Universidade de São Paulo Escola Politécnica Departamento de Engenharia Elétrica
Monografia apresentada à Escola Politécnica da Universidade de São Paulo como parte dos requisitos para a conclusão do Curso de Engenharia Elétrica.

IGOR TEIXEIRA LACERDA
LORENZO BIGHETTI ZAUITH
RODRIGO COSTA DE ARAÚJO

SISTEMA DE DETECÇÃO DE ACIDENTES E ALERTA AUTOMÁTICO PARA MOTOCICLISTAS

São Paulo
2025

IGOR TEIXEIRA LACERDA
LORENZO BIGHETTI ZAUITH
RODRIGO COSTA DE ARAÚJO

SISTEMA DE DETECÇÃO DE ACIDENTES E ALERTA AUTOMÁTICO PARA MOTOCICLISTAS

Orientador:
Márcio Lobo Netto

São Paulo
2025

** Dedicamos este trabalho a todos os motociclistas que inspiraram esta pesquisa e às nossas famílias pelo apoio incondicional durante todo o desenvolvimento deste projeto.*

ACKNOWLEDGMENTS

Agradecemos aos nossos colegas de turma e de laboratório, pelas discussões proveitosas e pela colaboração nas etapas de testes e validações. Às nossas famílias, pelo apoio emocional e motivação constantes, fundamentais para a conclusão desta monografia.

Agradecemos também à Escola Politécnica da USP e à Universidade de São Paulo pelas instalações e recursos disponibilizados, bem como aos professores do Departamento de Engenharia Elétrica que forneceram a base teórica utilizada neste trabalho.

Registramos nossa gratidão ao Prof. Dr. Márcio Lobo Netto pela orientação acadêmica fornecida durante o projeto.

O sucesso não é final, o fracasso não é fatal: é a coragem de continuar que conta.
(Winston Churchill)

RESUMO

Os acidentes motociclísticos representam uma das principais causas de mortalidade no trânsito brasileiro. Segundo a Associação Brasileira de Medicina do Tráfego (ABRAMET), a cada 39 minutos um motociclista perde a vida no trânsito brasileiro[1], e a taxa de mortalidade por quilômetro rodado é cerca de 30 vezes maior do que a de ocupantes de automóveis. A vulnerabilidade desses condutores agrava-se quando o acidente ocorre em vias de baixa circulação, áreas rurais ou estradas com pouca iluminação e monitoramento. Nesses cenários, o tempo até que alguém identifique o acidente e acione o resgate pode ultrapassar a chamada "hora de ouro", intervalo de até 60 minutos após um trauma grave, considerado crítico para a sobrevivência [35]. Estudos do *National Highway Traffic Safety Administration* (NHTSA) [14] indicam que vítimas de acidentes graves têm até 60% mais chances de sobreviver quando socorridas nos primeiros 30 minutos[3]. Diante desse quadro, este trabalho propõe o desenvolvimento de um sistema embarcado de baixo custo, fixado na estrutura da motocicleta, capaz de detectar automaticamente acidentes por meio de sensores inerciais, enviar alertas de emergência com localização geográfica em tempo real e, adicionalmente, registrar imagens do momento da colisão como apoio secundário para diagnóstico ou documentação do ocorrido. O dispositivo é composto por um microcontrolador Raspberry Pi Zero 2W, um microcontrolador ESP32-C3 Super Mini, um sensor GY-BMI160 (acelerômetro e giroscópio) e uma câmera grande angular de 160°, integrados em uma carcaça impressa em 3D projetada para resistir a impactos. A lógica de funcionamento baseia-se na análise de padrões de aceleração e variação angular para detectar quedas abruptas ou mudanças anormais de orientação, desencadeando automaticamente o protocolo de emergência sem necessidade de intervenção do usuário. A transmissão dos dados é feita via conexão Bluetooth, enviando as coordenadas do acidente e uma mensagem pré-configurada para contatos de emergência ou serviços especializados. A câmera, posicionada com ângulo amplo, atua como ferramenta auxiliar e pode registrar imagens de forma instantânea após o acidente, gerando evidência visual útil. O sistema é alimentado por bateria recarregável com autonomia de aproximadamente 10 horas de monitoramento contínuo. Com custo estimado em R$ 300,00 por unidade, o projeto visa tornar acessível uma tecnologia crítica para mitigar o impacto de acidentes motociclísticos, com potencial de aplicação em escala nacional por órgãos públicos, seguradoras e fabricantes. Além de seu caráter social, a proposta alinha-se às tendências de segurança ativa e resposta emergencial em tempo real, reforçando o papel da engenharia embarcada na preservação da vida.

Palavras-Chave: segurança veicular; acidentes de moto; sistemas embarcados; detecção automática; resposta emergencial.

