

**FACULDADE DE CIÊNCIAS E TECNOLOGIA DA UNIVERSIDADE DOS AÇORES PÓS-GRADUAÇÃO EM PROGRAMAÇÃO, ROBÓTICA E INTELIGÊNCIA ARTIFICIAL**

**Laboratório de Aplicações em Robótica e Aprendizagem**

**Exploração, Montagem e Programação de robô carro com controlo remoto inteligente**

**Gustavo Monteiro**

**Gustavo Monteiro**

**Exploração, Montagem e Programação de robô carro com controlo remoto inteligente**

Relatório de atividade laboratorial para avaliação de disciplina Laboratório de Aplicações em Robótica e Aprendizagem sob orientação do Prof. José Cascalho.

## RESUMO

Este relatório documenta o trabalho desenvolvido no âmbito da unidade curricular de Laboratório de Aplicações em Robótica e Aprendizagem, centrado no desenvolvimento de um robô carro controlável, utilizando o kit educacional AI Machine Learning Starter Kit da Weeemake como base tecnológica. Este projeto envolveu uma abordagem integrada de hardware e software, com foco em proporcionar uma plataforma funcional que combinasse mobilidade, visão computacional e comunicação em tempo real. Desde a montagem física do *chassi*, passando pela integração de modelo de deteção facial, motores, áudio, câmara, lcd, e módulo *Wifi*, até o desenvolvimento de servidores baseados em sockets UDP utilizando as linguagens Python e MaixPy, todas as etapas foram planeadas para garantir interatividade, modularidade e confiabilidade operacional. A solução implementada permitiu o controle remoto da movimentação do robô via interface web, e o streaming de vídeo ao vivo capturado pela câmara. Além da apresentação dos resultados obtidos, o relatório discute em profundidade as principais dificuldades técnicas enfrentadas durante o processo de desenvolvimento e propõe direções viáveis para a expansão futura do sistema, com foco na inteligência artificial incorporada e robótica autónoma.

## ABSTRACT

This report documents the work developed within the curricular unit "Laboratory of Applications in Robotics and Learning", focused on the development of a controllable car robot using the educational AI Machine Learning Starter Kit from Weeemake as the technological foundation. The project adopted an integrated hardware and software approach, aiming to deliver a functional platform that combines mobility, computer vision, and real-time communication. From the physical assembly of the chassis to the integration of face detection model, motors, audio, camera, lcd and *Wifi* module, and the development of UDP-based socket servers using the Python and MaixPy languages, all stages were carefully planned to ensure interactivity, modularity, and operational reliability. The implemented solution enabled remote control of the robot's movement via a web interface, and live video streaming captured by the onboard camera. In addition to presenting the results obtained, the report thoroughly discusses the main technical challenges encountered during development and proposes viable directions for future system expansion, focusing on embedded artificial intelligence and autonomous robotics.

## INTRODUÇÃO

Este relatório tem como objetivo documentar o trabalho desenvolvido no âmbito da unidade curricular de Laboratório de Aplicações em Robótica e Aprendizagem. Para este projeto, optei por desenvolver um protótipo funcional de robô carro utilizando o kit AI Machine Learning Starter Kit, integrando sensores, atuadores e módulos de comunicação. Implementar servidores UDP para controlar remotamente a movimentação do robô, receber alertas e realizar o streaming de vídeo em tempo real. Adicionalmente, explorar as possibilidades de reconhecimento de objetos via câmara e prover uma interface web acessível para controle e visualização do sistema.

## OBJETIVO

Exploração, Montagem e Programação de robô carro com controlo remoto inteligente

## CONTEXTO

### AI Machine Learning Starter Kit (Weeemake)

### O AI Machine Learning Starter Kit da Weeemake (Figura 1) é um kit educacional voltado ao ensino e experimentação em áreas como robótica, visão computacional e inteligência artificial. Com um design modular, ele permite que estudantes e desenvolvedores explorem conceitos fundamentais de hardware e software, facilitando a criação de protótipos inteligentes e interativos. O kit destaca-se por fornecer uma combinação poderosa entre um processador neuronal (K210) e um módulo de conectividade ESP32, tornando-o ideal para aplicações de inferência de IA local, isto é, sem dependência de servidores externos, e comunicação remota via rede Wifi. A sua arquitetura permite realizar desde tarefas simples de controle de motores até implementações mais avançadas como reconhecimento de objetos, streaming de vídeo, alerta sonoro e deteção de obstáculos em tempo real. O foco educacional do kit é complementado por uma série de ferramentas de desenvolvimento compatíveis, incluindo Weecode, um ambiente gráfico para programação por blocos, e MaixPy, uma linguagem semelhante ao MicroPython com extensões específicas para o hardware utilizado. Esta configuração possibilita uma transição suave entre níveis de complexidade, atendendo tanto a iniciantes quanto a utilizadores avançados.

