

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ  
CURSO DE ENGENHARIA DE COMPUTAÇÃO

BRUNO EMANUEL ZENATTI  
ENZO HOLZMANN GAIO  
FELIPE STILNER EUFRANIO  
IAN MASSANOBU SANTOS ISHIKAWA

**POLICE CAR CAM**

OFICINA DE INTEGRAÇÃO 2 – RELATÓRIO FINAL

CURITIBA

2025

**BRUNO EMANUEL ZENATTI  
ENZO HOLZMANN GAIO  
FELIPE STILNER EUFRANIO  
IAN MASSANOBU SANTOS ISHIKAWA**

## **POLICE CAR CAM**

Relatório Final da disciplina Oficina de Integração 2, do curso de Engenharia de Computação, apresentado aos professores que ministram a mesma na Universidade Tecnologica Federal do Paraná como requisito parcial para obtenção da aprovação na disciplina.

Orientador: Prof. Daniel Rossato De Oliveira

**CURITIBA**

**2025**

---

**Dados Internacionais de Catalogação na Publicação**

---

T137

Police Car cam/ . – 2025.  
48 f. : il. ; 30 cm

Orientador: Prof. Daniel Rossato De Oliveira.  
Oficina de Integração 2 – Relatório Final (Graduação em Engenharia de Computação) – Universidade  
Tecnológica Federal do Paraná. Curso de Engenharia de Computação. Curitiba, 2025.  
Bibliografia: f. 43-44.

1. Oficinas de Integração

CDD (22. ed.) 621.3

---

Biblioteca xxxxxx

## RESUMO

ZENATTI, Bruno; GAIÓ, Enzo; EUFRANIO, Felipe; ISHIKAWA, Ian. POLICE CAR CAM. 48 f. Oficina de Integração 2 – Relatório Final – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Curitiba, 2025.

O aumento da frota de veículos e dos casos de irregularidades no trânsito torna essencial o uso de tecnologias avançadas para auxiliar as forças de segurança. O projeto Police Car Cam propõe um sistema embarcado para leitura automática de placas veiculares, utilizando inteligência artificial e processamento de imagens. O objetivo é oferecer uma ferramenta eficiente para identificação de veículos irregulares, reduzindo a dependência de verificações manuais e tornando a fiscalização mais ágil e precisa.

A solução é composta por uma câmera acoplada a uma viatura e um hardware embarcado equipado com uma Neural Processing Unit (NPU), responsável pelo processamento das imagens capturadas. O sistema detecta e reconhece placas veiculares, comparando-as com um banco de dados local, atualizado periodicamente. Caso um veículo irregular seja identificado, alertas visuais e sonoros são emitidos para os agentes. Além disso, um mecanismo de gravação sob demanda permite o registro de vídeos e localização para futuras investigações.

O reconhecimento de placas é realizado por redes neurais convolucionais baseadas no modelo YOLO (You Only Look Once), conhecido por sua capacidade de detecção rápida e eficiente de objetos em imagens. A rede foi treinada utilizando um conjunto de dados diversificado, incluindo diferentes condições de iluminação e ângulos para aumentar sua robustez. O modelo segue uma abordagem de segmentação, primeiro localizando a placa na imagem e depois identificando os caracteres individualmente. Para otimizar o desempenho em hardware embarcado, o modelo foi convertido para um formato compatível com a Neural Processing Unit (NPU) do microcontrolador, garantindo inferências em tempo real. Essa otimização permite que a solução funcione de maneira independente, sem necessidade de processamento em nuvem, tornando o sistema mais rápido e confiável.

O Police Car Cam se apresenta como uma solução inovadora e eficaz para monitoramento veicular, trazendo avanços significativos para a segurança pública e a modernização dos processos de fiscalização.

**Palavras-chave:** Inteligência Artificial, Reconhecimento de Placas, Processamento de Imagens, Sistema Embocado, Redes Neurais Convolucionais, YOLOv3, Unidade de Processamento Neural

## ABSTRACT

ZENATTI, Bruno; GAIÓ, Enzo; EUFRANIO, Felipe; ISHIKAWA, Ian. POLICE CAR CAM. 48 f. Oficina de Integração 2 – Relatório Final – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Curitiba, 2025.

The increase in the vehicle fleet and the cases of traffic irregularities makes the use of advanced technologies essential to assist security forces. The Police Car Cam project proposes an embedded system for automatic license plate recognition, using artificial intelligence and image processing. The goal is to provide an efficient tool for identifying irregular vehicles, reducing the reliance on manual checks, and making monitoring more agile and precise.

The solution consists of a camera mounted on a police vehicle and an embedded hardware system equipped with a Neural Processing Unit (NPU), responsible for processing the captured images. The system detects and recognizes license plates, comparing them to a locally updated database. If an irregular vehicle is identified, visual and auditory alerts are issued to the agents. Additionally, an on-demand recording mechanism allows video and location data to be saved for future investigations.

License plate recognition is performed by convolutional neural networks based on the YOLO (You Only Look Once) model, known for its ability to quickly and efficiently detect objects in images. The network was trained using a diverse dataset, including different lighting conditions and angles to enhance its robustness. The model follows a segmentation approach, first locating the plate in the image and then identifying the characters individually. To optimize performance on embedded hardware, the model was converted into a format compatible with the microcontroller's Neural Processing Unit (NPU), ensuring real-time inferences. This optimization allows the solution to operate independently, without the need for cloud processing, making the system faster and more reliable.

The Police Car Cam presents itself as an innovative and effective solution for vehicle monitoring, bringing significant advancements to public safety and modernizing the monitoring processes.

**Keywords:** Artificial Intelligence, License Plate Recognition, Image Processing, Embedded System, Convolutional Neural Networks, YOLOv3, Neural Processing Unit

## LISTA DE FIGURAS

FIGURA 1	– Radxa ROCK 5C Lite .....	11
FIGURA 2	– Câmera .....	12
FIGURA 3	– Módulo GPS .....	12
FIGURA 4	– Display LCD .....	13
FIGURA 5	– Exemplo de emprego da rede YOLOv3 .....	14
FIGURA 6	– Exemplo de imagem extraída do dataset .....	15
FIGURA 7	– Projeto 3D .....	19
FIGURA 8	– Impressão 3D .....	19
FIGURA 9	– Projeto de Hardware .....	20
FIGURA 10	– Visão geral do sistema .....	22
FIGURA 11	– Estado de Atualização do Banco de Dados .....	23
FIGURA 12	– Estrutura de Tabelas dos Bancos de Dados .....	23
FIGURA 13	– Estado de Gravação de Imagens .....	25
FIGURA 14	– Estado de Detecção de Placas .....	26
FIGURA 15	– Exemplo de imagem anotada e visualizada no DarkMark .....	27
FIGURA 16	– Exemplo de placa anotada e visualizada no DarkMark .....	29
FIGURA 17	– Exemplo de possíveis valores para <i>paddings</i> .....	29
FIGURA 18	– Infraestrutura de software RKNN .....	30
FIGURA 19	– Resultado Final .....	36
FIGURA 20	– Legenda do Cronograma .....	39
FIGURA 21	– Plano de Projeto .....	39
FIGURA 22	– Elaboração do Blog .....	40
FIGURA 23	– Estrutura Física .....	40
FIGURA 24	– Prototipagem .....	40
FIGURA 25	– Desenvolvimento na Placa .....	41
FIGURA 26	– Integração das Partes .....	41
FIGURA 27	– Entrega do Projeto .....	41

## **LISTA DE TABELAS**

TABELA 1	– Desempenho do Modelo de Reconhecimento de Placas .....	33
TABELA 2	– Métricas de Desempenho da Segmentação .....	34
TABELA 3	– Métricas de desempenho dos modelos .....	34
TABELA 4	– Resultados de precisão, acurácia e taxa de erro por classe. ....	35
TABELA 5	– Lista de Materiais .....	42
TABELA 6	– Tempo Total Estimado e Levado .....	42
TABELA 7	– YOLOv3-Tiny modificada para Reconhecimento de Placas .....	45
TABELA 8	– YOLOv3-Tiny modificada para Segmentação de Caracteres .....	46
TABELA 9	– YOLOv3-Tiny modificada para Reconhecimento de Letras .....	47
TABELA 10	– Arquitetura da Rede YOLO baseada no Darknet .....	48

## SUMÁRIO

<b>1 INTRODUÇÃO .....</b>	<b>9</b>
1.1 MOTIVAÇÃO .....	9
1.2 OBJETIVOS .....	10
1.2.1 Objetivo Geral .....	10
1.2.2 Objetivos Específicos .....	10
<b>2 MATERIAIS .....</b>	<b>11</b>
2.1 RADXA ROCK 5C LITE .....	11
2.2 CÂMERA .....	11
2.3 MÓDULO GPS .....	12
2.4 DISPLAY LCD .....	13
2.5 BIBLIOTECA GPIOD .....	13
2.6 USO DE CNNS PARA DETECÇÃO DE INSTANCIAS - REDE YOLO ( <i>YOU ONLY LOOK ONCE</i> ) .....	13
2.7 UFPR-ALPR DATASET .....	14
2.8 FRAMEWORK DARKNET .....	16
2.9 RKNN-TOOLKIT2 .....	16
<b>3 DESENVOLVIMENTO .....</b>	<b>17</b>
3.1 VISÃO GERAL .....	17
3.1.1 Requisitos de software .....	17
3.1.2 Requisitos de hardware .....	17
3.1.3 Requisitos não funcionais .....	18
3.2 PROJETO MECÂNICO .....	18
3.3 PROJETO DE HARDWARE .....	19
3.3.1 Radxa ROCK 5C Lite .....	20
3.3.1.1 Neural Processing Unit .....	20
3.3.2 Módulo GPS NEO-6M .....	21
3.4 PROJETO DE SOFTWARE .....	21
3.4.1 Atualização do Banco de Dados Local .....	21
3.4.2 Gravação de imagem e geolocalização .....	24
3.4.3 Identificação das placas .....	24
3.4.4 Desenvolvimento e Treinamento das Redes .....	27
3.4.5 Inferência do Modelo na Placa .....	30
<b>4 RESULTADOS .....</b>	<b>33</b>
4.1 DESEMPENHO DO SISTEMA .....	33
4.1.1 Desempenho dos treinamentos .....	33
4.2 IMPLEMENTAÇÃO E INTEGRAÇÃO DOS COMPONENTES .....	35
4.3 TESTES PRÁTICOS .....	36
4.4 APRESENTAÇÃO DOS RESULTADOS .....	36
<b>5 CONCLUSÃO .....</b>	<b>37</b>
5.1 TRABALHOS FUTUROS .....	37
5.1.1 Melhorar a Detecção de Placas .....	37

5.1.2 Melhorar a Interação com o Usuário .....	38
5.1.3 Substituir o Módulo GPS .....	38
<b>6 CRONOGRAMA, CUSTOS E GESTÃO DO PROJETO .....</b>	<b>39</b>
6.1 CRONOGRAMA .....	39
6.2 CUSTOS .....	41
6.3 GESTÃO DO PROJETO .....	42
<b>REFERÊNCIAS .....</b>	<b>43</b>
<b>Apêndice A – CONFIGURAÇÃO REDES YOLO .....</b>	<b>45</b>

# 1 INTRODUÇÃO

O Police Car Cam consiste em um sistema embarcado que integra uma câmera acoplada a viaturas policiais e um hardware de alto desempenho equipado com uma Neural Processing Unit (NPU), responsável pelo processamento de imagens. Utilizando técnicas de aprendizado de máquina e redes neurais convolucionais, o sistema é capaz de detectar e reconhecer placas veiculares, comparando os dados capturados com um banco de dados atualizado periodicamente. Quando um veículo com irregularidades é identificado, o sistema emite alertas visuais e sonoros para os agentes em serviço, possibilitando uma abordagem mais ágil e eficiente.

## 1.1 MOTIVAÇÃO

O aumento da frota de veículos e a crescente incidência de infrações de trânsito, furtos e roubos de automóveis no Brasil representam desafios significativos para as autoridades de segurança pública (Correio Braziliense, 2025). O monitoramento e a fiscalização manuais dessas ocorrências são processos que demandam um alto volume de recursos humanos e materiais, tornando-se ineficientes diante do grande fluxo de veículos nas vias urbanas e rodoviárias. Além disso, as abordagens convencionais podem estar sujeitas a falhas humanas e limitações operacionais, dificultando a identificação rápida de veículos em situação irregular.

Nesse contexto, avanços tecnológicos, especialmente na área de inteligência artificial e processamento de imagens, apresentam soluções inovadoras para aprimorar a fiscalização do trânsito e a segurança pública. Com isso, o projeto Police Car Cam surge como uma alternativa eficiente para a automação da leitura de placas veiculares e a identificação de irregularidades em tempo real, reduzindo a dependência de processos manuais e ampliando a capacidade de monitoramento das autoridades competentes.

