**📡 Descrição do Código – wifi\_webserver\_leds\_simulacao.ino**

Este código implementa um sistema de **controlo remoto por Wi-Fi** para o Arduino UNO R4 WiFi, com **simulação do estado e movimento de um robô** através de 4 LEDs.

É possível **ligar/desligar o sistema e observar visualmente os estados** do robô (ON, Parar, Motor ESQ, Motor DIR) diretamente a partir de um browser, usando uma interface web mínima.

**🔁 Funcionamento passo-a-passo**

**🔹 1. Inicialização (setup())**

* Inicia comunicação com o Serial Monitor.
* Define os pinos D2, D4, D5 e D6 como saídas digitais para os LEDs:
  + D2 → LED VERDE → sistema ON
  + D4 → LED VERMELHO → modo PARAR
  + D5 → LED AMARELO → motor ESQUERDO
  + D6 → LED AZUL → motor DIREITO
* Apaga todos os LEDs ao início.
* Tenta ligar-se à rede Wi-Fi com as credenciais fornecidas.
* Quando a ligação for bem-sucedida, imprime o IP local no Serial Monitor.
* Inicia o Web Server na porta 80 (HTTP).

**🔹 2. Loop principal (loop())**

* Espera por um cliente (navegador ou app).
* Lê o pedido HTTP (/start, /stop, /status) enviado pelo browser.
* Interpreta o pedido e atualiza o estado do sistema.

**🔹 3. Comandos disponíveis**

| **Comando HTTP recebido** | **Ação executada** |
| --- | --- |
| /start | Liga LEDs Verde, Amarelo e Azul (simula robô em movimento). |
| /stop | Liga apenas o LED Vermelho (simula paragem total). |
| /status | Apenas mostra o estado atual no browser (sem alterar LEDs). |

**🔹 4. Interface web gerada**

* O Arduino responde com uma página HTML simples.
* Mostra o título do projeto, estado atual (ATIVO/INATIVO) e três botões:
  + ▶️ Start → envia /start
  + ⛔ Stop → envia /stop
  + ℹ️ Status → envia /status

**🔹 5. Fecho da ligação**

* Após enviar a página HTML, o Arduino fecha corretamente a ligação com client.stop();.
* Isto evita duplicações da interface no browser (problema comum quando client.stop() está ausente).

**🔦 Comportamento visual dos LEDs**

| **Estado do sistema** | **Verde (ON)** | **Vermelho (PARAR)** | **Amarelo (ESQ)** | **Azul (DIR)** |
| --- | --- | --- | --- | --- |
| Inicial | OFF | OFF | OFF | OFF |
| Após /start | ON | OFF | ON | ON |
| Após /stop | OFF | ON | OFF | OFF |

Uma imagem com texto, captura de ecrã, Tipo de letra, logótipo

Os conteúdos gerados por IA podem estar incorretos.

Uma imagem com texto, captura de ecrã, Tipo de letra, logótipo

Os conteúdos gerados por IA podem estar incorretos.

Uma imagem com interior, eletrónica, Engenharia eletrónica, vela

Os conteúdos gerados por IA podem estar incorretos.

**📘 Resumo técnico**

* Comunicação via Web Server HTTP (biblioteca WiFiS3.h)
* Interface simples via browser (sem app externa)
* LEDs simulam o estado e os motores do robô
* Ideal para testes de bancada e demonstrações pedagógicas
* Base futura para integrar sensores, motores reais ou lógica automática

Uma imagem com texto, captura de ecrã, software, Página web

Os conteúdos gerados por IA podem estar incorretos.

💡 **Este código pode ser expandido facilmente para incluir leitura da HuskyLens, controlo de motores reais com L298P, ou lógica baseada em sensores.**

**📷 Descrição do Código – huskylens\_uart\_read\_id.ino**

Este código estabelece a comunicação UART entre o **Arduino UNO R4 WiFi** e a **câmara HuskyLens**, utilizando a biblioteca oficial HUSKYLENS.h em conjunto com SoftwareSerial.

O objetivo é verificar a ligação física e a receção de dados da câmara via UART, imprimindo no **Serial Monitor** os dados dos objetos detetados: o **ID** e as **coordenadas do centro** (X/Y). Este é um passo essencial para validar a integração da visão computacional no robô **AzorCAR**.

**🔁 Funcionamento passo-a-passo**

**1. Inicialização (setup())**

* A comunicação com o monitor série é iniciada (Serial.begin(9600)).
* Define-se uma segunda porta série (em D8 e D9) com SoftwareSerial.
* Cria-se a instância da HuskyLens ligada a essa porta.
* O código tenta estabelecer ligação com a HuskyLens:
  + Se tiver sucesso, imprime: **"HuskyLens ligada com sucesso"**.
  + Caso contrário, imprime uma mensagem de erro.

