

# **DESENVOLVIMENTO DE PROTÓTIPO DE SEMÁFORO INTELIGENTE COM VISÃO COMPUTACIONAL E TINYML PARA OTIMIZAÇÃO ADAPTATIVA DO FLUXO DE VEÍCULOS**

**Autores:** Arthur Feitosa Nogueira, Gustavo Alves, Jhonatan Ricardo, Pietro Augusto e Samuel Deak Luiz

## **1. INTRODUÇÃO**

O controle de tráfego em ambientes urbanos permanece um desafio crítico de mobilidade e sustentabilidade. Sistemas semafóricos convencionais, baseados em temporização fixa, geram ineficiências significativas, como paradas desnecessárias em vias de baixo fluxo. Visando superar a rigidez desses modelos, este relatório técnico detalha a **Versão 2.0** de um semáforo inteligente, que evolui do sistema baseado em sensores infravermelhos (IR) para uma solução de **Visão Computacional** alimentada por Inteligência Artificial (IA) embarcada (TinyML).

O projeto utiliza um microcontrolador **ESP32-CAM** para realizar a detecção e classificação de veículos em tempo real. O objetivo é otimizar dinamicamente o fluxo em um cruzamento, priorizando uma via principal (Avenida Principal) e garantindo a abertura da via secundária (Rua Secundária) somente após a **detecção visual e confirmação de presença de um veículo**.

A principal inovação desta versão é a substituição da detecção por sensores de barreira pela análise de imagem. Essa abordagem elimina a necessidade de infraestrutura física intrusiva na via, como sensores indutivos ou de infravermelho instalados no pavimento ou em postes, necessitando apenas da câmera visual para monitoramento e tomada de decisão. O sistema foi implementado e validado em uma maquete, simulando um ambiente urbano em escala.

## **1. INTRODUCTION**

Traffic control in urban environments remains a critical challenge for mobility and sustainability. Conventional traffic light systems, based on fixed timing, generate significant inefficiencies, such as unnecessary stops on low-flow roads. Aiming to overcome the rigidity of these models, this technical report details **Version 2.0** of a smart traffic light, which evolves from the system based on infrared (IR) sensors to a **Computer Vision** solution powered by embedded Artificial Intelligence (AI) (TinyML).

The project uses an **ESP32-CAM** microcontroller to perform real-time vehicle detection and classification. The objective is to dynamically optimize the flow at an intersection, prioritizing a main road (Main Avenue) and ensuring the opening of the secondary road (Secondary Street) only after **visual detection and confirmation of a vehicle's presence**.

The main innovation of this version is the replacement of barrier sensor detection with image analysis. This approach eliminates the need for intrusive physical infrastructure on the road, such as inductive or infrared sensors installed on the pavement or poles, requiring only the

visual camera for monitoring and decision-making. The system was implemented and validated on a mock-up, simulating an urban environment to scale.

## 2. JUSTIFICATIVA (FOCO EM IA)

A transição para a Visão Computacional por meio da plataforma Edge Impulse é justificada pela busca por maior robustez, flexibilidade e capacidade de escalabilidade do sistema.

Enquanto o sensor IR fornece apenas uma detecção binária (presença/ausência de obstáculo em um ponto fixo), a IA embarcada no ESP32-CAM permite:

- **Detectão Não Invasiva:** Não requer contato físico com o ambiente da via, simplificando a instalação e manutenção.
- **Aumento de Precisão:** O modelo de Machine Learning pode ser treinado para classificar especificamente "carro", "moto" ou "caminhão", ignorando pedestres, animais ou detritos, o que não é possível com sensores IR passivos.
- **Inferência na Borda (Edge Computing):** O processamento da imagem é realizado diretamente no microcontrolador (TinyML), garantindo baixíssima latência na tomada de decisão do semáforo, sem depender de conectividade constante com a nuvem, o que é crucial para sistemas críticos de tráfego.

A IA se torna o cerne da inteligência do semáforo, permitindo a contagem e, futuramente, a medição da densidade de veículos para ajustes dinâmicos e mais refinados nos tempos de sinal.

## 3. MATERIAIS E MÉTODOS (HARDWARE E EDGE IMPULSE)

### 3.1. Hardware e Componentes

A unidade de processamento central (UCP) foi substituída pelo módulo **ESP32-CAM (ESP-32S com câmera OV2640)**, devido à sua capacidade de processamento de imagem, conectividade Wi-Fi e baixo custo.