## 1.2 OBJETIVOS

### 1.2.1 OBJETIVO GERAL

Os objetivos do projeto Police Car Cam são diversos e englobam tanto a melhoria na fiscalização de trânsito quanto o apoio às forças de segurança. O objetivo geral é desenvolver uma ferramenta confiável e eficaz para a identificação automática de veículos irregulares, otimizando os processos de monitoramento e redução de erros humanos.

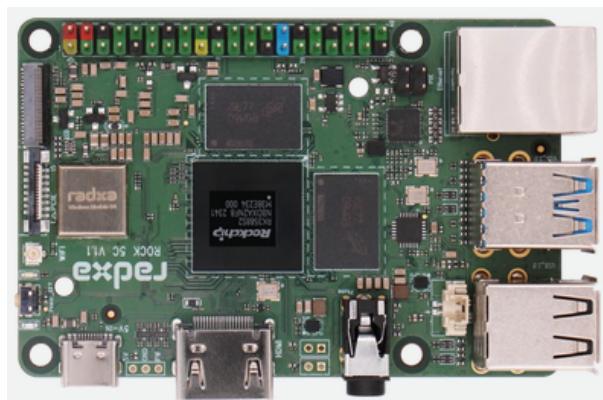
### 1.2.2 OBJETIVOS ESPECÍFICOS

- Implementar um sistema de captura de imagens utilizando uma câmera instalada na viatura policial;
- Desenvolver um algoritmo robusto para reconhecimento de placas utilizando inteligência artificial;
- Criar um banco de dados local contendo informações atualizadas sobre veículos em situação irregular;
- Integrar um módulo de comunicação para sincronização periódica do banco de dados local com um servidor remoto;
- Implementar mecanismos de alerta visual e sonoro para notificação instantânea dos agentes de segurança;
- Desenvolver um sistema de gravação de vídeos com geolocalização para registro e auditoria.

## 2 MATERIAIS

### 2.1 RADXA ROCK 5C LITE

Para realizar todo o processamento da identificação de imagem e das demais funcionalidades do projeto foi escolhida a *Single Board Computer* (SBC) Radxa ROCK 5C Lite. A principal motivação para a escolha desse hardware foi a presença de uma Neural Process Unit (NPU), a qual é focada em tarefas de processamento de Redes Neurais, e consequentemente em tarefas de Processamento de Imagens.



**Figura 1: Radxa ROCK 5C Lite**

**Fonte:** (Rockchip Electronics Co., Ltd, 2025)

### 2.2 CÂMERA

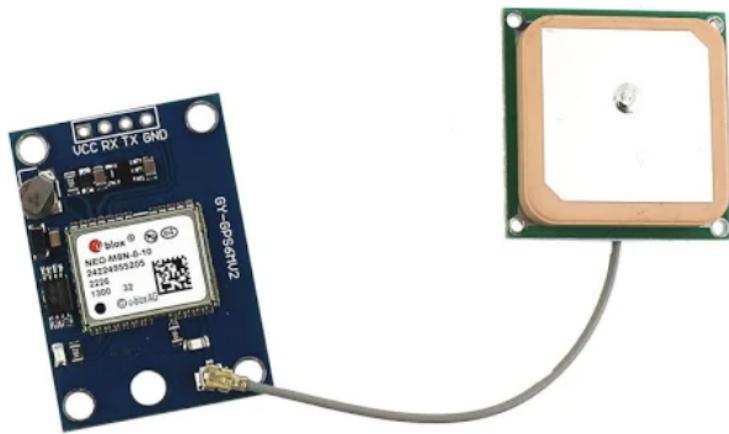
A webcam escolhida foi a Logitech C920e (LOGITECH, 2022), com resolução Full HD 1080p a 30fps. Sua conexão com a placa é feita via USB-A, de modo que facilita a integração com o restante do sistema. Uma imagem da câmera pode ser vista na Figura 2.



**Figura 2: Câmera**  
**Fonte: Autoria própria**

### 2.3 MÓDULO GPS

Para obter a localização da imagem a ser gravada, foi usado um módulo GPS NEO-6M, com uma antena externa, o qual possui comunicação via interface UART. Tal modelo foi escolhido devido a grande disponibilidade no mercado, além da fácil utilização, haja vista que é necessário somente conectá-lo a alimentação para iniciar o envio das coordenadas. O módulo GPS utilizado pode ser visualizado na Figura 3.



**Figura 3: Módulo GPS**  
**Fonte: Autoria própria**

## 2.4 DISPLAY LCD

A fim de mostrar a qual a placa comprometida, escolheu-se um display LCD 16x2 com adaptador i2c. Tal escolha foi motivada pelo baixo custo e facilidade de uso da interface i2c. Além disso, preferiu-se o i2c também pelo número reduzido de portas necessárias para operar o display em comparação com o uso sem o mesmo.



**Figura 4: Display LCD**

**Fonte:** Autoria própria

## 2.5 BIBLIOTECA GPIOD

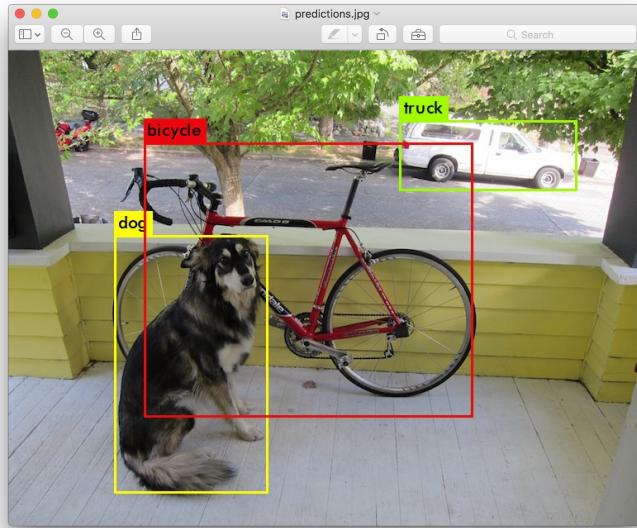
A fim de receber as informações do módulo GPS e realizar pooling para detectar a borda do botão foi utilizada a biblioteca GPIOD (GOLASZEWSKI, 2025) em python. Ela permite acessar as interfaces de todos os GPIO's da placa de forma simples e eficaz.

## 2.6 USO DE CNNS PARA DETECÇÃO DE INSTANCIAS - REDE YOLO (*YOU ONLY LOOK ONCE*)

Um Rede Neural Convolucional (CNN) (GOODFELLOW et al., 2016) é uma rede especializada no tratamento de dados que podem ser conformados em um plano n-dimensional, e possuem significado no ordenamento que são arranjados. Séries temporais e Imagens podem ser operadas por essas redes para que informações relevantes sejam extraídas da entrada original. O uso da operação matemática de Convolução é o que está sendo utilizado para extrair as informações relevantes.

Uma das descobertas inovadoras do ramo de Deep Learning foi a rede You Only Look Once (YOLO)(REDMON et al., 2016). A rede foi arquitetada para detecção de instâncias

em imagens com treinamento simplificado, alcançar detecção em tempo real. Os objetos são separados em classes e o sistema aprende a generalizar as representações das instâncias e reconhece-las, retornando ao usuário as coordenadas, classe e confiança identificadas pela rede, conforme um limiar de confiança.



**Figura 5: Exemplo de emprego da rede YOLOv3**

**Fonte:** Darknet (REDMON, 2013–2016a)

A rede calcula a confiança combinando a probabilidade da área de interesse ser um objeto conhecido com o mapa de probabilidades que separa as classes por área, gerando um *bounding-box* da classe mais provável. O aspecto mais interessante ao trabalho desenvolvido é que, utilizando camadas pré-treinadas, é possível expandir esse treinamento para detectar novas classes customizadas, o que é mais rápido e energéticamente eficiente que treinar completamente uma CNN. Esse método de *Transfer Learning* (TL) é possível, pois, as camadas de detecção de classes e objetos operam em cima de camadas que foram treinadas para extrair informações de imagens. Existem limitações relacionadas a esse método, porém em um tempo limitado e recurso de hardware limitado, o uso de TL possibilita o emprego dessas redes para aplicações especializadas com precisão razoável, como a referida nesse texto.

## 2.7 UFPR-ALPR DATASET

Para o treinamento das Redes, foi utilizado o *UFPR-ALPR dataset* (LAROCA et al., 2018). O dataset contém 4.500 imagens de carros em ambiente urbano, com as marcações

que referenciam a posição da placa, posição dos caracteres da placa e o caracteres respectivos da placa. O dataset foi sintetizado levando em conta diferentes perspectivas, ângulos, em ambientes urbanos e estradas variadas, diferentes incidências de luz do dia e obstrução da qualidade pela presença de sombras variadas. O objetivo é garantir que as placas fossem reconhecidas em ambientes variados e o modelo não se adaptasse a um nicho específico de plano de fundo, fenômeno conhecido como *overfitting*. Nesse fenômeno, a rede é treinada até encontrar uma métrica ótima no escopo de treinamento, porém a métrica esconde a ineficiência da rede de extrair sentido em ambientes diferentes do escopo de treinamento. O segundo aspecto importante é a divisão com que o dataset foi seccionado: 4/10 para treinamento; 4/10 para teste e 2/10 para validação. O treinamento e o teste foram preparados com instâncias



**Figura 6: Exemplo de imagem extraída do dataset**

**Fonte:** *UFPR-ALPR dataset* (LAROCA et al., 2018)

em ambientes relativamente similares, diferenciando apenas da forma em que os elementos são dispostos e as condições climáticas. A validação contém casos extremos, onde as imagens contêm placas muito sombreadas; ângulos de fotografia mais extremados, e principalmente, presença de elementos gráficos de texto, que colocaram a prova sistemas de segmentação e reconhecimento de placas mais antigos. A proposta presente no estudo que originou o dataset põe à prova abordagens que utilizaram datasets menos extensos (SILVA; JUNG, 2017)(SILVA; JUNG, 2018)(GONÇALVES et al., 2016). O último aspecto muito importante para a escolha desse dataset é o contexto em que as imagens foram capturadas. A perspectiva das fotografias é de uma camera presa à um painel de um carro, apontada para o vidro frontal do carro. O sistema proposto de ALPR também propõe-se estar posicionado na mesma posição. Isso garante que o contexto utilizados para treinamento não estejam muito diferente do contexto em que a rede será empregada, o que auxilia em superar um possível *overfitting* da rede.

## 2.8 FRAMEWORK DARKNET

O framework Darknet (REDMON, 2013–2016a) é uma ferramenta em código aberto para treinamento e execução de Redes Neurais, com suporte a aceleração gráfica com GPU. A biblioteca já está na sua terceira versão (CHARETTE, 2023–2025b), atualizada para drivers e bibliotecas de GPU mais atualizadas, sendo compatível com o CUDA na sua versão mais recente. Como o foco da ferramenta é o treinamento e emprego de Redes Neurais Convolucionais focadas em Visão Computacional, sua utilização será responsável por facilitar o desenvolvimento da cadeia de processamento da visão, da imagem até o reconhecimento dos caracteres. O framework foi empregado para o treinamento das Redes YOLO

## 2.9 RKNN-TOOLKIT2

O RKNN-Toolkit2 (Rockchip Electronics Co., Ltd, 2024b) é um framework de desenvolvimento otimizado para aceleração de redes neurais em dispositivos com NPUs (Neural Processing Units) da Rockchip. Ele permite a conversão de modelos treinados em frameworks populares, como TensorFlow, PyTorch, ONNX e Caffe, para um formato compatível com a NPU, o formato RKNN (Rockchip Neural Network), possibilitando inferências de alto desempenho com baixo consumo de energia.

### 3 DESENVOLVIMENTO

O desenvolvimento do *Police Car Cam* foi realizado com base em princípios de engenharia de software e hardware, garantindo uma solução integrada e eficiente. Dessa forma, nas próximas seções serão explanadas os processos envolvidos no desenvolvimento do projeto.

#### 3.1 VISÃO GERAL

Inicialmente, foram definidos os seguintes requisitos funcionais e não funcionais do projeto:

##### 3.1.1 REQUISITOS DE SOFTWARE

1. RF01: O sistema deve detectar as placas.
2. RF02: O sistema deve ser capaz de buscar a placa detectada em um banco de dados local.
3. RF04: O sistema deve gravar as imagens em um dispositivo de armazenamento.
4. RF05: Ao iniciar a gravação, o sistema deve armazenar a coordenada geográfica de início de gravação.