ABSTRACT

Motorcycle accidents are a leading cause of traffic fatalities in Brazil. According to the Brazilian Association of Traffic Medicine (ABRAMET), a motorcyclist dies every 39 minutes in Brazil, and the fatality rate per kilometer traveled is dozens of times higher than that of car occupants[1]. The vulnerability of these riders is exacerbated when accidents occur on low-traffic roads, rural areas, or poorly lit and monitored highways. In such situations, the time until an accident is identified and rescue is activated can exceed the "golden hour," a critical 60-minute window after severe trauma, vital for survival. Studies by the National Highway Traffic Safety Administration (NHTSA) indicate that severe accident victims have up to a 60% higher chance of survival when rescued within the first 30 minutes[3]. Given this scenario, this work proposes the development of a low-cost embedded system, mounted on the motorcycle's frame, capable of automatically detecting accidents using inertial sensors, sending real-time emergency alerts with geographical location, and additionally recording images of the collision moment as secondary support for diagnosis or documentation. The device comprises a Raspberry Pi Zero 2W microcontroller, an ESP32-C3 Super Mini microcontroller, a GY-BMI160 sensor (accelerometer and gyroscope), and a 160-degree wide-angle camera, all integrated into a 3D-printed enclosure designed to withstand impact. The system's operational logic is based on analyzing acceleration patterns and angular variations to detect abrupt falls or abnormal orientation changes, automatically triggering the emergency protocol without user intervention. Data transmission occurs via Bluetooth connection, sending accident coordinates and a pre-configured message to emergency contacts or specialized services. The wide-angle camera acts as an auxiliary tool, instantly capturing images during and after the accident, generating useful visual evidence. The system is powered by a rechargeable battery, providing approximately 10 hours of continuous monitoring. With an estimated cost of R$ 300.00 per unit, this project aims to make critical technology accessible to mitigate the impact of motorcycle accidents, with potential for national application by public agencies, insurance companies, and manufacturers. Beyond its social impact, the proposal aligns with trends in active safety and real-time emergency response, reinforcing the role of embedded engineering in preserving lives.

Keywords: vehicle safety; motorcycle accidents; embedded systems; automatic crash detection; emergency response.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

Associação Brasileira de Medicina do Tráfego

Bluetooth Low Energy

Central Processing Unit (Unidade Central de Processamento)

Família de microcontroladores ESP32 (RISC-V 160 MHz com Bluetooth 5.0)

Global Positioning System (Sistema de Posicionamento Global)

Inertial Measurement Unit (Unidade de Medição Inercial)

National Highway Traffic Safety Administration

Optical Character Recognition (Reconhecimento Óptico de Caracteres)

Requisito Funcional

Requisito Não Funcional

You Only Look Once (Algoritmo de detecção de objetos em tempo real)

Aceleração gravitacional padrão ($\approx 9{,}81\ \mathrm{m/s^2}$).

Velocidade angular (em radianos ou graus por segundo).

Variação de uma grandeza (diferença entre dois instantes ou estados).

1 INTRODUÇÃO

1.1 Antecedentes

A segurança no trânsito é um desafio crítico nas grandes cidades e rodovias. No contexto de transporte individual, os motociclistas se destacam pela alta exposição a riscos de acidentes graves. Nas últimas décadas, avanços em sistemas eletrônicos veiculares têm reduzido a mortalidade entre ocupantes de automóveis (por meio de airbags, freios ABS, etc.), porém motocicletas ainda carecem de dispositivos equivalentes de proteção ativa e de sistemas automáticos de alerta pós-acidente. Tecnologias de chamada de emergência automática, como o sistema *eCall* implementado em carros na União Europeia, comprovam que a rápida notificação de acidentes pode salvar vidas, diminuindo o tempo de resposta de equipes de resgate[4]. Entretanto, tais soluções ainda não são amplamente disponíveis para motocicletas.

Historicamente, a resposta a acidentes motociclísticos depende quase exclusivamente de terceiros presenciarem o evento e acionar socorro. Em áreas remotas ou horários de baixo movimento, um motociclista acidentado pode ficar longos períodos sem assistência. Esse atraso no atendimento médico reduz drasticamente as chances de sobrevivência, conforme destacam estatísticas da ABRAMET [28] e estudos internacionais [2]. Pesquisas de Park *et al.* [20] demonstram a relação direta entre tempo pré-hospitalar e desfechos em trauma, enquanto Brown *et al.* [21] evidenciam que, em casos de hemorragia, cada minuto adicional de atraso está associado a aumento significativo na mortalidade. Esse problema motivou pesquisas em sistemas de detecção automática de acidentes utilizando sensores inerciais, dispositivos móveis e técnicas de telecomunicação para envio de alertas.

Com o advento de sensores MEMS (*Micro-Electro-Mechanical Systems*) acessíveis e microcontroladores de baixo custo e baixo consumo, tornou-se viável a construção de dispositivos embarcados capazes de monitorar continuamente o movimento de um veículo e identificar padrões indicativos de colisão ou queda. Diversos trabalhos exploraram o uso de acelerômetros e giroscópios para esse fim[5, 6]. Em paralelo, a difusão de smartphones

com sensores embutidos levou ao surgimento de aplicativos de detecção de acidentes, embora com limitações (por exemplo, dependência de o smartphone estar fixo no veículo e ligado no momento do acidente). Nesse contexto, um dispositivo dedicado ao veículo, integrado ao motociclo, pode oferecer maior confiabilidade e autonomia na detecção de acidentes.

1.2 Motivação

A motivação principal deste projeto é reduzir o tempo entre a ocorrência de um acidente de moto e o atendimento à vítima. Conforme mencionado, a "hora de ouro" é crucial para aumentar a probabilidade de sobrevivência em traumas graves. No Brasil, onde milhares de motociclistas transitam diariamente em condições adversas, um sistema capaz de automaticamente perceber um acidente e alertar equipes de emergência ou contatos de confiança pode significar a diferença entre a vida e a morte em muitos casos.

