



PÓS GRADUAÇÃO - ESPECIALIZAÇÃO EM CIÊNCIA DE DADOS E  
INTELIGÊNCIA ARTIFICIAL

**SLEEP GUARD: Detecção de fadiga com Visão Computacional e  
acionamento de alarme**

**Larissa Freitas da Silva  
RA: 8093528**

Marília - SP

2025

**Larissa Freitas da Silva**  
**RA: 8093528**

**SLEEP GUARD: Detecção de fadiga com Visão Computacional e  
acionamento de alarme**

Trabalho de Conclusão de Curso apresentado  
de Pós-graduação em Ciência de Dados e  
Inteligência Artificial, da Universidade de  
Marília, como requisito parcial à obtenção do  
título de Especialista.

Orientador: Prof. Me. Henrique Leal Tavares.

Marília - SP

2025

## **AGRADECIMENTOS**

Agradeço principalmente a Deus por ter me orquestrado a este caminho em que hoje me encontro concluindo e satisfeita.

Gratidão à Universidade e aos professores pelo apoio, dedicação, resolução de problemas e entrega de soluções, principalmente à arte de ensinar e se doar.

Muito obrigada à minha família que cedeu com apoio o tempo de reuniões e confraternização para que eu pudesse aperfeiçoar meu desenvolvimento pessoal e profissional, além da ajuda em usá-los como usuários para testar meu projeto.

*Deus disse: — Eu irei com você e lhe darei a vitória.*

## **RESUMO**

O trabalho apresenta um sistema para a detecção de fadiga com o uso de Visão Computacional e acionamento de alarme via Arduino com um módulo de Buzzer ao receber alertas de fadiga. A detecção acontece graças aos landmarks da face obtidos pelo MediaPipe e métricas de conceitos estatísticos, sendo EAR (Taxa Eficaz Anual), MAR (Taxa mínima Aceitável de Retorno) e análise vertical para verificar a inclinação da cabeça. O sistema foi testado em tempo real apenas com Webcam, mas se mostrou funcional, considerando que seja uma base para melhorias como pré-processamento e refinamento da imagem, além de personalização de limiares.

Palavras-chave: Visão Computacional. Arduíno. Fadiga. Buzzer.

## **ABSTRACT**

This work presents a fatigue detection system using Computer Vision and an Arduino-based alarm with a Buzzer module triggered upon detecting signs of fatigue. Detection is achieved through facial landmarks obtained via MediaPipe and statistical metrics such as EAR (Eye Aspect Ratio), MAR (Mouth Aspect Ratio), and vertical analysis to assess head tilt. The system was tested in real time using only a webcam and proved functional, serving as a foundation for improvements like image preprocessing, threshold customization, and enhanced refinement.

**Keywords:** Computer Vision. Arduino. Fatigue. Buzzer.

## SUMÁRIO

<b>1 Introdução.....</b>	8
<b>2 Objetivo.....</b>	8
<b>3 Justificativa.....</b>	9
<b>4 Metodologia.....</b>	9
<b>5 Desenvolvimento.....</b>	9
5.1 Arduino Uno.....	9
5.2 Visão Computacional.....	10
5.3 Arduino Uno.....	11
5.4 Funcionamento.....	12
<b>6 Considerações Finais.....</b>	13
6.1 Teste de limiares e erros.....	13
6.2 Crescimento futuro.....	13
6.3 Integrações.....	14
6.4 Registros e ocorrências.....	14
6.5 Deep Learning e Inclusão de datasets.....	14
<b>7 Conclusão.....</b>	14

## **1 Introdução**

A sonolência no volante é uma das causas de acidentes no trânsito que colocam a vida de não só motoristas, mas também de passageiros em risco. Este projeto propõe o desenvolvimento com objetivo do uso de Visão computacional, um detector que ao analisar se o motorista está com sono ou fadiga aciona um alarme dependendo do grau de detecção.