**2. Loop principal (loop())**

* Em cada ciclo:
  + O Arduino envia um pedido de dados (huskylens.request()).
  + Se houver deteção:
    - Lê os dados disponíveis (huskylens.read()).
    - Imprime:
      * **ID do objeto/cor**
      * **X e Y do centro do objeto**
  + Se nada for detetado, imprime: **"Nada detetado."**
  + Se falhar a comunicação, imprime: **"Falha na comunicação com a HuskyLens."**

**🔌 Ligações físicas Arduino ↔ HuskyLens**

* **TX da HuskyLens** → **D8 do Arduino** (entrada RX via SoftwareSerial)
* **RX da HuskyLens** ← **D9 do Arduino** (saída TX)
* **GND da HuskyLens** → **GND do Arduino**
* **VCC**: **não ligado** ao Arduino, pois a câmara está a ser alimentada externamente via **powerbank USB**

⚠️ Nota: A ligação de **GND comum é obrigatória**, mesmo quando a alimentação da HuskyLens é feita por fonte externa.

**📋 Modo de operação da HuskyLens**

Este código é compatível com qualquer modo da HuskyLens que forneça um ID visível e posição, nomeadamente:

* Reconhecimento de cor (Color Recognition)
* Reconhecimento de objetos (Object Recognition)
* Leitura de tags (Tag Recognition)

Antes de iniciar o teste, deve-se selecionar um destes modos e garantir que o modelo foi previamente treinado e guardado.

**📘 Resumo técnico**

* Comunicação série via SoftwareSerial nos pinos D8 e D9
* Leitura de objetos com a biblioteca oficial HUSKYLENS.h
* Visualização dos dados no monitor série do Arduino IDE
* Funciona como base para controlar LEDs, motores ou decisões lógicas com base na visão
* Encaixa perfeitamente na arquitetura modular do AzorCAR

Uma imagem com texto, captura de ecrã, número, software

Os conteúdos gerados por IA podem estar incorretos.

Uma imagem com texto, interior, brinquedo, desenho

Os conteúdos gerados por IA podem estar incorretos.

Uma imagem com texto, computador, interior, brinquedo

Os conteúdos gerados por IA podem estar incorretos.

**🎯 Descrição do Código – huskylens\_led\_decision.ino – ETAPA 2**

Este código faz parte do projeto **AzorCAR** e implementa a lógica de reação automática do **Arduino UNO R4 WiFi** com base nos dados recebidos da câmara **HuskyLens**.

Através de comunicação UART (via SoftwareSerial), o Arduino lê o **ID da cor detetada** pela HuskyLens (modo “Color Recognition”) e acende **LEDs simuladores** de ações robóticas como: virar à esquerda, virar à direita e parar.

**🔁 Funcionamento geral**

**🟢 Inicialização**

* Define os pinos dos LEDs como saídas:
  + D2 → LED VERDE (sistema ativo)
  + D4 → LED VERMELHO (parar)
  + D5 → LED AMARELO (virar à esquerda)
  + D6 → LED AZUL (virar à direita)
* Liga o LED verde como indicador de sistema ligado.
* Estabelece a comunicação UART com a HuskyLens (baud rate: 9600).
* Imprime no Serial Monitor o estado da ligação.

**🔄 Loop principal**

* A cada 500 milissegundos:
  + O Arduino solicita um novo frame à HuskyLens.
  + Se for detetado um objeto com ID:
    - Apaga os LEDs anteriores.
    - Acende o LED correspondente à ação simulada:
      * ID 1 → amarelo → virar à esquerda
      * ID 2 → azul → virar à direita
      * ID 3 → vermelho → parar
  + Se não houver deteção:
    - Apaga todos os LEDs, exceto o LED verde (sistema pronto).

**📋 Mapeamento de ações**

| **Cor reconhecida pela HuskyLens** | **ID atribuído** | **Ação simulada** | **LED aceso** |
| --- | --- | --- | --- |
| Azul | 1 | Virar à esquerda | Amarelo (D5) |
| Verde | 2 | Virar à direita | Azul (D6) |
| Vermelho | 3 | Parar | Vermelho (D4) |

Nota: Os IDs dependem do treino da HuskyLens e podem variar. O utilizador deve confirmar no ecrã da câmara os IDs atribuídos a cada cor.