##### 3.1.2 REQUISITOS DE HARDWARE

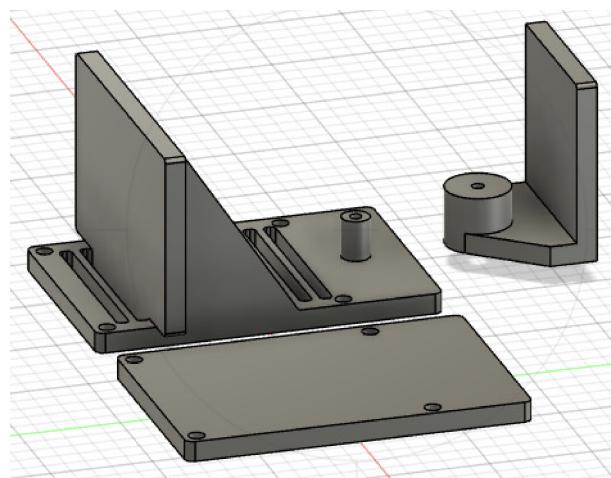
1. RF06: O sistema deve emitir um alerta sonoro ao detectar uma placa com avisos.
2. RF07: O sistema deve emitir um alerta visual em um display, evidenciando a placa detectada que seja suspeita.
3. RF08: Deve existir um botão para iniciar e encerrar gravações.
4. RF09: O sistema deve indicar no display que uma gravação está em andamento.

### 3.1.3 REQUISITOS NÃO FUNCIONAIS

1. RNF01: O sistema deve processar as imagens das placas localmente na placa Radxa ROCK 5C Lite.
2. RNF02: O sistema deve ser capaz de se conectar ao WiFi para atualizar o banco de dados local.
3. RNF03: O sistema deve se conectar a um banco de dados Postgress remoto quando estiver com acesso WiFi.
4. RNF04: O sistema deve ter suporte a microSD para armazenar as imagens gravadas.
5. RNF05: Um buzzer deve ser usado para emitir um alarme sonoro.
6. RNF06: Um display LCD deve ser usado para evidenciar a placa com problemas e indicar início de gravação.
7. RNF07: O sistema deverá se comunicar com o módulo GPS por interface serial.

## 3.2 PROJETO MECÂNICO

A parte mecânica do projeto é simples, restringindo-se a uma case impressa em 3D para suportar a placa Radxa ROCK 5C Lite, câmera, o LCD (responsável pelo feedback visual), o botão e o buzzer(responsável pelo feedback sonoro). Dessa forma, ela permite o suporte de uma webcam e a rotação no eixo horizontal da mesma. Além disso, possui uma região para fixação do painel LCD. É mostrada na Figura 7 o projeto da base a na Figura 8 sua impressão.



**Figura 7: Projeto 3D**

**Fonte: Autoria própria**

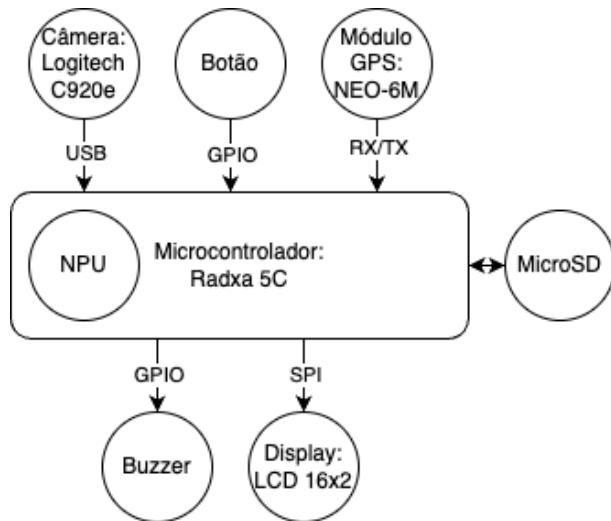


**Figura 8: Impressão 3D**

**Fonte: Autoria própria**

### 3.3 PROJETO DE HARDWARE

Assim como a parte mecânica, o projeto de hardware é simples. A Figura 9 apresenta o projeto do hardware desenvolvido com o microcontrolador, os periféricos e os protocolos utilizados em cada conexão.



**Figura 9: Projeto de Hardware**

**Fonte:** Autoria própria

### 3.3.1 RADXA ROCK 5C LITE

A Radxa ROCK 5C Lite é a unidade central de processamento do sistema Police Car Cam, responsável pela captura e análise de imagens veiculares. Sua principal vantagem é a presença de uma Neural Processing Unit (NPU), que otimiza a execução de algoritmos de inteligência artificial para detecção de placas em tempo real.

A placa gerencia a comunicação com dispositivos como o módulo GPS, o display LCD e o buzzer. O GPS fornece localização para registro de ocorrências, o display exibe informações relevantes e o buzzer alerta sobre veículos suspeitos.

Além disso, a conectividade Wi-Fi permite a atualização do banco de dados local, garantindo informações sempre atualizadas. Sua compatibilidade com periféricos e suporte a armazenamento via microSD tornam a implementação do sistema eficiente e adaptável.

#### 3.3.1.1 NEURAL PROCESSING UNIT

A *Neural Processing Unit* (NPU) é uma unidade de processamento especializada, projetada para acelerar cálculos relacionados a redes neurais e algoritmos de inteligência artificial. Diferente das Unidades Centrais de Processamento (CPUs) e Unidades de Processamento Gráfico (GPUs), as NPUs são otimizadas para operações altamente paralelizáveis, como multiplicações de matrizes e convoluções, comuns em redes neurais profundas.

A NPU integrada na Radxa ROCK 5C Lite permite a execução eficiente de inferências de modelos, em formato "rknn", de aprendizado de máquina diretamente no dispositivo, reduzindo a latência e o consumo de energia em comparação com a execução dessas tarefas em uma CPU ou GPU. Essa otimização é alcançada por meio de circuitos dedicados para operações matemáticas de baixa precisão, como cálculos em inteiros de 8 bits, preservando a precisão suficiente para a maioria dos modelos de inteligência artificial.

Além disso, ao processar tarefas de inteligência artificial localmente, a NPU reduz a necessidade de conexão constante com a nuvem. A presença de uma NPU na Radxa ROCK 5C Lite torna a placa uma excelente escolha para um projeto que envolve visão computacional.

### 3.3.2 MÓDULO GPS NEO-6M

O módulo GPS Neo-6M fornece as coordenadas geográficas necessárias para registrar a localização dos eventos. Ele utiliza a interface UART, operando com o padrão RS232, para transmitir dados conforme o protocolo NMEA (National Marine Electronics Association). Esse protocolo envia mensagens padronizadas em formato de texto, contendo informações de latitude, longitude, velocidade e tempo, garantindo uma comunicação simples e confiável com o sistema.

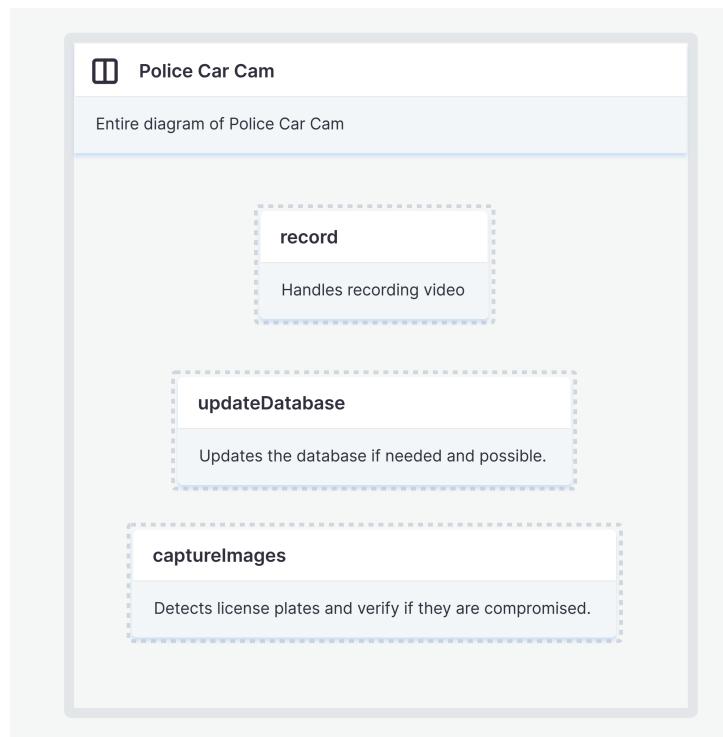
Devido a sua baixa resistência, o módulo GPS foi danificado diversas vezes ao longo do desenvolvimento. Por esse motivo, não foi possível finalizar o projeto utilizando o mesmo.

## 3.4 PROJETO DE SOFTWARE

Em contrapartida, o projeto de software é composta por diversas camadas. A seguir, serão apresentadas todas as funcionalidades que integram o Police Car Cam. Três funções operam de maneira concorrente, conforme ilustrado pelos estados na Figura 10, os quais serão detalhados nas subseções a seguir.

### 3.4.1 ATUALIZAÇÃO DO BANCO DE DADOS LOCAL

A atualização do banco de dados local tem como objetivo manter as informações sobre veículos suspeitos sempre atualizadas, garantindo que o sistema possa operar mesmo em ambientes sem conexão com a internet. Como as viaturas nem sempre terão acesso à rede, foi implementado um mecanismo que verifica periodicamente a conectividade e, sempre que houver acesso à internet, sincroniza os dados locais com o banco de dados remoto. Para



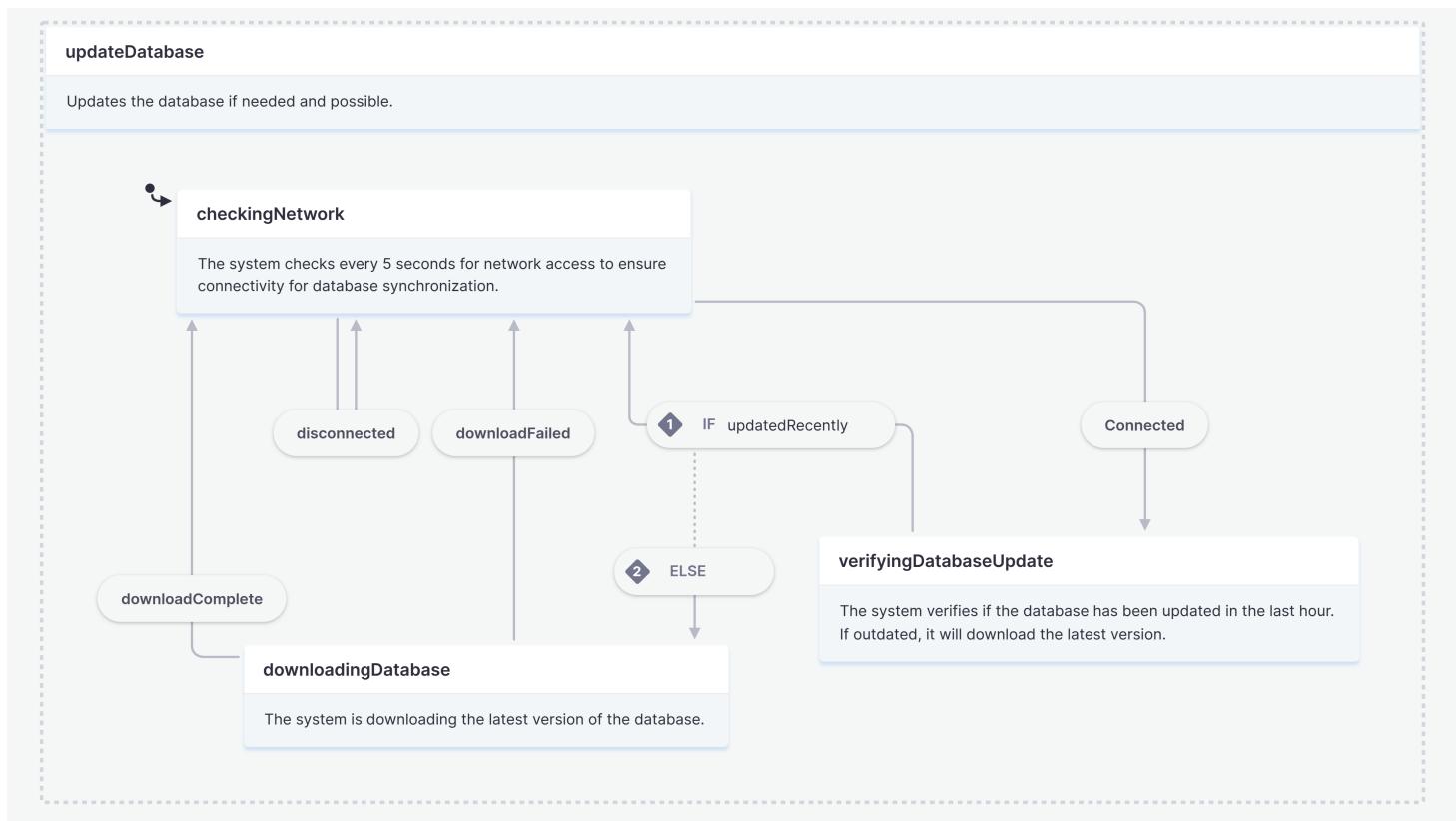
**Figura 10: Visão geral do sistema**

**Fonte: Autoria própria**

evitar atualizações excessivas em curtos períodos, um intervalo de espera é aplicado entre as sincronizações.