### Na figura 1, podemos observar os diferentes componentes do kit. Na esquerda, há vários sacos transparentes contendo parafusos, porcas e espaçadores organizados por tipo. Acima deles, estão ferramentas de montagem como uma chave Allen, uma chave de fenda pequena e uma chave de boca dupla. Um trilho metálico laranja perfurado para suportar as rodas e suportes metálicos também laranjas estão posicionados ao centro que servem como base. No canto superior esquerdo, há quatro motores amarelos do tipo DC com fios conectados a terminais rápidos. Ao lado, encontram-se quatro rodas mecanum com roletes inclinados e quatro rodas comuns pretas com o centro amarelo. No canto superior direito, está a placa-mãe ELF AIOT K210 protegida por uma estrutura de acrílico transparente, acompanhada de uma fonte de alimentação e um cabo USB tipo C. Próxima à placa, está a bateria de lítio 18650 de 6-12V, que fornece energia ao sistema. O conjunto é organizado e voltado para a montagem de um robô móvel com capacidades de inteligência artificial e visão computacional.



*Figura 1*

### Especificações Técnicas Relevantes

### O microcontrolador tem as seguintes características:

### Processamento:

### Kendryte K210: processador dual-core RISC-V a 64 bits com frequência de até 600 MHz, otimizado para tarefas de inferência em IA.

### ESP32: microcontrolador auxiliar com suporte a Wi‑Fi 2.4 GHz e Bluetooth, utilizado para comunicação em rede.

### Memória e Armazenamento:

### 8 MB de SRAM interna para processamento rápido de imagens e dados de sensores.

### Slot para cartão microSD de até 32 GB.

### 8 MB de memória flash onboard para scripts, bibliotecas e modelos.

### Visão Computacional:

### Câmera embutida de 2 MP com lente de 180°, capaz de capturar imagens em QVGA (320×240) a 30 FPS.

### Suporte à execução local de modelos como YOLOv2, MobileNet, TinyYOLOv3, e reconhecimento facial com aceleração via KPU (Kendryte Processing Unit).

### Periféricos e Conectividade:

### 4× portas RJ11 para sensores modulares.

### 4× portas DC motor independentes.

### 20× GPIOs (entradas/saídas digitais) e 2× canais I2C.

### 1× USB Type-C para alimentação, upload e comunicação serial.

### Wi‑Fi 802.11 b/g/n (2.4 GHz).

### Componentes adicionais:

### Tela LCD de 2.4" com suporte a exibição de imagens capturadas pela câmara ou mensagens do sistema.

### Microfone embutido e alto-falante com suporte à reprodução de ficheiros .wav.

### Botões físicos, chave de tensão (3.3 V/5 V) e botão de reset.

### APU (Audio Processing Unit) e FFT Accelerator dedicados a tarefas de som e análise de sinais.

### Desta forma, com estas especificações é possível realizar rastreio de blocos de cor (Color Block Tracking) em que robô segue objetos coloridos com base em deteção visual, reconhecimento de tags (Tag Recognition) em que o robô utiliza e reconhece marcas visuais (como QR codes ou tags coloridas) para tomar decisões, reconhecimento facial e rastrreamento: O robô pode detetar rostos humanos e ajustar a sua direção para segui-los, reconhecimento de ibjetos (Object Recognition) em que o robô classifica objetos usando modelos incorporados ou reconhecimento de voz: Implementação de comandos personalizados com capacidade de auto aprendizagem.

### Repositório do Projeto

### O [código-fonte](https://github.com/GustavoMonteiro02/Robotics_Project) do projeto está disponível publicamente. O repositório inclui todas as versões principais do código do robô e do servidor Flask, além dos recursos multimédia e modelos de inteligência artificial que foram utilizados para melhorar a interatividade com o utilizador e explorar as capacidades do kit. Os ficheiros refletem a evolução incremental do projeto e as tentativas de otimização durante o processo de desenvolvimento.

