	38,37	29,28 €	
NVIDIA Jetson AGX Xavier	YOLOPv2+YOLOv8	4,40	3,79	26,52	192,61 €	130%
	UFLDv2+YOLOv8	10,10	1,48	10,38	83,91 €	
ASUS Vivobook	YOLOPv2+YOLOv8	4,19	3,98	27,85	210,02 €	168%
	UFLDv2+YOLOv8	11,23	1,65	11,55	78,36 €	

Com base nos resultados pode-se ainda concluir definitivamente a inviabilidade do Raspberry Pi 5**.** Apesar de apresentar um custo por FPS baixo (cerca de 29 €), o aumento de desempenho foi de apenas 22% e, com uma distância de resposta de cerca de 38 metros, o dispositivo é incapaz de garantir a segurança mínima necessária. Assim, conclui-se que o dispositivo não possui a capacidade necessária para tornar o processamento de algoritmos significativamente rápido.

Outro aspeto de notar é o custo por *frame*, onde é notório que o sistema baseado em YOLOPv2 é economicamente ineficiente nas plataformas portáteis e embebidas. Na Jetson Xavier, o custo de cada *frame* processado era de 192.61 €, valor que desceu para

83.91 € com a otimização para UFLDv2. No ASUS *Vivobook*, a otimização permitiu uma redução de custos de processamento de cerca de 62% (de 210.02 € para 78.36 € por FPS).

Assim, torna-se evidente a escolha do *hardware* para o desenvolvimento do sistema ADAS: NVIDIA Jetson AGX Xavier**.** Embora o ASUS TUF *Gaming* A15 apresente os melhores resultados, a Jetson Xavier destaca-se como a solução de desenvolvimento ideal. Com um aumento de desempenho de 130% na otimização do sistema de deteção de faixas de rodagem e uma redução da distância de resposta para 10,38 metros, a Jetson provou que a sua capacidade permite o uso de um sistema de tempo real. Os resultados validam a escolha da fusão dos algoritmos UFLDv2 e YOLOv8 na Jetson. Esta combinação permitiu não só reduzir a distância de resposta em mais de 60% comparativamente à iteração anterior, mas também otimizar o rácio de custo por desempenho, tornando o sistema financeiramente mais viável e tecnicamente seguro para a implementação de um protótipo ADAS funcional.

3.3. Integração de LiDAR

Nesta etapa, o sistema de aquisição de dados visual foi fundido com um sistema laser com o objetivo de obter dados de distâncias com maior precisão que a câmara. Isto, pois, a câmara tem vulnerabilidades quando submetida a certos ambientes nomeadamente condições climatéricas adversas.

3.3.1. Princípio de Funcionamento

Um dispositivo LiDAR, acrónimo para Light Detection and Ranging or Laser Imaging, Detection and Ranging, utiliza tecnologia Time of Flight (ToF), onde o sensor emite um feixe de luz até alcançar um objeto, alcançando o objeto a luz é então refletida e regressa à fonte de emissão. Uma vez que a velocidade da luz é constante, o sensor é então capaz de calcular a distância com base no tempo decorrido entre a emissão e receção do sinal como se pode analisar na Figura 3-12 - Cálculo de ToF num LiDAR de um ponto.

.

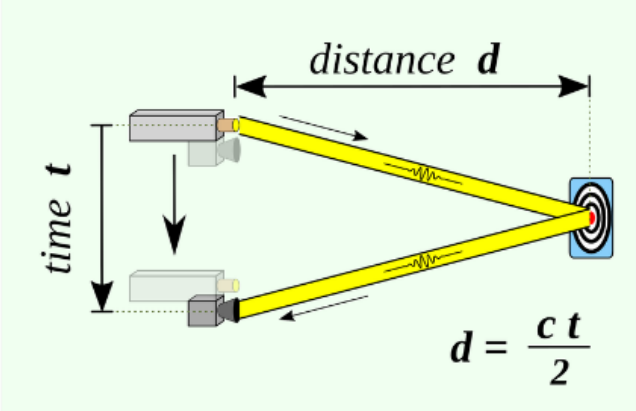


Figura 3-12 - Cálculo de ToF num LiDAR de um ponto. [44]

3.3.2. Interface com Sistema de Processamento

O dispositivo LiDAR utilizado no projeto, Garmin v3HP, tem dois modos de comunicação: I2C (*Inter-Integrated Circuit*) ou PWM (*Pulse Width Modulation*). Embora ter ambos os modos de comunicação disponíveis, a forma mais simples é através da comunicação I2C uma vez que a NVIDIA JETSON também possui este tipo de comunicação disponível nos seus terminais como mostra a Figura 3-13.

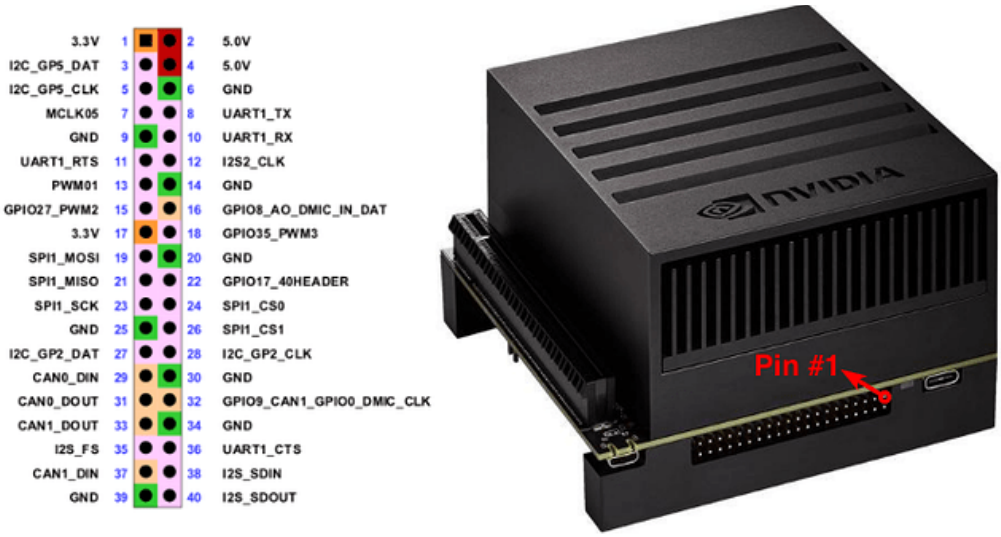


Figura 3-13 - Lista de terminais disponíveis na Nvidia Jetson Nano Xavier.

A instalação do componente é simples, uma vez que apenas precisa de quatro ligações à NVIDIA: alimentação de 5 volts, massa, I2C SDA e I2C SCL como mostra a Figura 3-14. Neste caso não é necessária a utilização de resistências uma vez que os cabos são relativamente curtos.

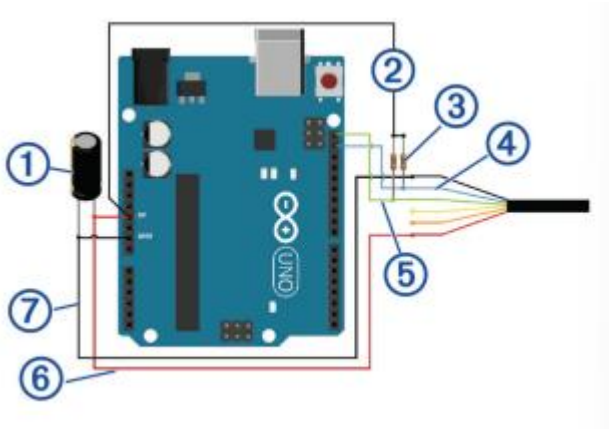


Figura 3-14 - Esquemático de ligações de um LiDAR Garmin v3HP a um sistema de processamento via I2C. [45]

Para aceder aos dados de distância do LiDAR, a NVIDIA comunica com o sensor através do I2C enviando um byte de comando para o LiDAR ativando assim o feixe laser. O LiDAR devolve então a medição dividida em dois bytes, aos quais consegue-se obter o valor utilizando a seguinte fórmula.

$$Distância[cm] = (data[0] \ll 8) + data[1]$$

Com a distância ao objeto diretamente à frente do LiDAR sendo calculada, basta criar um algoritmo para determinar a prioridade de dados vindos da câmara e do LiDAR.

3.4. Integração com Barramento CAN

A comunicação com a rede interna do veículo é um passo essencial para o projeto uma vez que permite analisar dados provenientes do veículo que são fundamentais para melhor compreender a situação em que o veículo se encontra em tempo real. Com dados provenientes do ambiente circundante do veículo e do próprio veículo, assim consegue-se melhor determinar a existência de possíveis riscos eminentes.

A importância estratégica da comunicação com o veículo baseia-se em diferentes aspetos, nomeadamente a contextualização dos alertas. O risco de embate de um obstáculo que se encontra a 10 metros do veículo difere para um veículo que se desloque a 20 km/h ou 80km/h uma vez que a distância de travagem diminui com o aumento de velocidade do veículo. Para contabilizar esse risco, uma das variáveis necessárias para contabilizar esse risco é a velocidade do veículo. Outro aspeto a monitorizar é a reação

do condutor perante situações de risco. Para tal, é necessário obter dados do pedal de travão e da posição do pedal de acelerador. Assim, o sistema consegue determinar se o condutor está ciente do risco iminente e se está a reagir ao mesmo.

3.4.1. Implementação no Sistema de Processamento

Para o desenvolvimento da interface entre o barramento CAN do veículo e do sistema de processamento, utilizou-se o simulador de linha CAN do Formula Student do Institudo Politécnico de Leiria (FSIPLeiria) como mostra na Figura 3-15 disponível no Departamento de Engenharia Automóvel na ESTG.

Figura 3-15 - Simulador de Linha CAN FSIPLeiria.

Este simulador permite adquirir e enviar dados através de um barramento CAN, algumas das variáveis que podem ser adquiridas são: velocidade de rotação de motor, pressão, temperatura, posição de pedal de acelerador e interruptor de travão. Contudo, é necessário algum tipo de dispositivo capaz de fazer a ponte entre os dados do barramento CAN e a NVIDIA. Para tal, decidiu-se utilizar um módulo CAN baseado no *transceiver* SNH65HVD230 de 3.3 V, como mostra a Figura 3-16. Decidiu-se utilizar este módulo porque a NVIDIA apenas recebe sinais de 3.3 V, logo um módulo de comunicação CAN típico de 5 volts não iria ser adequado.

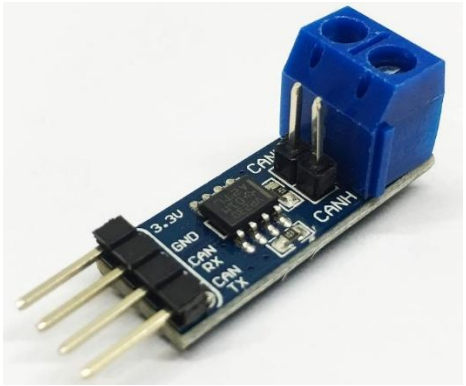


Figura 3-16 - Módulo CAN com transceiver SN65HVD230.

Para adquirir e descodificar a mensagem do barramento CAN na NVIDIA, utilizaram-se os terminais dedicados, como mostrado na Figura 3-13. Para tal, é necessário fazer a ativação da porta CAN na NVIDIA, configurando-a para a velocidade de comunicação de 500 kbps. Após isso, é necessário descodificar a mensagem CAN recebida, para isso é necessário saber a lista de ID da mensagem e a fórmula de descodificação de cada variável, disponível na Tabela 3-4.

Tabela 3-4 - Protocolo CAN do simulador FSIPLeiria

Variáveis	ID	Fórmula
RPM	0x60	[RPM] = B0 * 256 + B1
OP	0x60	[Psi] = (B2 * 256 + B3) / 10
ET	0x60	[ºC] = ((B4 * 256 + B5) / 10) - 40
TPS	0x6A	[%] = (B0 * 256 + B1) / 10
SPS	0x6A	[mm] = (B4 * 256 + B5) / 10
SW Brake	0x70	B0 / bit0 (status)
SW 1	0x70	B0 / bit1 (status)
SW 2	0x70	B0 / bit2 (status)
LED Brake	0x70	B0 / bit4 (status)
LED 1	0x70	B0 / bit5 (status)
LED 2	0x70	B0 / bit6 (status)
LED Brake	0x02A	B0 / bit0 (control)
LED 1	0x02A	B0 / bit1 (control)
LED 2	0x02A	B0 / bit2 (control)

Descodificada a mensagem enviada através da linha CAN, consegue-se integrar a informação proveniente da mesma no sistema de modo a identificar com maior precisão situações de risco para o condutor, distinguindo cenários de risco iminente de cenários de risco controlado pelo condutor.

3.5. Integração de Alertas Visuais e Sonoros

Com o sistema de aquisição de dados construído, resta desenvolver o sistema de alertas para o condutor. Os alertas devem ser suficientemente intuitivos e percetíveis para o condutor, garantindo a sua eficácia sem comprometer a atenção do condutor, ou seja, sem causar distrações.

3.5.1. Alertas Visuais

Assim, decidiu-se utilizar um conjunto de luzes LED de 12 V e um *buzzer* de 12 V para diferentes tipos de cenários.

Para tal, decidiu-se utilizar três fitas de LED com as seguintes funções:

- Fita LED azul: quando acesa indica que o sistema está a monitorizar objetos e faixas de rodagem;
- Fita LED vermelha 1: quando acesa indica uma situação de possível perigo por exemplo, quando a distância de segurança para o veículo da frente diminui;
- Fita LED vermelha 2: quando acesa indica uma situação de perigo iminente. Quando esta acende também permanece acesa a fita LED vermelha 1.

Para controlar estes LED utilizando a NVIDIA utilizam-se interruptores inteligentes que funcionam como uma ponte entre os sinais de baixa potência provenientes da NVIDIA com a alta potência dos LED. Para tal utilizam-se *Low Side Switch* BV1LB045FPJ-C fabricados pela Rohm (Figura 3-17), que consistem em interruptores inteligentes que quando recebem um sinal de baixa potência, fecham o circuito à massa, ou seja, os LED estão sempre ligados a 12 V mas interrompidos à massa pelos interruptores.

Figura 3-17 -Imagem ilustrativa de um *Low Side Switch* BV1LB045FPJ-C. [46]

3.5.2. Alertas Sonoros

Alertas sonoros também são essenciais para avisar o condutor de riscos para a condução e, para simplificar o sistema de alerta decidiu-se implementar os alertas sonoros em paralelo com os visuais.

Para tal utiliza-se um *buzzer* de vários tons com a referência RS 8377850 de 105 dB como mostra a Figura 3-18.

Figura 3-18 - Buzzer Multi-tons RS 8377850.

Este *buzzer* funciona a 12 V, e possui 3 níveis diferentes de tons que são ativados pela ligação à massa. Assim, este *buzzer* está ligado constantemente a 12 V tal como os LED e é ativado paralelamente em conjunto com os LED através dos interruptores inteligentes que são ativados pela NVIDIA.

4. Desenvolvimento do *software* e PCB

A eficácia de um sistema ADAS reside na sua capacidade de processar, em frações de segundo, um volume massivo de dados de modo a gerar uma resposta de segurança fidedigna. Se os sensores e a infraestrutura de comunicação representam o sistema sensorial e nervoso do projeto, o *software* constitui o centro cognitivo onde a inteligência artificial e a lógica de controlo convergem.

O desenvolvimento do *software* para este sistema enfrentou o desafio crítico da simultaneidade, ou seja, a necessidade de interpretar o ambiente visual através de redes neuronais complexas, enquanto se monitoriza ininterruptamente a dinâmica do veículo e a telemetria laser. Neste contexto, o *software* não atua apenas como um interpretador de dados, mas como um motor de decisão em tempo real, onde a robustez do código e a eficiência dos algoritmos determinam a fiabilidade da assistência prestada ao condutor. A arquitetura lógica aqui implementada reflete o esforço de transformar modelos matemáticos e fluxos digitais numa interface de proteção ativa, capaz de distinguir cenários de condução normal de situações de perigo iminente.

4.1. Arquitetura do *Software*

Para garantir que o sistema ADAS opere em tempo real, optou-se por uma arquitetura multitarefa (execução paralela). Esta abordagem permite que tarefas diferentes ocorram simultaneamente mantendo um tempo de reação do sistema rápido e preciso.

Assim, o código desenvolvido em linguagem P*ython* encontra-se dividido em dez secções diferentes:

1. ***Imports*:** Nesta secção é feita a importação de todas as bibliotecas auxiliares que permitem o uso de diferentes funções, nomeadamente das redes neuronais, câmara de profundidade, barramento CAN, entre outros.
2. ***Constants and Settings*:** Aqui são definidos todos os parâmetros fixos que definem o comportamento do sistema, nomeadamente ficheiros dos modelos das redes neuronais, configurações do I2C, barramento CAN, entre outros.
3. ***Global Variables*:** Onde são definidas todas as variáveis que são acedidas por diferentes tarefas do código, nomeadamente matrizes de dados.

4. **GPIO / CAN:** Nesta secção são configurados os pinos utilizados como saídas e inicializados os comandos de comunicação CAN e I2C.
5. ***Graphic Interface*:** Onde são criados os interfaces gráficos utilizados para controlar o sistema a nível de deteção de objetos e faixas de rodagem e ainda, variáveis de teste como gravações.
6. ***Threads* – LIDAR / CAN:** Onde é feita a comunicação com o barramento CAN e com o LiDAR através de comunicação I2C.
7. ***Models*:** Secção que contém as funções que carregam os modelos pré-treinados das redes neuronais de deteção de objetos e faixas de rodagem.
8. ***Dashboard*:** Secção responsável por transmitir toda a informação útil de forma visual em tempo real para um ecrã (testes).
9. ***Roadway and Object Detection*:** Contém as funções de processamento de imagens onde é feita a deteção de objetos e faixas de rodagem assim como parâmetros essências para cálculo e determinação de situações de risco,
10. ***Main Loop*:** Função principal do sistema de ciclo infinito onde são inicializadas todas as funções principais, adquiridos dados brutos, processamento de dados e determinação de cenários de risco.

O fluxo de execução do software inicia-se com a inicialização dos módulos de comunicação e dos *drivers* da NVIDIA, como se pode analisar na Figura 4-1. Após esta etapa, o sistema apresenta o menu de configurações ao utilizador, onde são definidas as preferências de operação e os modos de deteção. Com estas definições estabelecidas, o software avança para a configuração dos terminaias da Xavier e para o carregamento dos modelos das redes neuronais YOLOv8 e UFLDv2. Esta primeira fase conclui-se com o lançamento das tarefas de inicialização de comunicação, nomeadamente os protocolos I2C, para leitura do sensor LiDAR, e o protocolo CAN, que correm de forma independente para assegurar a fluidez dos dados.

