**Universidade dos Açores**

Pós-Graduação em Programação, Robótica e Inteligência Artificial (PRIA)

Relatório de Integração de Robótica e Inteligência Artificial

Carro Robótico com Visão Computacional usando HuskyLens

Autores: João Pavão, Tiago Martins

Docente: Prof. José Cascalho

Ponta Delgada, 30 de maio de 2025

# Resumo

Este relatório descreve o desenvolvimento de um projeto prático no âmbito da unidade curricular de Robótica e Inteligência Artificial, inserido na Pós-Graduação PRIA da Universidade dos Açores. O projeto consiste na construção de um carro robótico autónomo equipado com a câmara HuskyLens, capaz de realizar tarefas de perceção visual, navegação e tomada de decisão em tempo real, recorrendo a inteligência artificial embarcada.

São exploradas ligações entre os componentes físicos do projeto (Arduino UNO R4 WiFi ou Uno R3, ponte H L298P, motores DC, HuskyLens) e diversos algoritmos abordados na unidade curricular, incluindo algoritmos de procura (A\*, custo uniforme, procura em profundidade/largura), métodos de otimização (Trepa-colinas, algoritmos genéticos), classificação supervisionada, redes neuronais convolucionais e princípios de aprendizagem por reforço.

Este projeto será apresentado publicamente no evento AzoresBot, representando uma proposta de robótica educativa com IA acessível. Assume-se como ponto de partida para a evolução de um sistema autónomo mais avançado, recorrendo a plataformas open source e hardware como o Raspberry Pi, com integração de bibliotecas como TensorFlow, YOLO ou MediaPipe. A iniciativa destaca o potencial da visão computacional e da inteligência artificial no contexto da robótica prática e pedagógica.

**Palavras-chave:** Robótica Inteligente, Visão Computacional, HuskyLens, Arduino, IA Embarcada, Algoritmos de Procura, Otimização, Aprendizagem por Reforço, Raspberry Pi, Projeto Educacional, AzoresBot.

# Índice

[Resumo 2](#_Toc199154038)

[Índice 3](#_Toc199154039)

[1. Introdução 5](#_Toc199154040)

[2. Análise Detalhada da HuskyLens 6](#_Toc199154041)

[2.1 Características Físicas e Hardware 6](#_Toc199154042)

[2.2 Processador Kendryte K210 7](#_Toc199154043)

[2.3 Modos de Operação da HuskyLens 8](#_Toc199154044)

[2.4 Inteligência Artificial Embarcada e Classificação Supervisionada 9](#_Toc199154045)

[3. Relação com os Algoritmos de Inteligência Artificial 10](#_Toc199154046)

[3.1 Algoritmos de Procura (A\*, Guloso, Largura, Profundidade, Uniforme) 10](#_Toc199154047)

[3.2 Otimização Heurística e Algoritmos Genéticos 11](#_Toc199154048)

[3.3 Classificação Supervisionada e Redes Neuronais na HuskyLens 12](#_Toc199154049)

[3.4 Aprendizagem por Reforço (Q-Learning, DQN) 13](#_Toc199154050)

[3.5 Inferência e Restrições (Redes Bayesianas, CSP) 15](#_Toc199154051)

[4. Contribuição do Projeto para a Aprendizagem 16](#_Toc199154052)

[4.1 Integração prática de conceitos de IA 16](#_Toc199154053)

[4.2 Relevância pedagógica e educacional 17](#_Toc199154054)

[5. Plano de Interação do Robô com a HuskyLens 18](#_Toc199154055)

[6. Expansão do Projeto com Raspberry Pi 20](#_Toc199154056)

[7. Sistemas Operativos e Ambientes para Robôs Inteligentes 22](#_Toc199154057)

[8. Plataformas Open Source para Rovers e Veículos Autónomos 23](#_Toc199154058)

[9. Apresentação no AzoresBot e Perspetivas Futuras 24](#_Toc199154059)

[10. Bibliografia e Recursos 26](#_Toc199154060)

[10.1 Bibliografia 26](#_Toc199154061)

[10.2 Webgrafia 27](#_Toc199154062)

[10.3 Prompts utilizados (ChatGPT/OpenAI) 28](#_Toc199154063)

## 1. Introdução

O presente relatório visa documentar a integração de conceitos da disciplina de Robótica e Inteligência Artificial num projeto prático baseado na câmara AI HuskyLens. O projeto consiste na construção de um carro robótico autónomo com capacidades de visão computacional, utilizando como microcontrolador o Arduino, uma ponte H L298P, motores DC, rodas, estrutura em 3D e possibilidade de integração de sensores adicionais, como o sensor ultrassónico HC-SR04.