O fluxo de execução segue um modelo de máquina de estados (statechart), onde o estado *updateDatabase*, apresentado na Figura 11, é responsável por gerenciar a atualização dos dados. O processo ocorre em três etapas principais: primeiro, o sistema verifica a disponibilidade da conexão com a internet; em seguida, caso uma conexão esteja ativa, ele realiza a sincronização dos dados; por fim, um tempo de espera é imposto antes da próxima tentativa de atualização, prevenindo operações desnecessárias que poderiam sobrecarregar o sistema.

A estrutura do banco de dados foi projetada para armazenar informações detalhadas sobre veículos suspeitos, garantindo consultas rápidas e eficientes. O esquema das tabelas inclui registros de placas veiculares, status de irregularidade, timestamps de atualização e outros metadados relevantes para a fiscalização. Essa organização permite que o sistema realize buscas otimizadas e comparações rápidas entre as placas detectadas e os dados armazenados. A Figura 12 apresenta o diagrama do banco de dados, ilustrando a relação entre as tabelas e a forma como as informações são estruturadas.



**Figura 11: Estado de Atualização do Banco de Dados**

**Fonte:** Autoria própria

vehicle		vehicle_log	
id	bigint	id	bigint
plate	varchar	vehicle_id	bigint
model	varchar	severity	varchar
color	varchar	type	varchar
brand	varchar	description	varchar
year	bigint	created_at	date
owner	varchar	updated_at	date
created_at	date	deleted_at	date?
updated_at	date		
deleted_at	date?		

**Figura 12: Estrutura de Tabelas dos Bancos de Dados**

**Fonte:** Autoria própria

O banco de dados local foi implementado utilizando SQLite, uma solução leve e eficiente para armazenar e consultar dados diretamente no dispositivo embarcado. Já o banco de dados remoto utiliza PostgreSQL, uma alternativa robusta e escalável que suporta um grande volume de transações simultâneas. Essa arquitetura híbrida permite que o sistema funcione de maneira independente da conectividade e, ao mesmo tempo, mantenha os dados sempre atualizados quando possível.

### 3.4.2 GRAVAÇÃO DE IMAGEM E GEOLOCALIZAÇÃO

O sistema de gravação foi desenvolvido para capturar vídeos e registrar a geolocalização associada ao início da gravação. Existem duas formas de iniciar o processo: manualmente, por meio do acionamento de um botão, ou automaticamente, quando o sistema detecta uma placa suspeita. Em ambas as situações, as coordenadas geográficas do evento são obtidas pelo módulo GPS e adicionadas aos metadados do vídeo.

A execução do processo segue um fluxo baseado em máquina de estados (statechart). Uma função assíncrona monitora o estado do botão e, ao detectar sua ativação, inicia a gravação. O encerramento da gravação ocorre ao pressionar o botão novamente. A Figura 13 ilustra o estado de gravação e detalha seu funcionamento.

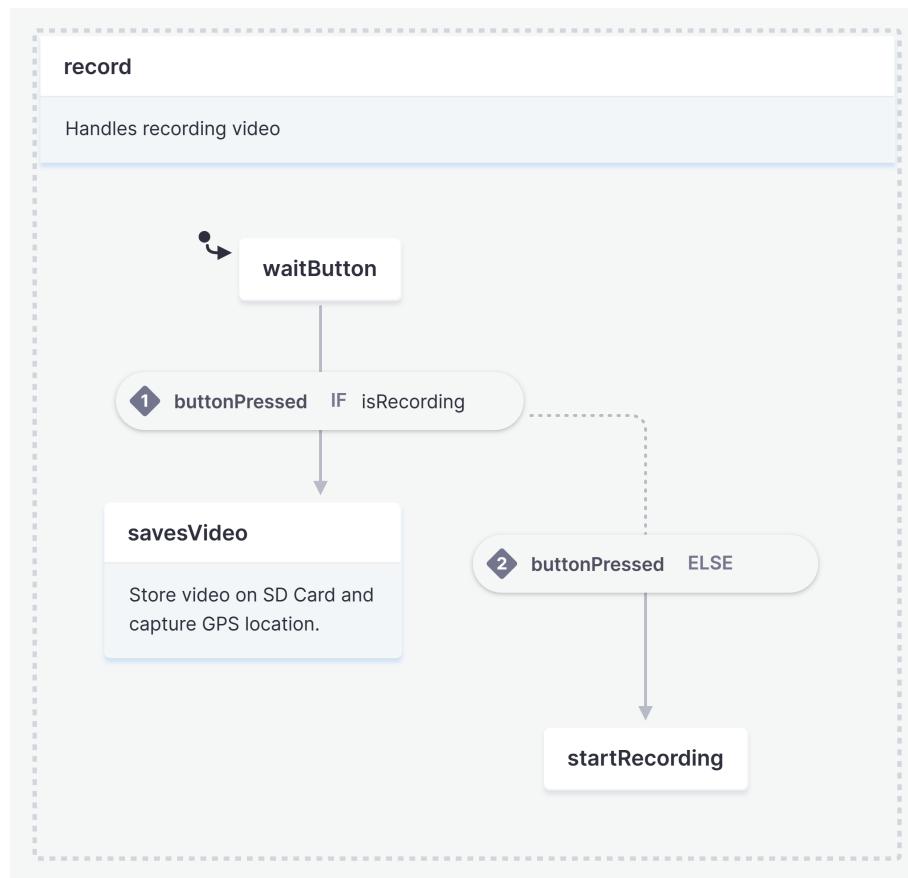
O sistema também possui um display LCD que informa em tempo real se a gravação está em andamento. Quando um vídeo está sendo capturado, o display exibe um indicativo visual, garantindo que o operador tenha consciência do estado atual do sistema.

### 3.4.3 IDENTIFICAÇÃO DAS PLACAS

Para identificar as placas e reconhecer seus caracteres, foram seguidas as etapas descritas no artigo (LAROCA et al., 2018):

1. Localização da placa
2. Recorte da placa
3. Segmentação dos caracteres
4. Identificação dos caracteres

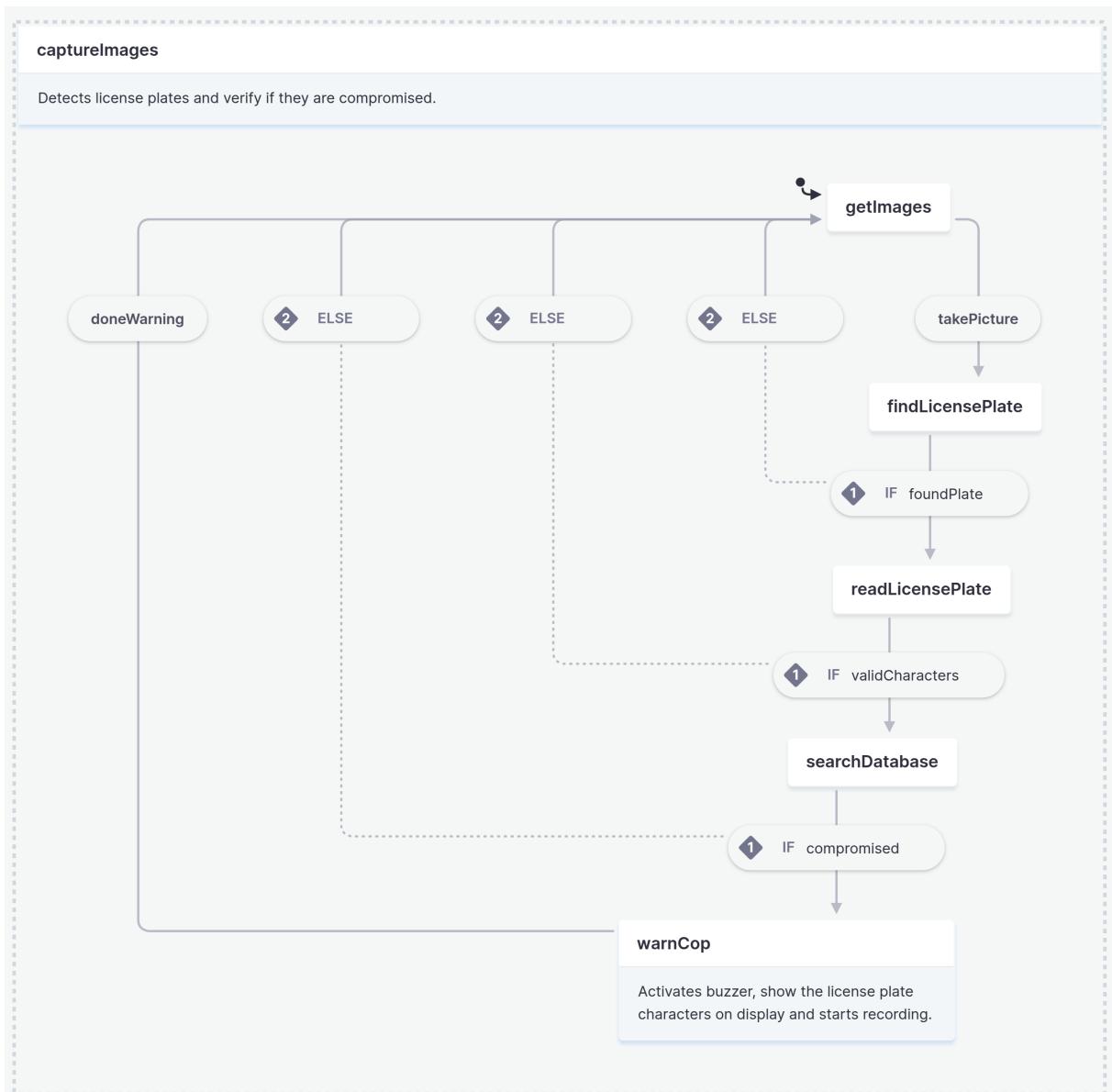
Para executar as etapas de localização, segmentação e identificação dos caracteres, foi necessário treinar um modelo próprio utilizando o framework Darknet para cada uma



**Figura 13: Estado de Gravação de Imagens**

**Fonte:** Autoria própria

delas. Após o treinamento inicial, o modelo foi convertido para um formato compatível com a Neural Processing Unit (NPU) presente na placa, por meio do RKNN Tool Kit – ferramenta fornecida pelo fabricante que possibilita a conversão de modelos desenvolvidos na Darknet para o formato RKNN, utilizado pela NPU. De forma semelhante, o estado que representa a captura de imagens, detecção e leitura das placas é ilustrado na Figura 14.



**Figura 14: Estado de Detecção de Placas**

**Fonte:** Autoria própria

### 3.4.4 DESENVOLVIMENTO E TREINAMENTO DAS REDES

Para o treinamento das Redes, serão utilizados quatro fontes principais: (LAROCA et al., 2018), (Laroca et al., 2021) , (REDMON, 2013–2016b), (CHARETTE, 2023–2025a). Em (LAROCA et al., 2018), é possível extrair a base da fundamentação teórica para o treinamento das redes, e a primeira definição para o número de ancoras que serão utilizadas para a detecção na rede YOLO. O número escolhido é  $A = 5$  que determinará quantos filtros a camada imediatamente anterior a camada de detecção usará. Na fonte (Laroca et al., 2021), é possível encontrar o detalhamento de como foi determinado no estudo o tamanho adequado da entrada da rede YOLO. As boas práticas determinadas em (CHARETTE, 2023–2025a), terminam por reforçar esse processo e revelam como apresentar resultados melhores para a detecção configurando o modelo. Por fim, em (REDMON, 2013–2016b) encontramos o tutorial de como utilizar o framework Darknet para treinar essas redes. A primeira rede foi treinada



**Figura 15: Exemplo de imagem anotada e visualizada no DarkMark**

**Fonte:** Autoria própria

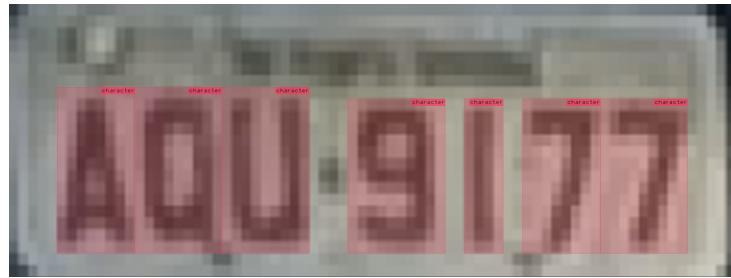
em cima do dataset sem nenhuma alteração nas imagens, apenas a separação e preparação dos *labels* específicos para esse treinamento, sendo utilizado um script simples em python para isso. Como a última versão do Darknet não contém a implementação da YOLOv2 e FAST-YOLO, utilizadas em (LAROCA et al., 2018), escolheremos os modelos mais leves que não comprometam o desempenho da rede. A rede escolhida foi a YOLOv3-tiny (REDMON;

FARHADI, 2018), que contém menos camadas que YOLOv3, mas entrega um modelo muito mais leve e rápido. Como base para o treinamento, foram utilizadas as 15 primeiras camadas pré-treinadas presentes no modelo *yolov3-tiny.conv.15* e foram treinadas apenas as camadas de detecção. Para as âncoras, foi empregado o algoritmo de clusterização *k-Means* para calcular os centroides das posições das placas em relação às imagens respectivas. A primeira camada de detecção receberá os dois últimos centroides, e a segunda, os três primeiros centroides. Calculando os filtros pela fórmula presente em (LAROCA et al., 2018), chega-se em 18 filtros para a camada mais externa e 12 filtros para a camada mais interna. Por último, a escolha do tamanho da entrada foi expandindo o tamanho da rede original para 416x448 pixels. Utilizando a ferramenta DarkMark (CHARETTE, 2019-2024) para visualização das imagens já anotadas, foi possível ver que o tamanho médio das instâncias anotadas não superava os 16x16 pixels necessários para a rede YOLO detectar o objeto adequadamente (CHARETTE, 2023–2025a). Aumentando em 32 pixels de altura, chegamos na dimensão média de 16 pixels para a altura das placas. Isso ocorre pois, as imagens em alta-definição do treinamento são reescaladas para serem operadas pela rede YOLO. Outro ponto importante é que a documentação recomenda fortemente a utilização de redes com tamanho inteiro proporcional a 32x32, conforme foi adotado pelo projeto.