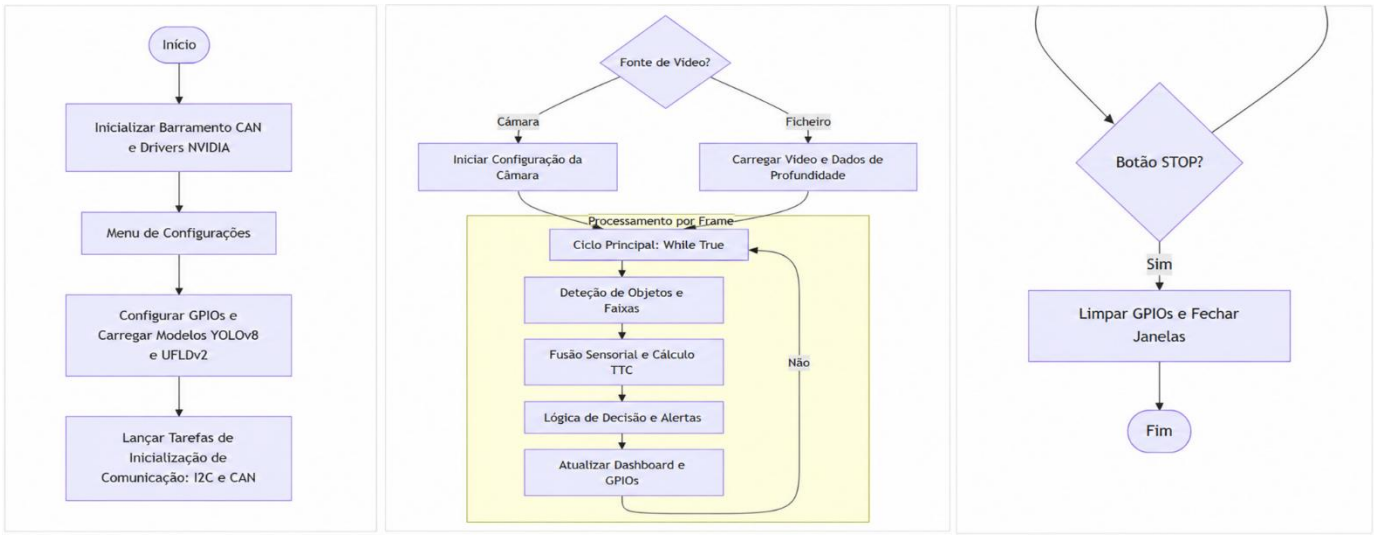


Figura 4-1 - Fluxograma geral do algoritmo desenvolvido.

A segunda parte do fluxo (Figura 4-1) foca-se na gestão da entrada de dados e no ciclo de execução principal do sistema ADAS. O programa avalia a fonte de vídeo escolhida no menu inicial, e procede com a configuração da câmara ou o carregamento de ficheiros de vídeo e dados de profundidade pré-gravados. Independentemente da fonte de dados, o fluxo converge para o ciclo principal onde ocorre o processamento por *frame*. Dentro deste bloco de processamento, o *software* executa sequencialmente a deteção de objetos e faixas de rodagem na imagem, realizando ainda, o cálculo do TTC [47] e a aplicação da lógica de decisão para determinar os alertas necessários com base no comportamento do condutor no cenário atual. Esta fase encerra-se com a atualização constante do *dashboard* visual e dos estados dos terminais GPIO.

A parte final do fluxograma (Figura 4-1) detalha o procedimento de saída e a terminação do programa. Durante a execução do ciclo de processamento, o *software* verifica continuamente a condição do botão de interrupção do processamento, presente na interface de controlo. Caso o botão não seja acionado, o fluxo de execução retrocede para o início do ciclo de processamento para analisar o *frame* seguinte. Se o utilizador optar por interromper o processo, o sistema sai do ciclo para executar as rotinas de finalização, que incluem a colocação de todos os terminais em estado 0, garantindo que nenhum alerta físico permanece ativo por erro, e o fecho de todas as janelas gráficas.

4.2. Desenvolvimento de Tarefas

A viabilidade de um sistema ADAS depende da eficiência individual de cada tarefa, garantindo que o tempo de processamento de uma função não crie bloqueios no fluxo principal. Por este motivo, cada componente do *software* foi desenvolvido sob princípios de otimização computacional, visando minimizar a latência e assegurar a continuidade da execução global do algoritmo.

4.2.1. Parâmetros de Inicialização

Nesta secção, são definidas todas as configurações e variáveis de controlo das diversas tarefas do *software*, como se pode ver na Figura 4-2.

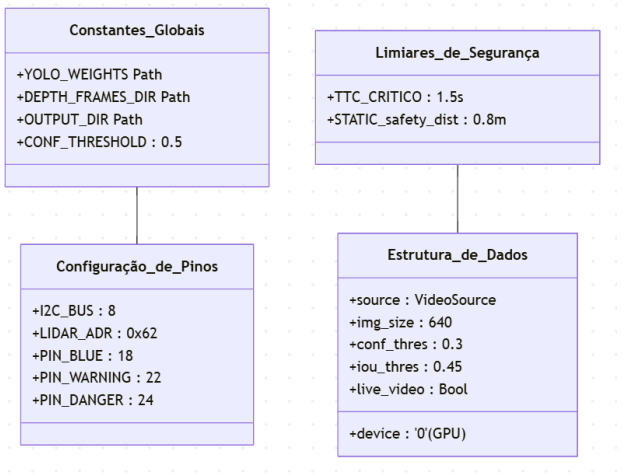


Figura 4-2 - Secção 2 de software de constantes e parâmetros de inicialização.

Esta fase de configuração começa pela gestão de modelos e dados, onde o sistema mapeia os caminhos para o modelo pré-treinado YOLOv8 e para as bases de dados de profundidade e vídeo utilizadas em testes de validação. Simultaneamente, estabelecem-se os critérios de filtragem para o processamento de imagem, destacando-se a definição de um limiar de confiança mínimo de 50%, o que garante que apenas objetos com alta probabilidade sejam processados, reduzindo potenciais falsos positivos no sistema.

No que respeita à interface com o utilizador, esta secção centraliza as configurações de *hardware*, nomeadamente os endereços de comunicação I2C necessários para a ativação e leitura do LiDAR, bem como a atribuição dos terminais responsáveis pelo acionamento das fitas de LED. Esta organização é complementada pela

definição de métricas de segurança críticas, como o tempo limite para colisão de 1,5 segundos e as distâncias mínimas de segurança em modo estático.

Dando continuidade à arquitetura, a secção 3 do software foca-se na implementação das variáveis globais, conforme ainda visto na Figura 4-2. Esta estrutura é fundamental para suportar o paradigma de execução multitarefa, funcionando como uma memória partilhada que permite a sua atualização em tempo real. Através deste mecanismo, dados críticos, tais como a velocidade do veículo e o estado dos pedais extraídos via barramento CAN, ou a distância ao objeto obtida pelo LiDAR, ficam instantaneamente disponíveis para todas as tarefas. Esta fluidez na troca de informação é o que permite ao algoritmo de decisão correlacionar a perceção visual com a dinâmica real do automóvel, assegurando uma resposta imediata perante cenários de risco.

4.2.2. Configuração de Comunicação

A operacionalidade do sistema de comunicação do sistema ADAS é estabelecida na quarta secção do *software*, onde se procede à configuração das interfaces de entrada e saída da NVIDIA Jetson Xavier AGX. Como ilustrado na Figura 4-3, o sistema inicializa os terminais no modo de numeração física da placa, definindo as saídas digitais para o controlo dos alertas visuais e garantindo que todos os periféricos iniciam em estado desligado por motivos de segurança.

Complementarmente, esta secção gere a ativação do barramento CAN através de uma função dedicada que executa comandos de baixo nível no sistema operativo. Este processo inclui o carregamento de *drivers* e a configuração da interface de rede automóvel com uma velocidade de comunicação de 500 kbps. Esta configuração é vital para assegurar que a Jetson consiga comunicar com o veículo sem erros de sincronização, permitindo a leitura ininterrupta da dinâmica do automóvel.

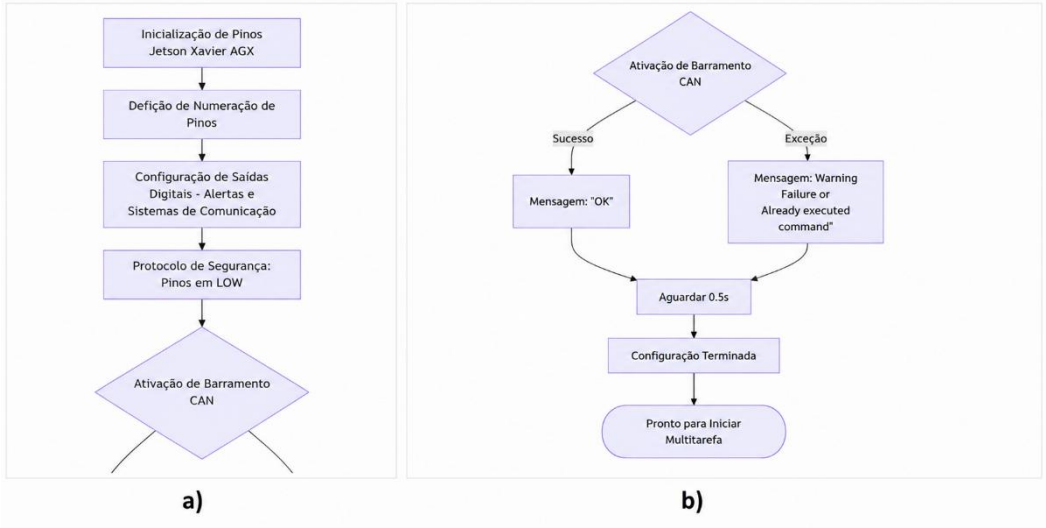


Figura 4-3 - Fluxograma de configuração de comunicação.

Caso a ativação do barramento CAN seja concluída com sucesso, a mensagem "OK" aparece, caso contrário, aparece uma mensagem de erro correspondente à causa da falha.

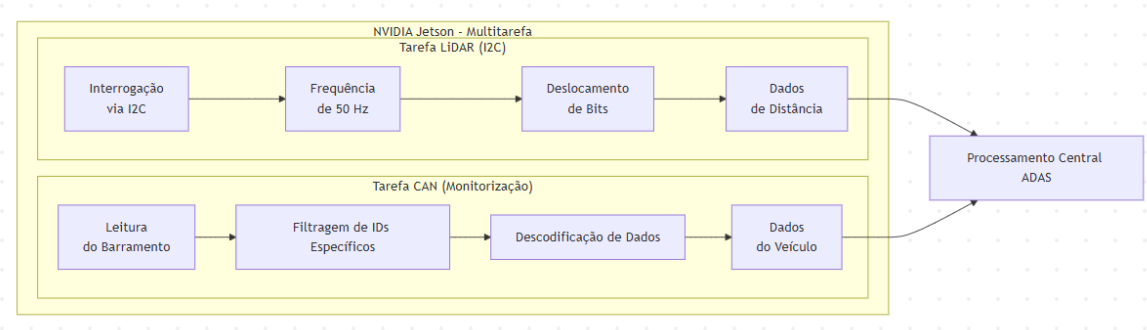


Figura 4-4 - Fluxograma de aquisição de dados via barramento I2C e CAN.

Para viabilizar o desempenho em tempo real, a sexta secção do código implementa o método de processamento multitarefa através de tarefas independentes, como se observa na Figura 4-4.

A tarefa dedicada ao LiDAR interroga o sensor via protocolo I2C a uma frequência de 50 Hz, utilizando operações de deslocamento de bits para reconstituir os dados de distância com precisão centimétrica. Simultaneamente, a tarefa de leitura CAN

monitoriza o barramento do veículo, filtrando e descodificando mensagens específicas de velocidade e estado dos pedais.

4.2.3. Interface Gráfico

A quinta secção do *software* foca-se na interação entre o utilizador e o sistema ADAS, utilizando a biblioteca Tkinter para criar uma interface intuitiva de configuração e controlo. Esta interface gráfica é explicada no Anexo 2 – Manual de Utilizador .

4.2.4. Redes Neuronais

A configuração e carregamento das redes neuronais é feita na secção 7, onde as mesmas são carregadas na GPU da NVIDIA. Esta etapa é fundamental pois garante que a elevada carga computacional exigida pelos modelos de deteção, seja processada com o mínimo de latência possível. Para o carregamento dos modelos, são usadas duas tarefas individuais: uma para o carregamento do modelo de deteção de objetos, outra para o carregamento do modelo de deteção de faixas de rodagem como se pode interpretar na Figura 4-5.

A tarefa de carregamento do modelo de deteção de objetos consiste numa função simples de carregamento do modelo pré-treinado para a CPU ou GPU da NVIDIA. Nesta tarefa, existe uma operação que permite otimizar o processo matemático (*fuse*), esta operação funde as camadas de convolução num único bloco matemático, permitindo assim reduzir o número de operações por *frame* e, consequentemente, obter uma resposta mais rápida a riscos detetados.

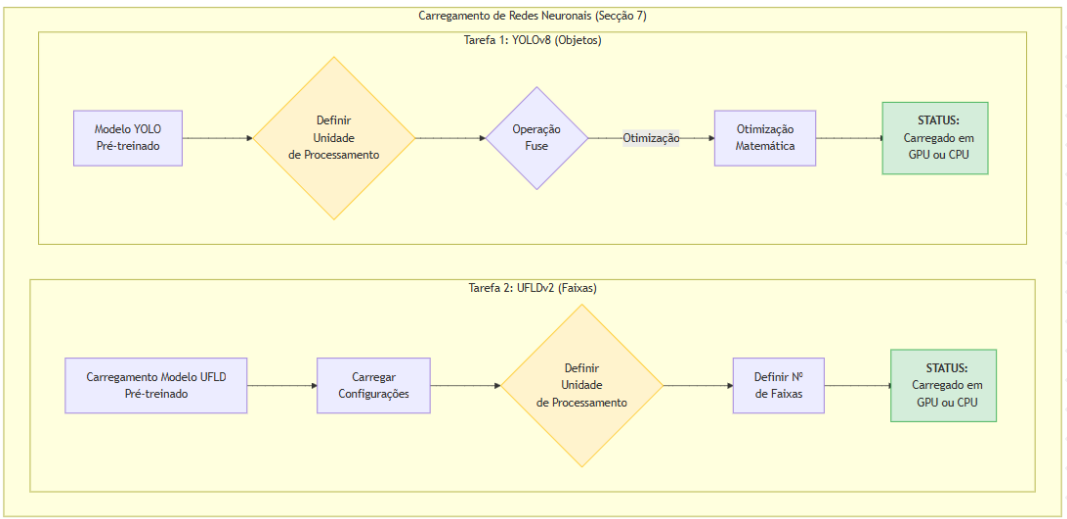


Figura 4-5 - Fluxograma com as tarefas de carregamento de modelo de deteção

A tarefa de carregamento do modelo de deteção de faixas de rodagem é muito semelhante à tarefa de carregamento do modelo YOLOv8, no sentido em que também consiste no carregamento de um modelo pré-treinado. Embora isso, também é feito o carregamento de um ficheiro de configurações onde é definido também o número de faixas a detetar.

A tarefa da Figura 4-6 constitui a função de perceção de objetos, sendo responsável por identificar obstáculos e calcular a sua distância real em relação ao veículo. O processo inicia-se com a aplicação do modelo YOLOv8 sobre os *frame*s de vídeo, configurado para filtrar apenas classes de interesse rodoviário, como veículos e peões. Para otimizar a análise de risco, o algoritmo estabelece um "corredor de segurança" central na imagem. Esta lógica permite ao sistema priorizar o processamento de objetos que se encontram na trajetória direta do veículo, ignorando elementos em faixas adjacentes que não representem perigo imediato. Para além disso, o sistema está limitado a detetar apenas certas classes, nomeadamente, carros, pedestres e semáforos.

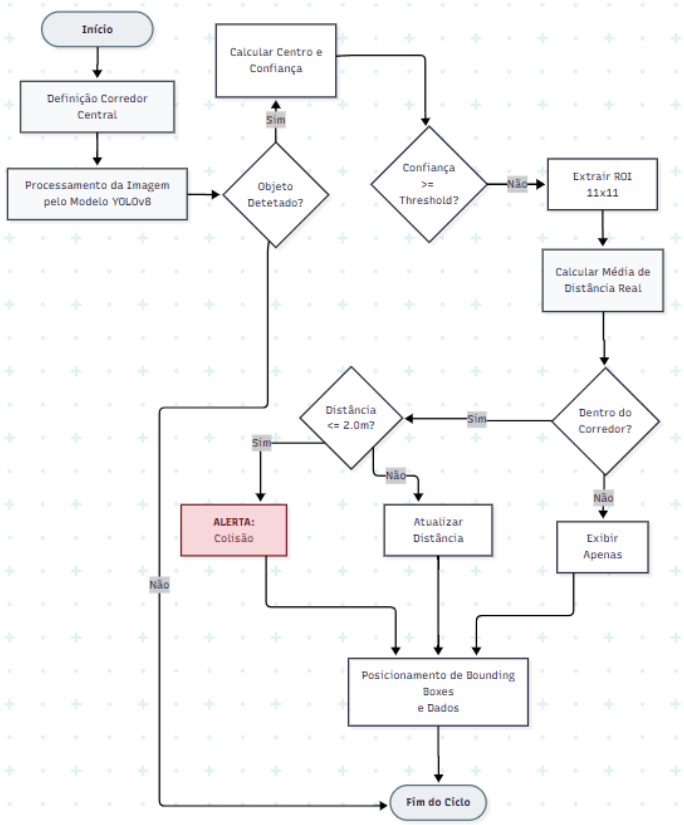


Figura 4-6 - Fluxograma de tarefa de deteção de objetos

Nesta função reside uma metodologia de fusão sensorial para a estimativa de distância ao objeto. Em vez de utilizar apenas a estimativa visual da rede neuronal, o *software* cruza as coordenadas das caixas delimitadoras (*bounding boxes*) com os dados matriciais da câmara de profundidade Intel RealSense. Para garantir a robustez da leitura e mitigar erros causados por reflexos ou oclusões parciais, o algoritmo não extrai apenas um ponto de profundidade único. Em vez disso, define uma região de interesse (ROI) de 10x10 *píxeis* no centro geométrico do objeto detetado, calculando a média dos valores válidos de profundidade.

Este valor médio é posteriormente convertido para metros, fornecendo uma distância fidedigna que é comparada com o limiar crítico de segurança. Caso um objeto seja validado dentro do corredor de segurança a uma distância inferior a 2 metros, a tarefa sinaliza um estado de colisão iminente. Esta informação é então transmitida ao ciclo principal do *software*, servindo para a ativação dos alertas visuais e sonoros, dependendo dos dados de velocidade do veículo e da reação do condutor monitorizados via barramento CAN.

A tarefa da Figura 4-7 constitui a tarefa de processamento visual para a monitorização da trajetória do veículo em relação às faixas de rodagem. O processo inicia-se com o pré-processamento do *frame* de vídeo, que é redimensionado para uma resolução aceitável para o modelo para permitir a inferência do modelo UFLDv2 na NVIDIA. Após a execução da rede neuronal, o *software* interpreta a saída matricial através de um cálculo de probabilidades, que identifica a posição mais provável das linhas de rodagem com base numa grelha de âncoras horizontais e verticais.