A HuskyLens é uma câmara de visão computacional equipada com modelos de Inteligência Artificial pré-treinados, permitindo o reconhecimento em tempo real de objetos, rostos, cores, linhas e códigos. Esta câmara possibilita a interação direta com algoritmos de classificação supervisionada e redes neuronais convolucionais (CNNs), dispensando a necessidade de programação complexa por parte do utilizador.

Além da aplicação prática da IA embarcada, o projeto serve de base para a exploração de diversos algoritmos clássicos de IA abordados na unidade curricular, tais como algoritmos de procura (A\*, BFS, DFS), otimização heurística (Trepa-colinas, Algoritmos Genéticos), aprendizagem por reforço (Q-Learning, DQN), e raciocínio simbólico (Redes Bayesianas e CSP). Este trabalho será também apresentado no evento AzoresBot e constitui um ponto de partida para a futura evolução do sistema para uma versão com maior autonomia, baseada em plataformas open source como o Raspberry Pi.

## 2. Análise Detalhada da HuskyLens

### 2.1 Características Físicas e Hardware

- Sensor de imagem: OV2640 (2.0 Megapixeis)

- Interface: UART / I2C / SPI

- Display integrado: ecrã LCD para visualização em tempo real

- Alimentação: 3.3V ou 5V

- Microcontrolador principal: Kendryte K210

### 2.2 Processador Kendryte K210

* Chip especializado em IA embarcada, com foco em visão computacional e reconhecimento de áudio.
* Dual-core RISC-V 64-bit, cada núcleo com FPU de precisão dupla (IEEE754).
* Inclui KPU (Unidade de Processamento Neural) dedicada para executar redes neuronais convolucionais (CNNs).
* Capaz de realizar inferência em tempo real com modelos quantizados de até 5.9 MB (ex: YOLOv2-tiny).
* 8 MB de memória SRAM interna, sendo 2 MB exclusivos para a IA.
* Baixo consumo de energia e operação em larga faixa de temperatura (-40 °C a 125 °C).
* Integra aceleradores de FFT, AES e SHA256, facilitando aplicações com segurança e análise de sinais.

### 2.3 Modos de Operação da HuskyLens

A HuskyLens é uma câmara de visão computacional com algoritmos de IA integrados e interface interativa. Ela suporta múltiplos modos de funcionamento autónomos, prontos para uso:

* **Reconhecimento Facial**
  + Deteta e reconhece rostos humanos com alta precisão. Permite registar múltiplas identidades.
* **Reconhecimento de Objetos**
  + Permite treinar objetos por exemplo visual, associando imagens captadas a etiquetas (rótulos) definidas pelo utilizador.
* **Deteção de Linhas (Line Following)**
  + Ideal para carros robóticos; identifica linhas curvas ou retas para seguimento autónomo.
* **Reconhecimento de Cores**
  + Deteta e classifica cores específicas aprendidas pelo utilizador com um simples toque.
* **Leitura de Códigos QR e de Barras**
  + Capaz de ler códigos QR e barras 1D (como EAN-13), permitindo comunicação via imagem.
* **Deteção de Objetos Pré-treinados (Object Tracking)**
  + Usa modelos integrados (ex: Tiny YOLO) para detetar categorias genéricas (pessoa, carro, etc).
* **Aprendizagem por Exemplos (Supervised Learning)**
  + O utilizador pode mostrar imagens diferentes para cada categoria e a HuskyLens aprende com base nessas amostras visuais.

### 2.4 Inteligência Artificial Embarcada e Classificação Supervisionada

A HuskyLens integra algoritmos de Inteligência Artificial embarcada, com foco em classificação supervisionada e visão computacional em tempo real, sem necessidade de programação externa ou ligação a servidores.

* A aprendizagem ocorre diretamente na câmara, através do botão físico ou via comunicação serial (I2C/UART), tornando a interação com o robô intuitiva e acessível.
* É possível treinar modelos diretamente com a imagem captada, atribuindo rótulos visuais a objetos, cores, rostos ou linhas, com feedback visual imediato.
* O processo de aprendizagem baseia-se em classificação supervisionada, onde a câmara associa cada imagem a um rótulo fornecido pelo utilizador.
* O reconhecimento é feito em tempo real, com apresentação gráfica no ecrã da câmara (retângulo delimitador, ID da classe, valor de confiança).

