Projeto de Reconhecimento de Placas de Veículos – Campus IFMS Campo Grande

Objetivo

Desenvolver um sistema automatizado de reconhecimento de placas veiculares para controlar o acesso de veículos ao campus do Instituto Federal de Mato Grosso do Sul – Campo Grande. O sistema visa aumentar a segurança, facilitar o monitoramento e permitir a abertura automática de portões para veículos previamente autorizados.

Etapas do Funcionamento

Captura da Imagem
Uma câmera posicionada na entrada do campus registra a imagem do veículo ao se aproximar.

Detecção da Placa
Utiliza-se o algoritmo YOLOv8n para localizar a placa na imagem capturada.

Correção de Perspectiva
Técnicas como MSER são aplicadas para corrigir distorções e inclinações da placa.

Reconhecimento de Caracteres
O OCR é realizado com PyTesseract, que converte a imagem da placa em texto.

Validação
O texto extraído é validado com expressões regulares (Regex) e comparado com um banco de dados de placas autorizadas.

Decisão e Ação
Se a placa estiver cadastrada, o sistema autoriza automaticamente a abertura do portão.

Tecnologias Utilizadas

Python: Linguagem principal do projeto, escolhida pela sua robustez e compatibilidade com bibliotecas de visão computacional.

OpenCV: Para pré-processamento de imagens (ajuste de brilho, contraste, remoção de ruído).

YOLOv8n: Detecção em tempo real da placa veicular.

MSER: Correção de distorções e inclinações.

PyTesseract OCR: Reconhecimento óptico dos caracteres.

Regex: Validação do formato da placa.

PostgreSQL + PGAdmin: Banco de dados para armazenar placas autorizadas e registrar acessos.

Raspberry Pi 4: Hardware embarcado para processamento local e controle do sistema.

Câmera IP com sensor IR: Equipamento de captura com suporte à visão noturna.

HTML + CSS + JS: Interface web para cadastro de placas e visualização de registros.

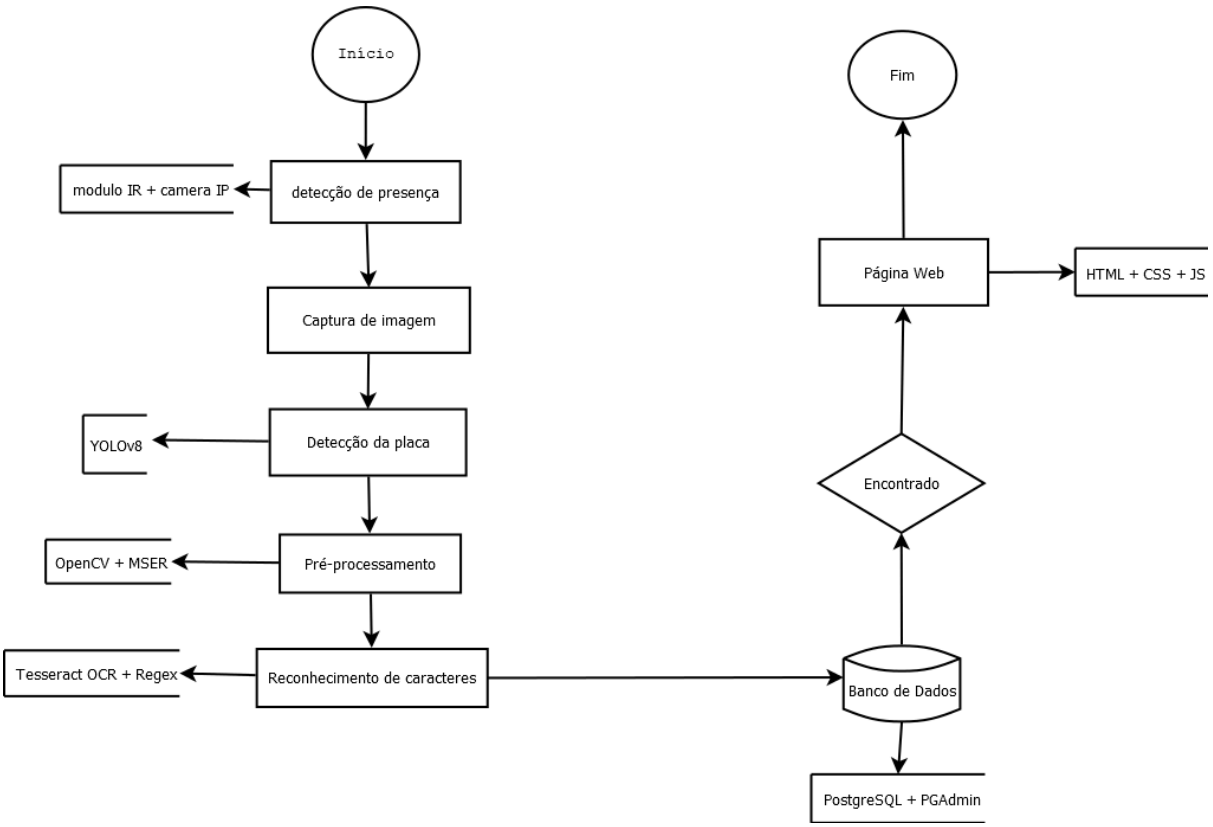
Aplicação no Campus

Cadastro de placas de veículos de servidores, alunos e visitantes autorizados.

Registro de data e hora de entrada dos veículos.

Geração de relatórios de acesso.

Integração com aplicativo móvel para gerenciamento remoto.



- **Câmera IP**
  Captura e envia imagens/vídeos para o Raspberry Pi.

  Custo: (R$125,00-R$280,00)

  Desempenho: melhor qualidade e ângulo ajustável, grava continuamente em rede.