Para o treinamento, foram utilizado separadamente dois dispositivos: Um notebook com uma placa NVIDIA RTX3060 6Gb, porém a placa interna sobreaquecia e o sistema era congelado forçadamente. Para contornar o problema, foi necessário ventilação externa forçada. O treinamento durou aproximadamente um dia e meio, onde foram realizadas 200 mil iterações

O segundo passo é a segmentação de caracteres. Agora, a placa detectada anteriormente foi recortada e recebe anotações sobre os sete caracteres. Os passos anteriores serão repetidos: Determina-se o tamanho da entrada da rede, número de âncoras, calcula-se o número de filtros na camada de detecção e aplica-se o algoritmo ao dataset para extração de centroides. O tamanho da entrada da rede foi escolhido como 256x96, já que as placas medem aproximadamente 240x80 (Laroca et al., 2021), foram escolhidos os menores múltiplos de 32 que aproxima o tamanho indicado. Os filtros são a mesma quantidade que anteriormente, já que ainda é uma classe e cinco âncoras. A rede pré-treinada utilizada será a mesma. Como a rede era significativamente menor, o treinamento é mais rápido.

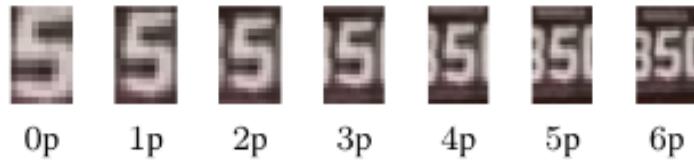
Para a rede de reconhecimento de caracteres, os mesmos passos anteriores foram executados. Foi utilizada a mesma rede e pré-treinamento, porém diferem no número de classes e número de filtros. Como existem 26 classes, uma para cada letra do alfabeto, o número de filtros será recalculado. Aplicando a fórmula, chegamos em 62 filtros para a camada de detecção



**Figura 16: Exemplo de placa anotada e visualizada no DarkMark**

**Fonte:** Autoria própria

mais interna e 93 filtros para a camada, ainda utilizando duas e três âncoras, respectivamente para cada camada de detecção. Um ponto importante para o treinamento foi o emprego de *paddings* para a fabricação do dataset utilizado para esse treino. As imagens antes recortadas na placa, agora seriam recortadas nas letras, o que poderia produzir quadros de detecção muito justos e piorar o desempenho da rede. Inspirados em (LAROCA et al., 2018), utilizamos alguns



**Figura 17: Exemplo de possíveis valores para *paddings***

**Fonte:** (LAROCA et al., 2018)

tamanhos de *paddings* para a mesma imagem, aumentando o tamanho da imagem em relação à letra à ser detectada e possibilitando a detecção contornar erros produzidos por variações na segmentação. Como a camada anterior produziria imagens com tamanho aproximado entre 16x96 e 32x96, foi definido como tamanho de entrada 32x96.

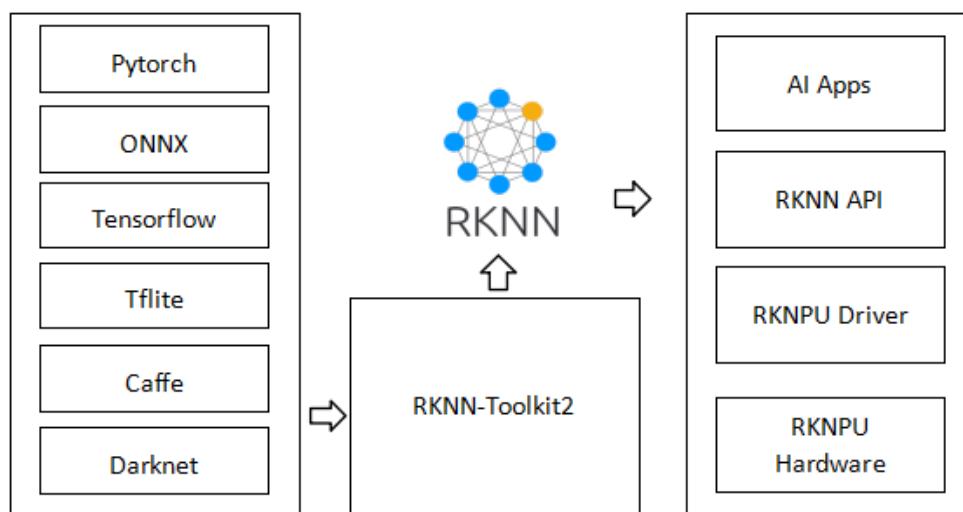
Não é necessário uma rede maior de entrada e maior em número de camadas, já que serão 26 classes comparadas às 80 classes do dataset MSCOCO (LIN et al., 2014) que o YOLOv3-tiny é originalmente treinado. Além disso, se o treinamento anterior garantir que sempre exista um objeto de letra a ser detectado na imagem devolvida ao próximo estágio, não é necessário a preocupação com redes mais complexas.

Por último, a rede de reconhecimento de dígitos numéricos: A rede é muito similar às anteriores, só que foram otimizadas camadas para minimizar o tamanho dos parâmetros da rede. Diferentemente do caso anterior, tem-se apenas 10 classes para detecção, o que facilita a tarefa. Nos outros passos, de reconhecimento da placa e segmentação dos caracteres, era estritamente

necessário uma alta precisão, garantida pela rede *YOLOv3-tiny*. Agora, será utilizado uma rede YOLO-v3-tiny-PRN (HAN et al., 2020), e como pré-treinamento, será empregado o modelo *yolov3-tiny.conv.11* (FUNKYFLAPJACKS, 2020). A primeira camada de detecção recebe 30 filtros e a segunda 45 filtros, respeitando as mesmas regras anteriores para âncoras.

### 3.4.5 INFERÊNCIA DO MODELO NA PLACA

Para que os modelos fossem inferidos diretamente na placa, tornou-se necessário convertê-los para o formato RKNN. Para esse fim, a API fornecida pelo fabricante (Rockchip Electronics Co., Ltd, 2024b) foi utilizada em um computador para realizar a conversão dos modelos para o formato compatível com a NPU RK3588. Em seguida, a API foi configurada na placa para carregar e executar os modelos convertidos. A Figura 18 ilustra o processo completo: um modelo originalmente desenvolvido para CPU/GPU é convertido, por meio da API (Rockchip Electronics Co., Ltd, 2024b), para o formato RKNN e, posteriormente, exportado para a placa, onde é processado pela NPU. Cada um desses passos serão explicados a seguir.



**Figura 18: Infraestrutura de software RKNN**

**Fonte:** (Rockchip Electronics Co., Ltd, 2024a)

Para converter modelos para o formato RKNN, há a etapa, chamada de quantização, para otimizar o desempenho na NPU. Esse processo consiste em mapear os valores contínuos dos pesos e ativações para um conjunto discreto de valores, geralmente com precisão reduzida (como inteiros sem sinal de 8 bits, uint8), o que resulta em uma inferência mais eficiente em termos de processamento e uso de memória.

Como a NPU é otimizada para operar com determinados tipos de valores, optou-se, neste projeto, pela conversão para uint8, com valores na faixa de 0 a 255.

A API disponibilizada pelo fabricante permite configurar diferentes algoritmos de quantização (Rockchip Electronics Co., Ltd, 2024a):

- **Normal:** Realiza uma quantização simples, mantendo uma relação direta entre os valores originais e os quantizados. Esse método é adequado para modelos que não são altamente sensíveis à redução de precisão.
- **KL Divergence:** Utiliza a divergência de Kullback-Leibler para definir os pontos de corte ideais durante a quantização, minimizando a discrepância entre as distribuições dos dados originais e quantizados. Esse algoritmo busca um equilíbrio entre eficiência computacional e manutenção da acurácia.
- **MMSE:** Baseado no erro quadrático médio (*Minimum Mean Squared Error*), esse método seleciona os parâmetros de quantização que minimizam o erro entre os valores originais e os quantizados, oferecendo uma abordagem mais refinada para modelos que exigem alta precisão.

Foram geradas versões do modelo utilizando todas essas abordagens, inclusive uma versão sem quantização. A escolha recaiu sobre o algoritmo KL Divergence, pois a versão sem quantização apresentava um tempo de processamento significativamente maior, com qualidade semelhante à obtida pelo MMSE. Além disso, o KL Divergence demonstrou maior capacidade na identificação correta das placas do que a quantização simples, mesmo com uma leve queda no desempenho de tempo, e obteve desempenho superior ao MMSE, com redução mínima na qualidade.

Diante desse cenário, o algoritmo KL Divergence revelou-se a opção mais vantajosa, permitindo processar um número maior de imagens no mesmo intervalo de tempo e compensando a ligeira perda de precisão em relação ao MMSE. Ademais, considerando que uma das principais causas de falhas na identificação de placas é a baixa qualidade das fotos, torna-se mais estratégico processar um volume maior de imagens, possibilitando a aplicação do modelo em diversas amostras, do que depender de poucas imagens de alta qualidade para um modelo mais sofisticado.

Para aprimorar a quantização, foi utilizado um conjunto de dados de calibração composto por imagens selecionadas do dataset original. Esse conjunto é essencial, pois fornece dados representativos do uso real, permitindo a determinação de parâmetros ideais para as

ativações e pesos do modelo. Dessa forma, obtém-se uma estimativa precisa da faixa de valores, minimizam-se os erros de quantização e mantém-se o desempenho e a robustez do modelo próximo ao original.

A quantidade de imagens necessárias para o dataset de calibração variou conforme o algoritmo e o modelo: aproximadamente 150 imagens foram utilizadas para os métodos Normal e KL Divergence, enquanto para o MMSE foram empregadas cerca de 70 imagens.

Para utilizar o modelo na placa, foi empregada a API fornecida pelo fabricante, na versão adaptada para a placa, RKNN-Toolkit2-Lite (Rockchip Electronics Co., Ltd, 2024b). Com essa API, foi desenvolvido o sistema ilustrado na Figura 14. Um aspecto importante do sistema é que, se em uma imagem o modelo não gerar um resultado satisfatório – ou seja, se o modelo de reconhecimento de placas não identificar nenhuma placa ou se o modelo de segmentação de caracteres não detectar os 7 caracteres esperados – o processamento é interrompido sem a execução dos demais modelos. Essa estratégia permite a captura de novas imagens, aumentando as chances de detectar e ler as placas corretamente.

## 4 RESULTADOS

O desenvolvimento do Police Car Cam demonstrou a viabilidade e eficácia do sistema proposto para reconhecimento automático de placas veiculares. Desde a fase de concepção até a integração final dos componentes, o projeto atingiu seus objetivos principais, resultando em uma ferramenta funcional e de grande potencial para aplicação prática.

### 4.1 DESEMPENHO DO SISTEMA

Os testes realizados indicaram que o modelo de inteligência artificial foi capaz de detectar placas veiculares com considerável precisão. O uso da rede YOLOv3-Tiny modificada permitiu um equilíbrio entre desempenho e tempo de resposta, garantindo que as identificações ocorressem em tempo real, aspecto fundamental para o uso em viaturas policiais.