Este tipo de abordagem permite integrar inteligência artificial em sistemas robóticos sem a complexidade habitual dos processos de treino, funcionando como um sensor de visão inteligente que comunica diretamente com o Arduino para permitir decisões autónomas baseadas na perceção visual.

## 3. Relação com os Algoritmos de Inteligência Artificial

### 3.1 Algoritmos de Procura (A\*, Guloso, Largura, Profundidade, Uniforme)

Apesar de o carro robótico com HuskyLens não utilizar explicitamente algoritmos de procura na sua implementação atual, é possível estabelecer analogias diretas com várias estratégias clássicas de procura ensinadas na disciplina de Inteligência Artificial.

* **Procura A\*:** Se o robô dispusesse de um mapa do ambiente (por exemplo, numa grelha ou labirinto), poderia utilizar A\* para encontrar o caminho mais curto entre dois pontos, com base numa função heurística (ex.: distância estimada até ao objetivo).
* **Procura Gulosa:** A reação do robô à deteção de objetos ou linhas pela HuskyLens pode ser comparada à lógica gulosa — seguir imediatamente o alvo com maior confiança, sem considerar todas as possibilidades globais.
* **Procura em Largura (BFS):** Aplicável em simulações, onde o robô tenta todas as ações possíveis a partir de um estado antes de aprofundar, útil para estratégias de exploração sistemática.
* **Procura em Profundidade (DFS):** Também útil em simulações, em que o robô se compromete com um caminho até ao fim antes de retroceder. Pode ser adaptado para planear sequências de ações.
* **Custo Uniforme:** Quando todas as ações do robô têm o mesmo custo (ex.: andar para a frente, virar), o algoritmo de custo uniforme torna-se equivalente à procura em largura.

Estas abordagens são particularmente relevantes para o planeamento de movimentos, navegação autónoma em ambientes estruturados (como labirintos ou trilhos), e podem ser integradas em versões futuras do projeto com mais capacidade de processamento (ex.: Raspberry Pi com ROS ou SLAM).

### 3.2 Otimização Heurística e Algoritmos Genéticos

O comportamento de um robô autónomo pode ser otimizado através de algoritmos heurísticos e evolucionários, sobretudo quando se pretende ajustar parâmetros sem um modelo matemático exato do ambiente. No projeto com HuskyLens, essas abordagens tornam-se especialmente úteis em contextos como seguimento de linha, resposta a objetos detetados e calibração do movimento.

* **Trepa-colinas (Hill Climbing):** Permite testar pequenas variações em parâmetros do robô (como velocidade dos motores ou limiares de confiança da HuskyLens) e manter as que melhoram o desempenho. É uma abordagem simples, mas eficaz em problemas com poucos picos locais.
* **Simulated Annealing:** Estende o algoritmo de trepa-colinas, introduzindo a possibilidade de aceitar temporariamente soluções piores para evitar mínimos locais. Pode ser aplicado, por exemplo, para encontrar a melhor trajetória em ambientes com obstáculos.
* **Algoritmo Genético:** Ideal para testar múltiplas combinações de parâmetros em paralelo. Cada “indivíduo” representa uma configuração do robô (ex.: tempo de resposta, ganhos de PID, distância mínima para parar). Após várias gerações, o sistema pode evoluir para uma solução mais robusta.

Estas técnicas são especialmente eficazes em cenários onde a programação manual seria demasiado demorada ou imprecisa. Apesar de ainda não terem sido implementadas no projeto atual, estão previstas para fases futuras de desenvolvimento, principalmente em versões com maior capacidade computacional como o Raspberry Pi.

Além disso, podem ser utilizadas em ambiente de simulação para encontrar os melhores comportamentos antes de testar no robô real, tornando-se ferramentas valiosas de otimização iterativa.

### 3.3 Classificação Supervisionada e Redes Neuronais na HuskyLens

A câmara HuskyLens é um dispositivo de IA embarcada que incorpora algoritmos de classificação supervisionada com redes neuronais convolucionais (CNNs), permitindo ao robô interpretar o mundo visual de forma autónoma, sem recurso a processamento externo.

**Classificação supervisionada:** A HuskyLens aprende com base em exemplos visuais fornecidos pelo utilizador. Durante o treino, o utilizador apresenta diferentes imagens (rostos, objetos, cores, linhas) e associa cada uma a um rótulo. O dispositivo armazena os vetores de características e utiliza-os para comparar futuras imagens em tempo real, numa lógica próxima ao K-Nearest Neighbors (KNN).