### Tabela: Evolução do Projeto e Limitações Encontradas

| **Etapa do Projeto** | **Funcionalidade Implementada** | **Limitações / Desafios Enfrentados** |
| --- | --- | --- |
| **1. Montagem Física** | Montagem do chassis, ligação dos motores, câmara e ecrã | Organização dos cabos para não interferirem com o movimento; necessidade de testes manuais por fase |
| **2. Conectividade Wi‑Fi** | Ligação do ESP32 à rede e obtenção de endereço IP | Wi‑Fi instável; necessidade de reinicializações frequentes; apenas compatível com redes 2.4 GHz |
| **3. Controlo via UDP** | Receção de comandos de direção e velocidade por socket UDP | Atrasos ocasionais na resposta; perdas de pacotes devido à natureza não fiável do protocolo UDP |
| **4. Movimento do Robô** | Implementação de comandos de movimentação (frente, trás, parar, etc.) | Dificuldade em manter trajetória reta; ajuste manual dos motores necessário |
| **5. Reprodução de Áudio** | Reprodução de áudios de sistema (ligar/desligar) e comandos vocais | Áudios de movimento removidos por bloquearem a execução do código durante a reprodução e copiados para a memória flash para que pudessem ser lidos |
| **6. Transmissão de Vídeo (UDP)** | Captura, compressão e envio de imagem JPEG para visualização remota | Fragmentação manual dos pacotes; perda de quadros em caso de congestionamento de rede |
| **7. Interface Web (Flask)** | Página HTML com botões de controlo, barra de velocidade e transmissão de imagem | Threads do Flask exigem arranque correto; atraso inicial na receção de vídeo |
| **8. Alertas Inteligentes** | Envio de mensagens contextuais (ex: “obstáculo à frente”) | Atrasos devido à concorrência entre threads de alertas e comandos |
| **9. Deteção de Objectos (IA)** | Utilização de modelo incorporado para identificar objectos e reagir a obstáculos | O ficheiro .kmodel não conseguia ser lido; execução bloqueia outras tarefas; |
| **10. Optimização e Estabilização** | Refactoring do código, libertação de memória e threads | Limitações do hardware; necessidade de reinicializações; ausência de multitarefa verdadeira |
| **11.Reconhecimento Facial** | Ajustar a direção do robô para mantero rosto encontrado centrado no lcd | Apesar de eficaz em condições de boa iluminação, a funcionalidade apresenta limitações, nomeadamente a sensibilidade à luz, a impossibilidade de executar outras tarefas em simultâneo |

## DESENVOLVIMENTO E RESULTADOS

## O desenvolvimento do robô carro foi realizado em etapas iterativas, focando inicialmente na montagem e teste de componentes individuais, seguido pela integração das funcionalidades principais: movimentação, visão computacional, câmara, lcd, audio comunicação remota e interface de controlo. Esta seção descreve cada uma dessas funcionalidades, os métodos de implementação adotados e as dificuldades encontradas ao longo do processo.

### Montagem do Robô

## A montagem física do robô foi realizada com base na estrutura modular fornecida pelo kit, composta por peças em acrílico, motores, suportes e conexões RJ11. O chassi principal foi montado para acomodar os quatro motores DC, a placa principal K210/ESP32, a câmara frontal e o display LCD (mostrado na figura 2).

## A alimentação foi realizada através de uma bateria ou cabo USB Type-C conectado a uma fonte externa ou computador, e a câmara e do display são embutidos na placa.

## Embora o processo de montagem tenha sido relativamente direto, foi necessário cuidado na disposição dos fios para evitar interferências com o movimento das rodas. A integração final exigiu testes individuais de cada componente antes da programação completa.

## Uma imagem com roda, Veículo de brincar, Peça de automóvel, pneu Os conteúdos gerados por IA podem estar incorretos.

*Figura 2*

### Movimentação e Controlo de Direção

### A movimentação do robô foi implementada por meio de quatro motores DC conectados às portas modulares da mainboard. Cada motor foi controlado individualmente via chamadas ao objeto WeDCMotor da biblioteca Weeemake, permitindo realizar movimentos para frente, trás, esquerda/direita, diagonais e parar como demonstrado na figura 3

### Uma imagem com diagrama, texto, captura de ecrã, design Os conteúdos gerados por IA podem estar incorretos.

*Figura 3*

### Comandos: Os comandos eram recebidos via UDP no formato COMMAND:X, onde X representava o código do movimento (eg: 8 = frente, 2 = trás, 4 = esquerda).

### Limitações encontradas: A coordenação entre os motores exigiu ajustes manuais em cada eixo, pois o robô apresentava tendência a desviar para um lado. Além disso, a resposta aos comandos dependia da estabilidade da conexão *Wifi*, o que gerava latência em ambientes com interferência.

### Ajuste de Velocidade Dinâmica

### A velocidade dos motores podia ser ajustada remotamente com o envio do comando SPEED:Y, onde Y representava um valor de 10 a 200. O sistema interpretava esse valor e atualizava a intensidade dos motores dinamicamente.