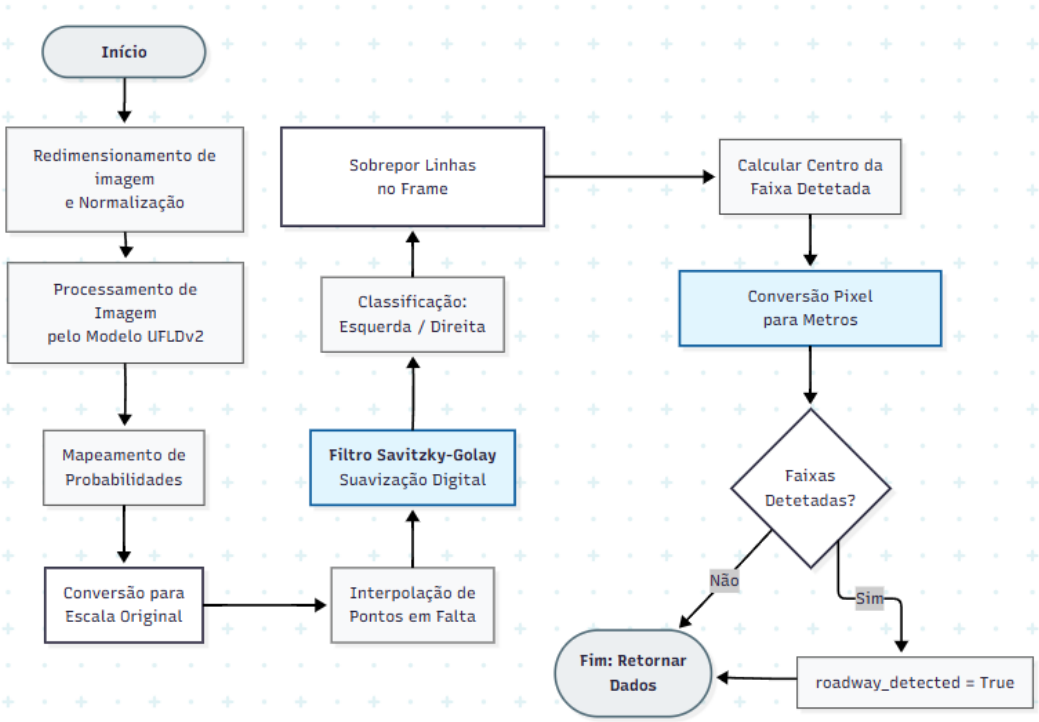


Figura 4-7 - Fluxograma de tarefa de deteção de faixas de rodagem

Uma fase crítica desta função é a conversão das coordenadas detetadas para as dimensões reais da imagem original. Como o modelo fornece pontos discretos, o algoritmo utiliza uma interpolação linear para preencher eventuais falhas de deteção e aplica o filtro de suavização de *Savitzky-Golay*. Este tratamento digital é fundamental para eliminar oscilações abruptas nas linhas desenhadas, garantindo uma representação visual estável e fidedigna da estrada, mesmo em situações de ruído visual ou pavimentos degradados.

Finalmente, o *software* utiliza a geometria das linhas esquerda e direita para calcular o posicionamento do veículo. Ao determinar o centro geométrico da faixa de rodagem e compará-lo com o centro da imagem (que representa o centro do automóvel),

o sistema obtém o desvio lateral em *píxeis*. Este valor é convertido para unidades métricas através de um fator de escala baseado na largura padrão das vias de rodagem (3,5 metros). O resultado deste cálculo permite ao sistema ADAS monitorizar, em tempo real, se o condutor está a transpor os limites de segurança da faixa, servindo de base para a ativação dos alertas visuais e sonoros.

4.2.5. *Dashboard* de Resultados

A tarefa da secção 8 (Figura 4-8) é responsável pela criação da interface homem-máquina, sobrepondo uma camada gráfica com dados do processamento de imagem sobre o fluxo de vídeo permitindo a monitorização em tempo real do sistema. O processo inicia-se com a criação de um painel lateral semitransparente, utilizando a sobreposição de um retângulo preto com opacidade ajustada, como se pode analisar na Figura 4-9. Neste painel, o *software* integra graficamente um assistente de manutenção na faixa através de um ícone dinâmico, cujas cores se alteram consoante o estado de segurança: as linhas surgem a cinzento quando não há deteção de faixas, a verde quando o veículo está centrado na faixa de rodagem e a vermelho sempre que o desvio lateral excede os limites de tolerância (0,5 metros). Abaixo deste ícone, o sistema apresenta a informação obtida via barramento CAN e sensores adjacentes, incluindo a velocidade em km/h, a distância ao objeto e o tempo para colisão. O sistema utiliza uma lógica de cores nestes textos para destacar situações críticas, como por exemplo para distâncias inferiores a 2 metros. Para além disso, o sistema converte estados lógicos em informação intuitiva, como a indicação de ativação dos travões e a posição do pedal de acelerador.

Finalmente, a função gere os avisos visuais de risco centralizados na imagem. Caso o algoritmo determine um risco de colisão iminente, é gerado um alerta intermitente com a mensagem "BRAKE IMMEDIATELY!", cujo efeito de piscar é controlado por uma função temporal para maximizar a atenção do utilizador. Adicionalmente, em situações onde a distância de segurança é insuficiente para a velocidade atual, o sistema projeta um aviso amarelo preventivo com a mensagem “INCREASE SAFETY DISTANCE”, instruindo o condutor a aumentar a distância de segurança, garantindo assim que o *dashboard* atue não apenas como um mostrador de dados, mas como um elemento ativo de auxílio à condução segura.

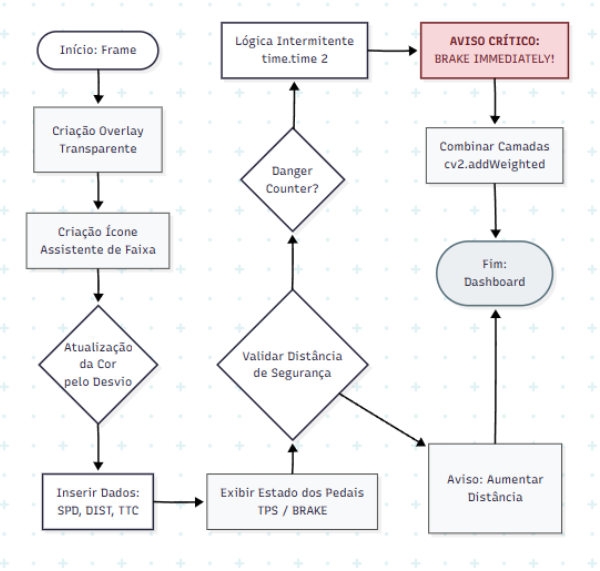


Figura 4-8 - Fluxograma de tarefa de apresentação de resultados no *Dashboard.*

O *dashboard* apresenta ainda no canto superior esquerdo, o valor de *frame*s por segundo a serem processados como mostra a imagem abaixo. Assim, o *dashboard* apresenta uma estrutura gráfica simples e intuitiva que se torna funcional ao permitir que o utilizador valide, com um único olhar, a correlação entre os dados sensoriais brutos e as decisões tomadas pelos algoritmos.

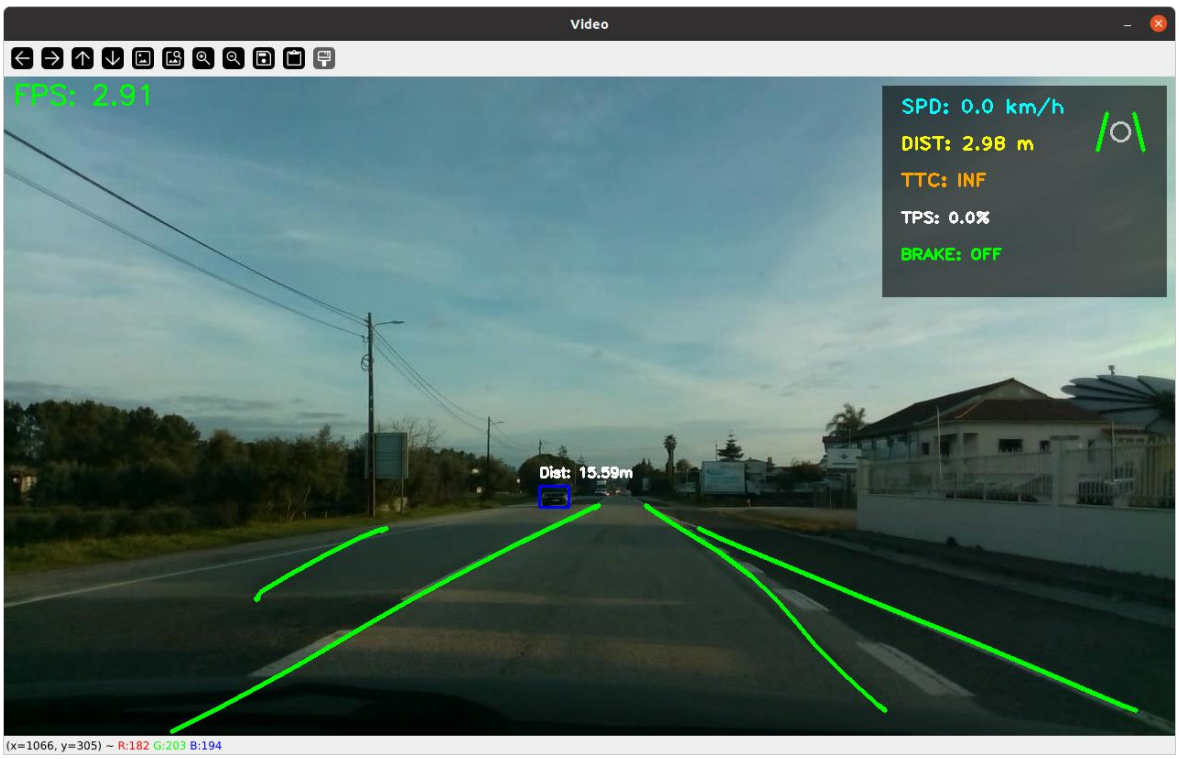


Figura 4-9 - *Dashboard* com apresentação visual de dados em tempo real.

4.2.6. Ciclo Principal

A tarefa de ciclo principal da secção 10, como se pode analisar na Figura 4-10, é onde se processam os cálculos de consideração de riscos no sistema ADAS. Para além disso, é nesta secção onde fluxo de execução é condicionado pelas escolhas efetuadas pelo utilizador nos menus de configuração iniciais. Aqui, o *software* interpreta se deve operar em tempo real utilizando a câmara Intel RealSense ou através de gravações para efeitos de teste, e define se o processamento incidirá sobre objetos, faixas de rodagem ou ambos. Uma vez estabelecidos estes parâmetros, são lançadas as tarefas paralelas para a monitorização do sensor LiDAR e do barramento CAN, garantindo que o estado dinâmico do veículo (velocidade, aceleração e travagem) seja integrado sem latência no ciclo de decisão.

A distância ao obstáculo frontal é calculada comparando os dados do LiDAR com a profundidade visual da câmara, selecionando-se o valor mínimo para o cálculo do TTC. O TTC é o indicador central de risco, sendo calculado pela razão entre a distância e a velocidade absoluta do veículo. Para este projeto, foram selecionados limiares de tempo específicos baseados em estudos de tempo de reação humana [47]:

- **TTC < 1.5 segundos:** Este valor foi escolhido como o limiar crítico por representar o tempo médio necessário para que um condutor atento perceba um perigo e inicie uma manobra de emergência. Abaixo deste valor, a colisão é considerada iminente se não houver intervenção imediata.
- **TTC entre 1.5s e 3.0s:** Definido como zona de aviso (aproximação rápida), permitindo uma reação preventiva e suave por parte do condutor.

A ponderação dos riscos é feita cruzando o TTC com os dados dinâmicos do barramento CAN. O sistema monitoriza a velocidade, considerando a desativação de alertas para velocidades abaixo de 10 km/h de modo a evitar falsos positivos em manobras de estacionamento por exemplo, assim como o estado dos pedais. Se o TTC for inferior a 1.5s e o sistema detetar, via CAN, que o condutor não está a pressionar o travão, o LED de perigo (vermelho) é ativado de forma fixa. No entanto, se o condutor já iniciou a reação de travagem, o sistema altera o alerta para um modo intermitente (piscar), servindo como confirmação de que a manobra de emergência está a ser monitorizada.

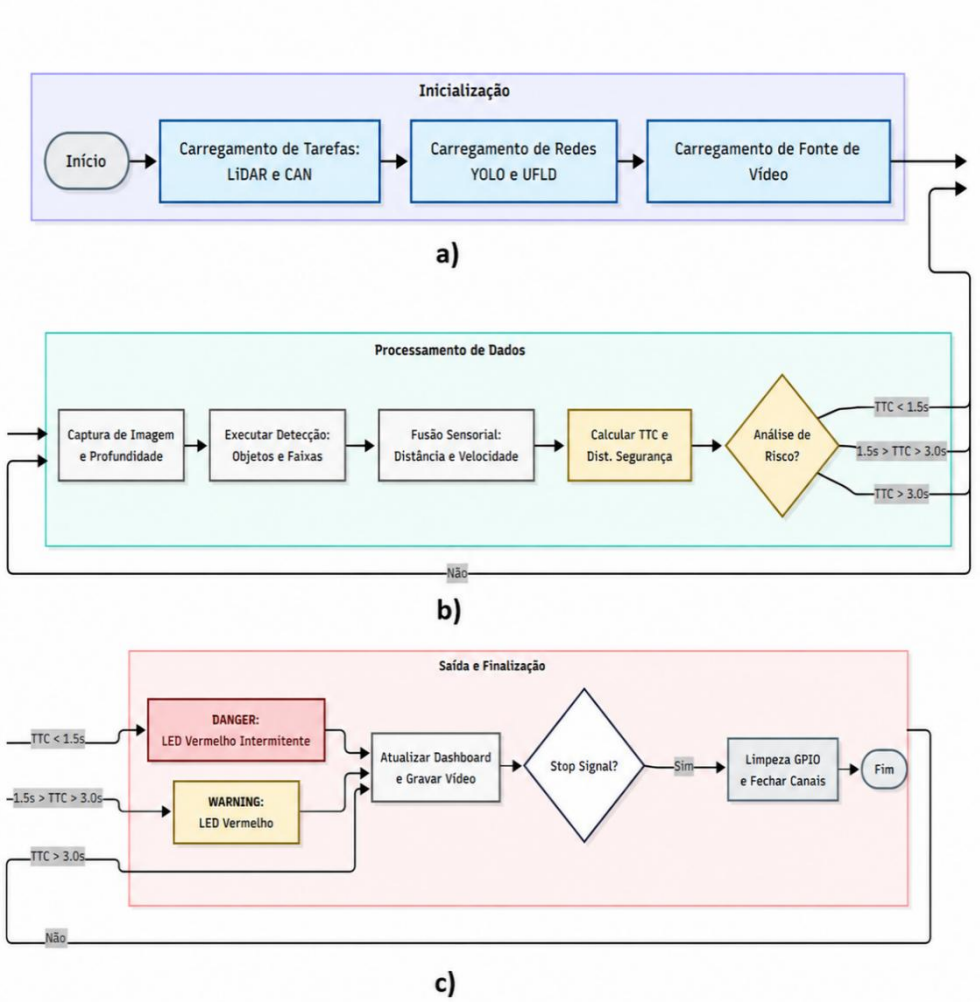


Figura 4-10 - Fluxograma do ciclo principal do algoritmo.

Relativamente ao auxílio de condução lateral, o sistema utiliza o desvio calculado em metros para graduar os avisos. Foi estabelecido um limiar de 0.4 metros para avisos amarelos (pisar a linha) e 0.6 metros para alertas críticos (transposição de faixa). Esta calibração impede que pequenas correções de trajetória ativem alertas desnecessários, mantendo a eficácia da assistência. Todo este conjunto de dados (velocidade, distância, pedal do travão, pedal de acelerador, desvio), converge para uma gestão de alertas inteligente que só valida o perigo através de contadores de persistência (4 *frames* consecutivos), garantindo que o sistema ADAS seja robusto contra ruídos sensoriais e ofereça uma proteção fidedigna em cenários reais.

4.3. Desenvolvimento de PCB

O teste do *hardware* para o projeto seguiu uma metodologia incremental, garantindo que cada interface de comunicação com cada componente fosse validada antes da sua integração final. Durante as fases de teste de componentes e de desenvolvimento do *software*, utilizou-se uma *breadboard* (placa de ensaio) como plataforma de prototipagem. Esta abordagem permitiu verificar a compatibilidade elétrica entre os componentes e a NVIDIA, ajustar os componentes e validar a comunicação CAN e I2C de forma flexível e segura.

4.3.1. Prototipagem em *Breadboard*

Como referido anteriormente, durante a fase de desenvolvimento do projeto, as ligações entre o sistema de processamento principal e os respetivos componentes associados foram realizadas com auxílio de uma *breadboard.*

As ligações de alimentação dos componentes foram feitas utilizando o terminal 1 de 3.3V, o terminal 2 de 5V, e os terminais de massa disponíveis na NVIDIA. Assim, permitiu-se que o LiDAR seja alimentado utilizando os 5V disponíveis pela NVIDIA. Ainda se referindo ao LiDAR, a comunicação foi estabelecida utilizando o protocolo I2C, utilizando os termianis 3 e 5 da NVIDIA. Para além disso, instalou-se um condensador de 680uF na alimentação do LiDAR conforme recomendado no manual e como pode ser visto na figura abaixo.

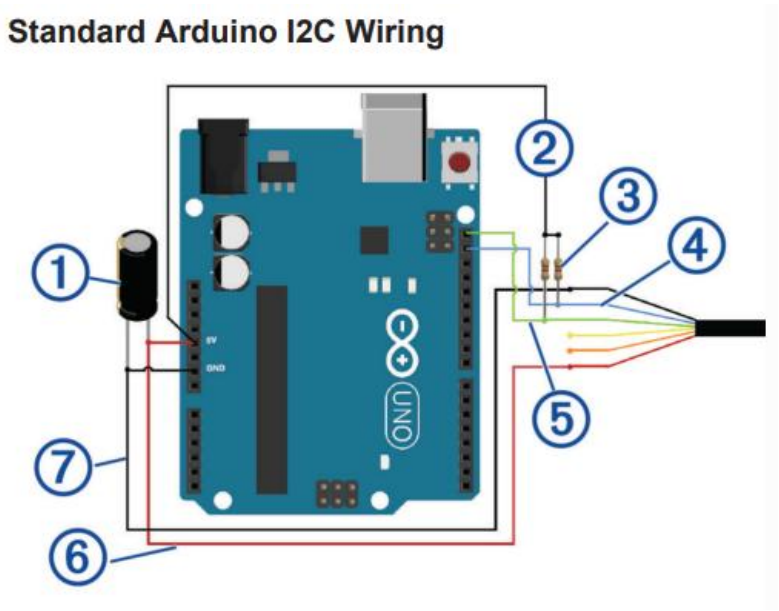


Figura 4-11 - Esquema de ligação do LiDAR Garmin v3HP usando o protocolo de comunicação I2C.