O processamento de imagens na Radxa ROCK 5C Lite mostrou-se eficiente, permitindo que o sistema analisasse e reconhecesse placas rapidamente. A otimização do modelo de detecção para a Neural Processing Unit (NPU) embutida contribuiu significativamente para a redução da latência, o que resultou em tempos de inferência menores e em um desempenho mais fluido.

#### 4.1.1 DESEMPENHO DOS TREINAMENTOS

Assim, após implementar o algoritmo de reconhecimento de placas usando `conf_thresh = 0.25` e `IoU threshold = 50 %`, chegamos a seguinte tabela de métricas:

**Tabela 1: Desempenho do Modelo de Reconhecimento de Placas**

Name	AvgPrecision	Accuracy	ErrorRate	Precision	Recall
license_plate (test)	95.3289	0.9014	0.0986	0.9533	0.9411
license_plate (val)	89.8353	0.8285	0.1715	0.9101	0.9000

Sendo assim, 89% de Precisão Média no dataset de validação são satisfatórios para a aplicação.

Para além da detecção das placas, foram necessárias mais oito horas de treinamento para a segmentação e realizadas 200 mil iterações e as seguintes métricas foram calculadas: Novamente, os 95% de Precisão Média no dataset de validação são satisfatórios para a

**Tabela 2: Métricas de Desempenho da Segmentação**

Name	AvgPrecision	Accuracy	ErrorRate	Precision	Recall
character(test)	95.3467	0.9239	0.0761	0.9657	0.9519
character(val)	95.3761	0.9254	0.0746	0.9627	0.9579

aplicação.

Seguindo o processo de desenvolvimento, agora serão apresentadas as métricas para a detecção das 26 letras:

Name	AvgPrecision	Accuracy	ErrorRate	Precision	Recall
A	99.9007	0.9771	0.0229	0.9688	0.9954
B	99.7767	0.9699	0.0301	0.9416	0.9942
C	98.8625	0.9735	0.0265	0.9276	0.9400
D	97.4744	0.9385	0.0615	0.7745	0.9733
E	98.8819	0.9195	0.0805	0.6608	1.0000
F	100.0000	0.9890	0.0110	0.8671	1.0000
G	93.6165	0.9820	0.0180	0.9609	0.8200
H	100.0000	0.9990	0.0010	0.9868	1.0000
I	87.9713	0.9222	0.0778	0.8772	0.6667
J	55.3560	0.8161	0.1839	0.9923	0.2867
K	16.2283	0.8614	0.1386	0.1358	0.2200
L	99.3287	0.9830	0.0170	0.8671	1.0000
M	85.0692	0.8024	0.1976	0.6333	0.7920
N	84.3306	0.8929	0.1071	0.9861	0.4733
O	94.7035	0.9549	0.0451	0.8373	0.9267
P	99.5704	0.9831	0.0169	0.9167	0.9900
Q	98.7512	0.9921	0.0079	0.9494	1.0000
R	31.7710	0.9215	0.0785	0.3729	0.1467
S	98.8044	0.9883	0.0117	0.9756	0.9756
T	70.9475	0.8704	0.1296	0.7886	0.4600
U	99.9920	0.9959	0.0041	0.9885	1.0000
V	98.4021	0.9498	0.0502	0.9013	0.9433
W	48.3119	0.8124	0.1876	0.8142	0.3311
X	91.2750	0.8433	0.1567	0.9270	0.6267
Y	99.9016	0.9386	0.0614	0.7719	1.0000
Z	97.2320	0.9578	0.0422	0.8540	0.9850
Media/Classe(%)	97.93825	95.235	4.765	88.925	95.83

**Tabela 3: Métricas de desempenho dos modelos**

Vemos na tabela que a maior parte das classes convergiu para valores superiores a 90%

de Precisão média, chegando em 97.9% de AvgPrecision por classe, o que é muito satisfatório para nosso objetivo.

Por fim, serão exibidas as métricas para a detecção do números

Name	AvgPrecision	Accuracy	ErrorRate	Precision	Recall
0	98.4628	0.9619	0.0381	0.9315	0.9154
1	95.5359	0.9039	0.0961	0.9274	0.8377
2	89.3854	0.9047	0.0953	0.7244	0.9738
3	97.8007	0.9607	0.0393	0.8492	0.9260
4	97.7796	0.9757	0.0243	0.9193	0.9531
5	98.4717	0.9763	0.0237	0.9539	0.9470
6	99.4827	0.9726	0.0274	0.9585	0.9729
7	86.0941	0.9231	0.0769	0.8148	0.8250
8	97.2548	0.9555	0.0445	0.9662	0.8853
9	95.4168	0.9216	0.0784	0.8941	0.8966
Media/Classe(%)	97.5172	95.81	4.19	92.335	92.07

**Tabela 4: Resultados de precisão, acurácia e taxa de erro por classe.**

Os resultados foram satisfatórios, alcançando 97% *AvgPrecision* médio por classe, o que é muito satisfatório. O treinamento foi iterado 100 mil vezes.

## 4.2 IMPLEMENTAÇÃO E INTEGRAÇÃO DOS COMPONENTES

A integração entre hardware e software foi bem-sucedida, refletindo a solidez do planejamento realizado. O sistema incorporou uma câmera para captura de imagens, um módulo de GPS para geolocalização das ocorrências e um display LCD para apresentar informações relevantes ao operador.

A interface física, projetada por meio de impressão 3D, acomodou todos os componentes de maneira organizada e funcional, garantindo que o dispositivo pudesse ser facilmente instalado e operado dentro de uma viatura.

O único contra tempo encontrado na integração dos componentes foi relacionado ao módulo GPS, o qual apresenta grande sensibilidade e foi queimado 2 vezes. Sendo assim, decidimos não usá-lo mais.

#### 4.3 TESTES PRÁTICOS

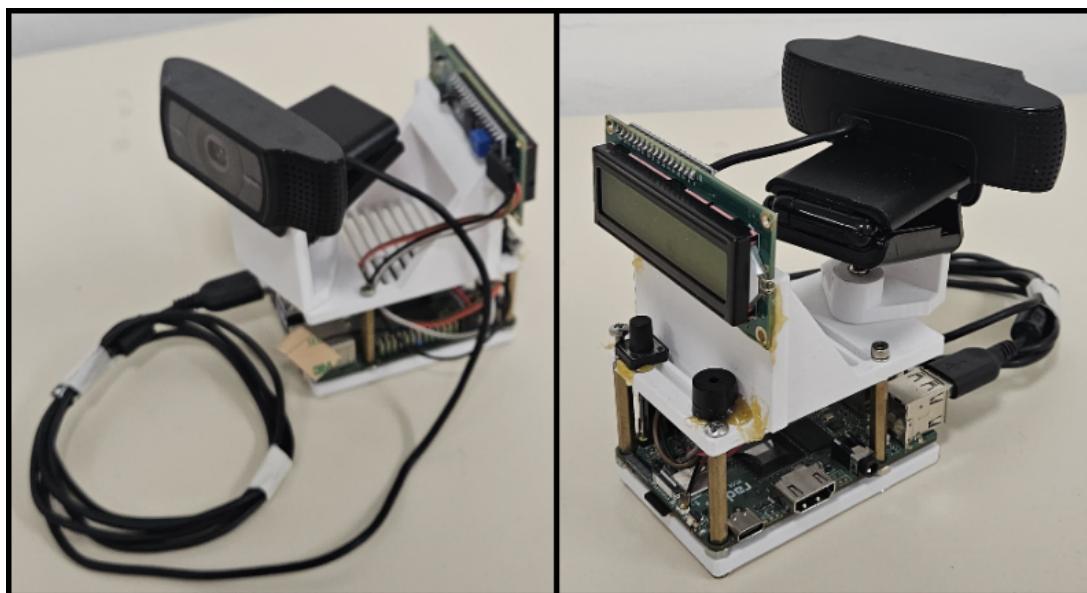
Durante os testes em ambiente real, o Police Car Cam demonstrou ser uma ferramenta promissora para o reconhecimento de veículos em situação irregular. O sistema foi capaz de identificar corretamente placas de diferentes modelos de veículos.

A interface simplificada permitiu que os operadores acionassem facilmente o sistema de gravação, registrando vídeos com geolocalização quando necessário. A funcionalidade de atualização do banco de dados local, sempre que uma conexão com a internet estava disponível, garantiu que as informações utilizadas pelo sistema estivessem sempre atualizadas.

Embora o módulo GPS tenha apresentado algumas limitações de durabilidade ao longo do desenvolvimento, o restante do sistema operou de forma estável e confiável, confirmando a robustez da solução.

#### 4.4 APRESENTAÇÃO DOS RESULTADOS

Para ilustrar os avanços alcançados, a Figura 19 apresenta as imagens do projeto finalizado, destacando o design compacto e a disposição dos componentes.



**Figura 19: Resultado Final**

**Fonte:** Autoria própria

Além disso, um vídeo demonstrativo foi produzido para apresentar o sistema. Esse vídeo pode ser acessado em <https://www.youtube.com/watch?v=UXRZuLZrKa8>

## 5 CONCLUSÃO

O projeto Police Car Cam atingiu seus objetivos principais, desenvolvendo um sistema eficiente para identificação de placas veiculares utilizando inteligência artificial e processamento de imagens. A implementação do sistema permitiu a detecção e reconhecimento de placas em tempo real, demonstrando sua viabilidade para auxiliar a fiscalização de veículos irregulares.

Durante o desenvolvimento, desafios técnicos foram superados, como a otimização dos modelos de inteligência artificial para execução em hardware embarcado e a integração dos diferentes componentes do sistema. Os testes realizados indicaram um desempenho satisfatório, com aceitável taxa de precisão na identificação das placas e uma resposta rápida do sistema.

Apesar dos resultados positivos, algumas limitações foram identificadas, especialmente em relação à qualidade das imagens capturadas e à comunicação entre sistema e usuário. Essas questões indicam oportunidades de aprimoramento e possíveis direções para trabalhos futuros.

### 5.1 TRABALHOS FUTUROS

O projeto Police Car Cam apresentou resultados promissores na identificação de placas veiculares utilizando inteligência artificial e visão computacional. No entanto, há diversas possibilidades de aprimoramento que podem elevar a precisão, robustez e usabilidade do sistema. Esta seção apresenta os principais pontos a serem trabalhados em versões futuras do projeto.

#### 5.1.1 MELHORAR A DETECÇÃO DE PLACAS

Para aumentar a acurácia do modelo de reconhecimento de placas, pretende-se expandir o conjunto de dados utilizados no treinamento. Além do UFPR-ALPR, serão incorporadas imagens coletadas em diferentes cenários. Além disso, será necessário contemplar

placas de motos e no formato Mercosul.

A atual implementação utiliza a arquitetura YOLOv3-tiny devido à sua leveza e eficiência. No entanto, modelos mais modernos, como YOLOv5, YOLOv8 e EfficientDet, serão testados para avaliar melhorias no equilíbrio entre tempo de inferência e precisão.

Atualmente, o sistema opera com a câmera Logitech C920e, que, embora funcional, pode ser substituída por modelos com maior resolução e melhor desempenho em baixa luminosidade. Sensores com HDR (High Dynamic Range) e estabilização óptica serão analisados para garantir capturas mais nítidas e precisas mesmo com o movimento do veículo.

### 5.1.2 MELHORAR A INTERAÇÃO COM O USUÁRIO

A substituição do display LCD 16x2 por uma tela maior, como um LCD TFT de 5 ou 7 polegadas, possibilitará uma interface gráfica mais intuitiva. Isso permitirá a exibição de informações adicionais, como imagens capturadas, status da conexão GPS e indicadores de alerta.

O suporte da câmera e da placa eletrônica será redesenhado para melhorar a estabilidade e minimizar vibrações, garantindo imagens mais nítidas. Além disso, o sistema de fixação permitirá ajustes no ângulo da câmera, adaptando-se melhor às necessidades operacionais.

### 5.1.3 SUBSTITUIR O MÓDULO GPS

O módulo NEO-6M demonstrou limitações de precisão e durabilidade. Como alternativa, serão testados módulos GPS de alta precisão, como o u-blox NEO-M8N ou ZED-F9P, que oferecem melhor captação de sinal e suporte para correção diferencial GNSS (RTK). Além disso, medidas de proteção contra interferências e vibrações serão implementadas para garantir maior vida útil ao componente.

## 6 CRONOGRAMA, CUSTOS E GESTÃO DO PROJETO

### 6.1 CRONOGRAMA

No início do projeto, foi elaborado um cronograma com o objetivo de distribuir as tarefas entre os membros da equipe de maneira equilibrada, garantindo que todos tivessem uma carga de trabalho semelhante e pudessem atuar nas atividades de sua preferência. Para facilitar a compreensão do cronograma, foi criada uma legenda simples para indicar o status de cada etapa ao longo das semanas, conforme ilustrado na Figura 20.