**Redes neuronais convolucionais (CNNs):** A inferência é feita internamente através de CNNs quantizadas e otimizadas para execução em tempo real no processador Kendryte K210. Este processador contém uma Unidade de Processamento Neural (KPU) dedicada, capaz de realizar reconhecimento visual de forma eficiente e com baixo consumo energético.

**Feedback visual direto:** A câmara fornece feedback visual no seu ecrã embutido, mostrando o retângulo de deteção, o ID da classe e o nível de confiança. Esta funcionalidade permite ao utilizador verificar rapidamente a qualidade do reconhecimento e realizar ajustes se necessário.

**Aplicação no robô:** As deteções visuais são enviadas ao Arduino via UART ou I2C, permitindo ao robô reagir com base na perceção visual: seguir uma linha, parar perante um objeto reconhecido, mudar de direção conforme a cor detetada ou ativar comportamentos com base na leitura de um código QR.

Esta abordagem permite incorporar inteligência adaptativa no robô sem necessidade de código complexo, tornando a HuskyLens uma poderosa ferramenta pedagógica para demonstrar na prática os princípios da IA supervisionada e da visão computacional.

### 3.4 Aprendizagem por Reforço (Q-Learning, DQN)

A Aprendizagem por Reforço (Reinforcement Learning – RL) é um paradigma poderoso da Inteligência Artificial, no qual um agente aprende a tomar decisões com base na interação com o ambiente. Embora o projeto atual com HuskyLens e Arduino ainda não utilize diretamente algoritmos de RL, a arquitetura do robô e a natureza da sua interação com o meio tornam esta abordagem altamente aplicável em fases futuras.

* **Princípio fundamental:** O agente (robô) executa ações num ambiente (ex.: sala, pista com linha) e recebe recompensas com base nas suas decisões. O objetivo é aprender uma política ótima que maximize a recompensa acumulada ao longo do tempo.
* **Estados, ações e recompensas no projeto:**
  + Estado: posição da linha no ecrã da HuskyLens, tipo de objeto detetado, distância a um obstáculo.
  + Ações: avançar, virar à esquerda/direita, parar.
  + Recompensa: manter-se no caminho correto, evitar colisões, alcançar um alvo visual.
* **Q-Learning:** É um dos algoritmos mais simples e eficazes para treinar o robô em tarefas discretas. Permite construir uma tabela de valor Q que associa cada par estado-ação ao retorno esperado. Pode ser usado, por exemplo, para treinar o robô a seguir uma linha mesmo com interrupções ou curvas inesperadas.
* **DQN (Deep Q-Network):** Uma extensão moderna do Q-Learning que utiliza redes neuronais profundas para estimar os valores Q, especialmente útil quando os estados são contínuos ou derivados de imagens. Em versões com Raspberry Pi ou uso de bibliotecas como Gymnasium ou TensorFlow, o robô poderá ser treinado com DQN para comportamentos mais complexos.
* **Ambientes de treino e simulação:** Antes de aplicar RL diretamente no robô físico, é possível utilizar ambientes simulados para treinar os modelos com segurança e maior velocidade, transferindo depois o conhecimento para o robô real (técnica conhecida como Sim2Real).

O uso de RL representa uma oportunidade de explorar comportamentos inteligentes emergentes, onde o robô não é programado diretamente com regras fixas, mas aprende através de tentativa, erro e recompensa. Esta abordagem liga diretamente os conceitos da disciplina a aplicações práticas com enorme valor didático.

### 3.5 Inferência e Restrições (Redes Bayesianas, CSP)

Além dos modelos de perceção e aprendizagem, o campo da Inteligência Artificial inclui métodos de raciocínio simbólico e inferência lógica, como as Redes Bayesianas e os Problemas de Satisfação de Restrições (CSP). Estas abordagens, vistas na unidade curricular (GAI2), têm aplicação direta ou futura em projetos robóticos que requerem tomada de decisão estruturada ou baseada em múltiplos fatores.