Para a ligação do *transceiver* de comunicação CAN, fez-se a alimentação do mesmo usando o terminal 1 de 3.3V, a massa da NVIDIA e utilizando os terminais 29 e 31 da linha CAN 0 disponível na NVIDIA. Este *transceiver* é posteriormente ligado ao simulador FSIPLeiria usando uma linha de comunicação CAN L e CAN H.

A câmara Intel RealSense, comunica com a NVIDIA utilizando um único cabo de ligação USB-C, que permite a comunicação com o sistema de processamento e a alimentação da mesma.

Finalmente, falta apenas as ligações dos alertas visuais e sonoros à NVIDIA que, para tal, foi necessário soldar os interruptores inteligentes numa placa de circuito impresso, com blocos de terminais de aperto, como mostra a Figura 4-12. Assim foi possível fazer a ligação dos interruptores inteligentes aos pinos 18, 22 e 24 da NVIDIA. Uma vez que o número de interruptores estaria limitado aquando da altura da montagem, as ligações do *buzzer* utilizado para alertas sonoros, foram feitas em paralelo com as dos alertas visuais. Tendo ainda em consideração que o sistema se vai implementar em veículos automóveis, foram selecionadas fitas LED de alimentação de 12V. Para efeitos de teste, foi utilizada uma fonte de alimentação externa para alimentar os LED.

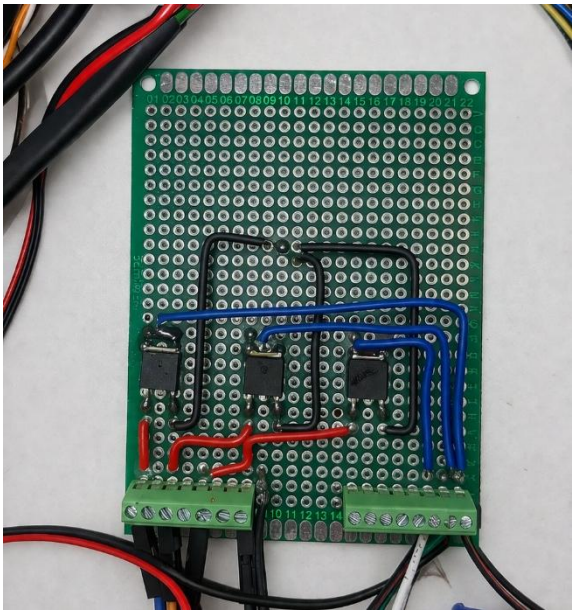


Figura 4-12 - PCB com interruptores inteligentes de controlo de alertas visuais e sonoros.

Todas estas ligações entre componentes resumiram-se na *breadboard* que podemos analisar na figura abaixo.

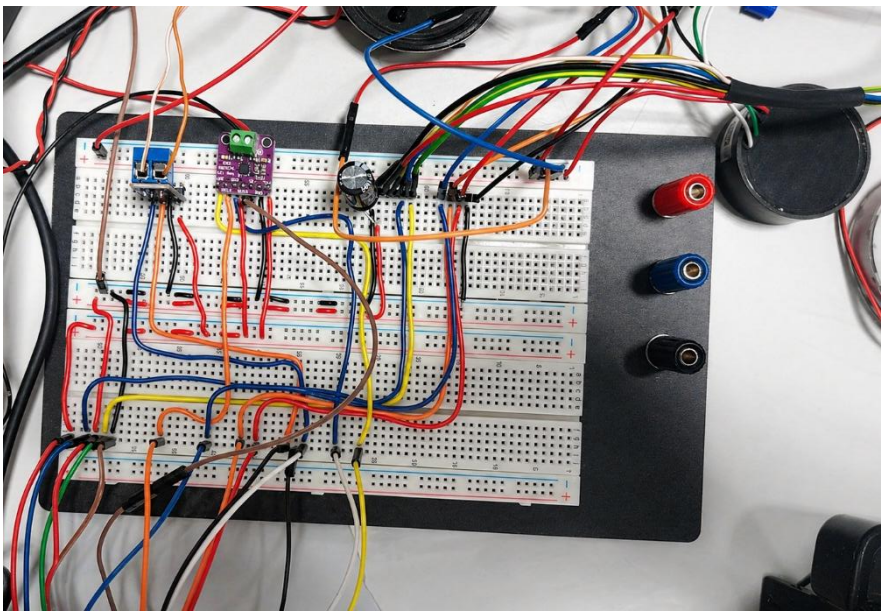


Figura 4-13 - *Breadboard* com as ligações de todos os componentes do sistema.

4.3.2. Conceção do Esquema Elétrico

A fase de conceção do esquema elétrico foi realizada utilizando o *software* de design eletrónico KiCad, permitindo a transição do protótipo de ensaio para um projeto de PCB (Placa de Circuito Impresso). A prioridade nesta etapa foi a centralização de todas

as ligações num único interface robusto e fiável. Para tal, decidiu-se pela utilização do conector 776087-5 da série AMPSEAL (Figura 4-14). Esta escolha justifica-se pela necessidade de acomodar o elevado número de entradas e saídas exigidas pelo sistema (Figura 4-15 e Figura 4-16), garantindo simultaneamente uma ligação segura e resistente, requisitos fundamentais para qualquer dispositivo destinado a uma aplicação em ambiente rodoviário.

Figura 4-14 - Conector automóvel 776087-5 AMPSEAL.

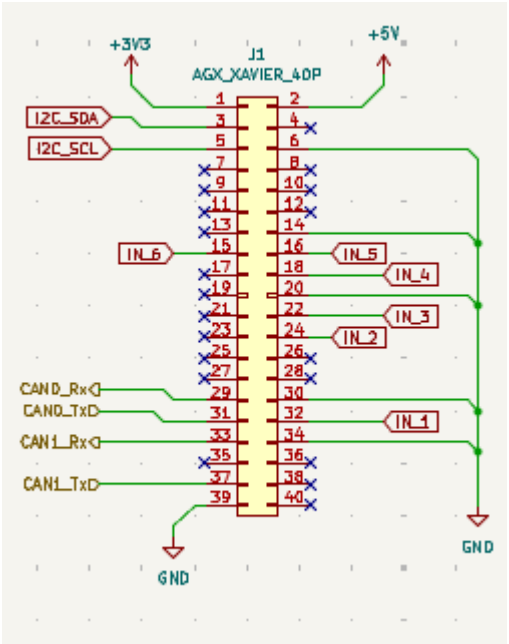


Figura 4-15 - Esquema de ligações da NVIDIA AGX Xavier à PCB.

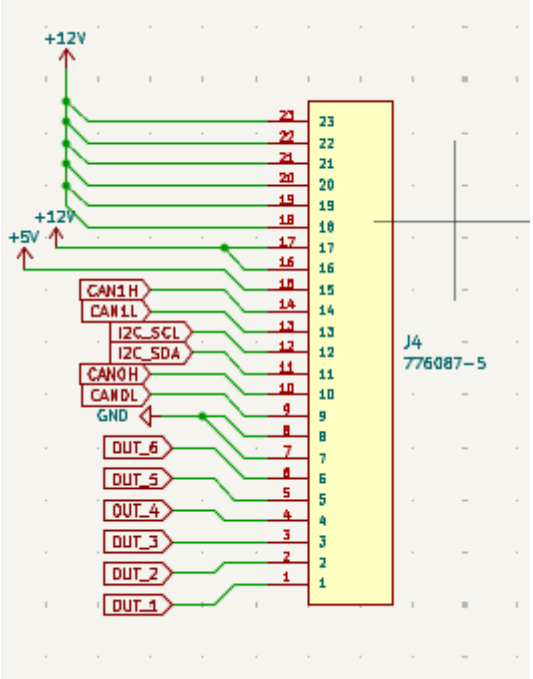


Figura 4-16 – Esquema de ligações do conector à PCB.

No desenvolvimento do esquema, foram integradas funcionalidades adicionais para aumentar a versatilidade e a segurança do *hardware*. Uma destas implementações foi a inclusão de três interruptores inteligentes adicionais aos usados no protótipo, que poderão ser utilizados para o controlo independente dos alertas sonoros produzidos pelo *buzzer,* como se pode analisar na Figura 4-17

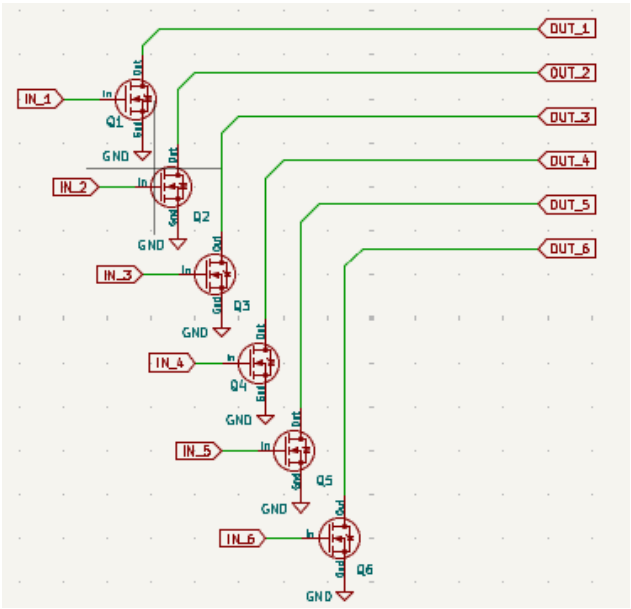


Figura 4-17 - Esquema de ligações dos interruptores inteligentes à PCB.

Além dos interruptores adicionais, o desenho da PCB contemplou ainda a expansão das capacidades de comunicação do sistema. Embora o protótipo inicial se focasse numa única interface de comunicação CAN, decidiu-se colocar à disposição a segunda linha CAN (CAN1) disponível na Xavier utilizando um *tranceiver* adicional (Figura 4-18). A disponibilização desta linha adicional permite que o sistema ADAS possa, futuramente, comunicar com outros módulos do veículo em simultâneo ou servir de ponte de diagnóstico entre diferentes redes de dados.

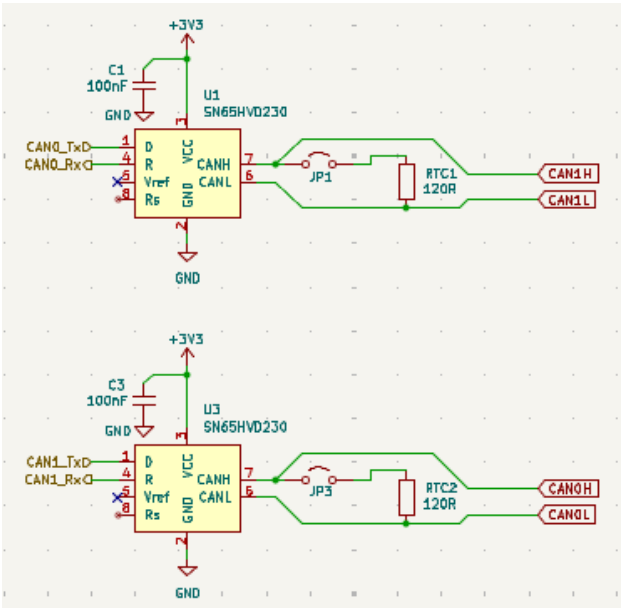


Figura 4-18 - Esquema de ligações dos *transceivers* CAN à PCB.

4.3.3. *Layout* e Dimensionamento de PCB

A fase de desenvolvimento da PCB focou-se na organização física dos componentes e no *routing* das pistas, respeitando rigorosamente as exigências elétricas de cada barramento e as normas de integridade de sinal. Para garantir a estabilidade do sistema, as espessuras das pistas foram alteradas de acordo com a sua função e a intensidade de corrente prevista.

Na linha de potência de 12V utilizou-se uma largura de 1 mm, suficiente para garantir a alimentação correta de todos os alertas visuais e sonoros. Já para as linhas de alimentação secundárias, nomeadamente para os barramentos de 5V, optou-se por pistas de 0.3 mm, enquanto as linhas de 3.3V foram dimensionadas para 0.25 mm. Para sinais e comunicação utilizou-se linhas com 0.2 mm de espessura. Na Figura 4-19 e Figura 4-20

pode-se analisar o layout de ligações da camada superior e inferior da PCB, respetivamente.

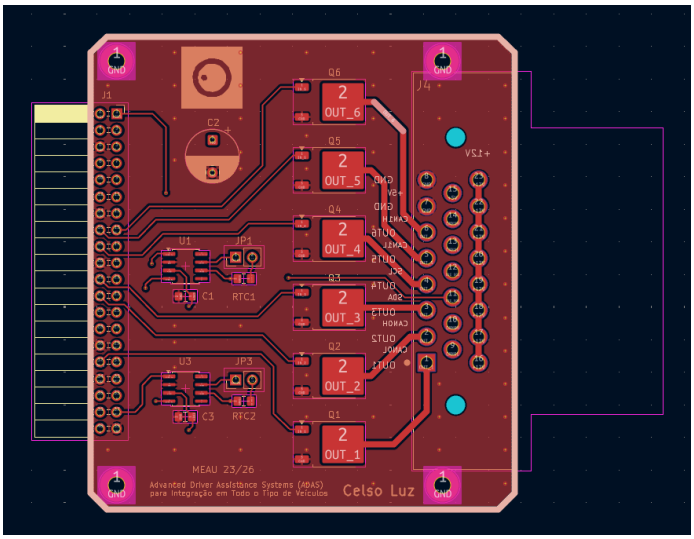


Figura 4-19 - *Layout* de ligações da camada superior da PCB.

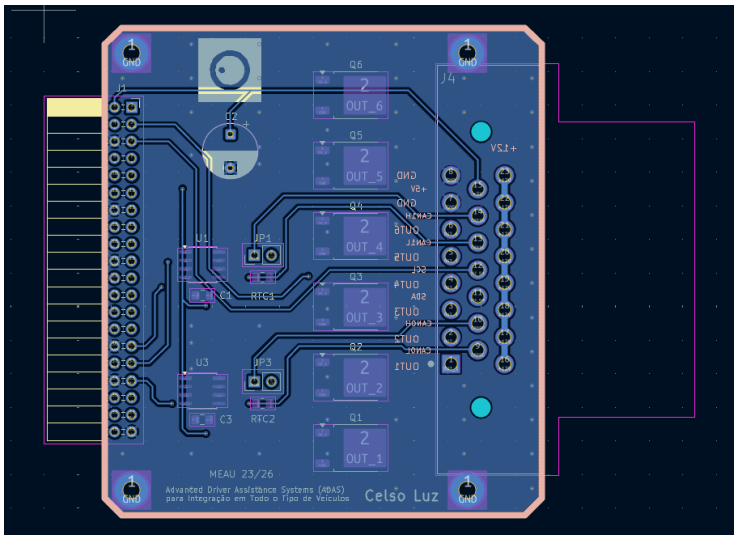


Figura 4-20 - *Layout* de ligações da camada inferior da PCB.

No que respeita à geometria das ligações, seguiu-se uma boa prática de design evitando-se o uso de ângulos de 90º. Em todo o projeto, foram utilizados exclusivamente ângulos de 45º, fator que poderia comprometer a comunicação de dados entre a Jetson e os periféricos. Além disso, a placa foi projetada com planos de massa em ambas as camadas. Esta técnica é fundamental para o bom desempenho do sistema, pois atua como uma blindagem contra ruído eletromagnético externo e fornece um caminho de retorno de baixa impedância para todas as correntes de sinal. Nas figuras seguintes é apresentado o resultado do desenvolvimento da PCB.

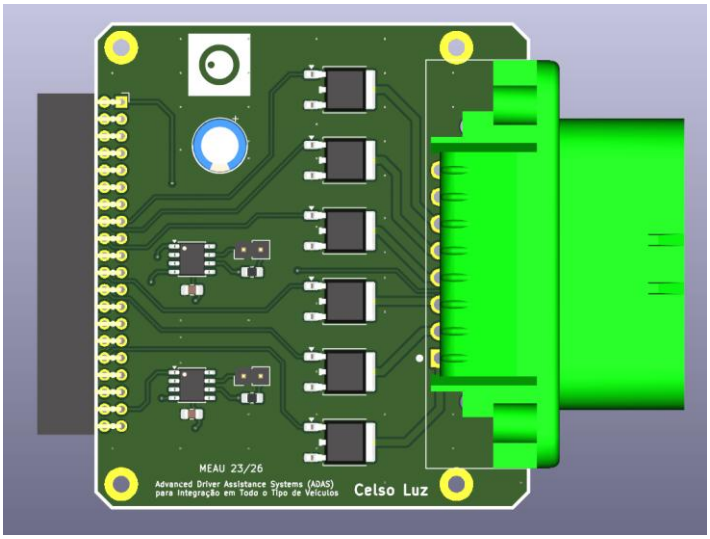


Figura 4-21 - Vista superior da PCB dimensionada.

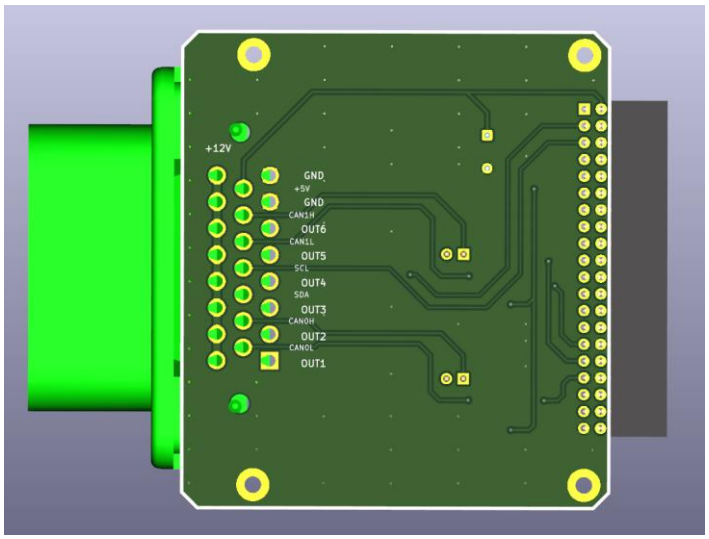


Figura 4-22 - Vista inferior da PCB dimensionada.

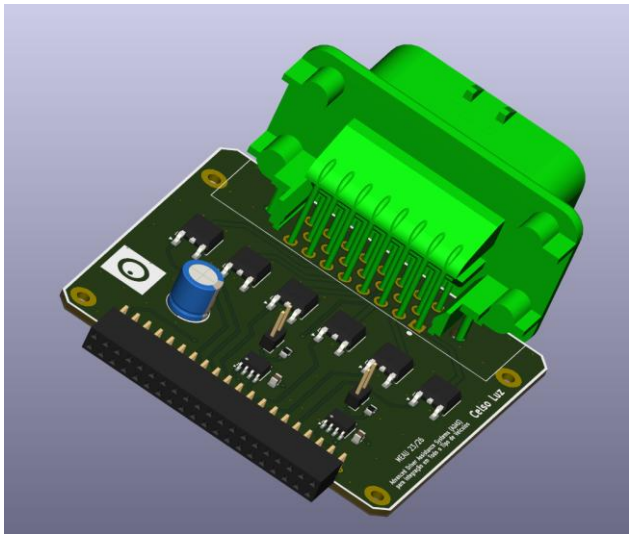


Figura 4-23 - Vista de perspetiva da PCB dimensionada.