Legenda:	<b>Finalizado</b>	<b>Em progresso</b>	<b>Atrasado</b>	<b>Iniciar tarefa</b>
----------	-------------------	---------------------	-----------------	-----------------------

**Figura 20: Legenda do Cronograma**

**Fonte:** Autoria própria

O cronograma foi dividido essencialmente em 7 partes, tendo em vista marcos para representar o desenvolvimento de cada etapa. A primeira foi a elaboração do plano de projeto, como ilustra a Figura 21.

PP: Plano de Projeto detalhado								PP	E1				E2	E3	E4	E5	Apresentação		
Atividade	Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/10	30/10	06/11	13/11	20/11	27/11	04/12	11/12	18/12	05/02	12/02	19/02	26/02
Levantamento de Requisitos	Todos	-	3	1	4	15													
Análise de Riscos	Enzo	Bruno	3	1	4	3													
Declaração de Escopo de Alto Nível	Felipe	Enzo	2	1	3	1													
Analizar Conhecimentos Necessários para o projeto	Todos	-	2	1	3	3													
Criação do Cronograma	Todos	-	6	2	8	10													
Analise de materiais e métodos	Bruno	Felipe	2	1	3	5													
Diagrama em Blocos	Felipe	Ian	2	1	3	2													
Elaboração dos Slides	Todos	-	5	2	7	2													
Compra dos Materiais	Todos	-	1	0	1	1													
<b>Tempo total</b>			26	10	35	42													

**Figura 21: Plano de Projeto**

**Fonte:** Autoria própria

Para o segundo marco, a intenção era finalizar a elaboração do blog do projeto até o dia 06/11/2024, conforme mostrado na Figura 22

											PP	E1				E2	E3	E4	E5	Apresentação	
<b>E1: Site/blog de acompanhamento</b>		Atividade	Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/10	30/10	06/11	13/11	20/11	27/11	04/12	11/12	18/12	05/02	12/02	19/02	26/02
Criação das seções	Todos	-			1	0	1	1													
Elaboração do Blog	Bruno	-			1	0	1	1													
<b>Tempo total</b>					2	0	2	2													

**Figura 22: Elaboração do Blog****Fonte: Autoria própria**

Na etapa seguinte, o objetivo foi finalizar a estrutura física do projeto. Embora tenha ocorrido um atraso na conclusão do projeto 3D do suporte para a câmera, a atividade foi concluída conforme o cronograma da Figura 23.

											PP	E1				E2	E3	E4	E5	Apresentação	
<b>E2: Estrutura física</b>		Atividade	Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/10	30/10	06/11	13/11	20/11	27/11	04/12	11/12	18/12	05/02	12/02	19/02	26/02
Projeto do suporte em 3D para placa ficar fixa no painel	Ian	Felipe			15	5	20	5													
Aquisição dos Componentes eletrônicos	Bruno	Enzo			5	2	7	2													
Projeto 3D para suporte câmera	Ian				5	2	7	2													
Atualizar o Blog	Todos	-			1	0	1	1													
<b>Tempo total</b>					26	9	35	10													

**Figura 23: Estrutura Física****Fonte: Autoria própria**

Em seguida, a prototipagem foi realizada dentro do prazo estabelecido (Figura 24).

											PP	E1				E2	E3	E4	E5	Apresentação	
<b>E3: Prototipagem</b>		Atividade	Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/10	30/10	06/11	13/11	20/11	27/11	04/12	11/12	18/12	05/02	12/02	19/02	26/02
Teste de captura de imagens	Felipe	Ian			10	3	13	4													
Teste de gravação de imagens	Bruno	Felipe			10	3	13	4													
Criação do Banco de Dados	Enzo	Bruno			3	1	4	3													
Atualizar o Blog	Todos	-			1	0	1	1													
<b>Tempo total</b>					24	7	31	12													

**Figura 24: Prototipagem****Fonte: Autoria própria**

Posteriormente, iniciou-se a etapa que demandou maior dedicação dos integrantes.

Apesar de um leve atraso em um dos objetivos, a entrega ocorreu uma semana depois (Figura 25).

Na fase final anterior à entrega, foram realizadas atividades de finalização (Figura 26).

Por fim, concluíram-se as etapas relativas ao relatório, à apresentação e à gravação do vídeo de demonstração do protótipo (Figura 27).

						PP	E1															
				Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/10	30/10	06/11	13/11	20/11	27/11	04/12	11/12	18/12	05/02	12/02	19/02	26/02
E4: Desenvolvimento na placa																						
Atividade	Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/10	30/10	06/11	13/11	20/11	27/11	04/12	11/12	18/12	05/02	12/02	19/02	26/02			
Criação dos Diagramas	Ian	Ian	5	2	7	4																
Atualização do Banco de Dados via WiFi	Felipe	Enzo	3	1	4	3																
Criação do código inicial de identificação de placas	Todos	-	5	2	7	10																
Desenvolvimento do código de reconhecimento de placa	Todos	-	10	3	13	25																
Configuração do Cartão SD	Enzo	Felipe	3	1	4	2																
Acionamento de Buzzer e Botão	Bruno	Ian	2	1	3	2																
Configuração do Display LCD	Enzo	Enzo	2	1	3	2																
Configuração do Módulo GPS	Felipe	Bruno	2	1	3	3																
Alimentação dos Componentes	Enzo	Bruno	3	1	4	1																
Passar imagens da câmera para RADXA	Enzo	Bruno	5	2	7	5																
Atualizar o Blog	Todos	-	1	0	1	1																
<b>Tempo total</b>					<b>41</b>	<b>15</b>	<b>56</b>	<b>58</b>														

Figura 25: Desenvolvimento na Placa

Fonte: Autoria própria

						PP	E1															
				Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/10	30/10	06/11	13/11	20/11	27/11	04/12	11/12	18/12	05/02	12/02	19/02	26/02
E5: Integração das partes																						
Atividade	Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/10	30/10	06/11	13/11	20/11	27/11	04/12	11/12	18/12	05/02	12/02	19/02	26/02			
Código de reconhecimento de placas rodando na RADXA	Todos	-	8	3	11	15																
Atualização do banco de dados na RADXA	Todos	-	8	3	11	5																
Integração do GPS com a gravação de imagens	Todos	-	3	1	4	3																
Atualizar o Blog	Todos	-	1	0	1	1																
<b>Tempo total</b>					<b>20</b>	<b>7</b>	<b>27</b>	<b>24</b>														

Figura 26: Integração das Partes

Fonte: Autoria própria

						PP	E1															
				Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/10	30/10	06/11	13/11	20/11	27/11	04/12	11/12	18/12	05/02	12/02	19/02	26/02
Demonstração do funcionamento do protótipo																						
Atualizar o Blog	Todos	-	1	0	1	1																
Apresentar	Todos	-	1	0	1	1																
<b>Tempo total</b>					<b>2</b>	<b>0</b>	<b>2</b>	<b>2</b>														
RT: Relatório Técnico																						
Atividade	Responsável	Assistente	Duração prevista	Margem de erro	Total	Tempo levado	23/10	30/10	06/11	13/11	20/11	27/11	04/12	11/12	18/12	05/02	12/02	19/02	26/02			
Gravação do vídeo de demonstração	Todos	-	3	1	4	5																
Escrita do relatório final	Todos	-	30	10	40	30																
<b>Tempo total</b>					<b>33</b>	<b>11</b>	<b>44</b>	<b>35</b>														

Figura 27: Entrega do Projeto

Fonte: Autoria própria

## 6.2 CUSTOS

Em relação aos custos, pode-se dizer que foi um bom orçamento, tendo em vista que entregou o esperado e seu benefício compensa. Na tabela 5 é possível ver o custo de cada material, que, no total, resultou em R\$790,00.

Os custos apresentados tiveram algumas diferenças se comparados aos custos estimados iniciais, haja vista que a equipe já possuía alguns itens e o módulo GPS apresentou problemas e necessitou ser trocado.

**Tabela 5: Lista de Materiais**

<b>Quantidade</b>	<b>Produto</b>	<b>Custo</b>
1	Radxa ROCK 5C Lite	R\$ 600,00
1	Buzzer	R\$ 10,00
1	Display LCD 16x2	R\$ 50,00
2	Módulo GPS	R\$ 50,00
1	Botão	R\$ 5,00
1	Estrutura 3D	R\$ 25,00

### 6.3 GESTÃO DO PROJETO

A gestão do projeto foi conduzida com sucesso, garantindo uma distribuição eficiente das tarefas entre os integrantes, além de promover uma interação e integração satisfatórias entre as partes, como demonstrado na Tabela 6.

**Tabela 6: Tempo Total Estimado e Levado**

	<b>Tempo Total Estimado</b>		<b>Tempo Total Levado</b>	
	<b>Responsável</b>	<b>Assistente</b>	<b>Responsável</b>	<b>Assistente</b>
<b>Felipe</b>	72	40	46	16
<b>Ian</b>	64	26	44	12
<b>Bruno</b>	57	22	47	15
<b>Enzo</b>	56	17	49	8

## REFERÊNCIAS

- CHARETTE, S. **DarkMark - C Code Run.** 2019–2024. <http://www.ccoderun.ca/darkmark/>.
- CHARETTE, S. **Darknet/YOLO FAQ - C Code Run.** 2023–2025. [https://www.ccoderun.ca/programming/yolo\\_faq/](https://www.ccoderun.ca/programming/yolo_faq/).
- CHARETTE, S. **Hank AI, Inc.'s Darknet.** 2023–2025. <https://github.com/hank-ai/darknet>.
- Correio Braziliense. 2025. Acessado em: 12 fev. 2025. Disponível em: <<https://www.correiobraziliense.com.br/brasil/2022/12/5057545-564-mil-veiculos-foram-roubados-ou-furtados-no-brasil-em-2021.html>>.
- FUNKYFLAPJACKS. **Yolo BBD.** 2020. <https://github.com/FunkyFlapjacks/bbd/tree/master>.
- GOLASZEWSKI, B. **GPIOD: A Python library for GPIO management.** 2025. Acessado em: 14 fev. 2025. Disponível em: <<https://pypi.org/project/gpiod/>>.
- GONÇALVES, G. R. et al. A benchmark for license plate character segmentation. **CoRR**, abs/1607.02937, 2016. Disponível em: <<http://arxiv.org/abs/1607.02937>>.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning.** [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.
- HAN, X. et al. Sensor drift detection based on discrete wavelet transform and grey models. **IEEE Access**, v. 8, p. 204389–204399, 2020.
- LAROCA, R. et al. A robust real-time automatic license plate recognition based on the yolo detector. In: **2018 International Joint Conference on Neural Networks (IJCNN).** [S.l.: s.n.], 2018. p. 1–10.
- Laroca, R. et al. An efficient and layout-independent automatic license plate recognition system based on the YOLO detector. **IET Intelligent Transport Systems**, v. 15, n. 4, p. 483–503, 2021. ISSN 1751-956X.
- LIN, T. et al. Microsoft COCO: common objects in context. **CoRR**, abs/1405.0312, 2014. Disponível em: <<http://arxiv.org/abs/1405.0312>>.
- LOGITECH. **Webcam HD C920E.** 2022. Acessado em: 15 fev. 2025. Disponível em: <<https://www.farnell.com/datasheets/4420290.pdf>>.
- REDMON, J. **Darknet: Open Source Neural Networks in C.** 2013–2016. <http://pjreddie.com/darknet/>.

REDMON, J. **Train a Classifier on CIFAR-10**. 2013–2016. <https://pjreddie.com/darknet/train-cifar/>.

REDMON, J. et al. **You Only Look Once: Unified, Real-Time Object Detection**. 2016. Disponível em: <<https://arxiv.org/abs/1506.02640>>.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. **arXiv**, 2018.

Rockchip Electronics Co., Ltd. **RKNN SDK User Guide**. 2024. Acessado em: 15 fev. 2025. Disponível em: <<https://github.com/airockchip/rknn-toolkit2/tree/master/doc>>.

Rockchip Electronics Co., Ltd. **RKNN-Toolkit2 API Reference**. 2024. Acessado em: 15 fev. 2025. Disponível em: <<https://github.com/airockchip/rknn-toolkit2/tree/master/doc>>.

Rockchip Electronics Co., Ltd. **Radxa Rock 5C**. 2025. Acessado em: 12 fev. 2025. Disponível em: <<http://radxa.com/products/rock5/5c/>>.

SILVA, S. M.; JUNG, C. R. Real-time brazilian license plate detection and recognition using deep convolutional neural networks. In: **2017 30th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)**. [S.l.: s.n.], 2017. p. 55–62.

SILVA, S. M.; JUNG, C. R. License plate detection and recognition in unconstrained scenarios. In: **2018 European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2018. p. 580–596.