Neste caso, se o motorista permanecer com os olhos fechados por 5 segundos é acionado um alarme de 10 beeps, se for analisado que o motorista está com piscadas frequentes e/ou bocejo é acionado 1 beep e caída de cabeça é acionado 2 beeps.

Inicialmente foi considerada a ideia de aprimoramento com um possível classificador de áudio que analisaria o som do ambiente para subentender se realmente o motor está ligado (se referenciando com o barulho do motor, tráfego e som ambiente), mas depois de estudos foi percebido que podem existir maneiras mais qualificadas para tal melhoria que estarão à grosso modo, sendo mencionadas nas Considerações Finais deste relatório, além de outras considerações valiosas.

## **2 Objetivo**

Mas como um sistema de Visão Computacional pode ser utilizado para detectar sonolência em motoristas e alertar eles de forma eficaz?

O objetivo inicial para esta questão seria detectar os olhos fechados por 5 segundos e criar um alerta para isso, significando sonolência, além de detecção de fadiga que possa analisar piscadas frequentes, caída de cabeça ou bocejos, explorando também possibilidades futuras, como, classificar se o veículo está em movimento além de integração, por exemplo, em sistemas de segurança em tempo real, um alerta, no qual seria emitido um som, seria o ideal pensado para o presente projeto.

A proposta principal é o sistema receber stream de vídeo, onde serão processados os frames em tempo real e por meio da detecção facial de landmarks que colaboram para que sejam feitos técnicas com cálculos com EAR, MAR e inclinação de cabeça, que em junção dos limiares acionam um alarme via arduino que podem representar riscos, detectando assim a fadiga e/ou sonolência de acordo com os pontos faciais detectados.

### **3 Justificativa**

A prevenção de acidentes é um assunto delicado que abrange vários fatores, não só no meio tecnológico e social, visando trazer mais confiabilidade na segurança do motorista.

A fadiga é um estado de exaustão física e mental que reduz a capacidade de concentração, além de diminuir os reflexos, A National Sleep Foundations relatou por meio de uma pesquisa que a privação do sono é capaz de causar comprometimentos cognitivos semelhante à embriaguez, sendo que ficar 24 horas sem dormir equivale a um nível de álcool no sangue de 0,10% que seria um valor acima do limite permitido para motoristas em muitos países (NATIONAL SLEEP FOUNDATION; Novembro, 2023).

### **4 Metodologia**

Foram adotadas abordagens de Visão Computacional (com uso do MediaPipe para analisar as imagens em tempo real) e Fundamentos de Inteligência Artificial (com as lógicas e regras para as detecções).

Houve a captura de imagens em tempo real por vídeo de WebCam, a detecção de pontos faciais com o MediaPipe, que utiliza modelos do Google, abordagem de métricas para obter a validação dos casos de uso requeridos com uso dos limiares, comunicação do Arduino Uno que intermediou o sistema com o buzzer ativo, além de testes e ajustes.

### **5 Desenvolvimento**

#### **5.1 Arduino Uno**

Para a parte física, foi utilizado um módulo de buzzer ativo, em que de início era ligado ao protoboard para fazer as conexões com os jumpers ao arduino, mas posteriormente o protoboard foi descartado. Logo para o uso de acionamento de alarme para a parte física foram utilizados 3 jumpers, um arduino e um módulo de buzzer ativo. E assim que o processo é executado e feita a análise de Visão computacional o arduino aciona o alarme.

