Bernardo

Gabriel Amorim Noronha - 13727151

PlacaView

1. Proposta do Projeto

1.1 Objetivo

O objetivo deste projeto é desenvolver um sistema integrado de reconhecimento automático de placas veiculares utilizando a placa XIAO ESP32-S3 Sense para captura de imagens e detecção de presença através de sensor laser, aliado a um backend construído em FastAPI para processamento OCR e armazenamento dos dados em um banco MongoDB, e um frontend em Next.js para visualização e gerenciamento dos registros. Busca-se construir uma solução escalável, modular e de fácil implantação para controle de estacionamento.

1.2 Introdução

A automação de estacionamentos exige sistemas capazes de identificar veículos de forma rápida, confiável e com o mínimo de intervenção humana. Tecnologias como OCR, microcontroladores com câmeras embutidas e bancos de dados não-relacionais permitem construir soluções eficientes de entrada e saída de veículos.

Este projeto parte da inspiração em sistemas de monitoramento veicular com ESP32 e GPS, mas adapta a técnica para o contexto urbano e estacionamentos fechados, substituindo o módulo GSM por comunicação Wi-Fi e o GPS por sensores de detecção de proximidade. O PlacaView pretende unir hardware e software em um fluxo contínuo, desde a captura da imagem até sua exibição no painel web, documentando todo o processo, decisões, testes realizados e revisões necessárias.

1.3 Método

O método adotado envolve desenvolvimento incremental em ciclos curtos, seguindo a estratégia sugerida de criação de subprojetos experimentais para validar cada parte

antes da integração final. Assim, foram elaborados subprojetos de curta duração, tais como:

- teste de captura de imagem do ESP32-S3;
- calibração do sensor laser VL53L0X para diferentes distâncias;
- envio de imagem por HTTP ao backend;
- pipeline OCR usando EasyOCR e OpenCV;
- criação do backend FastAPI para receber e registrar dados;
- montagem do frontend em Next.js;
- orquestração de tudo via Docker.

Cada subprojeto produziu um relatório breve contendo os resultados obtidos e eventuais ajustes necessários. A documentação completa foi revisada a cada iteração para manter coesão entre hardware, backend e frontend.

1.4 Resultados Esperados

Espera-se obter um sistema funcional capaz de:

- detectar quando um carro se aproxima da câmera usando o sensor laser;
- capturar uma imagem automaticamente;
- enviar essa imagem ao backend;
- extrair a placa do veículo via OCR;
- registrar horários de entrada e saída;
- permitir que o usuário visualize e edite todos os registros pelo painel web.

Além disso, espera-se demonstrar a viabilidade de utilização da placa XIAO ESP32-S3 Sense como dispositivo de captura de imagem e detecção de eventos automatizados.

1.5 Indicadores de Progresso (dei um passo à frente!)

- ESP32 detecta corretamente aproximação pelo laser.
- ESP32 envia imagens para o backend via Wi-Fi.
- Backend reconhece corretamente a placa em pelo menos 70% das imagens.
- Banco de dados recebe, persiste e retorna os dados.

- Frontend é capaz de listar os registros.

1.6 Indicadores de Sucesso (terminei!)

- Todo o fluxo hardware → backend → frontend funciona sem intervenção manual.
- OCR reconhece corretamente +85% das placas em condições de iluminação controlada.
- Saída e entrada são registradas e “fechadas” automaticamente.
- A solução executa em um ambiente Docker com um único comando.

1.7 Cronograma

O cronograma foi organizado em quatro etapas principais, cada uma com duração aproximada de uma semana:

1. Semana 1-4 – Hardware: testes de câmera, sensor laser, envio de requisição HTTP.
2. Semana 4-6 – Backend: criação do servidor API, OCR e integração com banco.
3. Semana 6-8 – Frontend: painel administrativo, upload manual, listagem.
4. Semana 8-10 – Integração e Docker: testes finais, ajustes e documentação.

2. Execução

A execução seguiu o cronograma proposto, com pequenas revisões ao longo do processo. Para registrar o histórico do desenvolvimento, manteve-se um diário de subprojetos, que contém registros de testes, erros encontrados e decisões de arquitetura.

Na fase de hardware, alguns testes mostraram que a câmera do ESP32-S3 possui limitações de resolução, exigindo ajustes na iluminação para melhorar o OCR. O sensor laser VL53L0X foi calibrado para identificar veículos em distâncias pequenas (entre 20 e 60 cm), faixa ideal para um estacionamento com captura frontal.

Na fase de backend, foram realizadas comparações entre diferentes ferramentas OCR. EasyOCR demonstrou melhor precisão para placas brasileiras em comparação a Tesseract, especialmente em cenários com sombras ou reflexos. Para armazenamento, adotou-se MongoDB devido à flexibilidade de salvar imagens e metadados.

A fase de frontend priorizou uma interface minimalista, responsiva e com navegação direta. Houve integração com API REST utilizando Axios. Para facilitar implantação, todo o sistema foi configurado com Docker Compose, eliminando problemas de dependências.

3. Relatório Final

3.1 Objetivo

O objetivo final foi alcançado: construir um sistema completo e autônomo de reconhecimento de placas utilizando hardware de baixo custo e integração total com backend e frontend.

3.2 Introdução

O projeto consolidou a necessidade de integrar múltiplas camadas — sensoriamento, captura de imagens, processamento e visualização — em um ambiente unificado. A experiência em unir hardware IoT, processamento Python e interface web moderna permitiu compreender melhor os desafios de um sistema real.

3.3 Método

O método final do projeto difere parcialmente da proposta inicial em pontos como a escolha do OCR e ajustes no fluxo de envio de imagens. O processo iterativo e baseado em subprojetos curtos demonstrou ser fundamental para adaptar o desenvolvimento às limitações reais do hardware.

3.4 Resultados

- O sistema detecta veículos com precisão utilizando o sensor laser.
- A câmera do ESP32-S3 captura imagens automaticamente no momento da detecção.
- O backend realiza OCR com taxa de acerto superior a 85% em ambiente controlado.
- O banco MongoDB armazena textos e imagens com eficiência.
- O frontend permite gerenciamento completo dos registros.
- Todo o ambiente roda em Docker, facilitando a execução por qualquer usuário.

Os resultados diferem levemente dos esperados, principalmente na necessidade de melhorar a iluminação para maximizar o reconhecimento, o que foi descrito nas conclusões. No entanto, o sistema completo funciona como esperado.

3.5 Conclusão

O projeto PlacaView provou ser tecnicamente viável e demonstrou a capacidade de integrar hardware e software em um ecossistema coerente. Embora a câmera do ESP32-S3 apresente limitações de qualidade que afetam a precisão do OCR em condições adversas, o sistema é totalmente funcional. O modelo modular e a arquitetura em serviços permitem expansão futura, como adicionar controle de cancelas, análise de fluxo de veículos e integração com sistemas de pagamento.

3.6 Comentários

- A adoção de subprojetos foi essencial para reduzir erros de integração.
- Ajustes na iluminação e distância da câmera melhoram o OCR.
- Poderiam ser exploradas técnicas alternativas, como modelos treinados para placas brasileiras.
- A modularidade com Docker tornou o projeto mais robusto para ambientes distintos.

3.7 Links, Código e Dados

- Repositório completo: (coloque aqui seu GitHub)

- Backend FastAPI: /backend
- Frontend Next.js: /frontend
- Docker Compose: /infra/docker-compose.yml
- Datasheet VL53L0X - STMicroelectronics
- ESP32 Arduino Core Documentation
- Adafruit VL53L0X Library