Além do aspecto vital, há também motivação social e econômica. Acidentes de moto representam custos elevados ao sistema de saúde e à seguridade (licenças médicas, indenizações, etc.), e mitigá-los ou minimizar suas consequências traz benefícios coletivos. Segundo dados da Agência Brasil [29], a frota brasileira de motocicletas cresceu 42% entre 2015 e 2024, alcançando 35 milhões de unidades, enquanto as mortes de motociclistas correspondem a 38,6% dos óbitos no trânsito. Tecnologias de segurança ativa para motocicletas estão atrasadas em comparação às dos automóveis, de forma que este projeto busca contribuir para preencher essa lacuna, alinhando-se às iniciativas de *smart cities* e mobilidade mais segura.

Do ponto de vista tecnológico, o projeto também é motivado pela oportunidade de integração de diferentes áreas da Engenharia Elétrica e da Computação: sistemas embarcados, sensores inerciais, processamento de sinais, comunicação sem fio e desenvolvimento móvel. Implementar uma solução funcional exigiu aplicar e aprimorar conhecimentos teóricos em cada um desses domínios, o que justifica academicamente a empreitada como um Trabalho de Conclusão de Curso desafiador e enriquecedor.

1.3 Relevância

A relevância deste trabalho pode ser avaliada em múltiplas dimensões. Sob o prisma da Engenharia, trata-se de um esforço de inovação incremental, combinando tecnologias

existentes de forma original para solucionar um problema real. Embora sensores e comunicações sem fio sejam amplamente usados em outros contextos, sua aplicação focada em segurança de motociclistas ainda é limitada. O sistema proposto diferencia-se por buscar um equilíbrio entre baixo custo e alto desempenho, viabilizando sua adoção em larga escala, inclusive por órgãos públicos (por exemplo, em frotas de motofrete ou programas de seguridade para motociclistas) e empresas seguradoras para redução de sinistralidade.

Em termos sociais, o projeto atende a uma demanda urgente de saúde pública. Segundo a ABRAMET, os acidentes com motos já configuram uma epidemia silenciosa nas emergências hospitalares. Cada minuto de redução no tempo de resposta do resgate pode aumentar significativamente a chance de sobrevivência e reduzir sequelas permanentes nas vítimas. Assim, o impacto potencial de um sistema que agilize a chegada do socorro é enorme, salvando vidas e diminuindo o sofrimento de famílias.

No âmbito acadêmico e de pesquisa, esta monografia documenta os desafios e soluções encontrados no desenvolvimento de um sistema *cyber-physical* (ciberfísico) complexo em pequena escala. A documentação minuciosa de falhas, decisões de projeto e desempenho obtido contribui para a literatura de projetos similares, servindo de referência para futuros desenvolvimentos de segurança veicular e IoT (*Internet of Things*) aplicada ao trânsito.

1.4 Declaração do Problema

Dado o exposto, o problema específico que este trabalho se propõe a resolver é: **Como detectar automaticamente um acidente envolvendo uma motocicleta e notificar, de forma confiável e imediata, os contatos de emergência ou serviços de resgate, fornecendo a localização exata e informações contextuais do evento?**

Essa declaração de problema envolve diversos subproblemas técnicos interconectados. Primeiramente, é necessário determinar como distinguir, via sensores embarcados, um acidente verdadeiro de eventos cotidianos como buracos, frenagens bruscas normais ou inclinações em curvas. Além disso, deve-se garantir que o alerta chegue rapidamente a um destinatário, mesmo em condições adversas como sinal celular fraco ou quando o motociclista estiver inconsciente. Por fim, cabe investigar quais informações adicionais podem ser fornecidas para melhorar a eficiência do socorro ou registro do incidente, como imagens do local ou identificação de placa de veículo envolvido.

Em resumo, o problema central reside na criação de um *sistema de detecção de acidentes motociclísticos* que seja autônomo, confiável e reproduzível, cobrindo desde a mon-

itoração do veículo até a comunicação pós-acidente.

1.5 Declaração da Necessidade

A necessidade de tal sistema torna-se evidente pelos números alarmantes e casos recorrentes de motociclistas que ficam sem atendimento em tempo hábil. No dia-a-dia, muitos motociclistas trafegam sozinhos por trajetos pouco movimentados; um deslize, colisão ou mesmo mal súbito pode deixá-los caídos sem que ninguém testemunhe imediatamente. Assim, existe uma necessidade clara de um "anjo da guarda eletrônico" para esses condutores: um dispositivo sempre alerta que, em caso de acidente, automaticamente chame por ajuda.

Do ponto de vista dos usuários (motociclistas e seus familiares), a tranquilidade proporcionada por saber que existe um sistema de alerta automático é um fator de necessidade intangível, relacionado à segurança psicológica. Para os serviços de emergência, um aviso prévio com coordenadas precisas agiliza a logística de atendimento. Para órgãos de trânsito e seguradoras, há necessidade de dados mais confiáveis sobre acidentes para análises estatísticas e apuração de causas; um sistema embarcado pode registrar dados do momento do acidente, atendendo também a essa necessidade secundária.

Em suma, a necessidade abrange múltiplos públicos: para motociclistas e familiares, representa a garantia de socorro rápido em caso de acidente, mesmo quando o condutor estiver incapacitado de pedir ajuda; para autoridades e serviços médicos, significa o recebimento de alerta geolocalizado que permite otimizar o tempo-resposta; e para seguradoras e pesquisadores, constitui o registro de informações do acidente, incluindo dinâmica e imagens, que auxilia em processos de indenização e estudos de segurança viária.