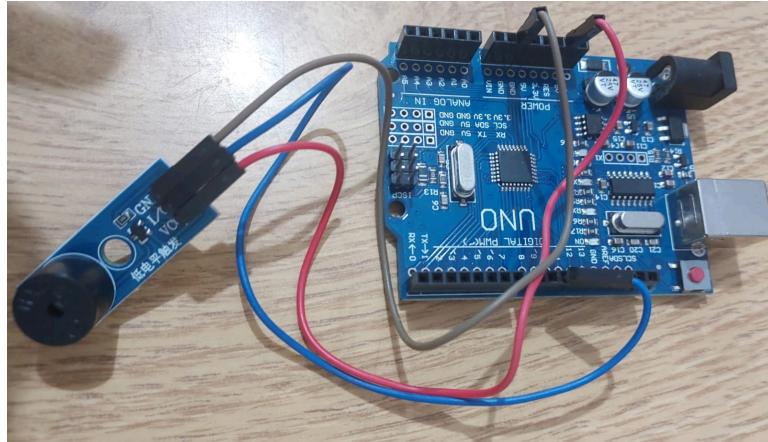


Figura 1: Foto de componentes principais utilizados (não descarta uso de conector USB).

Fonte: autora.

## 5.2 Visão Computacional

Para a Visão Computacional, principal funcionamento para a proposta do projeto, foi usado o *MediaPipe FaceMesh* com o pacote de modelos do *FaceLandmarker* da Google AI for Developers, onde foram recolhidos os principais pontos para uso:

Região	Índices	Descrição	Motivação
Nariz	[1]	Ponta do nariz	QUEDA DA CABEÇA
Olho Esquerdo	[362, 385, 387, 263, 373, 380]	Pontos principais para olho esquerdo	VERIFICAÇÃO DE OLHOS FECHADOS
Olho Direito	[33, 160, 158, 133, 153, 144]	Pontos principais para olho direito	
Boca	[13, 14, 78, 308]	Pontos principais para boca	BOCEJO

Figura 2: Imagem de tabela explicativa dos pontos faciais landmarks escolhidos. Fonte: autora.

Os pontos são pesquisados e consultados, e podem ser visualizados também em uma prévia no Guia de detecção de pontos de referência do rosto, a task permite a aprendizagem de aplicações assim como aplicação de filtros e efeitos faciais, criação de avatares virtuais, possibilita o uso de Machine Learning, geração de pontos faciais tridimensionais, pontuações *blendshape* para inferência das superfícies faciais (de modo tridimensional) detalhadas em tempo real, além de transformações para renderização de efeitos.

De acordo com o mapeamento disposto na plataforma, pode ser analisado a maior parte dos pontos que seriam essenciais para as análises de imagem feitas, segue a imagem destacando os pontos usados em verde.