## APÊNDICE A – CONFIGURAÇÃO REDES YOLO

**Tabela 7: YOLOv3-Tiny modificada para Reconhecimento de Placas**

Camada	Filtros	Tamanho	Stride	Ativação	Entrada / Saída
Convolucional	16	$3 \times 3$	1	Leaky	$416 \times 448 \times 3 \rightarrow 416 \times 448 \times 16$
MaxPool	-	$2 \times 2$	2	-	$416 \times 448 \times 16 \rightarrow 208 \times 224 \times 16$
Convolucional	32	$3 \times 3$	1	Leaky	$208 \times 224 \times 16 \rightarrow 208 \times 224 \times 32$
MaxPool	-	$2 \times 2$	2	-	$208 \times 224 \times 32 \rightarrow 104 \times 112 \times 32$
Convolucional	64	$3 \times 3$	1	Leaky	$104 \times 112 \times 32 \rightarrow 104 \times 112 \times 64$
MaxPool	-	$2 \times 2$	2	-	$104 \times 112 \times 64 \rightarrow 52 \times 56 \times 64$
Convolucional	128	$3 \times 3$	1	Leaky	$52 \times 56 \times 64 \rightarrow 52 \times 56 \times 128$
MaxPool	-	$2 \times 2$	2	-	$52 \times 56 \times 128 \rightarrow 26 \times 28 \times 128$
Convolucional	256	$3 \times 3$	1	Leaky	$26 \times 28 \times 128 \rightarrow 26 \times 28 \times 256$
MaxPool	-	$2 \times 2$	2	-	$26 \times 28 \times 256 \rightarrow 13 \times 14 \times 256$
Convolucional	512	$3 \times 3$	1	Leaky	$13 \times 14 \times 256 \rightarrow 13 \times 14 \times 512$
MaxPool	-	$2 \times 2$	1	-	$13 \times 14 \times 512 \rightarrow 13 \times 14 \times 512$
Convolucional	1024	$3 \times 3$	1	Leaky	$13 \times 14 \times 512 \rightarrow 13 \times 14 \times 1024$
Convolucional	256	$1 \times 1$	1	Leaky	$13 \times 14 \times 1024 \rightarrow 13 \times 14 \times 256$
Convolucional	512	$3 \times 3$	1	Leaky	$13 \times 14 \times 256 \rightarrow 13 \times 14 \times 512$
Convolucional	12	$1 \times 1$	1	Linear	$13 \times 14 \times 512 \rightarrow 13 \times 14 \times 12$
YOLO	-	-	-	-	Detecta objetos (Saída 1)
Route	-	-	-	-	Concatena camadas -4
Convolucional	128	$1 \times 1$	1	Leaky	$13 \times 14 \times 512 \rightarrow 13 \times 14 \times 128$
Upsample	-	-	2	-	$13 \times 14 \times 128 \rightarrow 26 \times 28 \times 128$
Route	-	-	-	-	Concatena camadas -1, 8
Convolucional	256	$3 \times 3$	1	Leaky	$26 \times 28 \times 256 \rightarrow 26 \times 28 \times 256$
Convolucional	18	$1 \times 1$	1	Linear	$26 \times 28 \times 256 \rightarrow 26 \times 28 \times 18$
YOLO	-	-	-	-	Detecta objetos (Saída 2)

**Tabela 8: YOLOv3-Tiny modificada para Segmentação de Caracteres**

<b>Camada</b>	<b>Filtros</b>	<b>Tamanho</b>	<b>Stride</b>	<b>Ativação</b>	<b>Entrada / Saída</b>
Convolucional	16	$3 \times 3$	1	Leaky	$256 \times 96 \times 3 \rightarrow 256 \times 96 \times 16$
MaxPool	-	$2 \times 2$	2	-	$256 \times 96 \times 16 \rightarrow 128 \times 48 \times 16$
Convolucional	32	$3 \times 3$	1	Leaky	$128 \times 48 \times 16 \rightarrow 128 \times 48 \times 32$
MaxPool	-	$2 \times 2$	2	-	$128 \times 48 \times 32 \rightarrow 64 \times 24 \times 32$
Convolucional	64	$3 \times 3$	1	Leaky	$64 \times 24 \times 32 \rightarrow 64 \times 24 \times 64$
MaxPool	-	$2 \times 2$	2	-	$64 \times 24 \times 64 \rightarrow 32 \times 12 \times 64$
Convolucional	128	$3 \times 3$	1	Leaky	$32 \times 12 \times 64 \rightarrow 32 \times 12 \times 128$
MaxPool	-	$2 \times 2$	2	-	$32 \times 12 \times 128 \rightarrow 16 \times 6 \times 128$
Convolucional	256	$3 \times 3$	1	Leaky	$16 \times 6 \times 128 \rightarrow 16 \times 6 \times 256$
MaxPool	-	$2 \times 2$	2	-	$16 \times 6 \times 256 \rightarrow 8 \times 3 \times 256$
Convolucional	512	$3 \times 3$	1	Leaky	$8 \times 3 \times 256 \rightarrow 8 \times 3 \times 512$
MaxPool	-	$2 \times 2$	1	-	$8 \times 3 \times 512 \rightarrow 8 \times 3 \times 512$
Convolucional	1024	$3 \times 3$	1	Leaky	$8 \times 3 \times 512 \rightarrow 8 \times 3 \times 1024$
Convolucional	256	$1 \times 1$	1	Leaky	$8 \times 3 \times 1024 \rightarrow 8 \times 3 \times 256$
Convolucional	512	$3 \times 3$	1	Leaky	$8 \times 3 \times 256 \rightarrow 8 \times 3 \times 512$
Convolucional	12	$1 \times 1$	1	Linear	$8 \times 3 \times 512 \rightarrow 8 \times 3 \times 12$
YOLO	-	-	-	-	Detecta objetos (Saída 1)
Route	-	-	-	-	Concatena camadas -4
Convolucional	128	$1 \times 1$	1	Leaky	$8 \times 3 \times 512 \rightarrow 8 \times 3 \times 128$
Upsample	-	-	2	-	$8 \times 3 \times 128 \rightarrow 16 \times 6 \times 128$
Route	-	-	-	-	Concatena camadas -1, 8
Convolucional	256	$3 \times 3$	1	Leaky	$16 \times 6 \times 256 \rightarrow 16 \times 6 \times 256$
Convolucional	18	$1 \times 1$	1	Linear	$16 \times 6 \times 256 \rightarrow 16 \times 6 \times 18$
YOLO	-	-	-	-	Detecta objetos (Saída 2)

**Tabela 9: YOLOv3-Tiny modificada para Reconhecimento de Letras**

<b>Camada</b>	<b>Filtros</b>	<b>Tamanho</b>	<b>Stride</b>	<b>Ativação</b>	<b>Entrada / Saída</b>
Convolucional	16	$3 \times 3$	1	Leaky	$32 \times 96 \times 3 \rightarrow 32 \times 96 \times 16$
MaxPool	-	$2 \times 2$	2	-	$32 \times 96 \times 16 \rightarrow 16 \times 48 \times 16$
Convolucional	32	$3 \times 3$	1	Leaky	$16 \times 48 \times 16 \rightarrow 16 \times 48 \times 32$
MaxPool	-	$2 \times 2$	2	-	$16 \times 48 \times 32 \rightarrow 8 \times 24 \times 32$
Convolucional	64	$3 \times 3$	1	Leaky	$8 \times 24 \times 32 \rightarrow 8 \times 24 \times 64$
MaxPool	-	$2 \times 2$	2	-	$8 \times 24 \times 64 \rightarrow 4 \times 12 \times 64$
Convolucional	128	$3 \times 3$	1	Leaky	$4 \times 12 \times 64 \rightarrow 4 \times 12 \times 128$
MaxPool	-	$2 \times 2$	2	-	$4 \times 12 \times 128 \rightarrow 2 \times 6 \times 128$
Convolucional	256	$3 \times 3$	1	Leaky	$2 \times 6 \times 128 \rightarrow 2 \times 6 \times 256$
MaxPool	-	$2 \times 2$	2	-	$2 \times 6 \times 256 \rightarrow 1 \times 3 \times 256$
Convolucional	512	$3 \times 3$	1	Leaky	$1 \times 3 \times 256 \rightarrow 1 \times 3 \times 512$
MaxPool	-	$2 \times 2$	1	-	$1 \times 3 \times 512 \rightarrow 1 \times 3 \times 512$
Convolucional	1024	$3 \times 3$	1	Leaky	$1 \times 3 \times 512 \rightarrow 1 \times 3 \times 1024$
Convolucional	256	$1 \times 1$	1	Leaky	$1 \times 3 \times 1024 \rightarrow 1 \times 3 \times 256$
Convolucional	512	$3 \times 3$	1	Leaky	$1 \times 3 \times 256 \rightarrow 1 \times 3 \times 512$
Convolucional	62	$1 \times 1$	1	Linear	$1 \times 3 \times 512 \rightarrow 1 \times 3 \times 62$
YOLO	-	-	-	-	Detecta objetos (Saída 1)
Route	-	-	-	-	Concatena camadas -4
Convolucional	128	$1 \times 1$	1	Leaky	$1 \times 3 \times 512 \rightarrow 1 \times 3 \times 128$
Upsample	-	-	2	-	$1 \times 3 \times 128 \rightarrow 2 \times 6 \times 128$
Route	-	-	-	-	Concatena camadas -1, 8
Convolucional	256	$3 \times 3$	1	Leaky	$2 \times 6 \times 256 \rightarrow 2 \times 6 \times 256$
Convolucional	93	$1 \times 1$	1	Linear	$2 \times 6 \times 256 \rightarrow 2 \times 6 \times 93$
YOLO	-	-	-	-	Detecta objetos (Saída 2)

**Tabela 10: Arquitetura da Rede YOLO baseada no Darknet**

<b>Tipo</b>	<b>Filtros</b>	<b>Tamanho</b>	<b>Stride</b>	<b>Ativação</b>	<b>Entrada / Saída</b>
Entrada	-	-	-	-	$32 \times 96 \times 3$
Convolucional	16	$3 \times 3$	1	Leaky	$32 \times 96 \times 3 \rightarrow 32 \times 96 \times 16$
MaxPool	-	$2 \times 2$	2	-	$32 \times 96 \times 16 \rightarrow 16 \times 48 \times 16$
Convolucional	32	$3 \times 3$	1	Leaky	$16 \times 48 \times 16 \rightarrow 16 \times 48 \times 32$
MaxPool	-	$2 \times 2$	2	-	$16 \times 48 \times 32 \rightarrow 8 \times 24 \times 32$
Convolucional	64	$3 \times 3$	1	Leaky	$8 \times 24 \times 32 \rightarrow 8 \times 24 \times 64$
MaxPool	-	$2 \times 2$	2	-	$8 \times 24 \times 64 \rightarrow 4 \times 12 \times 64$
Convolucional	128	$3 \times 3$	1	Leaky	$4 \times 12 \times 64 \rightarrow 4 \times 12 \times 128$
MaxPool	-	$2 \times 2$	2	-	$4 \times 12 \times 128 \rightarrow 2 \times 6 \times 128$
Convolucional	256	$3 \times 3$	1	Leaky	$2 \times 6 \times 128 \rightarrow 2 \times 6 \times 256$
MaxPool	-	$2 \times 2$	2	-	$2 \times 6 \times 256 \rightarrow 1 \times 3 \times 256$
Convolucional	512	$3 \times 3$	1	Leaky	$1 \times 3 \times 256 \rightarrow 1 \times 3 \times 512$
MaxPool	-	$2 \times 2$	1	-	$1 \times 3 \times 512 \rightarrow 1 \times 3 \times 512$
Convolucional	1024	$3 \times 3$	1	Leaky	$1 \times 3 \times 512 \rightarrow 1 \times 3 \times 1024$
Convolucional	256	$1 \times 1$	1	Leaky	$1 \times 3 \times 1024 \rightarrow 1 \times 3 \times 256$
Convolucional	512	$3 \times 3$	1	Leaky	$1 \times 3 \times 256 \rightarrow 1 \times 3 \times 512$
Convolucional	30	$1 \times 1$	1	Linear	$1 \times 3 \times 512 \rightarrow 1 \times 3 \times 30$
YOLO	-	-	-	-	$1 \times 3 \times 30 \rightarrow$ Saída
Route	-	-	-	-	Concatena saída de -4
Convolucional	128	$1 \times 1$	1	Leaky	$1 \times 3 \times 512 \rightarrow 1 \times 3 \times 128$
Upsample	-	-	2	-	$1 \times 3 \times 128 \rightarrow 2 \times 6 \times 128$
Route	-	-	-	-	Concatena saída de -1, 8
Convolucional	256	$3 \times 3$	1	Leaky	$2 \times 6 \times 256 \rightarrow 2 \times 6 \times 256$
Convolucional	45	$1 \times 1$	1	Linear	$2 \times 6 \times 256 \rightarrow 2 \times 6 \times 45$
YOLO	-	-	-	-	$2 \times 6 \times 45 \rightarrow$ Saída