1.6 Requisitos de Marketing

Antes de partir para os requisitos técnicos, é importante traduzir as necessidades acima em requisitos de alto nível, do ponto de vista do usuário final e do mercado. Esses requisitos de marketing orientam as características gerais que o produto deve possuir para ser viável e atrativo. Os requisitos de marketing levantados para o sistema proposto são apresentados a seguir em formato de texto corrido, evitando listas extensivas.

As necessidades de marketing identificadas para o sistema incluem um preço acessível, com custo unitário em torno de R$300–400 para viabilizar ampla adoção. A instalação

deve ser simples, permitindo acoplamento direto no guidão ou chassi com mínima intervenção técnica. O sistema deve apresentar alta confiabilidade com baixo índice de falsos alarmes e mecanismos de confirmação manual ou automatizada. A autonomia de funcionamento deve alcançar 8–10 horas em uso típico. A integração com smartphone é essencial para envio de alertas utilizando Bluetooth e internet móvel. O equipamento deve ser robusto, com resistência a impactos, poeira e chuva. Por fim, deve oferecer valor adicional através de telemetria, estatísticas de pilotagem ou rastreamento que agreguem benefícios percebidos além do alerta de acidente.

As necessidades de marketing identificadas para o sistema incluem um preço acessível – com custo unitário em torno de R$300–400 – para viabilizar ampla adoção; instalação simples e acoplamento direto no guidão ou chassi, permitindo que o próprio usuário instale o dispositivo com mínima intervenção técnica; confiabilidade, com baixo índice de falsos alarmes e mecanismos de confirmação manual ou automatizada; autonomia de funcionamento de 8–10 horas em uso típico; integração com smartphone para envio de alertas, utilizando Bluetooth e internet móvel; robustez, com resistência a impactos, poeira e chuva; e valor adicional, oferecendo telemetria, estatísticas de pilotagem ou rastreamento que agreguem benefícios percebidos além do alerta de acidente. Foram definidos os requisitos de engenharia do sistema de detecção de acidentes e alerta automático para motociclistas. Esses requisitos foram divididos em requisitos funcionais (RF) e requisitos não-funcionais (RNF), conforme resumido nas Tabelas 1 e 2. Os requisitos funcionais descrevem as funções e comportamentos que o sistema deve realizar, enquanto os requisitos não-funcionais estabelecem restrições e parâmetros de qualidade. não-funcionais estabelecem restrições e atributos de qualidade do sistema.

1.6.1 Requisitos Funcionais

Na Tabela 1, são listados os principais requisitos funcionais levantados para o projeto, codificados como RF-01, RF-02, etc.:

Os requisitos funcionais acima garantem que o sistema desempenhe as funções essenciais para detectar e reportar acidentes de forma autônoma e confiável. A implementação de cada um deles é detalhada nos capítulos de desenvolvimento. Vale notar que o RF-08 (telemetria) é um *plus* funcional que agrega valor, embora não seja estritamente necessário para o atendimento emergencial imediato.

1.6.2 Requisitos Não-Funcionais

Além das funções, foram estabelecidos requisitos não-funcionais que o sistema deve satisfazer, apresentados na Tabela 2. Esses RNFs dizem respeito ao desempenho, restrições físicas, usabilidade e outros atributos de qualidade:

Os requisitos não-funcionais delineados acima guiaram as decisões de projeto em termos de escolha de componentes (por exemplo, seleção de um microcontrolador de ultra-baixo consumo para atender RNF-01), calibração do algoritmo de detecção (RNF-03) e arquitetura de software (RNF-05 e RNF-07). Durante o desenvolvimento, a satisfação de cada RNF foi avaliada e, quando necessário, ajustes foram realizados para cumpri-los (conforme será discutido nos Resultados).

1.7 Árvore de Objetivos

Tendo definidos os requisitos, é útil apresentá-los de forma hierárquica sob a perspectiva dos objetivos do projeto. A **árvore de objetivos** a seguir ilustra os objetivos gerais e específicos do sistema, relacionando-os aos requisitos correspondentes:

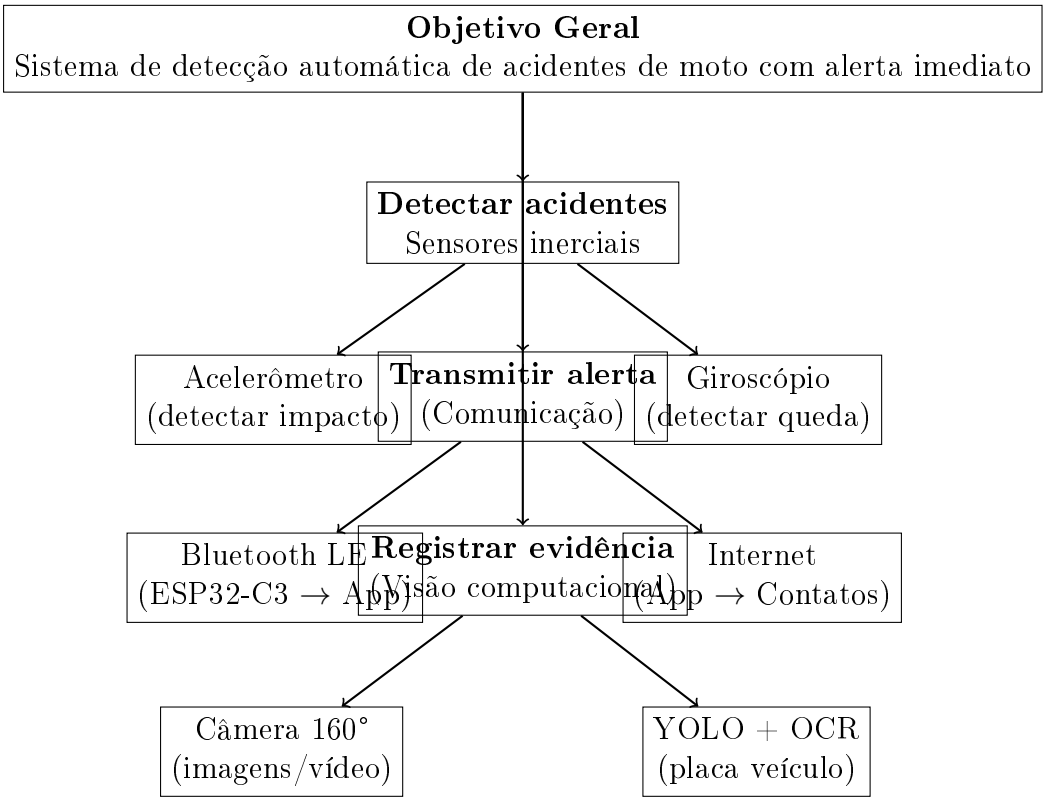

Figure 1: Árvore de objetivos do projeto, relacionando objetivos específicos aos requisitos correspondentes.

Conforme a figura 1, o objetivo geral de prover um *alerta imediato de acidente* desdobra-se em objetivos específicos: (1) **detectar acidentes** de forma confiável, o que remete ao uso combinado de acelerômetro e giroscópio para identificar quedas ou colisões; (2) **transmitir o alerta** de maneira rápida aos destinatários apropriados, envolvendo tanto a comunicação local (Bluetooth) quanto o envio remoto (internet); e (3) **registrar evidências** visuais que enriqueçam as informações do acidente, objetivo atendido pelo módulo de câmera e algoritmos de visão computacional (detecção de placa via YOLO + OCR). Cada um desses objetivos específicos se vincula a requisitos funcionais e não-funcionais descritos nas seções anteriores.

Table 1: Requisitos Funcionais do Sistema

Código	Descrição do Requisito Funcional
RF-01	**Detecção de evento crítico**: O dispositivo embarcado (ESP32-C3 + sensor inercial) deve monitorar continuamente aceleração e rotação da motocicleta, identificando padrões típicos de queda ou colisão brusca.
RF-02	**Acionamento do módulo de captura**: Ao detectar um evento de acidente (critério de detecção atendido), o ESP32-C3 deverá gerar um sinal de ativação via GPIO para "acordar" o microcontrolador Raspberry Pi Zero 2W, que normalmente permanece em modo de baixo consumo.
RF-03	**Captura de imagens/vídeo**: O Raspberry Pi, ao ser despertado, deverá realizar o registro do contexto do acidente. Isso inclui capturar, em modo *burst*, uma sequência curta de imagens (ou um vídeo de poucos segundos) através da câmera grande angular, de forma a obter registro visual do momento do impacto e instantes subsequentes.
RF-04	**Transmissão ao smartphone**: Os dados capturados (imagens ou vídeo, e dados do evento como aceleração pico, horário, etc.) deverão ser enviados do dispositivo embarcado para o smartphone do usuário via conexão Bluetooth Low Energy (BLE).
RF-05	**Processamento no aplicativo móvel**: O aplicativo móvel (smartphone) deve receber os dados do acidente e então executar processamento adicional. Especificamente, deve: (a) Extrair de imagens recebidas a placa de eventual veículo envolvido, usando algoritmo de detecção de objetos (p. ex. YOLO) seguido de OCR para ler os caracteres da placa; (b) Compilar uma mensagem de alerta contendo localização geográfica (via GPS do smartphone), horário e, opcionalmente, uma foto do acidente com a placa destacada.
RF-06	**Envio de alerta**: O aplicativo deverá então enviar automaticamente um alerta de emergência aos destinatários predefinidos (contatos de confiança do motociclista e/ou serviços de resgate). Esse alerta pode ser realizado via mensagem SMS, e-mail ou integração com aplicativos de mensagem (*ex.:* utilizando a API do Telegram), incluindo as informações relevantes (localização, identificação do veículo, etc.).
RF-07	**Interação com o usuário para confirmação**: O sistema deve incluir uma forma do usuário cancelar o alerta nos casos de falso acionamento ou acidentes sem gravidade. Por exemplo, o aplicativo móvel pode exibir um aviso sonoro e visual por alguns segundos antes de disparar a notificação, permitindo que o motociclista consciente cancele o envio. Caso não haja cancelamento em tempo hábil, assume-se emergência real e o alerta é concluído.
RF-08	**Registro de dados contínuo (telemetria)**: Adicionalmente, o sistema deverá registrar dados de telemetria do percurso (acelerações extremas, velocidade estimada, eventos de frenagem brusca) em uma base de dados remota ou local. Esses dados podem ser usados para análise posterior do comportamento do piloto e para fornecimento de evidências em caso de acidentes (trajetória antes do acidente, por exemplo).