### Desafio enfrentado: O uso de sockets UDP exigiu controle de concorrência para evitar conflitos entre pacotes de velocidade e direção recebidos simultaneamente.

### Deteção de Obstáculos com Visão Computacional

### O kit inclui uma câmara de 2MP embutida no K210, capaz de capturar imagens e processá-las em tempo real. Um modelo pré-treinado de deteção de objetos (20class.kmodel) foi utilizado para identificar a presença de obstáculos em três zonas da imagem: frente, esquerda e direita.

### Função de segurança: Se o robô detetasse um objeto à sua frente ou na direção do movimento solicitado, ele não executava o comando e gerava um alerta.

### Limitação: O sistema não permitia a execução do modelo em simultâneo com a captura de imagem e o envio de vídeo, exigindo a priorização entre funcionalidades e o modelo apenas detetava certos objetos tendo sido retirado posteriormente.

### Streaming de Vídeo em Tempo Real

### A imagem capturada pela câmara é comprimida com qualidade ajustável (e.g. JPEG, QVGA) e fragmentada em pacotes de até 1024 bytes, enviados via UDP para um servidor Flask. O servidor reconstrói os frames e exibe o vídeo numa interface web com atualizações contínuas.

### Desafios enfrentados:

### O protocolo UDP não garante a ordem ou entrega dos pacotes, causando travamentos ou perda de frames quando a rede estava congestionada.

### Uma vez que apenas um socket de envio podia operar de forma confiável, tentar enviar imagens e alertas simultaneamente por sockets distintos causava falhas intermitentes.

### Interface Web para Controlo e Monitorização

### Foi desenvolvida uma página HTML com Flask para permitir o controlo remoto do robô via navegador (figura 4). A interface incluía botões para movimentos (figura 5), slider para controle de velocidade e transmissão vídeo em tempo real.

### Facilidade de uso: A interface era responsiva, compatível com dispositivos móveis e não exigia instalação de software adicional.

### Problema identificado: Em alguns casos, o servidor Flask demorava a iniciar as threads de receção de vídeo e alertas, exigindo reinicializações manuais.

### Uploaded image

*Figura 4*

|  |  |
| --- | --- |
| Botão de Input | Ação |
| ↑ | Quando o botão de input ↑ (seta para cima) é pressionado, o robô avança, com velocidade positiva nos motores traseiros (dc1 e dc4) e velocidade negativa nos motores dianteiros (dc3 e dc2), fazendo com que todas as rodas girem no sentido de avanço. |
| ↓ | Quando o botão ↓ (seta para baixo) é pressionado, o robô recua, com velocidade negativa nos motores traseiros (dc1 e dc4) e velocidade positiva nos motores dianteiros (dc3 e dc2), movimentando todas as rodas no sentido inverso ao avanço. |
| ← | Ao pressionar a seta ← (esquerda), o robô roda para a esquerda sobre o próprio eixo. Isso é feito com os motores do lado esquerdo (dc4 e dc3) girando em sentido contrário (positivo em dc4 e positivo em dc3) aos motores do lado direito (dc1 e dc2), que giram com velocidade negativa, causando rotação em sentido anti-horário. |
| → | Quando a seta → (direita) é pressionada, o robô roda para a direita sobre o próprio eixo, com os motores do lado esquerdo (dc4 e dc3) girando para a frente (velocidade positiva em dc1 e negativa em dc3) e os motores do lado direito (dc1 e dc2) girando para trás (velocidade negativa em dc4 e positiva em dc2), provocando rotação no sentido horário. |
| ↖ | Quando o botão **↖** (diagonal para frente e esquerda) é pressionado, o robô desloca-se para a frente e para a esquerda. Isso é feito com dc1 (traseiro direito) e dc3 (dianteiro esquerdo) parados, enquanto dc2 (dianteiro direito) gira para a frente com velocidade positiva e dc4 (traseiro esquerdo) gira para frente com velocidade negativa. |
| ↗ | Quando o botão **↗** (diagonal para frente e direita) é pressionado, o robô desloca-se para a frente e para a direita. Neste movimento, dc1 (traseiro direito) gira para frente com velocidade positiva e dc3 (dianteiro esquerdo) gira para frente com velocidade negativa, enquanto dc2 (dianteiro direito) e dc4 (traseiro esquerdo) permanecem parados. |
| **↙** | Quando o botão **↙** (diagonal para trás e esquerda) é pressionado, o robô desloca-se para trás e para a esquerda. Isso é feito com dc1 (traseiro direito) girando para trás com velocidade negativa e dc3 (dianteiro esquerdo) girando para trás com velocidade positiva, enquanto dc2 e dc4 ficam parados. |
| ↘ | Quando o botão **↘** (diagonal para trás e direita) é pressionado, o robô desloca-se para trás e para a direita. Para isso, dc2 (dianteiro direito) gira para trás com velocidade positiva e dc4 (traseiro esquerdo) gira para trás com velocidade negativa, enquanto dc1 e dc3 permanecem parados. |
| o | Quando o botão "o" é pressionado, o robô para imediatamente, com todos os motores (dc1, dc4, dc3 e dc2) a receberem velocidade nula, fazendo com que as rodas deixem de se movimentar. |
| TURN OFF | Quando o botão TURN OFF é pressionado, o robô também para com velocidade nula em todos os motores, tal como no botão “o”, mas além disso executa o procedimento de desligamento do sistema. |