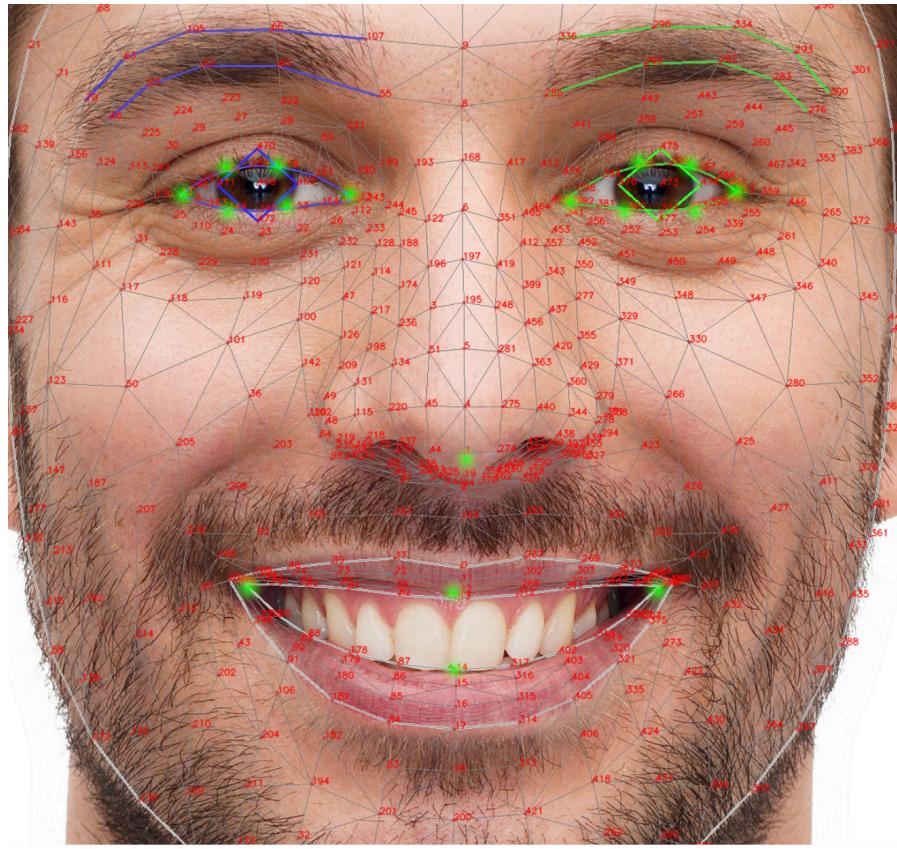


Figura 3: Mapeamento de landmarks utilizados com destaque em verde.

Fonte:

[https://storage.googleapis.com/mediapipe-assets/documentation/mediapipe\\_face\\_landmark\\_fullsize.png](https://storage.googleapis.com/mediapipe-assets/documentation/mediapipe_face_landmark_fullsize.png)

### 5.3 Funcionamento geral

Da linguagem, foi usada a linguagem Python para o sistema principal e linguagem C++ para o Arduino IDE, além das bibliotecas para o sistema ter o funcionamento esperado, sendo as principais o *Pytorch*, *OpenCV* e *MediaPipe* para Visão Computacional, serial para conexão com Arduino, entre outras libs.

Dos casos de uso para a análise foram usados os cálculos; de *EAR* (Eye Aspect Ratio) que usa a fórmula  $EAR = (\|p2 - p6\| + \|p3 - p5\|) / (2 * \|p1 - p4\|)$ , sendo uma métrica para analisar os olhos se estão fechados de acordo com a geometria, de modo geral, se o valor for alto os olhos estão fechados, se o valor for baixo, os olhos estarão abertos.

Após o cálculo de *EAR* foi usado também o cálculo de *MAR* (Mouth Aspect Ratio, sendo sua fórmula  $MAR = (\|p1-p2\|) / (\|p1-p3\| + \|p2-p3\|)$ ), em que de modo geral o MAR de valor alto mostra que a boca está aberta e se o valor estiver mais baixo indicaria que a boca está fechada, neutra ou mais relaxada, de acordo com o limiar indicado.

## 5.4 Aspectos para análise

E através de apenas cálculos existem uma gama de ideias voltados a ergonomia, com o uso de Visão Computacional que permitem a análise de movimentos esportivos, de posições de ioga, de exercícios, de movimentos para pacientes em fisioterapia, de direção no volante, entre outros, sendo assim neste caso foi feito uma análise para a inclinação da cabeça com o uso do ponto principal do rosto, sendo o nariz para a complementação do alerta de fadiga, sendo que se o ponto inicial cair (ponto considerando queda vertical do nariz), significa a queda de cabeça.

O próximo passo é mostrar qual seria o objetivo do que foi apresentado em um fluxograma, para melhor compreensão:

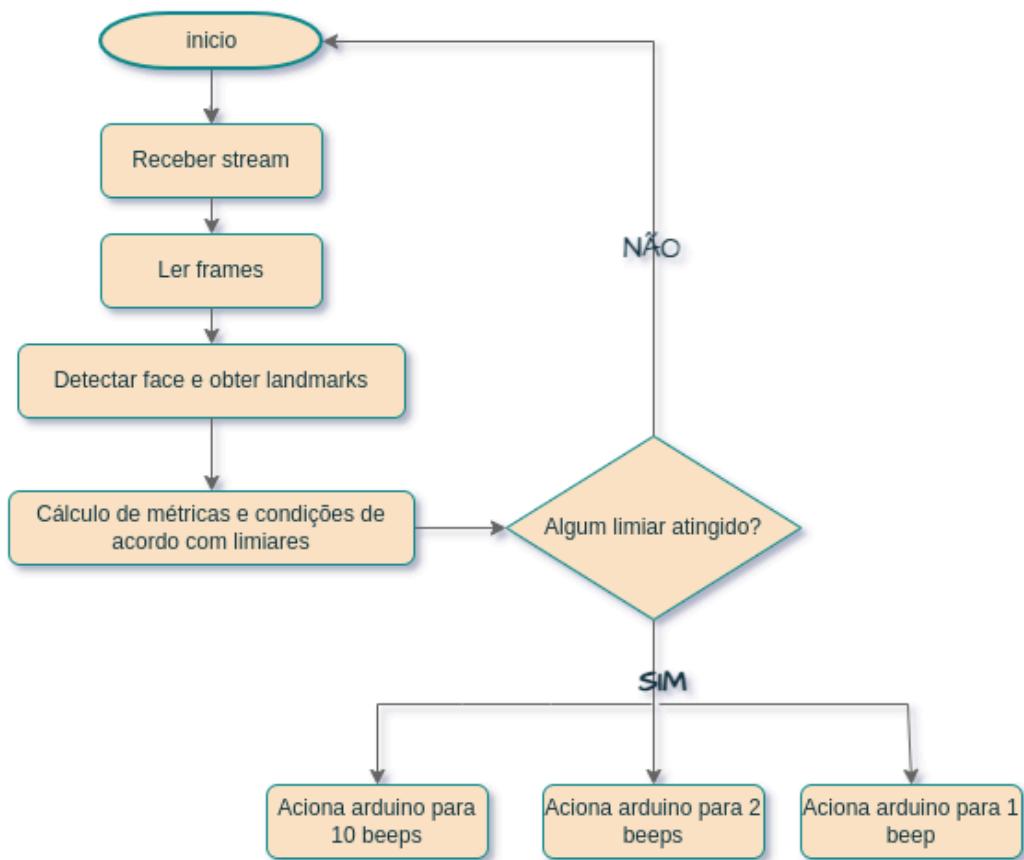


Figura 4: Fluxograma de processo. Fonte: autora.

A condições para o alarme são:

Alarme	Condição	Beeps
<b>Sonolência</b>	<b>Olhos fechados por 5 segundos contínuos</b>	10 beeps 
<b>Fadiga com queda de cabeça</b>	<b>Cabeça inclinada 3x em 30s com limiar de ângulo &gt; 30°</b>	2 beeps 
<b>Fadiga (por piscadas e/ou bocejo)</b>	<b>10 piscadas em 15s ou 8 piscadas + 1 bocejo</b>	1 beep 

Tabela 1: Explicação de finalidade de acionamentos de alarme por condições.

## 6 Considerações Finais

### 6.1 Teste de limiares e erros

Teste de limiares e erros: Durante os testes para todos os casos, percebe-se que foi essencial uma webcam para ser analisado e testado em tempo real, pois escolher um limiar adequado exige muitos testes e ajustes. Mas o limiar para cálculo de EAR se mostrou mais sensível, testado com outras pessoas, percebe-se que a luminosidade, ou falta de luz; o formato dos olhos, o uso de acessórios no rosto às vezes podem influenciar no valor de EAR de tal maneira. Um sistema mais robusto poderia incluir um controle de ajuste de limiares de forma personalizada para que os usuários se sentissem mais seguros e o cálculo de EAR agisse de forma mais precisa diante de cada situação.