  Compatibilidade: conecta via Ethernet/Wi-Fi, fácil integrar com OpenCV (RTSP/HTTP).

- **Módulo IR**
  Detecta passagem de veículos ou ativa a captura (sensor de presença).

  Custo: (R$10–R$40).
  Desempenho: complementa a câmera em baixa luz; reduz uso contínuo do yolo.

  Compatibilidade: conecta direto no GPIO do Raspberry.

- **YOLOv8n (Nano)**
  Detecta placas em tempo real (localização).

  Custo: open-source gratuito.

  Desempenho: versão “n” é mais leve, ideal para hardware limitado.

  Compatibilidade: roda em Python, integra facilmente com OpenCV.

- **OpenCV + MSER**
  Pré-processa a imagem, recorta a área da placa, melhora contraste e ruído.

  Custo: biblioteca open-source gratuita.

  Desempenho: otimizada, muito rápida mesmo em hardware pequeno.

  Compatibilidade: roda em Python, se integra com YOLO e Tesseract.

- **Tesseract OCR**
  Transforma imagem da placa em caracteres (texto).

  Custo: open-source gratuito.

  Desempenho: reconhece texto multi-idioma; ótimo para placas.

  Compatibilidade: fácil integrar via Python.

- **Regex**

  Valida se o texto lido segue padrão de placa brasileira (Mercosul).

  Custo: nativo em Python.

  Desempenho: extremamente rápido.

  Compatibilidade: funciona diretamente com saída do Tesseract.

- **Raspberry Pi**
  Unidade central: recebe câmera + sensor IR, processa YOLO, pré-processamento, OCR e banco de dados.

  Custo: (R$400-R$900)

  Desempenho: potente o suficiente para tarefas leves/moderadas se bem otimizado.

  Compatibilidade: integra sensores, câmera IP, banco de dados e Python em um só dispositivo.

- **HTML + CSS + JS**

  Interface web para visualizar dados, cadastros, estatísticas.

  Custo: grátis, você já domina.
  Desempenho: roda em qualquer navegador (PC ou celular).

  Compatibilidade: integra com backend em Python ou PHP.

- **PostgreSQL + PGAdmin**

  Armazena dados de entrada/saída, histórico, veículos cadastrados.

  Custo: open-source gratuito.

  Desempenho: robusto, escalável, aceita grandes volumes de dados.

  Compatibilidade: integra facilmente com Python e web apps.

- **Python**

  Configura YOLOv8, OpenCV, MSER, Tesseract OCR e backend da aplicação.

  Custo: open-source gratuito.

  Desempenho: enorme ecossistema de bibliotecas para visão computacional, bancos de dados e web.

  Compatibilidade: linguagem nativa do YOLO, OpenCV e Tesseract.

Principais soluções open-source ou com código disponível encontradas

YOLOv8n (Nano)

- **Vantagens:** open-source, leve, roda em tempo real em hardware limitado; fácil integração com Python e OpenCV.

- **Desvantagens:** menor precisão que versões maiores; pode falhar em situações de baixa iluminação ou placas muito desgastadas.

OpenCV + MSER

- **Vantagens:** biblioteca robusta e gratuita; otimizada para processamento rápido; enorme comunidade de suporte; pré-processamento melhora a precisão do OCR.

- **Desvantagens:** exige ajustes finos nos parâmetros para cada cenário; não resolve sozinho a detecção (depende de modelos adicionais).

Tesseract OCR

- **Vantagens:** open-source, suporte a múltiplos idiomas; boa precisão em imagens limpas; fácil integração com Python.

- **Desvantagens:** sensível a imagens com baixa qualidade/contraste; pode precisar de pré-processamento pesado para funcionar bem.

Regex (Python)

- **Vantagens:** nativo em Python, extremamente rápido, útil para validar padrões como placas Mercosul; zero custo.

- **Desvantagens:** não corrige erros de OCR (apenas valida); precisa ser configurado manualmente conforme o padrão.

PostgreSQL + PGAdmin

- **Vantagens:** banco de dados robusto, escalável, gratuito; excelente para grandes volumes de dados; PGAdmin facilita administração.

- **Desvantagens:** curva de aprendizado maior em comparação a bancos mais simples; requer configuração inicial mais detalhada.

Python

- **Vantagens:** linguagem open-source, versátil, vasta quantidade de bibliotecas para IA, visão computacional, bancos de dados e web; grande comunidade.

- **Desvantagens:** performance inferior a linguagens compiladas em tarefas muito pesadas; pode exigir otimizações (threads, uso de GPU).

HTML + CSS + JS

- **Vantagens:** totalmente abertas; compatíveis com qualquer navegador; permitem criar interfaces acessíveis em PC ou celular.

- **Desvantagens:** demandam integração com backend para funcionalidades avançadas; nível de segurança depende da implementação.

Componentes do Projeto (Raspberry Pi + Acessórios)

Categoria	Componente	Função / Observação
Processamento	Raspberry Pi 4 8GB	Centro do sistema; processa YOLO, OpenCV e Tesseract.
Alimentação	Fonte oficial Raspberry Pi (5V 3A) ou nobreak pequeno	Evita quedas de energia.
Armazenamento	Cartão microSD rápido (≥32GB) ou SSD via USB	Sistema + banco de dados.
Câmera	Câmera IP Full HD com IR (stream RTSP)	Captura vídeo das placas.
Sensor de presença (PIR)	Aciona captura quando veículo se aproxima	Reduz processamento ocioso.
Rede	Wi-Fi integrado do Pi ou adaptador Ethernet	Conexão com rede e banco.
Outros (opcional)	Case + dissipador + cooler para o Pi	Ajuda na refrigeração e protege.