Table 2: Requisitos Não-Funcionais do Sistema

Código	Descrição do Requisito Não-Funcional
RNF-01	**Autonomia de energia**: O sistema embarcado deve operar por pelo menos 10 horas contínuas sem recarga de bateria, considerando até 2 detecções de acidente por dia. Esse requisito visa atender o uso típico diário (trajetos de trabalho, viagens curtas) sem depender de recarga frequente.
RNF-02	**Robustez física**: A carcaça do dispositivo deve proteger os componentes contra impactos de colisão e vibrações intensas. Espera-se que o equipamento continue funcional (ou ao menos preserve os dados capturados) mesmo após o acidente. Além disso, deve ter algum grau de proteção ambiental, sendo resistente a poeira e respingos d'água (equivalente a IP54 ou superior).
RNF-03	**Desempenho em detecção**: O algoritmo de detecção deve equilibrar sensibilidade e especificidade, acionando em casos de acidente verdadeiro ($> 95\%$ de detecções) e evitando falsos alarmes em situações não críticas ($< 5\%$ de falsos positivos). Esses valores servem como meta de projeto. Estudos mostram ser comum adotar limiares na faixa de 3–4 g de aceleração para distinguir colisões[7], ajustados experimentalmente para minimizar disparos indevidos.
RNF-04	**Tempo de resposta**: O intervalo de tempo entre a detecção do acidente e o envio do alerta ao contato de emergência deve ser o menor possível. A meta estabelecida foi de no máximo 30 segundos para envio da primeira notificação (mensagem de socorro básica com localização). A disponibilização de dados complementares (imagens, vídeo) pode ocorrer em segundo plano, em até 2 minutos, sem comprometer o aviso inicial. Essa rapidez garante alinhamento com a necessidade de atendimento imediato.
RNF-05	**Integração e usabilidade do aplicativo**: O aplicativo móvel deve ser intuitivo e de operação simples. Deve iniciar automaticamente com o sistema do smartphone e permanecer rodando em segundo plano enquanto o usuário estiver pilotando, conectando-se ao dispositivo via BLE sem requerer intervenção manual a cada uso. Em caso de acidente, a interface deve avisar claramente o usuário e solicitar confirmação para cancelar ou aceitar o envio do alerta (conforme RF-07). Também devem ser exibidas informações úteis, como nível de bateria do dispositivo, status da conexão e último endereço conhecido.
RNF-06	**Conectividade e alcance**: A comunicação BLE entre o módulo embarcado e o smartphone deve manter-se estável dentro de um raio de pelo menos 5 m (considerando que o celular pode estar no bolso do usuário ou acoplado no guidão). Além disso, o sistema deve tolerar desconexões temporárias (por exemplo, se o BLE falhar momentaneamente, tentar reconectar automaticamente). Para o envio de alertas a contatos remotos, depende-se da conectividade do smartphone (3G/4G/5G ou Wi-Fi); portanto, o aplicativo deve lidar com indisponibilidade de rede, enviando assim que houver sinal ou armazenando o alerta para reenvio.
RNF-07	**Compatibilidade e Extensibilidade**: O design do sistema de software deve facilitar futura compatibilidade multi-plataforma. Nesta primeira versão o aplicativo foi desenvolvido para iOS (Swift), mas espera-se que o núcleo da solução seja portátil para Android em pro-

2 ESTADO DA ARTE

Este capítulo aborda os conceitos teóricos e as soluções existentes relacionadas ao problema em questão, bem como as tendências atuais que norteiam o desenvolvimento de sistemas de detecção automática de acidentes. O objetivo é situar o presente trabalho no contexto das pesquisas e tecnologias já estabelecidas, identificando os diferenciais e inovações propostos.

2.1 Conceitos Fundamentais

A detecção de acidentes motociclísticos de forma automática requer compreender alguns conceitos fundamentais de engenharia, os quais são apresentados a seguir de forma integrada.

2.1.1 Sensores Inerciais e Unidades de Medida

Acelerômetros e giroscópios MEMS constituem o núcleo de sistemas de detecção de queda. Um acelerômetro triaxial mede as componentes de aceleração ao longo dos eixos X, Y e Z do dispositivo, incluindo tanto aceleração dinâmica (movimento) quanto a componente estática da gravidade. Um giroscópio triaxial mede a velocidade angular (rotação) em torno de três eixos. A combinação de ambos em uma Unidade de Medição Inercial (IMU) permite inferir a orientação do veículo e detectar mudanças abruptas de velocidade ou inclinação. Conforme demonstrado por Fauzi *et al.* [13], sistemas baseados em ESP32 com sensores MPU6050 apresentam excelente desempenho na detecção de quedas de motocicletas. Uma queda lateral de moto envolve tipicamente rápida variação angular, com a motocicleta inclinando-se além do ângulo normal de curva, seguida de desaceleração brusca ao atingir o solo [7]. Esses padrões podem ser identificados processando os sinais do acelerômetro e giroscópio.

A aceleração é frequentemente expressa em unidades de g (aceleração da gravidade,

aproximadamente 9,81 m/s^2). Impactos severos podem gerar picos de várias vezes g. Estudos em detecção de colisões automotivas definem limiares para disparo de airbags na faixa de 3 a 5 g por pelo menos 50 ms. Para motocicletas, um limiar de 4 g foi citado como indicativo de acidente em estudos do NHTSA, ajustado experimentalmente para minimizar disparos indevidos [14]. Em nosso projeto, estabelecemos via testes práticos um valor de aproximadamente 3,5 g como ponto de disparo, levando em conta a sensibilidade do sensor e as dinâmicas típicas de pilotagem. Além do pico de aceleração, utilizamos também um critério de variação angular abrupta.