#### Componentes Utilizados:

- **1x Módulo ESP32-CAM:** CPU e plataforma de Visão Computacional.
- **1x Módulo FTDI ou ESP32-CAM-MB:** Necessário para a programação serial do ESP32-CAM.
- **6x LEDs (2 VERMELHOS, 2 AMARELOS E 2 VERDES):** Para a sinalização do tráfego.
- **6x Resistores 220R:** Para segurança dos LEDs
- **1x Protoboard 400 Pontos:** Para montagem e prototipagem do circuito.
- **Jumpers:** Para conexões.

### Conexões e Integração ao Sistema:

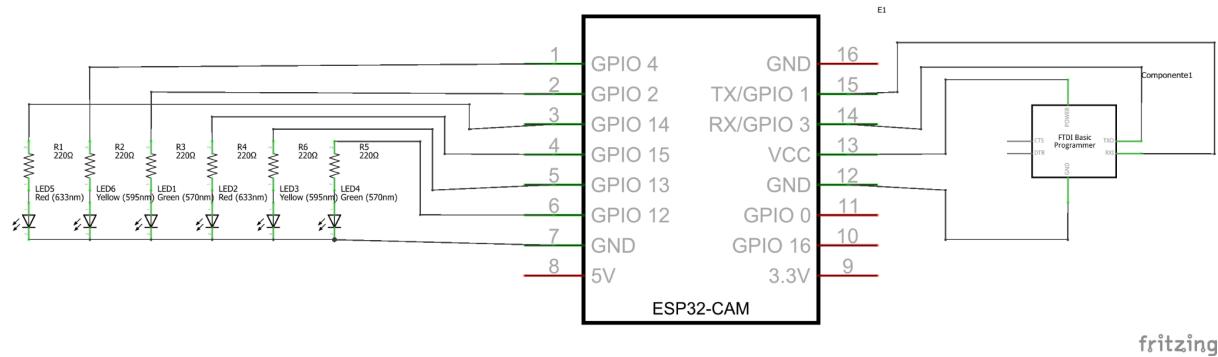
O ESP32-CAM, por operar em lógica de 3.3V, requer atenção especial nas conexões. Os LEDs de cada semáforo são conectados aos pinos de **GPIO (General Purpose Input/Output)** do ESP32, que atuam como saídas digitais para acionamento.

Semáforo	Cor do LED	Pino GPIO Sugerido (Saída)
Principal (1) ▾	Verde ▾	GPIO 2
Principal (1) ▾	Amarelo ▾	GPIO 4
Principal (1) ▾	Vermelho ▾	GPIO 14
Secundário (2) ▾	Verde ▾	GPIO 12
Secundário (2) ▾	Amarelo ▾	GPIO 13
Secundário (2) ▾	Vermelho ▾	GPIO 15

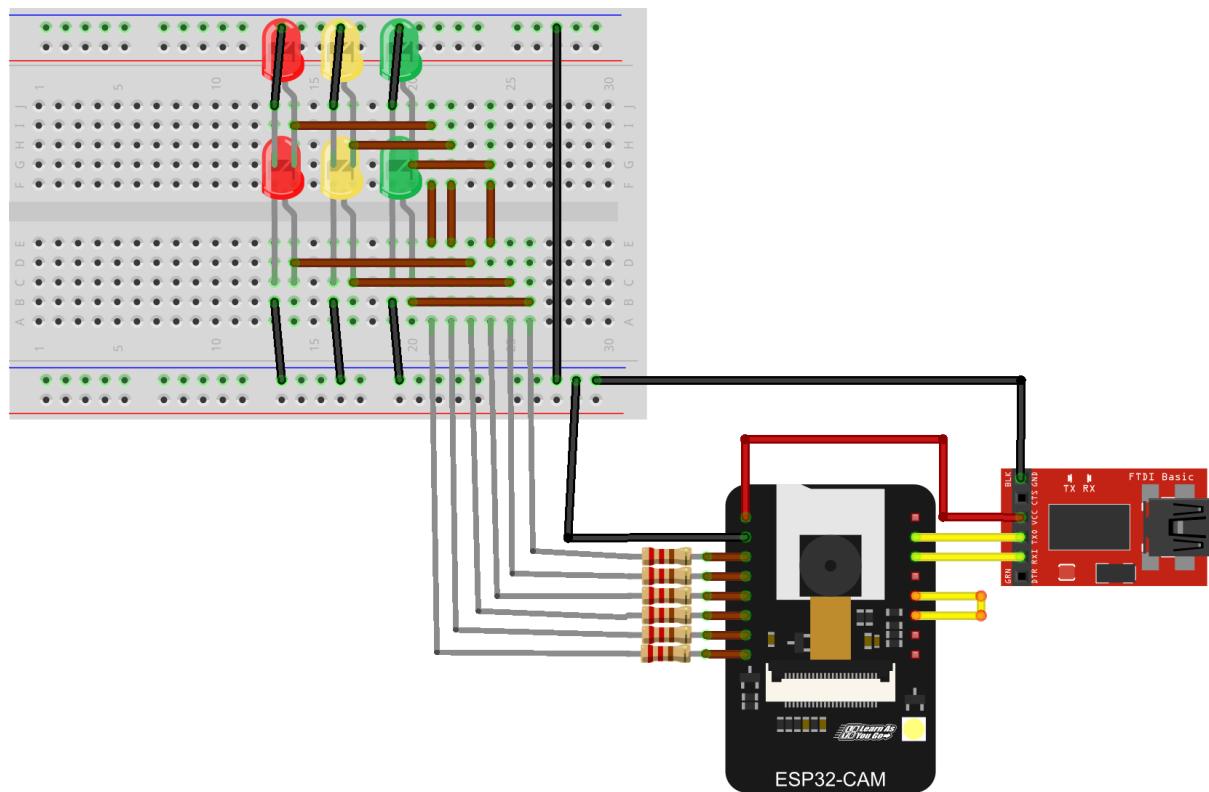
*Obs.:* Os pinos GPIO 0 e GPIO 2 são críticos para o modo de programação e *boot*, respectivamente, e devem ser manuseados com cuidado.