**🔌 Ligações físicas**

**HuskyLens**

* TX → D8 do Arduino (RX via SoftwareSerial)
* RX ← D9 do Arduino (TX via SoftwareSerial)
* GND → GND do Arduino
* VCC → *Não ligado ao Arduino* (a câmara está alimentada por powerbank USB)

**LEDs**

* D2 → LED verde (sistema ativo)
* D4 → LED vermelho (parar)
* D5 → LED amarelo (esquerda)
* D6 → LED azul (direita)
* Cada LED está ligado com uma resistência de ~220 ohms

**📘 Resumo técnico**

* Comunicação serial UART com a HuskyLens usando SoftwareSerial (pinos D8 e D9)
* Leitura de dados com as funções huskylens.request() e huskylens.read()
* Reação imediata com digitalWrite() consoante o ID lido
* Delay de 500 ms entre cada leitura
* Uso da biblioteca oficial HUSKYLENS.h da DFRobot

**🎓 Aplicação pedagógica**

Este código é indicado para simulações e introdução ao controlo reativo baseado em visão. Permite:

* Integrar sensores inteligentes com Arduino
* Praticar comunicação entre módulos via UART
* Visualizar comportamentos reativos de um robô
* Preparar a lógica de decisão antes de controlar motores reais

Uma imagem com texto, interior, Dispositivo eletrónico, multimédia

Os conteúdos gerados por IA podem estar incorretos.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Os conteúdos gerados por IA podem estar incorretos.

Uma imagem com texto, interior, captura de ecrã, computador

Os conteúdos gerados por IA podem estar incorretos.

Uma imagem com eletrónica, texto, Dispositivo eletrónico, computador

Os conteúdos gerados por IA podem estar incorretos.

**📄 AzorCAR –Webserver com HuskyLens e LEDs Etapa - 3 FINAL**

Este código implementa uma solução de **controlo remoto por Wi-Fi com visão artificial HuskyLens**, baseada no **Arduino UNO R4 WiFi**, com simulação visual por LEDs. A arquitetura permite o controlo e a leitura do robô **via browser**, ao mesmo tempo que processa os dados de reconhecimento da HuskyLens via **UART (SoftwareSerial)**.

**🔁 Funcionamento geral**

1. **Inicialização (setup()):**
   * Configura os pinos dos LEDs (D2, D4, D5, D6).
   * Tenta conectar à rede Wi-Fi (WiFi.begin()).
   * Mostra o IP atribuído, ou avisa se estiver em modo offline.
   * Inicia a comunicação com a HuskyLens a 9600 bps.
   * Por omissão, **o modo de visão (modoVisaoAtivo) é iniciado como false** — ou seja, precisa de ser ativado via browser (exceto se o programador decidir forçá-lo manualmente).
2. **Servidor Web HTTP:**
   * Funciona na porta 80 (browser).
   * Permite três comandos:
     + /start → ativa a visão (LED verde ligado).
     + /stop → desativa a visão (LED vermelho ligado).
     + /status → mostra o estado atual sem alterar nada.
3. **Modo de visão ativo:**
   * Quando o modoVisaoAtivo == true, o Arduino tenta ler continuamente os dados da HuskyLens.
   * O ID detetado é interpretado assim:
     + ID 1 → virar à **esquerda** → LED **amarelo**
     + ID 2 → virar à **direita** → LED **azul**
     + ID 3 → **parar** → LED **vermelho**
   * Se não for detetado nenhum ID válido, apenas o LED **verde** fica ligado (sistema ativo).
4. **Modo offline (sem Wi-Fi):**
   * O sistema mostra no monitor:

Uma imagem com texto, captura de ecrã, Tipo de letra, branco

Os conteúdos gerados por IA podem estar incorretos.

* + Neste caso, **o modo de visão permanece inativo** por padrão (modoVisaoAtivo = false), obrigando o utilizador a decidir se quer forçar o início da leitura automática.

**🔦 LEDs – Simulação visual**

| **LED (pino)** | **Cor** | **Função simulada** |
| --- | --- | --- |
| D2 | Verde | Sistema ativo (ligado) |
| D4 | Vermelho | Paragem total |
| D5 | Amarelo | Motor esquerdo (virar ESQ) |
| D6 | Azul | Motor direito (virar DIR) |

**📘 Notas técnicas**

* A **comunicação com a HuskyLens** usa a biblioteca HUSKYLENS.h e SoftwareSerial nos pinos D8 (RX) e D9 (TX).
* O baud rate da câmara deve estar ajustado a **9600 bps** (configurado via menu da HuskyLens).
* A rede Wi-Fi pode ser doméstica ou hotspot de telemóvel.
* Quando sem rede, o sistema **funciona parcialmente offline** (visão desligada).

💡 Este código serve como **ponte entre a visão artificial e o controlo remoto básico**, podendo evoluir para comandos de motores reais com L298P e integração com ROS ou automação por IA.

Uma imagem com captura de ecrã, Retângulo, quadrado, texto

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

Os conteúdos gerados por IA podem estar incorretos.

Uma imagem com texto, computador portátil, interior, captura de ecrã

Os conteúdos gerados por IA podem estar incorretos.

Uma imagem com eletrónica, computador portátil, texto, computador

Os conteúdos gerados por IA podem estar incorretos.

Uma imagem com texto, captura de ecrã, software, Tipo de letra

Os conteúdos gerados por IA podem estar incorretos.