### 6.2 Crescimento futuro

O projeto pode evoluir com adoção de datasets especializados, como o uso do DMD (Driver Monitoring Dataset), contendo outros serviços, que dependem do ângulo que a câmera estará posicionada, mas assim podendo ser observadas situações como, distração no celular e comportamento na direção. Sendo afirmado que “Os acidentes de trânsito de hoje são principalmente devido a erro humano”.

Melhorias com pré-processamento: A precisão de leitura na imagem pode ser melhorada com pré processamento de imagem, por exemplo, ativar a análise apenas se houver uma maior movimentação, após analisados alguns frames, além

de serem adicionados filtros como Gaussian Blur, Thresholding, máscara, dilatação, contorno, entre outros métodos.

### **6.3 Integrações**

Integração com dispositivos, como por exemplo MDVRs (Mobile Digital Video Recorder) para segurança de frotas veiculares.

### **6.4 Registros e ocorrências**

Registros de ocorrências: Dependendo da robustez pode também ser criado um registro de ocorrências que pode capturar frames enquanto é acionado o alarme.

### **6.5 Deep Learning e Inclusão de datasets**

Entrando em foco por um momento na necessidade de promover mais segurança aos motoristas, o artigo Identificação de distração de motorista com um conjunto de redes neurais convolucionais, publicado em 13 de fevereiro de 2019, mostra que não é apenas a fadiga a causa de acidentes de trânsito, sendo que outras causas podem ser atividades que desvie atenção na direção, considerando causas visuais (“tirar os olhos da estrada”), manuais (tirar mãos do volante) e cognitivas (tirar o foco da direção). Por isso a de estudar a importância de aderir mais funcionalidades, como por exemplo, inclusão de datasets e classificações de comportamento.

## **7 Conclusão**

A análise por forma mais analítica de modo a encontrar maneiras estatísticas, como geometria espacial, trabalhar com as matrizes, pode fazer com que diminua a possibilidade de uso de datasets, ao menos neste caso, em que é uma base simples de um programa que pode vir a ter mais robustez.

Mas o que vem ao caso é essa opção de não ter precisado do uso de um dataset que pode acabar se tornando custoso, mesmo que possa trazer mais exatidão, pode também trazer mais gastos.

No entanto há a possibilidade de mais gastos, mas que tragam mais robustez e precisão, sendo que mesmo assim pode ser testado maneiras de não se precisar de um ou mais datasets.

Por enquanto trazer mais precisão, como pré-processamento já pode contribuir ainda mais para cenários reais e com maior precisão.

## **Referências**

NATIONAL SLEEP FOUNDATION. *Drowsy Driving*. Disponível em: <https://www.sleepfoundation.org/drowsy-driving>. Acesso em: abr. 2025.

LEARN OPENCV. *Driver Drowsiness Detection Using MediaPipe in Python*. Disponível em: <https://learnopencv.com/driver-drowsiness-detection-using-medialpipe-in-python/>. Acesso em: abr. 2025.

CHEN, Tianyang et al. *Driver Drowsiness Detection Using Lightweight Convolutional Neural Network with Visual Attention*. arXiv, 2023. Disponível em: <https://arxiv.org/html/2306.10159v4>. Acesso em: abr. 2025.

ABDELBAKY, Mohamed et al. *Driver Distraction Identification Using Deep Learning*. *Journal of Advanced Transportation*, 2019. Disponível em: <https://onlinelibrary.wiley.com/doi/10.1155/2019/4125865>. Acesso em: abr. 2025.

IEEEEXPLORE. *Driver Fatigue Detection System Based on Convolutional Neural Network*. IEEE, 2024. Disponível em: <https://ieeexplore.ieee.org/document/10462528>. Acesso em: abr. 2025.

GOOGLE AI. *MediaPipe Face Landmarker*. Google Developers, 2024. Disponível em: [https://ai.google.dev/edge/mediapipe/solutions/vision/face\\_landmarker?hl=pt-br](https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker?hl=pt-br). Acesso em: abr. 2025.