### 3.2. Adaptação do Diagrama Elétrico

O diagrama elétrico é adaptado da configuração original do Arduino para o ESP32-CAM. A principal alteração é a substituição do microcontrolador e a conexão das saídas digitais para o acionamento dos LEDs do semáforo. O ESP32-CAM é programado via **Módulo FTDI** (utilizando uma **Baud Rate** de 115200) ou um ESP32-CAM-MB e deve ser configurado no modo de *flash* (GPIO 0 em *LOW*) antes do *upload* do código. A alimentação pode ser feita via pino VCC (5V ou 3.3V, dependendo do módulo e da fonte). Segue abaixo o esquema elétrico:



fritzing



fritzing

### 3.3. Metodologia de IA com Edge Impulse (TinyML)

O treinamento do modelo de Visão Computacional foi realizado na plataforma **Edge Impulse**, seguindo as etapas de Machine Learning para Microcontroladores (TinyML):

- 1. Coleta de Dataset:** Foram tiradas centenas de fotos da maquete, variando as condições (com e sem veículos, diferentes posições, iluminação)

The screenshot shows the Edge Impulse web interface. On the left, there's a sidebar with navigation links: Dashboard, Devices, Data acquisition, Experiments, EON Tuner, and Impulse design. Below these are sections for Upgrade Plan and View plans. The main area is titled 'Dataset' and shows a table of training samples. The columns are 'SAMPLE NAME', 'LABELS', and 'ADDED'. The first few rows are labeled 'carro\_...' and have the 'Carro' label. To the right, a preview window titled 'carro\_1762881923844' displays a small image of a toy car with a bounding box around it, labeled 'Carro'. Below the preview are various control buttons.

- 2. Rotulagem (Labeling):** No Edge Impulse, as imagens foram rotuladas (ex: "carro", "vazio"). A rotulagem precisa é fundamental para o sucesso da inferência.

This screenshot shows the Edge Impulse interface during the labeling process. The left sidebar includes 'Experiments', 'EON Tuner', 'Impulse design' (with options like 'Create impulse', 'Image', 'Object detection', 'Retrain model', and 'Live classification'), and 'Upgrade Plan'. The main area has a 'Dataset' header with 'Training (303)', 'Test (82)', and 'Post-processing (0)' tabs. A modal dialog is open over the training sample list, prompting 'Enter label' with a large question mark icon. A text input field contains the label 'Carro'. At the bottom of the dialog are 'Cancel' and 'Set label' buttons. The background shows a preview of a toy car image with a bounding box and the label 'Carro'.

3. **Criação do Impulso:** Definiu-se o *pipeline* de processamento, incluindo *Image Data* (redimensionamento para 96x96 ou 160x160) e a etapa de *Classification* (Rede Neural Convolucional - CNN).