* **Redes Bayesianas:** São modelos probabilísticos que representam relações causais entre variáveis. No contexto do robô com HuskyLens, podem ser usadas para combinar diferentes fatores sensoriais:
  + Exemplo: Se o robô detetar um objeto azul com alta confiança e estiver a menos de 15 cm do mesmo (medido por sensor ultrassónico), então a probabilidade de ser “o alvo certo” é elevada, e o robô pode decidir parar.
  + Isto permite um comportamento mais flexível e adaptativo do que decisões baseadas apenas em regras fixas (IF-THEN).
* **Problemas de Satisfação de Restrições (CSP):** Aplicáveis quando o robô tem de planear ações que respeitem certas condições simultâneas.
  + Exemplo: “O robô só pode virar à esquerda se o caminho estiver livre e a cor detetada for vermelha.”
  + Este tipo de raciocínio é fundamental para programar sequências de tarefas complexas ou missões com múltiplas etapas (como desafios no AzoresBot).
* **Backtracking e propagação de restrições:** São estratégias úteis para resolver labirintos, decidir trajetórias em ambientes com obstáculos, ou definir planos com prioridades (ex: “recolher primeiro os objetos verdes antes dos azuis”).

Estas técnicas aumentam a capacidade de decisão simbólica do robô, permitindo integrar perceção visual com lógica formal e probabilística. Embora não estejam implementadas no protótipo atual, são perfeitamente compatíveis com versões futuras baseadas em plataformas como o Raspberry Pi, onde há recursos computacionais para modelar conhecimento, planear e inferir.

## 4. Contribuição do Projeto para a Aprendizagem

### 4.1 Integração prática de conceitos de IA

desenvolvimento do carro robótico com a câmara HuskyLens permitiu a aplicação direta de diversos conceitos abordados na disciplina de Inteligência Artificial. Ao longo do projeto, foi possível explorar, observar e interpretar o funcionamento de algoritmos clássicos e modernos em contexto real, promovendo uma compreensão mais profunda dos seus princípios.

Foram integrados conceitos como:

* Classificação supervisionada, através do treino visual com HuskyLens;
* Reconhecimento em tempo real com redes neuronais convolucionais (CNNs);
* Analogia com algoritmos de procura e planeamento de caminhos;
* Possibilidade de otimização heurística e evolução de parâmetros do sistema;
* Perspetiva de aplicação de aprendizagem por reforço e raciocínio simbólico em versões futuras.

Este tipo de abordagem prática demonstrou ser eficaz para consolidar o conhecimento teórico, ao permitir a experimentação e observação de resultados tangíveis num ambiente físico controlado.

### 4.2 Relevância pedagógica e educacional

Além do valor técnico e científico, o projeto assume uma forte dimensão pedagógica. A utilização de hardware acessível, como o Arduino e a HuskyLens, aliada a conceitos fundamentais de IA, torna este tipo de sistema ideal para atividades educacionais em contextos escolares e universitários.

A HuskyLens, com a sua interface intuitiva e capacidades de IA embarcada, proporciona uma experiência de aprendizagem imediata e motivadora para os alunos. O facto de não exigir conhecimentos avançados de programação torna-a uma ferramenta inclusiva, capaz de estimular o interesse pela robótica, pela inteligência artificial e pela engenharia de sistemas.

O projeto reforça também o potencial das abordagens práticas e baseadas em projetos (learning by doing), permitindo aos alunos compreenderem não só os algoritmos, mas também os desafios e decisões envolvidos na sua implementação em robôs reais.

## 5. Plano de Interação do Robô com a HuskyLens

A interação entre o robô e a câmara HuskyLens constitui o núcleo funcional do projeto, permitindo a execução de comportamentos autónomos baseados na perceção visual. Esta integração combina os dados obtidos pela câmara com decisões programadas no Arduino, formando um sistema robótico capaz de interpretar e reagir ao ambiente.

A lógica de controlo do robô é baseada na deteção visual de elementos previamente treinados na HuskyLens (ex.: rostos, objetos, cores, linhas) e na execução de ações apropriadas com base nos dados recebidos, através de estruturas condicionais e regras de decisão.

**Estruturas de Decisão e Comportamentos Reativos**

O Arduino recebe, via UART ou I2C, as informações da HuskyLens (como a posição X/Y do objeto, ID da classe e valor de confiança) e executa decisões com base em condições definidas pelo programador. Estas decisões seguem, em geral, uma lógica IF–THEN–ELSE, que pode ser estendida para estruturas mais avançadas, como máquinas de estados finitos (finite state machines).