5. Implementação e Análise de Resultados

Após a fase de desenvolvimento e prototipagem, torna-se imperativo validar a eficácia do sistema ADAS através da simulação de condições de condução em diferentes cenários. A avaliação de desempenho foi estruturada em diferentes cenários operacionais, visando testar a robustez do algoritmo e a precisão da fusão sensorial.

5.1. Testes da Taxa de Processamento

Como referido anteriormente, a eficácia de um sistema ADAS é intrinsecamente dependente da sua capacidade de resposta em tempo real. Nesta secção, avalia-se a estabilidade computacional do protótipo através da taxa de processamento em *frames per second*.

Uma vez que o sistema não estaria ainda pronto a instalar num veículo, o sistema foi testado em condições de simulação em dois cenários diferentes: usando a câmara em tempo real ou utilizando uma gravação de vídeo. No primeiro cenário, a câmara estaria a monitorizar um vídeo de circulação de um veículo em faixa de rodagem num ecrã. Contudo, as informações de distância adquiridas pela câmara e pelo LiDAR, não seriam coerentes com o cenário estabelecido, então foram ignoradas nesta fase de testes. No segundo cenário, o sistema é testado utilizando uma gravação de vídeo, onde o algoritmo processa os *frames* e os dados de profundidade relativos ao mesmo vídeo.

Assim, alcançaram-se os resultados apresentados na Tabela 5-1 onde se pode analisar que o sistema alcança uma taxa de processamento de cerca de 10 FPS, no melhor cenário, ou seja, quando o algoritmo processa dados em tempo real, apenas alertando o utilizador usando alertas visuais (LEDs) e sonoros (*buzzer*).

Tabela 5-1 - Tabela de testes de taxas de processamento do sistema final em diferentes cenários

Algoritmo	Dashboard	Entrada de Vídeo	Taxa de Processamento
Objetos + Faixas de Rodagem	Ativo	Câmara	8.50
		Gravação	9.10
	Não ativo	Câmara	10.10
		Gravação	10.64
Objetos	Ativo	Câmara	14.87
		Gravação	18.01
	Não ativo	Câmara	17.70
		Gravação	20.59
Faixas de Rodagem	Ativo	Câmara	13.77
		Gravação	14.26
	Não ativo	Câmara	14.63
		Gravação	15.66

5.2. Teste em Ambiente de Simulação

Inicialmente, a validação do sistema foi efetuada através da utilização de dados pré-adquiridos, recorrendo a sequências de vídeo e mapas de profundidade reais. Esta metodologia permitiu a correção de possíveis anomalias num ambiente controlado, garantindo a estabilidade necessária antes da transição para os testes dinâmicos em cenário real.

5.2.1. Análise de Resposta do Sistema

Nesta fase, foram simuladas diversas situações de condução através da manipulação controlada das variáveis obtidas através do barramento CAN. Ao alterar manualmente a velocidade do veículo, foi possível observar a variação dinâmica do TTC e a consequente ativação dos diferentes níveis de alerta (visual e sonoro).

O primeiro cenário testado, foi com o veículo em circulação numa estrada com uma velocidade de cerca de 90km/h onde o LiDAR deteta um objeto a 1.82 metros de distância, mas onde a câmara, não deteta nenhum objeto a essa distância. Neste cenário extremo, o sistema deteta que o condutor não reage, uma vez que o travão não está ativo e a posição do acelerador é superior a 0. Assim, perante esta situação, o sistema ativa o sistema de alerta visual com a mensagem que podemos ver na Figura 5-1, ativação dos LED vermelhos e ativação de um som intermitente pelo *buzzer*.

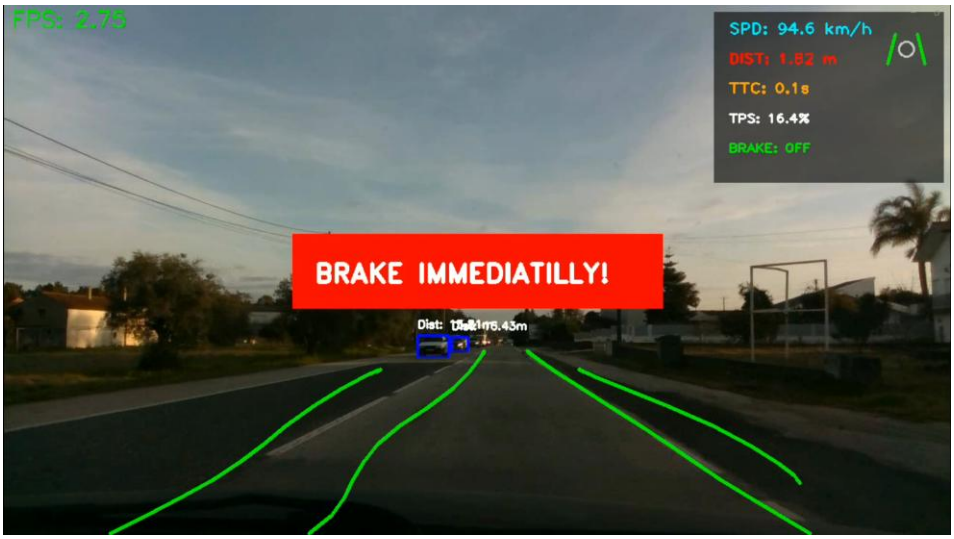


Figura 5-1 - *Dashboard* de resultados perante um cenário extremo em simulação.

Adicionalmente, testou-se a integração do estado dos pedais, nomeadamente a ativação do travão, para verificar a capacidade do sistema em suprimir alertas redundantes quando o condutor já demonstra uma reação defensiva. Esta análise permitiu aferir a sensibilidade do sistema perante cenários críticos de aproximação.

No cenário seguinte (Figura 5-2), o veículo encontra-se a circular a uma velocidade mais moderada de cerca de 40 km/h. Apesar disso, o sistema deteta um objeto a uma distância reduzida de cerca de 8 metros, o que, consequentemente, traduz-se num TTC de 0.7 s (crítico). Apesar do aparente cenário de risco iminente, uma vez que o sistema deteta que o condutor está a reagir, ou seja, o travão está a ser ativado, o sistema considera que a situação está sob o controlo do condutor da viatura.

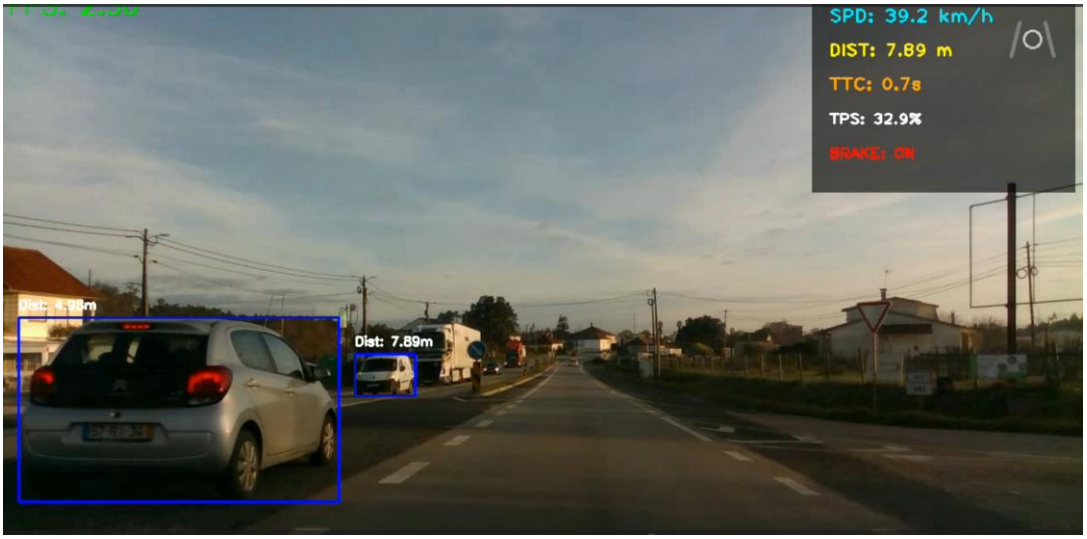


Figura 5-2 - *Dashboard* de resultados perante um cenário controlado pelo condutor em simulação.

Nesta situação (Figura 5-3), o veículo encontra-se a circular a uma velocidade relativamente reduzida, e o LiDAR deteta um objeto a cerca de 11 metros do veículo, o TTC calculado é de cerca de 1.7 segundos. Este cenário é considerado como uma situação de aviso, então o sistema apresenta uma mensagem de alerta para aumentar a distância de segurança e é ligado apenas um dos alertas visuais vermelhos em conjunto com um alerta sonoro moderado.

Figura 5-3 - *Dashboard* de resultados perante um cenário de alerta em simulação.

5.2.2. Discussão de Resultados

A análise dos resultados obtidos nos ensaios de simulação permite validar a eficácia da lógica de decisão do sistema, demonstrando a sua capacidade de processar e reagir a diferentes cenários de risco. Através da manipulação controlada das variáveis do

barramento CAN, verificou-se que o algoritmo consegue interpretar corretamente a relação entre a velocidade e a distância, atribuindo os alertas de acordo com a gravidade da situação. No cenário de alta velocidade (90 km/h) com um objeto detetado pelo LiDAR a curta distância, o sistema evidenciou a importância da fusão sensorial. O facto de o LiDAR ter identificado um obstáculo que a câmara ignorou comprova que a redundância de sensores é vital para a segurança ativa, garantindo que o alerta crítico seja acionado mesmo perante falhas de deteção visual.

Por outro lado, a inclusão do estado dos pedais na lógica de decisão revelou-se um dos pontos mais robustos do projeto. Como observado no cenário de 40 km/h com um TTC crítico de 0.7 segundos, a capacidade do sistema em reconhecer a ativação do travão e suprimir alertas redundantes é fundamental para garantir que o ADAS intervenha apenas quando não há uma reação humana defensiva. Da mesma forma, o cenário de aviso com um TTC de 1.7 segundos confirmou a capacidade do sistema em fornecer alertas graduais, incentivando uma condução preventiva antes que a situação evolua para um risco de colisão.

Contudo, é fundamental ressalvar que estes resultados foram obtidos num ambiente de simulação com dados pré-adquiridos e manipulação estática de variáveis. Embora a lógica de decisão tenha respondido conforme esperado, o desempenho observado não é totalmente fidedigno em relação a uma operação real de estrada.

5.3. Teste em Ambiente Real

A validação final do sistema ADAS foi realizada através de testes dinâmicos num cenário de condução real, utilizando um veículo de passageiros Volvo V70 (modelo de 2007) como plataforma de testes. Esta etapa permitiu avaliar o comportamento dos algoritmos de fusão sensorial, a robustez da deteção de objetos em tempo real e a fiabilidade da aquisição de dados do veículo sob condições adversas de vibração e luminosidade variáveis.

5.3.1. Configuração Inicial e Sistema de Alimentação

Numa primeira fase, procedeu-se à instalação de todos os componentes no habitáculo do veículo. Tanto a câmara como o LiDAR foram fixados no interior do veículo, recorrendo a suportes de ventosa aplicados no vidro para-brisas. Esta abordagem

visava proteger o *hardware* das condições climatéricas e simplificar a instalação do sistema.

Figura 5-4 - Câmara e LiDAR instalados no interior do veículo.

Todos os componentes eletrónicos, incluindo a NVIDIA, o ecrã de monitorização e todos os sensores, foram alimentados a partir da porta OBD do veículo. Para além disso, a porta OBD foi utilizada para a ligação ao barramento CAN de dados do veículo, derivando as linhas CAN-H e CAN-L para o *transceiver*.

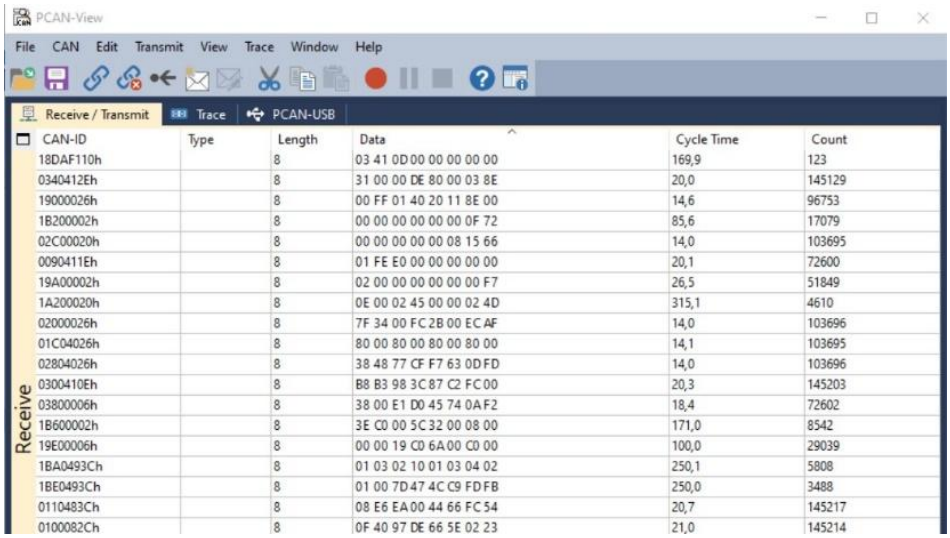
Os avisos visuais e sonoros foram aplicados no tablier do veículo para alertar o condutor relativamente aos perigos durante a condução, tal como se pode visualizar na Figura 5-5.

Figura 5-5 - Alertas visuais aplicados no veículo de testes.

5.3.2. Descodificação do Barramento CAN

Antes de iniciar os testes, foi necessário realizar uma fase de descodificação da rede CAN do veículo (Figura 5-6), uma vez que as variáveis necessárias para o algoritmo não são disponibilizadas de forma direta pelos ID usados anteriormente na fase de desenvolvimento.

Primeiramente, utilizou-se o *candump* na Jetson para perceber se existiria comunicação com o veículo monitorizando o tráfego do barramento. Através de testes estáticos (pressionar o pedal de travão) e utilizando os PID gerais, identificaram-se os ID correspondentes a: velocidade do veículo aplicando a respetiva resolução matemática para obter o valor em km/h com precisão e o estado do interruptor do travão obtendo o bit específico de estado que transita de 0 para 1 no exato milissegundo em que o condutor pressiona o pedal, dado crítico para anular falsos alarmes de colisão frontal.



CAN-ID	Type	Length	Data	Cycle Time	Count
18DAF110h		8	03 41 0D 00 00 00 00 00	169,9	123
0340412Eh		8	31 00 00 DE 80 00 03 8E	20,0	145129
19000026h		8	00 FF 01 40 20 11 8E 00	14,6	96753
1B200002h		8	00 00 00 00 00 00 0F 72	85,6	17079
02C00020h		8	00 00 00 00 00 08 15 66	14,0	103695
0090411Eh		8	01 FE E0 00 00 00 00 00	20,1	72600
19A00002h		8	02 00 00 00 00 00 00 F7	26,5	51849
1A200020h		8	0E 00 02 45 00 00 02 4D	315,1	4610
02000026h		8	7F 34 00 FC 2B 00 EC AF	14,0	103696
01C04026h		8	80 00 80 00 80 00 80 00	14,1	103695
02804026h		8	38 48 77 CF F7 63 0D FD	14,0	103696
0300410Eh		8	B8 B3 98 3C 87 C2 FC 00	20,3	145203
03800006h		8	38 00 E1 D0 45 74 0A F2	18,4	72602
1B600002h		8	3E C0 00 5C 32 00 08 00	171,0	8542
19E00006h		8	00 00 19 C0 6A 00 C0 00	100,0	29039
1BA0493Ch		8	01 03 02 10 01 03 04 02	250,1	5808
1BE0493Ch		8	01 00 7D 47 4C C9 FD FB	250,0	3488
0110483Ch		8	08 E6 EA 00 44 66 FC 54	20,7	145217
0100082Ch		8	0F 40 97 DE 66 5E 02 23	21,0	145214

Figura 5-6 - Mensagem CAN obtida através porta OBD.

5.3.3. Testes de Estrada

Durante os primeiros ensaios de estrada com a configuração inicial, detetou-se uma falha crítica no sistema de leitura de distância: o sensor LiDAR apresentava leituras de proximidade constantes e erráticas, impossibilitando o cálculo do TTC corretamente e assumindo constantemente situações de perigo elevado.

Após uma análise de dados, verificou-se que o feixe de laser infravermelho do LiDAR sofria refração e reflexão ao atravessar o vidro laminado do para-brisas do

veículo. O sensor interpretava o próprio vidro como um obstáculo imediato a poucos centímetros de distância.

Para contornar esta limitação física, foi necessário alterar a disposição da montagem do protótipo, onde a câmara foi mantida no interior do habitáculo e o LiDAR foi relocalizado para o exterior do veículo, sendo fixado na grelha frontal (para-choques). Esta alteração exigiu a passagem de uma cablagem mais longa desde o compartimento do motor até ao habitáculo aplicando resistências em paralelo no sinal I2C, garantindo que o sinal não sofria interferências provocadas pelo comprimento de cablagem.

Figura 5-7 - Dispositivo LiDAR fixado na frente do veículo de teste.

Com o LiDAR posicionado na grelha frontal, o feixe laser passou a ter uma linha de vista desimpedida para a estrada, eliminando as falsas leituras e permitindo avançar com sucesso para os testes de estrada definitivos.

No segundo teste de estrada, foi possível concluir que o sistema é bastante eficaz em diversas situações. Na situação seguinte, é possível perceber que um veículo à frente do veículo de testes, estava a travar para fazer mudança de direção, em que o condutor não estava ainda consciente do perigo iminente uma vez que que o pedal de travão não estava a ser pressionado. O sistema calculou um TTC de 0.8 s com base na velocidade do veículo e a distância do veículo à frente, dados que se podem consultar no canto superior direito (Figura 5-8) onde mostra a distância calculada pela câmara e pelo LiDAR respetivamente.