The screenshot shows the Edge Impulse web interface for creating a machine learning model. The left sidebar includes options like Dashboard, Devices, Data acquisition, Experiments, EON Tuner, and Impulse design. Under Impulse design, there's a sub-menu for Create impulse, Image, Object detection, Retrain model, and Live classification. A 'View plans' button is also present. The main workspace displays a pipeline with four stages: 'Image data' (red card), 'Image' (white card), 'Object Detection (Images)' (purple card), and 'Output features' (green card). The 'Image data' stage has input axes set to 96x96 and a resize mode of 'Fit'. The 'Image' stage is named 'Image' and has an input axis of 'image'. The 'Object Detection (Images)' stage is named 'Object detection' and has input features checked. The 'Output features' stage is named '1 (Carro)'. A large 'Save Impulse' button is located at the bottom right of the workspace.

4. **Mudança das imagens para escala de cinza:** O ESP-32 CAM, por ser um hardware de baixo custo, é recomendado a mudança das imagens para escala de cinza, para que se atinja maior desempenho da IA.

The screenshot shows the Edge Impulse interface focusing on the 'Image' stage parameters. The left sidebar includes options like Experiments, EON Tuner, and Impulse design (Create impulse, Image, Object detection, Retrain model, Live classification). A 'View plans' button is also present. The main workspace shows the 'Image' stage parameters. Under 'Parameters', the 'Color depth' dropdown is set to 'Grayscale'. A 'Save parameters' button is located at the bottom right of the parameter panel. To the right, there are sections for 'Raw features' (showing raw binary data), 'DSP result' (showing a grayscale image of a car), 'Processed features' (showing numerical feature values), and 'On-device performance'.

5. **Treinamento do Modelo:** Foi utilizado um modelo otimizado para *on-device* inferência, como o **MobileNetV2** (ou um modelo customizado e eficiente em termos de memória), treinado para classificar as imagens. O treinamento envolveu diversas **Épocas** até se atingir uma precisão satisfatória (acima de 90%).

The screenshot shows the Edge Impulse web interface during the training phase. On the left sidebar, under 'Impulse design', there's a 'Upgrade Plan' section with a 'View plans' button. The main area is divided into two main sections: 'Neural Network settings' and 'Training output'.

**Neural Network settings:**

- Training settings:** Number of training cycles: 60, Use learned optimizer: off, Learning rate: 0.001, Training processor: CPU, Data augmentation: on.
- Advanced training settings:** (dropdown menu)
- Neural network architecture:** Shows the input layer with 9,216 features.

**Training output:**

- Model:** Model version: Quantized (int8).
- Last training performance (validation set):** F1 SCORE: 99.4%.
- Confusion matrix (validation set):**

	BACKGROUND	CARRO
BACKGROUND	100.0%	0.0%
CARRO	0%	100%
F1 SCORE	1.00	0.99
- Metrics (validation set):**

METRIC	VALUE
Precision (non-background)	0.99
Recall (non-background)	1.00
F1 Score (non-backeround)	0.99

6. **Deploy (Implantação):** Após o treinamento e validação, o modelo foi exportado como uma **Biblioteca Arduino (Edge Impulse library)**. Este código é então integrado à IDE do Arduino e *flasheado* no ESP32-CAM.

The screenshot shows the Edge Impulse web interface during the deployment phase. The left sidebar remains the same as the training interface.

**Configure your deployment:** You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. [Read more.](#)

**Search deployment options:** (input field)

**DEFAULT DEPLOYMENT:** **Arduino library** An Arduino library with examples that runs on most Arm-based Arduino development boards.

**MODEL OPTIMIZATIONS:** Model optimizations can increase on-device performance but may reduce accuracy.

**Run this model:** Scan QR code or launch in browser to test your prototype.

A large QR code is displayed for testing.

**Launch in browser** button.

O processo de **Inferência** é a execução do modelo treinado, realizada **na borda (on-device)**, sem necessidade de comunicação externa após o *deploy*. A câmera captura o *frame*, a UCP executa a inferência e devolve a probabilidade de cada *label* (ex: 85% 'carro', 15% 'vazio').

## 4. LÓGICA DE CONTROLE

A lógica de controle implementada no *firmware* do ESP32-CAM gerencia o fluxo de tráfego com base no resultado da inferência da IA. O sistema opera em um cruzamento de duas vias de mão única (Avenida Principal e Rua Secundária).