Exemplos de comportamentos reativos implementados ou previstos:

* Se uma linha for detetada, o robô ativa o modo de seguimento e ajusta a direção com base na posição da linha.
* Se um rosto conhecido for reconhecido, o robô pode parar, acenar com um servo motor ou emitir um som.
* Se uma cor específica for detetada, o robô pode alterar o seu trajeto ou realizar uma ação especial (ex.: mudar de velocidade).
* Se um código QR for lido, pode ser interpretado como uma instrução visual para seguir uma nova rota ou iniciar uma missão.

Este tipo de interação promove uma forma de comportamento emergente simples, em que a inteligência do robô deriva da combinação entre perceção visual e reação motora.

**Reações a Estímulos Visuais**

A HuskyLens oferece suporte a múltiplos modos visuais, o que permite definir diferentes tipos de estímulos que podem desencadear ações específicas no robô. Entre os modos utilizados ou previstos no projeto destacam-se:

* Deteção de linhas para navegação e seguimento automático;
* Reconhecimento de rostos para interação social ou autenticação visual;
* Reconhecimento de objetos personalizados, treinados visualmente pelo utilizador;
* Identificação de cores para controle de fluxo ou sinalização de zonas;
* Leitura de códigos QR para comandos simbólicos e instruções visuais;
* Rastreamento de objetos móveis, aplicável a perseguição ou evasão de alvos.

A flexibilidade destes modos permite desenhar experiências interativas e educativas altamente envolventes, adequadas para demonstrações em eventos como o AzoresBot, bem como para ambientes de aprendizagem formal ou informal.

## 6. Expansão do Projeto com Raspberry Pi

A arquitetura atual do projeto, baseada em Arduino e na câmara HuskyLens, foi concebida para ser simples, acessível e eficiente em termos pedagógicos. No entanto, a crescente complexidade das tarefas de inteligência artificial e a necessidade de maior capacidade de processamento visual e tomada de decisão tornam natural a evolução para uma plataforma mais poderosa, como o Raspberry Pi.

A adoção do Raspberry Pi permitiria transformar o carro robótico num verdadeiro agente inteligente autónomo, com capacidade para executar modelos personalizados de IA, integrar múltiplos sensores e processar vídeo em tempo real.

**Visão Computacional Avançada com Modelos Personalizados**

Com o Raspberry Pi, seria possível substituir ou complementar a HuskyLens por uma câmara convencional (Pi Camera ou USB Webcam) e utilizar bibliotecas como:

**OpenCV:** Para pré-processamento de imagem, deteção de formas, contornos, movimento e extração de características;

**YOLO (You Only Look Once):** Para deteção de objetos com elevada performance e em tempo real;

**MediaPipe:** Para reconhecimento de gestos, mãos, poses corporais e segmentação de objetos;

**TensorFlow / Keras:** Para treinar e executar modelos personalizados de classificação ou regressão, tanto em imagens como em dados multissensoriais.

Estes recursos oferecem ao robô a capacidade de lidar com tarefas mais complexas, como seguir múltiplos alvos, identificar objetos em movimento, distinguir entre diferentes tipos de sinais visuais ou tomar decisões baseadas em contextos variáveis.

**Aprendizagem em Tempo Real e IA Descentralizada**

O Raspberry Pi também possibilita a implementação de modelos de aprendizagem por reforço, inferência probabilística e planeamento com restrições, com ou sem recurso à cloud. Isso inclui:

* Execução local de modelos treinados previamente (offline learning);
* Aprendizagem contínua durante a navegação do robô (online learning);
* Integração de múltiplos sensores (ultrassónicos, IMU, GPS, etc.) para decisões baseadas em fusão sensorial;
* Criação de mapas do ambiente com SLAM (Simultaneous Localization and Mapping).

Além disso, com o suporte a redes Wi-Fi e Bluetooth, o Raspberry Pi pode comunicar com dispositivos móveis ou servidores externos, facilitando a análise remota, recolha de dados e teleoperação.

**Aproveitamento de Plataformas Open Source**

Importa destacar que já existem diversos projetos de rovers open source, como o Open Source Rover da NASA/JPL (github.com/nasa-jpl/open-source-rover), entre outros, que oferecem documentação completa de hardware e software. O objetivo deste projeto não é reinventar soluções existentes, mas sim aprender a partir dessas bases, adaptando e aplicando os conhecimentos adquiridos na unidade curricular de Inteligência Artificial e Robótica.

A adoção desta filosofia permite acelerar o desenvolvimento, seguir boas práticas de engenharia e focar a aprendizagem nos algoritmos, integração e controlo inteligente dos sistemas.