*Tabela 1*

### Uma imagem com texto, captura de ecrã, Tipo de letra, file Os conteúdos gerados por IA podem estar incorretos.

*Figura 5*

### Reprodução de Áudio e Feedback Local

### Para indicar o estado do sistema, ficheiros de áudio como "turnon.wav" e "turnoff.wav" eram executados pelo módulo APU na inicialização e no desligar do robô. Após configurar corretamente os pinos do I2S os áudios foram copiados manualmente da memória SD para a memória flash.

### Limitação observada: O áudio só funcionava após copiados da memória SD para a memória flash. Áudios de movimento removidos por bloquearem a execução do código durante a reprodução e copiados para a memória flash

### Controlo do Robô através do reconhecimento facial

### O controlo do robô por reconhecimento facial foi implementado através de um modelo de deteção de rostos (face.kmodel, treinado pela weeemake) executado localmente no processador K210. A câmara capta imagens em tempo real, e sempre que um rosto é detetado, o robô ajusta a sua direção para manter esse rosto centrado no campo de visão. Se o rosto estiver à esquerda ou à direita da imagem, o robô gira respetivamente nessa direção; se estiver centrado, avança; e se estiver demasiado próximo, recua. Esta lógica permite ao robô seguir um rosto de forma reativa e autónoma.

### Limitação observada: Apesar de eficaz em condições de boa iluminação, a funcionalidade apresenta limitações, nomeadamente a sensibilidade à luz, a impossibilidade de executar outras tarefas em simultâneo.

### 

## CONCLUSÃO

O desenvolvimento deste projeto proporcionou uma experiência prática valiosa na montagem e programação de um robô carro físico, aprofundando o meu conhecimento em sistemas incorporados, redes de comunicação e visão computacional. Através da integração entre hardware e software, foi possível compreender os desafios reais da robótica, como a gestão de recursos limitados, instabilidades de rede e restrições no processamento de imagem em tempo real.

Ao longo do processo, exploraram-se diferentes estratégias de controlo remoto e resposta a estímulos do ambiente, recorrendo tanto a sensores físicos quanto à análise de imagens via câmara através de modelos de inteligência artificial. A implementação de uma interface web para controlo e visualização do robô demonstrou a importância da usabilidade e da comunicação eficiente entre sistemas heterogéneos e o controlo do robô através do reconhecimento facial demonstrou a capacidade dos modelos de inteligência artificial e os seus possíveis usos na robótica. Um dos maiores desafios enfrentados foi garantir a estabilidade da transmissão de vídeo via UDP, especialmente diante das limitações de largura de banda e fragmentação de pacotes.

Como próximos passos, seria interessante aprofundar o uso de redes neuronais incorporadas na placa para a deteção de objetos, experimentar protocolos mais robustos como WebRTC para streaming e testar o comportamento do robô em ambientes mais complexos e dinâmicos. Além disso, há espaço para otimizar o consumo de energia e explorar a integração com plataformas como o ROS, ampliando as possibilidades de colaboração entre múltiplos robôs em rede.

## BIBLIOGRAFIA

1. <https://weeemake.notion.site/AI-Machine-Learning-Starter-Kit-d98cc47e6d4544b68ab03a6cc48987e3>
2. <https://pt.weeemake.com/product/ai-machine-learning-starter-kit-ai-education.html>
3. <https://youtu.be/jwFJEjryWy8?si=mqku9px_QnU0eiDj>
4. [Welcome to Flask — Flask Documentation (3.1.x)](https://flask.palletsprojects.com/en/stable/)
5. [K210/model at master · LZBUAV/K210](https://github.com/LZBUAV/K210/tree/master/model)