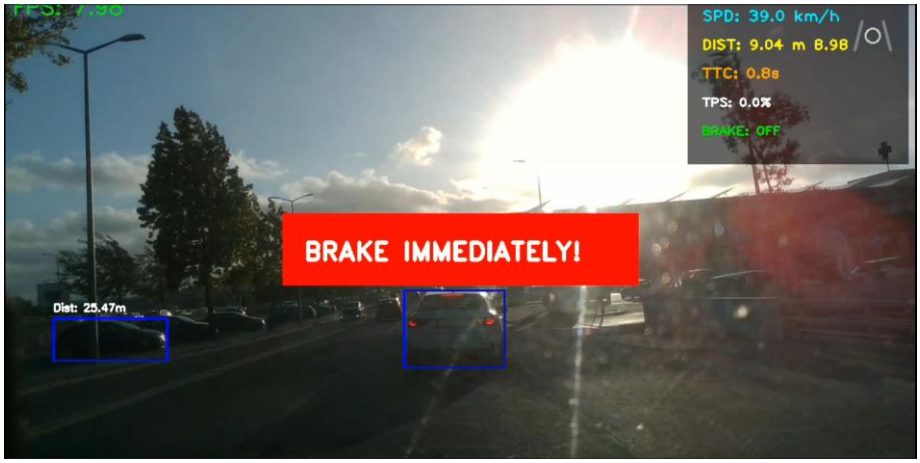


Figura 5-8 - Situação de perigo iminente detetada pelo sistema

Na Figura 5-9 ainda se consegue observar que todos os alertas visuais se encontram ativos, isto porque em situações de perigo ambos os LED vermelhos ativam para maior atenção do condutor.

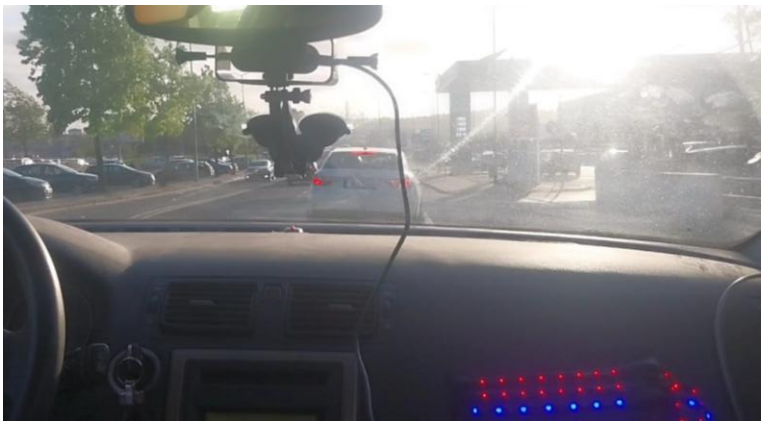


Figura 5-9 - Situação de perigo iminente alertada pelos avisos visuais

Nas figuras abaixo, na mesma situação de perigo pode-se analisar que o sistema percebe que o condutor já está a reagir à situação, pressionando o pedal de travão. Para além disso, pode-se analisar que apenas um dos LED vermelho agora está ativo, isto porque ainda é uma situação de perigo, mas agora a ser reagida pelo condutor.

Figura 5-10 - Situação de perigo iminente detetada pelo sistema com o condutor a reagir.

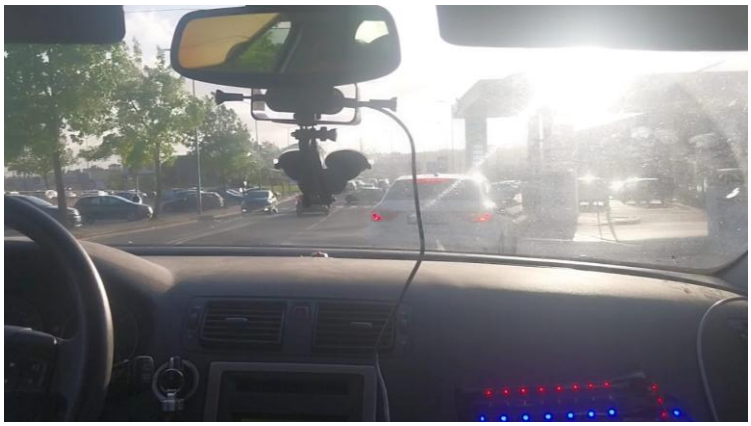


Figura 5-11 - Situação de perigo iminente alertada pelos avisos visuais (condutor a reagir)

Noutra situação durante os testes, o veículo circulava num parque de estacionamento com a identificação de diversos objetos, alguns relativamente perto do veículo, mas nenhum foi considerado numa emergência uma vez que não se encontravam na faixa de rodagem do veículo.

Figura 5-12 - Deteção de múltiplos objetos não enquadrados numa situação de perigo.

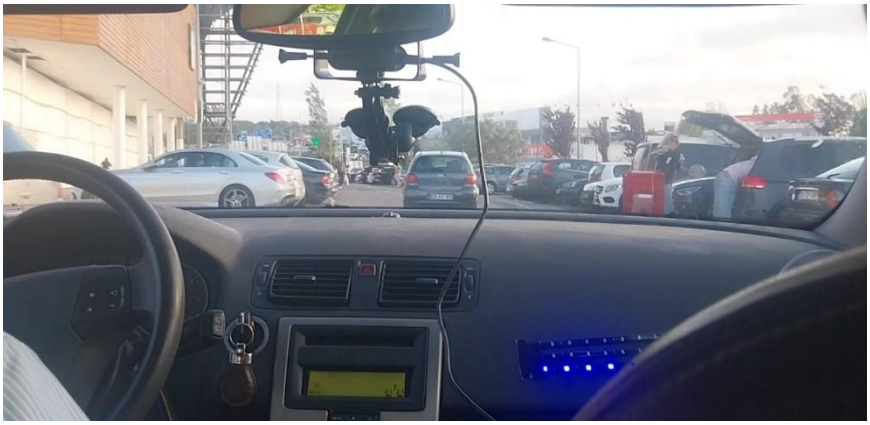


Figura 5-13 - Alertas visuais não ativos apesar de deteção de múltiplos objetos

Outra situação crítica para o bom funcionamento do sistema, é a deteção de peões, situação essa que foi possível comprovar a sua eficácia. Nas figuras abaixo, é possível analisar que o sistema detetou dois peões numa situação de perigo iminente. Na Figura 5-14, pode-se perceber que a câmara detetou os peões a cerca de 20 metros, mas o LiDAR a uma distância igual ou inferior a 4 metros. Isto indica que o LiDAR tem falhas no seu funcionamento derivadas às vibrações do veículo, necessitando assim de um algoritmo de filtragem dos valores obtidos. Já na Figura 5-15, o LiDAR já deteta os peões à semelhança da câmara.

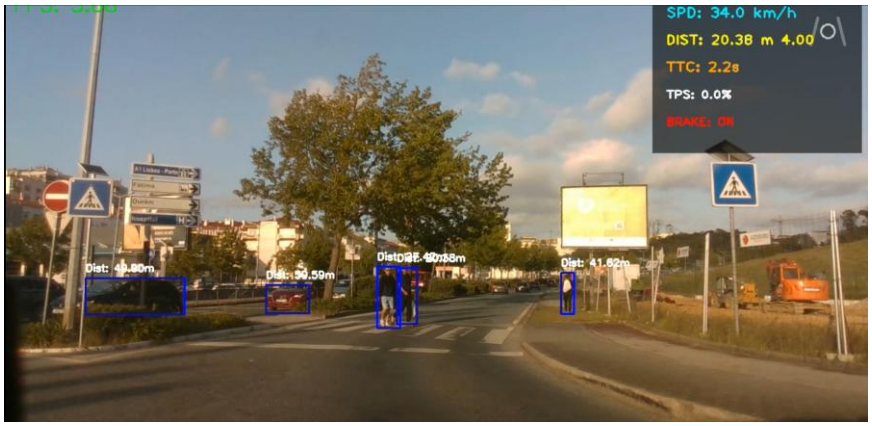


Figura 5-14 - Situação de perigo iminente envolvendo peões detetada pelo sistema.

Figura 5-15 - Situação de perigo iminente envolvendo peões detetada pelo sistema com alerta visual.

Embora todas estas situações de sucesso, também existem situações de falsos positivos, onde o sistema deteta situações de perigo iminente, nomeadamente situações onde o veículo à frente está bastante próximo do veículo de testes durante algum tempo e o sistema considera sempre uma emergência, como se pode observar na figura seguinte.

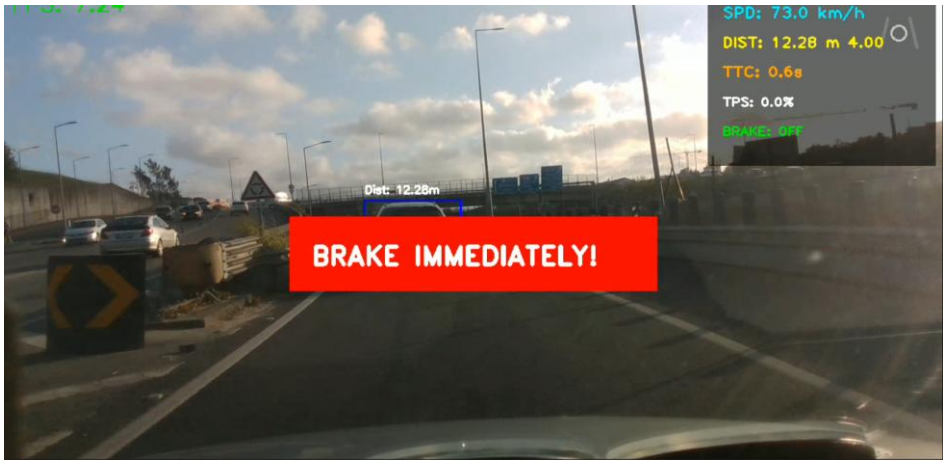


Figura 5-16 - Situação de falso positivo do algoritmo.

O algoritmo também apresenta algumas falhas em ambiente noturno. Na seguinte imagem podemos observar uma situação onde o algoritmo não deteta um veículo na mesma faixa de rodagem por encandeamento de um veículo na direção oposta. Resultando numa situação de perigo detetada apenas quando se está bastante próximo do veículo. Nesta situação o LiDAR deveria detetar o objeto a embater, no entanto existem várias variáveis nesta situação que não o permitiram, nomeadamente: o facto de ser o LiDAR de um único ponto, a estrada ser de inclinação gradual e o posicionamento do veículo na faixa de rodagem.

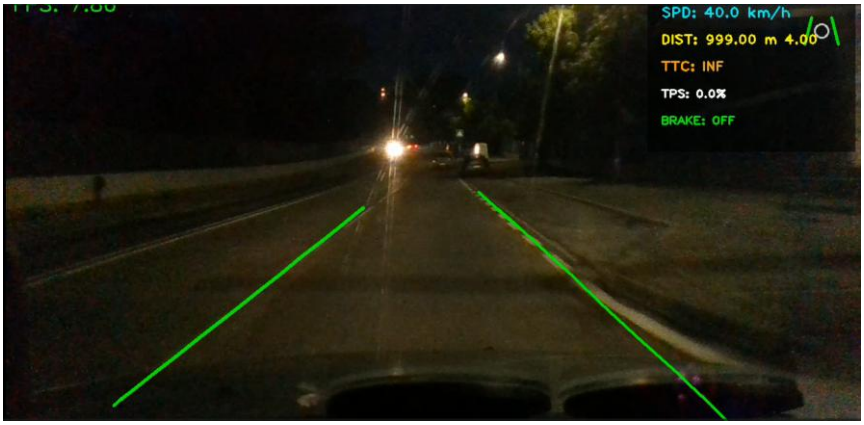


Figura 5-17 - Situação de não deteção de perigo iminente.

Como se pode ainda analisar na Figura 5-18, o sistema detetou o veículo a apenas alguns metros de embate, neste caso não fornecendo tempo suficiente ao condutor para reação.

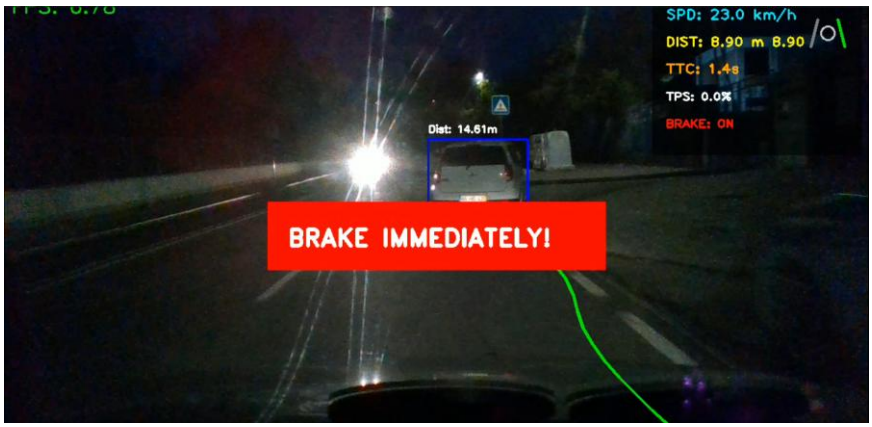


Figura 5-18 - Situação de perigo iminente em ambiente noturno.

6. Conclusões e Trabalho Futuro

6.1. Conclusões

O desenvolvimento de sistemas avançados de assistência à condução (ADAS) representa atualmente um dos maiores desafios e oportunidades na engenharia automóvel, permitindo elevar os padrões de segurança ativa através da integração de sensores e inteligência artificial. O objetivo central deste projeto foi a conceção, implementação e validação de um sistema capaz de monitorizar a trajetória do veículo e prevenir colisões em tempo real, adaptável em quaisquer veículos.

Para alcançar este propósito, o projeto integrou com sucesso redes neuronais de *Deep Learning*, nomeadamente o modelo YOLOv8 para a deteção de obstáculos e o UFLDv2 para a deteção de faixas de rodagem. A arquitetura de *software* desenvolvida permitiu a fusão sensorial de dados provenientes de uma câmara e de um sensor LiDAR, cruzando-os com a informação obtida do veículo via barramento CAN. No âmbito do *hardware*, a transição da prototipagem em *breadboard* para uma PCB dedicada com conectores de padrão automóvel permitiu tornar o sistema mais compacto e prático a nível de integração e instalação no veículo.

Pela análise dos resultados e testes efetuados, concluiu-se que a utilização de um sistema dedicado como a Jetson Xavier é fundamental para manter taxas de processamento estáveis e rápidas enquanto se executam múltiplas redes neuronais em simultâneo. A lógica de decisão baseada no TTC revelou-se eficaz, sendo que o limiar crítico de 1.5 segundos permitiu distinguir com precisão situações de risco iminente de manobras de aproximação normal. Verificou-se ainda que a integração dos dados obtidos através da linha CAN, é crucial para evitar alertas redundantes, permitindo ao sistema adaptar a sua resposta visual e sonora à reação real do condutor.

No entanto, o sistema apresentou algumas limitações em cenários de curvas de raio muito reduzido e em condições de oclusão severa, onde a deteção de faixas de rodagem feita pelo modelo UFLDv2 perde alguma precisão geométrica. De salientar também que, embora a fusão sensorial aumente a robustez do sistema, a calibração entre as diferentes coordenadas espaciais dos sensores exige um rigor extremo para evitar discrepâncias nas medições de distância.

Os testes de estrada foram igualmente vitais para validar a importância da utilização do barramento CAN. A integração do estado do interruptor do pedal de travão revelou-se um dado crítico, pois o algoritmo ADAS passou a discriminar se o condutor já se encontrava a reagir ativamente ao perigo. Esta validação permitiu ao sistema comutar dinamicamente a intensidade dos alertas, reduzindo a ativação para apenas um LED vermelho, o que eliminou avisos redundantes e mitigou a fadiga de alarmes ao condutor. Adicionalmente, o sistema demonstrou excelente assertividade em zonas de parque de estacionamento, filtrando múltiplos objetos fora da trajetória estrutural da faixa de rodagem, e exibiu eficácia na deteção de pedestres em cenários diurnos.

Contudo, o ambiente real expôs limitações cruciais que enriquecem o desenvolvimento deste projeto. A montagem inicial do LiDAR no interior do habitáculo revelou-se inviável devido à reflexão do feixe laser infravermelho ao atravessar o vidro laminado, exigindo a relocalização do sensor para a grelha frontal do para-choques. Verificaram-se também discrepâncias pontuais nas leituras de distância do LiDAR em relação à câmara durante a aproximação a objetos, causadas pelas vibrações do veículo que desalinhavam temporariamente o feixe de ponto único. Por fim, o algoritmo manifestou tendência para manter alertas de emergência contínuos em situações de trânsito intenso onde o veículo da frente circulava muito próximo durante períodos prolongados. O cenário noturno revelou-se o mais desafiante, registando-se uma falha crítica de não deteção de um objeto devido ao encandeamento da câmara pelos faróis de veículos em sentido oposto. Nestas condições, o LiDAR falhou na deteção precoce do obstáculo devido à inclinação gradual da via e ao posicionamento do veículo na faixa, reduzindo a distância de deteção para escassos metros antes do possível impacto.

6.2. Trabalhos Futuros

Como seguimento e otimização deste trabalho, sugerem-se vários trabalhos de desenvolvimento futuro com o intuito de colmatar as limitações identificadas. Primeiramente, propõe-se a implementação de algoritmos de filtragem nos dados do LiDAR de modo a suavizar e eliminar os ruídos e leituras erráticas provocadas pelas vibrações do veículo. Adicionalmente, revela-se pertinente a expansão da aquisição de dados do barramento CAN para abranger os sinais de mudança direção do veículo, o que permitiria ao sistema interpretar os piscas e silenciar os alertas de transição de faixa quando a manobra é intencional, em ambiente de auto-estrada.

Relativamente ao LiDAR, sugere-se a substituição do LiDAR de ponto único por um sensor LiDAR de duas dimensões. Esta alteração permite a cobertura completa do plano da estrada, eliminando as falhas de leitura pelas condições adversas.

Por fim, recomenda-se o desenvolvimento de um algoritmo para a gestão de falsos positivos, para que o sistema consiga interpretar situações de trânsito intenso ou cenários de "pára-arranca", evitando a saturação e o acionamento desnecessário de alertas visuais e sonoros quando a velocidade relativa entre os veículos é nula ou residual.

Referências Bibliográficas

[1] Autoridade Nacional Segurança Rodoviária, "RELATÓRIO MAIO 2024," 8 agosto 2024. [Online]. Available: http://www.ansr.pt/Documents/RelatorioSinistralidade24heFiscalizacaoRodoviariaMaio2024.pdf. [Acedido em janeiro 2025].

[2] World Health Organization, "Road traffic injuries," 13 dezembro 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries. [Acedido em Janeiro 2025].

[3] L. Masello, B. Sheehan, G. Castignami, D. Shannon e F. Murphy, "On the impact of advanced driver assistance systems on driving distraction and risky behaviour: An empirical analysis of irish commercial drivers," p. 18, 12 janeiro 2023.

[4] European Comision, "NEW RULES ON VEHICLE SAFETY," 5 julho 2024. [Online]. Available: https://single-market-economy.ec.europa.eu/document/download/cd243af9-c877-401e-9f69-d7d4ab6a90c6_en?filename=Fact%20Sheet%20General%20Safety%20regulations_July%202024.pdf. [Acedido em Janeiro 2025].

[5] EURO NCAP, "Assisted Driving 2023," outubro 2023. [Online]. Available: https://www.euroncap.com/assessments/mercedes-benz/e-class/a016/. [Acedido em junho 2025].