## 7. Sistemas Operativos e Ambientes para Robôs Inteligentes

A transição do projeto para uma plataforma computacional mais avançada, como o Raspberry Pi, requer uma escolha adequada do sistema operativo. Esta escolha influencia diretamente a capacidade do robô executar algoritmos de inteligência artificial, processar imagens, integrar sensores e atuar de forma autónoma.

Entre os sistemas operativos open source mais utilizados para robótica e IA destacam-se:

* **Raspberry Pi OS:** leve, estável e amplamente suportado, ideal para projetos educativos com sensores, OpenCV, Python e IA leve.
* **Ubuntu Server/Desktop:** compatível com ROS 1 e ROS 2, permite correr frameworks como TensorFlow, Keras, PyTorch e Docker.
* **DietPi:** distribuição ultra-leve baseada em Debian, ideal para robôs headless ou aplicações com recursos limitados.
* **ROS (Robot Operating System):** não é um sistema operativo completo, mas sim um middleware robótico amplamente adotado, instalado normalmente sobre Ubuntu. Essencial para integração de múltiplos módulos robóticos.
* **BalenaOS:** sistema voltado para IoT, com suporte a containers e deploy remoto, útil em robôs conectados em rede.

A escolha do sistema operativo deve ser feita com base nas necessidades do projeto, nos recursos computacionais disponíveis e na experiência da equipa com ferramentas como Python, ROS ou frameworks de IA. Esta camada de software será fundamental para suportar a futura evolução do projeto, com aprendizagem por reforço, visão computacional avançada e navegação autónoma.

## 8. Plataformas Open Source para Rovers e Veículos Autónomos

O desenvolvimento de robôs móveis e rovers autónomos beneficia de uma vasta comunidade open source, que disponibiliza plataformas completas de hardware e software. Em vez de desenvolver todos os componentes do zero, é possível aprender e adaptar soluções já testadas, acelerando o desenvolvimento e focando a aprendizagem nos algoritmos e na integração de sistemas.

Algumas das principais plataformas open source disponíveis incluem:

* **NASA/JPL Open Source Rover:** réplica educacional do rover marciano, com documentação completa para montagem e programação, ideal para projetos escolares e universitários.
* **DonkeyCar:** carro autónomo baseado em Raspberry Pi e TensorFlow, treinado por comportamento humano. Excelente para aprendizagem de redes neuronais e visão computacional.
* **Duckietown:** ecossistema educacional com robôs e ambientes urbanos em miniatura, desenvolvido no MIT, focado em condução autónoma em cidades.
* **JetRacer (NVIDIA):** baseado em Jetson Nano, combina velocidade com capacidade de inferência com redes neuronais profundas em tempo real.
* **ArduRover (ArduPilot):** plataforma modular com navegação por GPS, útil para rovers de campo, agrícola ou terrestre.
* **Autoware e Apollo:** frameworks robustas para veículos autónomos reais, com suporte para ROS, sensores LIDAR, câmaras e planeamento avançado.

Estas plataformas oferecem código-fonte aberto, manuais de montagem e software pronto a usar, permitindo a qualquer equipa educacional ou entusiasta entrar no mundo da robótica e IA com base sólida.

Neste projeto, adota-se a filosofia de não reinventar a roda, mas sim aprender a partir destas soluções, explorando o seu funcionamento, adaptando módulos e construindo conhecimento prático a partir da engenharia já desenvolvida. Esta abordagem permite centrar os esforços na aprendizagem, criatividade e integração crítica dos conceitos de Inteligência Artificial e Robótica em ambientes reais.

## 9. Apresentação no AzoresBot e Perspetivas Futuras

A apresentação do projeto no AzoresBot representa não só uma oportunidade de demonstrar o funcionamento do carro robótico com IA embarcada, mas também um momento de partilha, validação e reflexão sobre o potencial educativo e científico da robótica inteligente.

A participação neste evento visa mostrar de forma prática como é possível integrar visão computacional, decisão autónoma e controlo físico num sistema acessível, usando tecnologias como o Arduino e a HuskyLens. Durante a apresentação, o robô demonstrará comportamentos como:

Seguimento de linha com deteção visual;

Reação a objetos ou rostos previamente treinados;

Execução de ações em resposta a estímulos visuais (cores, códigos QR, etc.).

Esta demonstração reforça a aplicabilidade dos algoritmos de Inteligência Artificial estudados na unidade curricular, ao permitir que sejam observados em funcionamento real, fora do ambiente puramente teórico ou simulado.