### 4.1. Fluxograma de Operação

1. **Estado Padrão (Prioridade):**
  - **Avenida Principal (Semáforo 1): VERDE.**
  - **Rua Secundária (Semáforo 2): VERMELHO.**
2. **Detectção de Veículo (Gatilho):**
  - O ESP32-CAM monitora continuamente o *feed* de vídeo focado na Rua Secundária.
  - **Gatilho de Ação:** Se a **Inferência da IA** detectar o *label* "carro" com uma **confiança (probabilidade) maior que 70%**.
3. **Transição de Fechamento da Via Principal:**
  - Ao detectar o gatilho, o Semáforo 1 (Avenida Principal) inicia o ciclo de fechamento:
    - **VERDE > AMARELO** (Tempo de retardo: 2 segundos).
    - **AMARELO > VERMELHO**.
4. **Abertura da Via Secundária:**
  - Imediatamente após o Semáforo 1 fechar (Vermelho), o Semáforo 2 (Rua Secundária) abre:
    - **VERMELHO > VERDE**.
5. **Período de Abertura e Retorno:**
  - O Semáforo 2 permanece em **VERDE** por um tempo pré-determinado (X segundos) ou enquanto a Inferência da IA continuar a reportar a presença de "carro" com alta confiança.
  - **Retorno ao Padrão:** Após o período de abertura (ou se a detecção de "carro" cair abaixo de 70% por um período de segurança), o Semáforo 2 fecha:
    - **VERDE > AMARELO** (Tempo de retardo: 2 segundos).
    - **AMARELO > VERMELHO**.
  - O Semáforo 1 retorna imediatamente para o estado **VERDE**, reiniciando o ciclo no Estado Padrão.

## 5. REPRESENTAÇÃO VISUAL

Para a apresentação e validação visual do projeto, foi construída uma maquete em uma base de isopor de 60x60 cm. O asfalto foi representado por E.V.A. de cor cinza e áreas verdes foram decoradas com grama E.V.A. A câmera foi colocada como um poste com suas fiações ligadas aos postes dos leds.

### 5.1 Projeto da maquete em 3D



## 6. CONCLUSÃO

O desenvolvimento do protótipo de semáforo inteligente com ESP32-CAM e TinyML (Edge Impulse) demonstra a viabilidade e eficácia da Visão Computacional para sistemas adaptativos de gerenciamento de tráfego. O projeto alcançou o objetivo de otimizar o fluxo, substituindo a detecção simplista por IR por uma **Inferência** de IA mais precisa e adaptável, que opera de forma eficiente **na borda**.

A implementação valida o potencial do TinyML como uma solução de baixo custo para problemas complexos de infraestrutura urbana, eliminando a dependência de sensores físicos invasivos e fornecendo dados ricos para a tomada de decisão do controlador. Trabalhos futuros devem se concentrar no aprimoramento do modelo de IA para contagem de veículos, o que permitiria o ajuste dinâmico do tempo de abertura dos sinais (em vez de tempo fixo X), e na integração de módulos de comunicação para a criação de uma rede semafórica urbana robusta.

## REFERÊNCIAS

1. PARANÁ. Secretaria da Educação. **Aula 04: Semáforo inteligente.** 2021. Disponível em:  
[https://aluno.escoladigital.pr.gov.br/sites/alunos/arquivos\\_restritos/files/documento/2021-07/aula04\\_semaforo\\_inteligente\\_m2.pdf](https://aluno.escoladigital.pr.gov.br/sites/alunos/arquivos_restritos/files/documento/2021-07/aula04_semaforo_inteligente_m2.pdf). Acesso em: 14 ago. 2025.
2. PARANÁ. Secretaria da Educação. **Aula 30: Sensor de obstáculo.** 2021. Disponível em:  
[https://aluno.escoladigital.pr.gov.br/sites/alunos/arquivos\\_restritos/files/documento/2021-05/aula\\_30\\_sensor\\_obstaculo.pdf](https://aluno.escoladigital.pr.gov.br/sites/alunos/arquivos_restritos/files/documento/2021-05/aula_30_sensor_obstaculo.pdf). Acesso em: 14 ago. 2025.
3. ESPRESSIF SYSTEMS. **ESP32 Series Datasheet.** Versão 4.3. Xangai: Espressif Systems, 2023. Disponível em:  
[https://www.espressif.com/sites/default/files/documentation/esp32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf).
4. GHAZAL, B.; ELKADRI, R.; AL KHALIL, K.; BAZZI, A. **Smart traffic light control system.** In: THIRD INTERNATIONAL CONFERENCE ON ELECTRICAL, COMPUTER, COMMUNICATIONS, AND MECHATRONICS ENGINEERING (ICECCME), 2023. IEEE, 2023.
5. SANDLER, Mark et al. **MobileNetV2: Inverted Residuals and Linear Bottlenecks.** In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR), 2018. Disponível em: <https://arxiv.org/abs/1801.04381>.