[6] EURO NCAP, "How To Read The Stars," 2023. [Online]. Available: https://www.euroncap.com/en/about-euro-ncap/how-to-read-the-stars/. [Acedido em 05 outubro 2024].

[7] EURO NCAP, "Assisted Driving Highway & Interurban Assist Systems Test & Assessment Protocol V2.0," 2024.

[8] spiceworks, "What Is ADAS (Advanced Driver Assistance Systems)? Meaning, Working, Types, Importance, and Applications," 15 Julho 2022. [Online]. Available: https://www.spiceworks.com/tech/iot/articles/what-is-adas/. [Acedido em Março 2025].

[9] SAE International, "SAE Blog," 3 Maio 2021. [Online]. Available: https://www.sae.org/blog/sae-j3016-update. [Acedido em Fevereiro 2025].

[10] Audi MediaCenter, "Driver assistance systems," Audi MediaCenter, 13 fevereiro 2017. [Online]. Available: https://www.audi-mediacenter.com/en/audi-

technology-lexicon-7180/driver-assistance-systems-7184. [Acedido em 10 outubro 2024].

[11] Mercedes-Benz, "Mercedes-Benz Intelligent Drive," Mercedes-Benz, 2024. [Online]. Available: https://www.mercedes-benz.pt/passengercars/technology/advanced-driver-assistance-systems.html. [Acedido em 10 outubro 2024].

[12] European Parliament, "Self-driving cars in the EU: from science fiction to reality," 14 Janeiro 2019. [Online]. Available: https://www.europarl.europa.eu/topics/en/article/20190110STO23102/self-driving-cars-in-the-eu-from-science-fiction-to-reality. [Acedido em Fevereiro 2025].

[13] A. E. Sharkawy, M. Asfoor e M. Yacoub, "Comprehensive evaluation of emerging technologies of advanced driver assistance systems: An overview," setembro 2021.

[14] Y. H. X. H. H. W. B. Z. Zhiyang Guo, "A Survey on Deep Learning Based Approaches for Scene," 15 fevereiro 2021. [Online]. Available: https://www.researchgate.net/publication/349322006_A_Survey_on_Deep_Learning_Based_Approaches_for_Scene_Understanding_in_Autonomous_Driving. [Acedido em Março 2025].

[15] datacamp, "An Introduction to Convolutional Neural Networks (CNNs)," datacamp, 14 novembro 2023. [Online]. Available: https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns. [Acedido em 15 novembro 2024].

[16] A. K. Shetty, I. Saha, R. M. Sanghvi, S. A. Save e Y. J. Patel, "A Review: Object Detection Models," 2 abril 2021.

[17] D. K. Gurveen Kaur, "Lane Detection Techniques: A Review," 10 fevereiro 2015.

[18] J. Fu, "Lane Departure Warning System for Autonomous Driving," 19 Abril 2017. [Online]. Available: https://junshengfu.github.io/driving-lane-departure-warning/. [Acedido em dezembro 2024].

[19] M. Kolek, "Fine-Tuning YOLO for Road Sign and Object Detection on Polish roads," 7 agosto 2024. [Online]. Available: https://medium.com/@mikolaj.kolek/fine-tuning-yolo-for-road-sign-and-object-detection-on-polish-roads-71a366e9a876. [Acedido em Fevereiro 2025].

[20] C. Han, Q. Zhao, S. Zhang, Y. Chen, Z. Zhang e J. Yuan, "YOLOPv2:rocket:: Better, Faster, Stronger for Panoptic driving Perception," 30 agosto 2022. [Online].

Available: https://github.com/CAIC-AD/YOLOPv2/blob/main/README.md. [Acedido em abril 2025].

[21] mobileye, mobileye, 2024. [Online]. Available: https://ims.mobileye.com/fleets/au/. [Acedido em 8 outubro 2024].

[22] V. Anh, "Advanced driver-assistance system on Jetson Nano," 10 setembro 2020. [Online]. Available: https://www.vietanh.dev/blog/2020-09-10-adas-jetson-nano-intro-and-hardware. [Acedido em 8 outubro 2024].

[23] V. V. H. S. K. K. Ramachandra A.C., "INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY," 18 outubro 2022. [Online]. Available: https://indjst.org/articles/advanced-driving-assistance-system-for-cars-using-raspberry-pi. [Acedido em Julho 2025].

[24] Rapsberry Pi Ltd, "Rapsberry Pi AI Camera," setembro 2024.

[25] logitech, "C920 HD Pro Webcam," 2024. [Online]. Available: https://www.logitech.com/en-eu/shop/p/c920-pro-hd-webcam.960-001055#key-features. [Acedido em 18 novembro 2024].

[26] Intel RealSense, "Intel RealSense Depth Camera D435i," 2023. [Online]. Available: https://www.intelrealsense.com/depth-camera-d435i/. [Acedido em 04 novembro 2024].

[27] Tejal, "LiDAR : Light Detection And Ranging Sensor — Chapter 1," 15 março 2021. [Online]. Available: https://medium.com/analytics-vidhya/lidar-light-detection-and-ranging-sensor-chapter-1-68cad17c8847. [Acedido em 29 novembro 2024].

[28] GARMIN, "LIDAR-Lite v3HP," GARMIN, abril 2020. [Online]. Available: https://www.garmin.com/en-US/p/578152. [Acedido em 4 novembro 2024].

[29] Slamtec Co., Ltd, "RPLIDAR A2," 2024. [Online]. Available: https://www.slamtec.com/en/Lidar/A2. [Acedido em 29 novembro 2024].

[30] LIVOX, "MID-40/MID-100 SPECS," 2024. [Online]. Available: https://www.livoxtech.com/mid-40-and-mid-100/specs. [Acedido em 29 novembro 2024].

[31] MINISFORUM, "Minisforum EM780," 2024. [Online]. Available: https://store.minisforum.com/products/minisforum-em780. [Acedido em 18 novembro 2024].

[32] A. S. L. Machado e B. R. R. Oliveira, "O Sistema OBD (On-Board Diagnosis)," [Online]. Available:

https://www.ave.dee.isep.ipp.pt/~mjf/act_lect/SIAUT/Trabalhos%202007-08/Trabalhos/SIAUT_OBD.pdf. [Acedido em Maio 2025].

[33] S. Sharma, "Understanding CAN Bus: A Comprehensive Guide," 7 novembro 2023. [Online]. Available: https://www.wevolver.com/article/understanding-can-bus-a-comprehensive-guide. [Acedido em 3 dezembro 2024].

[34] K. Katsanos, "Advanced driver assistance system with accident recognition and OpenCV," *Advanced driver assistance system with accident recognition and OpenCV,* 16 fevereiro 2023.

[35] Microchip Technology Inc., "MCP2515," 2024. [Online]. Available: https://www.microchip.com/en-us/product/MCP2515#product-purchase. [Acedido em 02 dezembro 2024].

[36] Techno Vision, "ASUS TUF Gaming A15 (2023) FA507NV," 2023. [Online]. Available: https://vision-techno.com/en/product/380/ASUS+TUF+Gaming+A15+%282023%29+FA507NV/. [Acedido em 9 fevereiro 2026].

[37] LaptopMedia, "ASUS VivoBook 15 (X512JP-WB711)," [Online]. Available: https://laptopmedia.com/laptop-specs/asus-vivobook-15-x512jp-wb711/gaming-performance/. [Acedido em 9 fevereiro 2026].

[38] NVIDIA DEVELOPER, "Jetson Nano," 2024. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano. [Acedido em 18 novembro 2024].

[39] LeadTek, "NVIDIA Jetson Xavier Developer Kit," [Online]. Available: https://www.leadtek.com/eng/products/ai_hpc(37)/nvidia_jetson_xavier_developer_kit(10807)/detail. [Acedido em 9 fevereiro 2026].

[40] Rapsberry Pi Ltd, "Rapsberry Pi 5," agosto 2024.

[41] Raspberry Pi, "Raspberry Pi computer hardware," [Online]. Available: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html. [Acedido em 9 fevereiro 2026].

[42] EETimes, "ADAS Front Camera: Demystifying Resolution and Frame-Rate," 3 julho 2016. [Online]. Available: https://www.eetimes.com/adas-front-camera-demystifying-resolution-and-frame-rate/. [Acedido em Junho 2025].

[43] Qin, Z. a. Wang, H. a. Li e Xi, "Ultra-Fast-Lane-Detection-V2," 2022. [Online]. Available: https://github.com/cfzd/Ultra-Fast-Lane-Detection-v2. [Acedido em maio 2025].

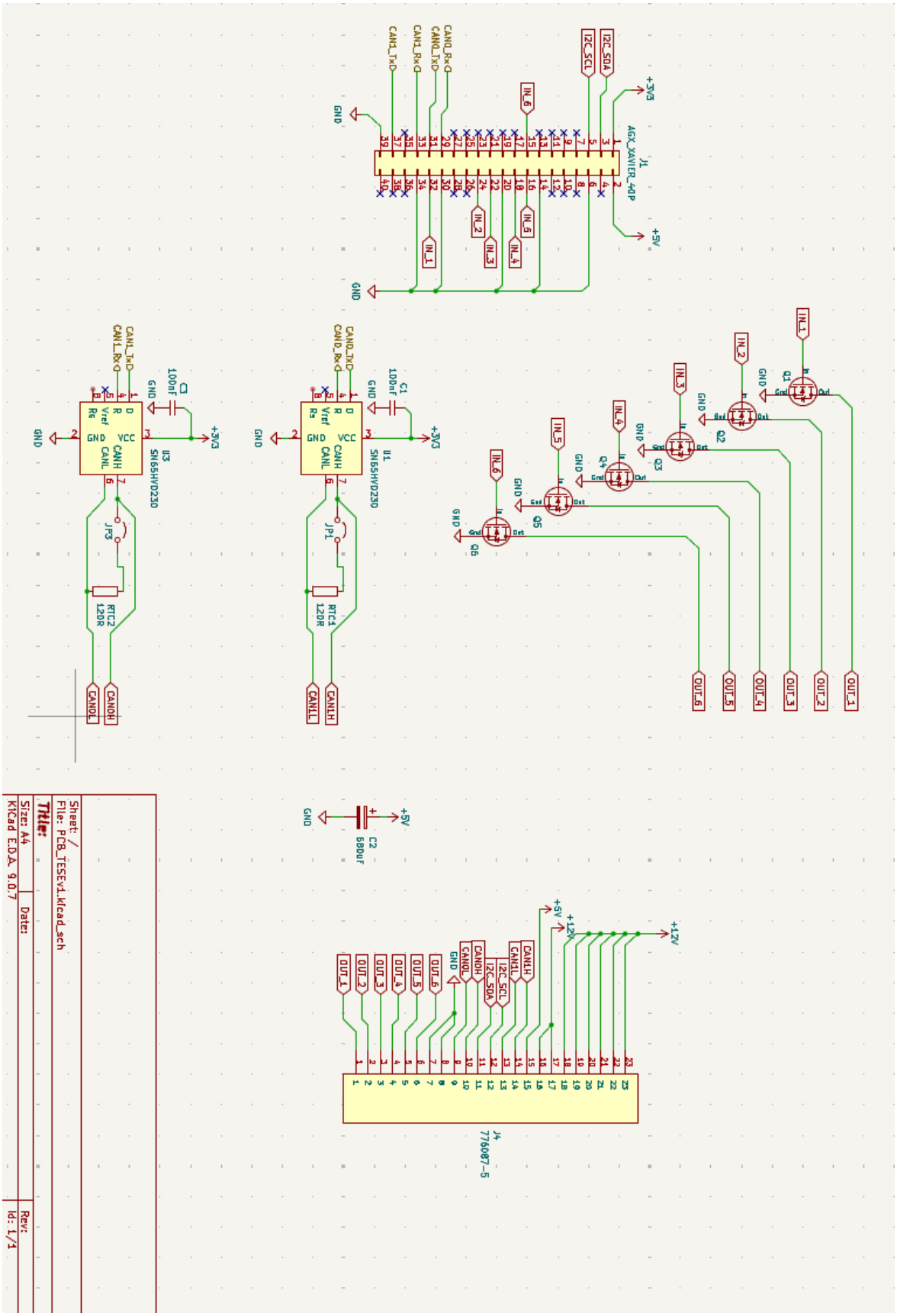
[44] C. Joseph, “Understanding LiDAR,” 30 janeiro 2024. [Online]. Available: https://www.virtanatech.com/blog/understanding-lidar. [Acedido em 21 Abril 2026].

[45] Garmin Ltd, “LIDAR-Lite v3HP,” 2024. [Online]. Available: https://www.garmin.com/en-US/p/578152/pn/010-01722-10#specs. [Acedido em 29 novembro 2024].

[46] ROHM, “Automotive IPD 1ch Low Side Switch,” 2015. [Online]. Available: https://fscdn.rohm.com/en/products/databook/datasheet/ic/power/ipd/bv1lb045fpj-c-e.pdf. [Acedido em Janeiro 2026].

[47] R. v. d. Horst e J. Hogema, “TIME-TO-COLLISION AND COLLISION AVOIDANCE,” Janeiro 1994. [Online]. Available: https://www.researchgate.net/publication/237807114_TIME-TO-COLLISION_AND_COLLISION_AVOIDANCE_SYSTEMS. [Acedido em Maio 2025].

Anexo 1 – Esquema Global da PCB



Anexo 2 – Manual de Utilizador

Índice

1. Visão Geral

Este manual descreve a interface gráfica desenvolvida para configurar e controlar o sistema ADAS antes e durante a execução do processamento de imagem. Inclui instruções passo-a-passo para os menus de pré-execução e o mecanismo de interrupção.

2. Requisitos Mínimos

- Sistema operativo: Ubuntu 20.04
- Dependências Python: tkinter, opencv-python, pyrealsense2, python-can, Jetson.GPIO.
- Permissões: acesso a dispositivos de vídeo e de escrita em disco.
- *Hardware*: NVIDIA Jetson AGX Xavier, sensor LiDAR, câmara Intel RealSense (opcional), barramento CAN.

3. Inicialização do *Software* e Menu de Configuração

3.1 Modo de Execução

No início do processamento, o sistema apresenta um menu de configuração intuitivo. Esta interface permite ao utilizador definir o modo de execução do sistema quer seja apenas deteção de objetos, de faixas ou ambas, como é apresentado na Figura 0-1 com as seguintes opções:

1. *Objects*
2. *Roadway*
3. *Both*

Neste menu, utilizador deve inserir o número correspondente e clicar em “OK”. Em caso de erro de seleção de uma opção válida, a seguinte mensagem de erro aparece: "Invalid Mode. Choose 1, 2 or 3." E o sistema é desligado.

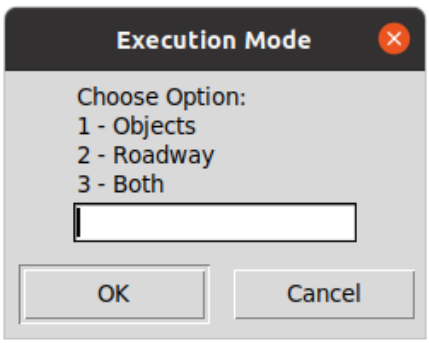


Figura 0-1 - Menu de seleção de modo de execução.

3.2 Seleção da Fonte de Vídeo

Após o primeiro menu, o sistema pergunta se o utilizador pretende usar uma câmara para aquisição de imagens em tempo real como mostra a Figura 0-2.

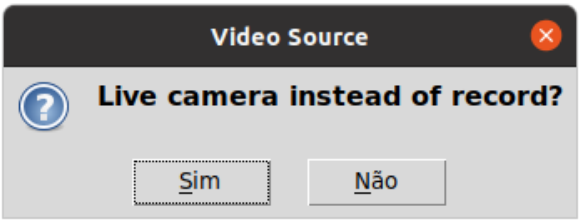


Figura 0-2 - Menu de configuração de fonte de vídeo.

3.2.1 Seleção de Ficheiro de Vídeo

Caso o utilizador não opte por utilizar uma câmara para aquisição de imagens em tempo real, uma nova janela surge para seleção de ficheiro de vídeo para processamento no sistema como mostra a Figura 0-3.

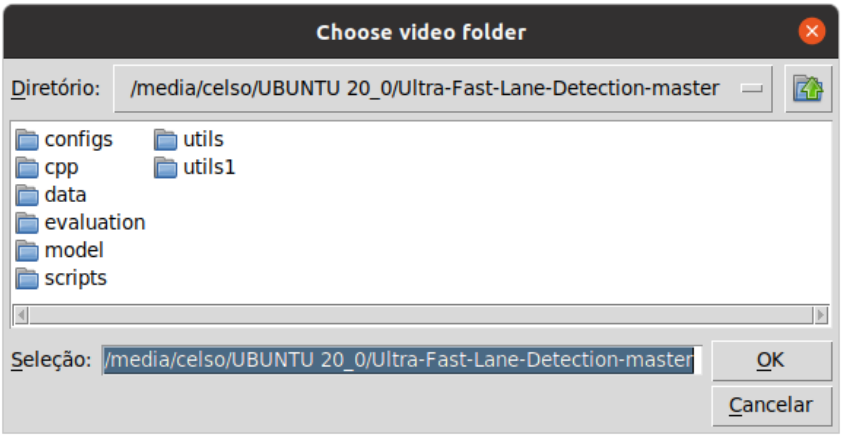


Figura 0-3 - Menu de seleção de ficheiro de vídeo.

Aqui, caso nenhum ficheiro seja selecionado e o utilizador clique em “OK”, uma mensagem de erro aparece e o sistema é encerrado.

3.3 Opção de gravação e nome do ficheiro

Caso a fonte de video selecionada no menu anterior seja a câmara, o sistema pergunta se o utilizador pretende gravar o *dashboard* de resultados como mostra a Figura 0-4.

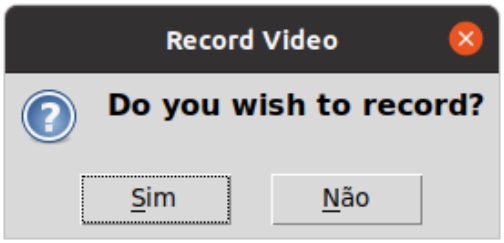


Figura 0-4 - Menu de ativação de gravação de *Dashboard* de resultados.

Se 'Sim', o utilizador deve indicar o nome do ficheiro e a pasta de destino, como mostra na Figura 0-3 e Figura 0-5.

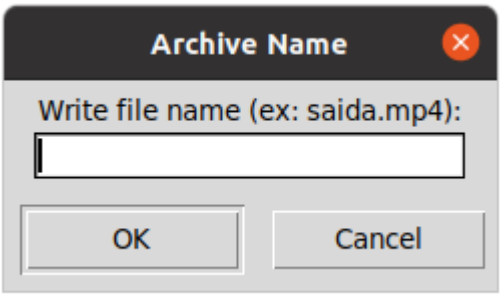


Figura 0-5 - Menu de seleção de nome de ficheiro de saída.

3.4 Visualização em tempo real

Após a configuração dos menus anteriores, o sistema pergunta se o utilizador pretende visualizar o processamento em tempo real como mostra a figura seguinte.