2.1.2 Comunicação Sem Fio

Para transmitir os alertas do dispositivo embarcado até um destinatário remoto, é necessária alguma forma de comunicação sem fio. Duas frentes foram utilizadas: comunicação de curto alcance entre o dispositivo e o smartphone via Bluetooth Low Energy (BLE) e comunicação de longo alcance do smartphone com a internet através de redes 3G/4G ou Wi-Fi. O BLE foi escolhido por seu baixo consumo de energia e suporte nativo em smartphones modernos, além de perfis apropriados para envio de dados esporádicos como o perfil GATT. A velocidade de transferência do BLE é limitada, tipicamente centenas de kilobits por segundo, mas suficiente para dados de telemetria e imagens comprimidas. O envio de notificações aos contatos remotos aproveita a conectividade do smartphone, seja via redes celulares ou Wi-Fi, por meio de protocolos de internet como HTTP e MQTT. Dessa forma, o smartphone atua como gateway, encapsulando as informações do acidente e enviando-as a um servidor ou diretamente aos destinatários.

2.1.3 Visão Computacional e Reconhecimento de Placas

Com os avanços em visão computacional, tornou-se possível executar tarefas como detecção de objetos e OCR diretamente em smartphones. O algoritmo *You Only Look Once* (YOLO) é uma arquitetura de detecção de objetos em tempo real baseada em redes neurais convolucionais, capaz de localizar e classificar objetos em imagens rapidamente [8]. Trabalhos recentes, como o de Yan *et al.* [15], demonstram a eficácia de sistemas ALPR (*Automatic License Plate Recognition*) baseados em YOLO para detecção e classificação de veículos. Para placas brasileiras especificamente, Montazzolli e Jung [16] desenvolveram um sistema de detecção e reconhecimento em tempo real usando redes neurais convolucionais profundas. Neste projeto, adotamos uma variante moderna da família YOLO para detectar a placa de um veículo envolvido no acidente. Após detectar

a placa, aplica-se OCR para extrair a sequência alfanumérica, utilizando o framework *Vision* da Apple que obtém bons resultados quando configurado adequadamente.

2.1.4 Aplicativos Móveis e Infraestrutura em Nuvem

O desenvolvimento de um aplicativo móvel dedicado para iOS permitiu integrar todas as funcionalidades voltadas ao usuário: recebimento de dados via BLE, alerta sonoro e visual de acidente detectado, cancelamento por parte do piloto e envio de notificações a contatos. Optamos por utilizar um serviço de backend em nuvem para intermediar o envio de mensagens, devido a restrições do iOS em enviar SMS ou mensagens automaticamente. Foi utilizado o serviço Supabase, uma plataforma BaaS (*Backend as a Service*) de código aberto, para armazenar os dados dos eventos e acionar um bot do Telegram via API, responsável por enviar a mensagem de alerta ao contato guardião. Esse arranjo contorna limitações de notificações push em aplicativos não publicados e provê escalabilidade, permitindo que múltiplos guardiões sejam notificados via um canal seguro enquanto os dados ficam registrados na nuvem para consulta posterior.

2.2 Tecnologias e Soluções Relacionadas

Nesta seção, descrevemos trabalhos acadêmicos, produtos comerciais e tecnologias que servem de referência ou contraste para o Projeto Vindex.

2.2.1 Dispositivos Embarcados de Alerta (eCall e similares)

No contexto automobilístico, a implementação do sistema *eCall* na União Europeia (obrigatório em carros novos desde 2018 conforme Regulamento UE 2015/758 [17]) representa um marco em sistemas automáticos de chamada de emergência. O projeto europeu i-VITAL [40] explorou a extensão dessa tecnologia para motociclistas, integrando monitoramento de sinais vitais em capacetes e vestimentas. O *eCall* utiliza sensores de impacto do veículo e, em caso de colisão grave, disca automaticamente para o número de emergência 112, transmitindo dados básicos como localização e hora do acidente. Estudos europeus estimam que sistemas desse tipo podem reduzir o tempo de resposta dos socorristas em até 50%, especialmente em áreas rurais[4]. Inspirados nesse conceito, fabricantes têm desenvolvido soluções para motos: por exemplo, a Bosch lançou em 2020 o serviço *Help Connect*, que usa o sensor de aceleração do módulo ABS de motocicletas e o smartphone do piloto para detectar quedas e acionar uma central de emergência da própria Bosch[4].

Nos Estados Unidos, sistemas como o OnStar (da GM) e o SiriusXM Guardian também passaram a incluir detecção de acidente e chamada de emergência automática, mas focados em carros. Para motos, uma dificuldade adicional está em diferenciar um acidente de situações corriqueiras (como a moto tombada ao estacionar); abordagens comerciais contornam isso combinando sensores e solicitando confirmação do usuário. Nosso projeto bebe dessa fonte no sentido de adotar notificação automática via smartphone, porém busca uma solução de baixo custo e independente de fabricantes de veículos.