**Perspetivas Futuras**

O projeto atual constitui uma fase introdutória de um sistema mais ambicioso, que poderá evoluir para um carro autónomo avançado, com maior grau de inteligência, perceção e planeamento. As etapas futuras incluem:

Substituição ou extensão do microcontrolador Arduino com um Raspberry Pi, permitindo processamento local de IA mais avançada;

Implementação de modelos personalizados de classificação e deteção de objetos, utilizando bibliotecas como TensorFlow, YOLO ou MediaPipe;

Exploração de algoritmos de Aprendizagem por Reforço (ex: Q-Learning, DQN), com treino em simulação e validação no robô real;

Integração de múltiplos sensores (ultrassónicos, IMU, GPS) para fusão sensorial e navegação em ambientes reais;

Adaptação e estudo de plataformas open source já existentes, como o Open Source Rover da NASA/JPL ou o DonkeyCar, com o objetivo de aprender com base em modelos testados.

A filosofia do projeto continuará centrada na educação, acessibilidade e código aberto, incentivando a criação de ferramentas didáticas que combinem criatividade, rigor técnico e impacto pedagógico. A longo prazo, o projeto poderá dar origem a uma plataforma open source educacional em português, adaptada ao contexto nacional, e aberta à participação de escolas, clubes de robótica e instituições de ensino superior.

## 10. Bibliografia e Recursos

### 10.1 Bibliografia

**Inteligência Artificial e Machine Learning**

1. Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
2. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O’Reilly Media.
3. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.
4. Liu, Y. (2024). *Python Machine Learning by Example: With Real-World Use Cases* (4th ed.). Packt Publishing.
5. Mzili, T. (2025). *Innovations in Optimization and Machine Learning*.
6. Ryan, M. (2025). *Machine Learning for Tabular Data: XGBoost, Deep Learning, and AI*.
7. Selvi, G. (2025). *IoT and Machine Learning for Smart Applications*.

**Visão Computacional com Python e OpenCV**

1. Chen, J. (2023). *Learn OpenCV with Python by Examples: Implement Computer Vision* (2nd ed.).
2. Chesterfield, G. (2024). *Advanced Image Processing with Python and OpenCV*.
3. Mohaideen, K. (2024). *Industrial Vision Systems with Raspberry Pi: Python and OpenCV*.
4. Nuti, G. (2024). *Neural Network Computer Vision with OpenCV 5*.
5. Beyeler, M. (2017). *Machine Learning for OpenCV: A Practical Introduction to the World of Machine Learning and Image Processing using OpenCV and Python*. Packt Publishing.
6. Laganière, R. (2011). *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Publishing.

**Arduino, Sensores e Robótica Educativa**

1. Ziemann, V. (2023). *A Hands-On Course in Sensors Using the Arduino and Raspberry Pi* (2nd ed.).
2. Guillen, G. (2024). *Sensor Projects with Raspberry Pi: IoT and Digital Image Processing* (2nd ed.).

**Raspberry Pi e Robótica Avançada**

1. Cazemier, F. (2024). *The Comprehensive Guide to Build Raspberry Pi 5 Robotics Projects*.
2. STEM School. (2025). *Raspberry Pi 101: The Ultimate Beginner’s Guide*.
3. Li, C. (2024). *Record Weather Data with Arduino and Solar Power: Use Sensors Effectively*.

### 10.2 Webgrafia

- [Wiki Oficial da HuskyLens - DFRobot]

(<https://wiki.dfrobot.com/HUSKYLENS_V1.0_SKU_SEN0305_SEN0336>)

- [Datasheet Kendryte K210]

(<https://cdn.hackaday.io/files/1654127076987008/kendryte_datasheet_20181011163248_en.pdf>)

- [HuskyLens WIKI Document.pdf]

(<https://raw.githubusercontent.com/DFRobot/Wiki/master/SEN0305/res/HuskyLens%20WIKI%20Document.pdf>)

### 10.3 Prompts utilizados (ChatGPT/OpenAI)

- 'Cria uma estrutura de relatório para um projeto com a HuskyLens relacionado com Machine Learning.'

- 'Explica o funcionamento do processador Kendryte K210.'

- 'Compara a aprendizagem supervisionada da HuskyLens com KNN.'

- 'Sugere planos de interação entre um robô Arduino e a HuskyLens para demonstração num evento.'

- 'Cria uma versão estendida do relatório destacando IA embarcada e futuras aplicações com Raspberry Pi.'