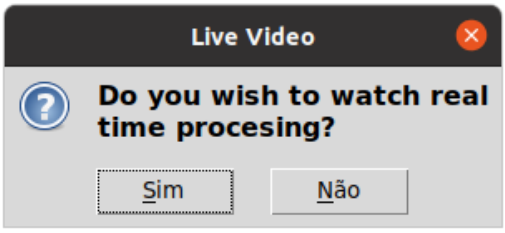


Figura 0-6 - Menu de configuração de visualização de processamento em tempo real.

4. Botão de interrupção (STOP)

Paralelamente ao processamento, o *software* inicia uma tarefa independente para a gestão do botão de interrupção (Figura 0-7). Esta funcionalidade é uma medida de interrupção que, ao ser acionada, altera a variável global de controlo, que por sua vez desencadeia um protocolo de encerro imediato. Este protocolo garante que, antes do fecho do programa, todos os GPIOS sejam colocados num estado baixo, prevenindo que os

alertas visuais e sonoros permaneçam ativos ou que ocorram consumos energéticos desnecessários após a paragem do sistema.

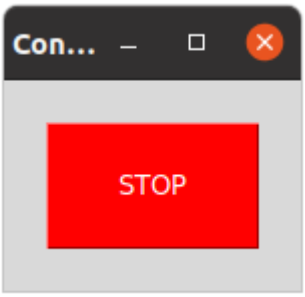


Figura 0-7 - Botão de interrupção do sistema.

Durante o protocolo de encerro, o sistema encerra todas as tarefas, encerra as janelas de visualização de vídeo e gravação, terminando assim o programa com segurança.

5. Notas de Segurança

Este sistema deve ser utilizado apenas em ambientes controlados, onde se deve garantir que todos os testes são realizados com supervisão e medidas de segurança adequadas.

Anexo 3 – Guia de Instalação e Configuração: Sistema ADAS em NVIDIA Jetson AGX Xavier

1. Introdução

Este guia tem como objetivo fornecer instruções detalhadas para a instalação e configuração de um sistema ADAS implementado numa NVIDIA Jetson AGX Xavier. Este manual fornece informações desde a instalação de um sistema operativo na NVIDIA até à implementação do sistema.

2. Requisitos de *hardware* e software

O presente capítulo descreve as especificações técnicas e as ferramentas fundamentais para a instalação do sistema ADAS proposto. A correta definição dos requisitos de *hardware* e software é um passo crítico para assegurar o cumprimento dos objetivos de desempenho, garantindo a interoperabilidade entre os sensores, a unidade de processamento e a lógica de controlo em tempo real.

2.1 *Hardware* Necessário

Nesta secção, são detalhados os componentes físicos que constituem a arquitetura do protótipo. A integração destes elementos, desde a unidade de processamento central até aos sensores de profundidade e interface física, forma a base do sistema de segurança ativa desenvolvido. Os requisitos de *hardware* necessário são:

- NVIDIA Jetson AGX Xavier
- Fonte de alimentação
- Câmara Intel RealSense D435i
- Sensor LiDAR Garmin LIDAR-Lite V3HP
- LEDs e *buzzer*
- PCB do sistema ADAS
- Computador com sistema operativo Ubuntu 20.04 LTS

2.2 Software Necessário no Computador

A preparação e o suporte à instalação na unidade embebida exigem um ambiente de trabalho específico no computador anfitrião. Esta secção enumera os requisitos de

software e de armazenamento indispensáveis para realizar a instalação do sistema operativo na NVIDIA, que são os seguintes:

- Ubuntu 20.04 LTS
- NVIDIA SDK Manager
- Espaço livre em disco: mínimo 50 gigabytes

3. Flash do Sistema Operativo

Diferente de uma arquitetura computacional convencional, a NVIDIA Jetson Xavier AGX requer um processo de instalação de software designado por *flashing*, realizado a partir de um computador externo. Este procedimento permite gravar a imagem do sistema operativo e as bibliotecas JetPack diretamente na memória da unidade. O processo é mediado pelo software NVIDIA SDK Manager. Para garantir o sucesso da operação, a unidade deve ser colocada em modo de recuperação, permitindo a comunicação de baixo nível entre o computador externo e a Xavier.

Assim, o procedimento de instalação do sistema operativo na NVIDIA segue os seguintes passos:

1. Desligar a Jetson Xavier.
2. Premir e manter o botão "Force Recovery".
3. Premir o botão "Power" enquanto mantém o anterior premido (Figura 0-1).
4. Ligar o cabo USB-C ao computador externo.
5. No computador externo, abrir o SDK Manager, iniciar sessão e selecionar "Jetson AGX Xavier" com JetPack 5.1.1.
6. Selecionar "OS + SDK" e clicar em "Flash".

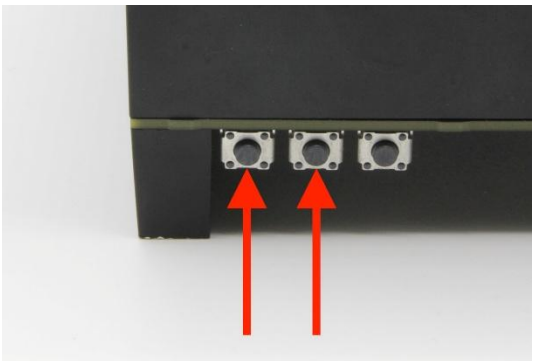


Figura 0-1 - Procedimento de ativação do *Recovery Mode* na NVIDIA AGX Xavier

Em caso de dúvidas nesta fase da instalação deve-se consultar o tutorial de instalação disponível em: https://www.youtube.com/watch?v=wACG73lWWe0 .

4. Configuração Inicial da Jetson

Após o primeiro arranque do sistema operativo, deve se proceder à configuração da conta de utilizador e à atualização dos repositórios. Este passo é crítico para garantir que a base do sistema Linux está estável e segura antes da instalação das bibliotecas requeridas para o funcionamento das redes neuronais.

Em primeiro lugar, deve-se proceder à atualização de repositórios e *drivers* do sistema. A sincronização com os servidores oficiais da NVIDIA e do Ubuntu permite obter as versões mais recentes dos *drivers* e correções de segurança. Para tal, deve-se executar o seguinte código no terminal: sudo apt update && sudo apt upgrade -y. Após este comando, deve aparecer um conjunto de dados como os da figura seguinte.

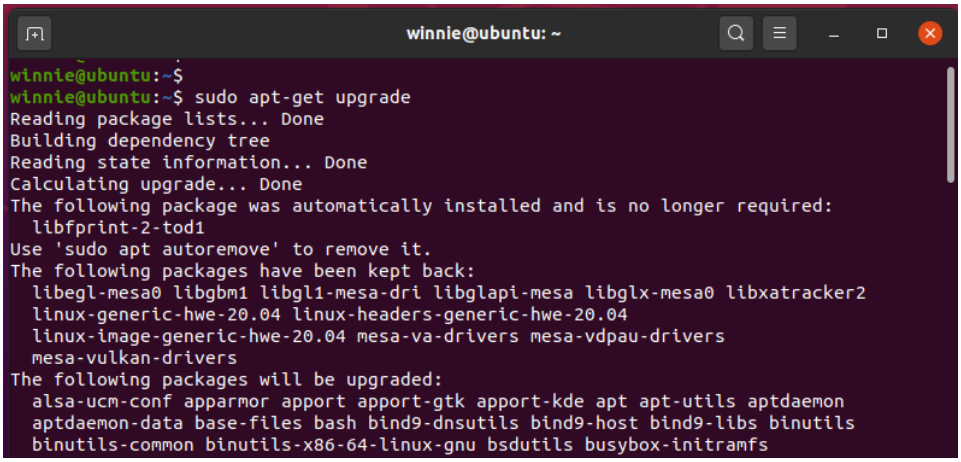


Figura 0-2 - Atualização dos *drivers* da NVIDIA via terminal.

Para suportar a compilação de bibliotecas Python e a gestão do algoritmo, é necessária a instalação de um conjunto de utilitários de sistema. Para tal, deve-se executar o seguinte código no terminal: sudo apt install git build-essential python3-pip python3-venv curl -y.

5. Instalação e Configuração do Ambiente de Desenvolvimento (Spyder)

Para a edição e execução do algoritmo ADAS, sugere-se a utilização do programa Spyder. O procedimento de instalação é realizado diretamente através dos repositórios oficiais do Ubuntu, o que garante a compatibilidade com as bibliotecas de sistema da Jetson. Para tal, deve-se executar o seguinte comando no terminal: sudo apt install spyder3 -y.

Ao abrir o Spyder pela primeira vez (Figura 0-3), deve confirmar-se que o interpretador está a localizado no Python 3 da Jetson (geralmente em /usr/bin/python3). Isto garante que todas as bibliotecas instaladas anteriormente sejam reconhecidas corretamente.

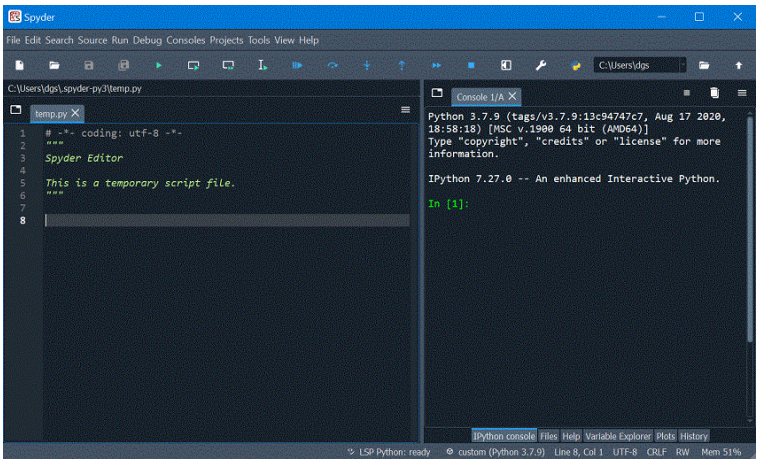


Figura 0-3 - Janela inicial do programa Spyder

6. Integração de *Hardware* e Ligações Periféricas

Após a preparação do ambiente de software, deve proceder-se à montagem física do protótipo. Esta fase exige rigor na manipulação dos componentes para garantir a integridade dos sinais elétricos e a correta comunicação entre os sensores e a unidade de processamento.

6.1. Acoplamento da PCB à NVIDIA Jetson Xavier

A PCB foi desenvolvida para ligar diretamente com a placa de expansão de 40 pinos da NVIDIA Jetson Xavier AGX.

O acoplamento deve ser realizado com a unidade desligada da alimentação. A PCB deve ser alinhada verticalmente com os pinos da Jetson, garantindo que todos os conectores fêmea encaixam sem esforço excessivo, evitando a deformação dos pinos de cobre.

Com a PCB devidamente fixada, os restantes componentes devem ser conectados seguindo a arquitetura de portas dedicada:

1. **Sensor LiDAR:** O LiDAR deve ser ligado ao conector específico na PCB, que utiliza o protocolo de comunicação I2C (Bus 8). É imperativo respeitar a ordem dos cabos (VCC, GND, SDA, SCL) conforme indicado na serigrafia da placa.

2. **Câmara Intel RealSense D435i:** Embora o processamento seja feito na Jetson, a câmara deve ser ligada diretamente a uma das portas USB 3.1 Tipo-C utilizando o cabo específico da mesma. Esta ligação direta é necessária para suportar a elevada taxa de transferência de dados do fluxo de imagens de cor e profundidade.
3. **Alertas (LEDs e Buzzer):** Estes componentes devem ser ligados a cada interruptor inteligente disponível na PCB.

O utilizador deve consultar o PINOUT (Figura 0-4) da ficha de expansão, disponível na parte inferior da PCB, antes de realizar qualquer ligação. Este passo é de extrema importância, uma vez que cada pino tem uma função dedicada (alimentação, massa ou sinal). Uma inversão, por exemplo, entre o pino de alimentação (5V) e um pino de dados (SDA/SCL), resultará em danos irreversíveis nos pinos de comunicação I2C da Jetson Xavier.

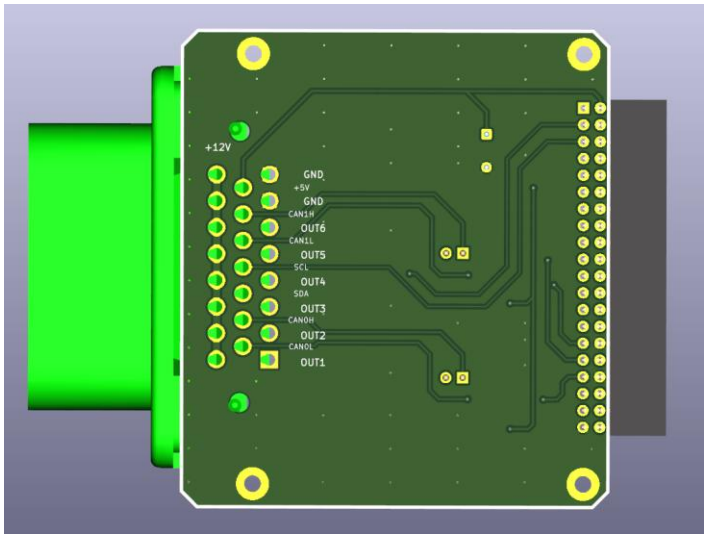


Figura 0-4 - Vista de baixo da PCB

7. Ativação e Configuração do Barramento CAN

A NVIDIA possui controladores CAN integrados, no entanto, por questões de versatilidade de *hardware*, os pinos correspondentes no barramento de 40 pinos encontram-se, por defeito, configurados como GPIO. Para estabelecer comunicação com o barramento da viatura, é imperativo realizar uma reconfiguração da matriz de pinos do processador, de modo a ativar as funções de *hardware* específicas para cada saída CAN.

A alteração das definições de *hardware* é efetuada através da ferramenta utilitária jetson-io.py, que permite modificar os pinos do sistema de forma intuitiva. Para tal, deve-se seguir a seguinte ordem:

1. **Acesso às definições:** No terminal da Jetson, deve executar-se o comando: sudo /opt/nvidia/jetson-io/jetson-io.py onde irá aparecer uma interface gráfico como mostra a Figura 0-5.

2. **Ativação:** Dentro da interface gráfica, deve-se seguir o caminho:

    - ***Configure Jetson 40pin Header*:** Seleciona o barramento de 40 pinos.

    - ***Configure header pins manually:*** Permite a ativação individual de funções.

    - **Ativar can0 e can1:** Neste passo, os pinos físicos 29, 31, 37 e 33 deixam de ser portas lógicas simples e passam a estar ligados aos controladores CAN internos.

3. **Guardar alterações:** Selecionar Save and reboot. O sistema irá gerar um ficheiro de configuração (*overlay*) e reiniciar para aplicar as novas funções de *hardware*.

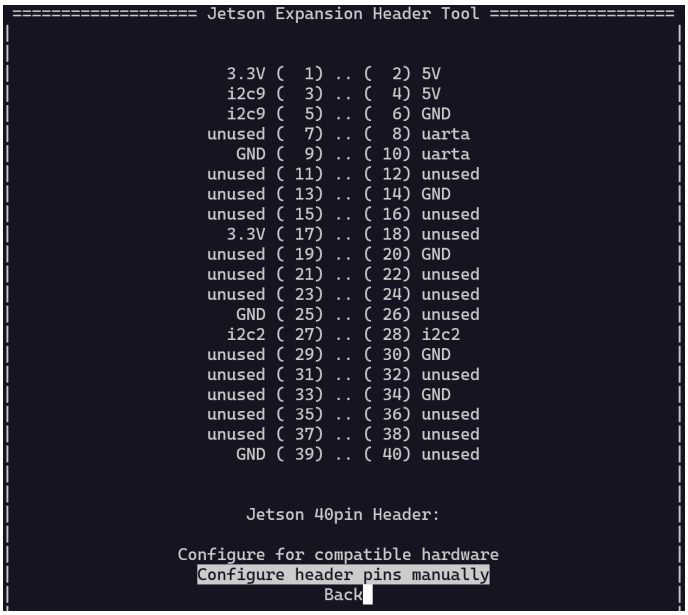


Figura 0-5 - Interface gráfica de configuração de pinos da NVIDIA

Uma vez com os barramentos CAN ativos, é necessário definir as propriedades da rede (*bitrate*) para que o sistema consiga interpretar os sinais diferenciais provenientes da PCB. Para tal, deve-se executar os seguintes comandos:

- sudo modprobe mttcan;
- sudo ip link set can0 type can bitrate 500000 restart-ms 100;
- sudo ip link set can0 up.

Estes comandos ativam o driver oficial da NVIDIA para o controlador CAN, definem a velocidade de transmissão de dados e ativa a leitura de dados de linha CAN na linha selecionada, respetivamente.

8. Ambiente Python e Dependências

Para que o algoritmo ADAS funcione corretamente, o sistema operativo deve estar equipado com um conjunto específico de bibliotecas. E

As bibliotecas listadas abaixo são fundamentais e obrigatórias. Sem elas, o algoritmo não conseguirá inicializar os drivers da câmara, ler os dados do LiDAR ou interpretar as mensagens do veículo via CAN:

- ultralytics;
- pyrealsense2;
- smbus2;
- python-can;
- opencv-python;
- numpy;
- scipy.

No terminal da Jetson, deve-se executar respetivamente os seguintes comandos:

- pip3 install --upgrade pip
- pip3 install ultralytics pyrealsense2 smbus2 python-can opencv-python numpy scipy

Um erro comum em sistemas NVIDIA Jetson é tentar instalar o PyTorch via comando pip normal. Para que o algoritmo utilize os núcleos CUDA da GPU e não sobrecarregue o processador (CPU), a instalação deve ser feita com ficheiros específicos. Assim, devem

ser utilizadas os ficheiros (.whl) oficiais da NVIDIA. Estes ficheiros binários são pré-compilados para a arquitetura ARM da Jetson. Assim, deve-se executar os seguintes comandos:

- pip3 install torch-2.0.0+nv23.05-cp38-cp38-linux_aarch64.whl
- pip3 install torchvision

9. Estrutura do projeto e organização de ficheiros

Finalmente, para que o sistema ADAS opere corretamente, a organização dos ficheiros no armazenamento da NVIDIA deve seguir uma estrutura rigorosa. O algoritmo foi desenvolvido assumindo uma hierarquia de diretórios específica, permitindo que as funções de carregamento de modelos e utilitários localizem os recursos necessários de forma automática.

Recomenda-se que a pasta principal do projeto, seja colocada no diretório pessoal do utilizador (ex: /home/usr/TESE/). Esta localização facilita a gestão de permissões de leitura e escrita, tanto para a execução do código como para a exportação dos vídeos de saída.

A alteração do nome de qualquer uma destas pastas ou ficheiros sem a correspondente atualização na secção de "Constants and Settings" do código resultará numa falha de inicialização.

Após a descompactação de todos os ficheiros nas localizações determinadas previamente, o sistema estará prono a ser utilizado. Assim, o sistema pode ser acionado utilizando diretamente o programa Spyder.