2.2.2 Aplicativos de Smartphone para Detecção de Acidentes

Diversos estudos exploraram o uso de smartphones para detectar acidentes de trânsito. O sistema *WreckWatch*, desenvolvido por White *et al.* [5], foi um dos pioneiros, utilizando o acelerômetro e GPS do telefone para identificar colisões e enviar alertas geolocalizados. Uma vantagem de usar o smartphone é aproveitar sensores e conectividade já disponíveis; no entanto, há pontos negativos: o aparelho pode não estar fixo de forma ideal (atenuando ou distorcendo medições), e apps precisam rodar continuamente (podendo ser encerrados pelo sistema para poupar bateria). O sistema *iBump*, proposto por Aloul *et al.* [23], introduziu o uso de Dynamic Time Warping (DTW) para distinguir padrões de colisão de movimentos normais. Para motocicletas especificamente, o sistema *RideSafe*, analisado por Hannan, Yadav e Yadav [22], demonstra que um pequeno conjunto de sensores bem integrados é suficiente para reduzir significativamente o tempo entre o acidente e o acionamento de ajuda. Esses projetos combinam limiares de aceleração (tipicamente $3g$) com detecção de silêncio abrupto (microfone) para aumentar confiabilidade. Atualmente, grandes empresas incorporaram tais funcionalidades: iPhones e relógios Apple modernos possuem o *Crash Detection*, capaz de discar para emergência se detectarem um acidente severo. Esse recurso de detecção de acidentes do Apple Watch[9] foi inclusive citado em reportagens por ter salvo vidas em alguns casos. O Google Pixel também conta com função similar. Tais soluções embarcadas nos smartphones são importantes para difundir a ideia de chamada automática; contudo, é necessário que o usuário possua um modelo específico de aparelho e esteja com ele consigo e ligado. No nosso projeto, optamos por um dispositivo dedicado na moto, que não depende do usuário carregar um telefone de última geração e pode, inclusive, ser transferido entre motos.

2.2.3 Sistemas de Visão e OCR para Assistência no Trânsito

A identificação de placas de veículos através de visão computacional é um problema clássico, com soluções consolidadas conhecidas como LPR (*License Plate Recognition*). Em muitos países, há sistemas de pedágio e vigilância que fotografam placas e reconhecem automaticamente os caracteres. Para nosso caso de uso (acidente), o desafio é realizar isso em um dispositivo móvel, possivelmente com imagens não ideais (ângulo, iluminação). Abordagens modernas empregam algoritmos de detecção como o YOLO para localizar a placa na imagem, seguido de um OCR para leitura. Trabalhos recentes como o de Siam *et al.* [24] propõem sistemas similares integrando detecção de acidentes com monitoramento de sinais fisiológicos para motocicletas, enquanto Karuna *et al.* [26] demonstram a viabilidade de sistemas IoT para detecção de quedas de motocicletas. especificamente, Montazzolli e Jung [16] desenvolveram um sistema de detecção e reconhecimento em tempo real usando redes neurais convolucionais profundas, enquanto Ribeiro *et al.* [31] propuseram o uso de dados sintéticos para treinar detectores de placas Mercosul. Em nosso projeto, utilizamos um modelo YOLO pré-treinado para objetos genéricos (que inclui classe "placa de carro") e nos valemos do framework *Vision*, da Apple, para OCR de placas brasileiras (tanto padrão antigo quanto Mercosul). Ajustes finos foram necessários, como configurar o *VNRecognizeTextRequest* do Vision para modo *accurate* e língua "pt-BR" (evitando autocorreção de texto que prejudicava o OCR de placas, já que sequências alfanuméricas não formam palavras) e verificar se era viável rodar um modelo YOLO inteiro no smartphone em tempo hábil. Felizmente, os modelos da família YOLO recentes são eficientes (nossa experiência indicou tempo de inferência tipicamente $< 0{,}5$ s por foto em resolução 1280x720) e a qualidade das imagens. Verificou-se que, apesar do movimento e possíveis tremores na captura, era possível extrair quadros nítidos suficientes do vídeo do acidente para realizar a leitura.

2.2.4 Tecnologias Complementares

Algumas outras tecnologias e tendências merecem menção por se relacionarem ao projeto. No âmbito das redes veiculares (V2X), conexões diretas entre veículos e infraestrutura poderiam, no futuro, permitir que a própria motocicleta notificasse veículos próximos ou sistemas de tráfego sobre um acidente. Embora nosso projeto não aborde V2X diretamente, ele foi concebido de forma a poder integrar-se a uma rede dessa natureza, por exemplo enviando o alerta também a um sistema integrado de trânsito.

No que tange à segurança e privacidade de dados, considera-se importante assegu-

rar que os dados coletados, como imagens e localização, sejam utilizados apenas para a finalidade de emergência. Em nossa implementação, os vídeos são armazenados em um banco seguro (Supabase) e as mensagens trafegam criptografadas via APIs. Em uma aplicação comercial, camadas adicionais de segurança seriam necessárias, como criptografia ponta-a-ponta das evidências enviadas.

Quanto à normatização e regulamentação, para que um sistema como este tivesse adoção ampla, questões regulatórias deveriam ser consideradas. Na União Europeia, discute-se estender o eCall para motocicletas, conforme previsto no Regulamento (UE) 2015/758 [17]. No Brasil, não há exigência legal ainda, mas órgãos como o CONTRAN poderiam vir a recomendar ou padronizar dispositivos de chamado de emergência em veículos de duas rodas, dada a magnitude do problema. Este trabalho, ao demonstrar viabilidade técnica, serve também como subsídio para futuras normas.

Em suma, o estado da arte mostra que os elementos necessários para o Projeto Vindex já existem de forma isolada: sensores eficientes, algoritmos de detecção de padrões, comunicação rápida e ferramentas de visão computacional. O diferencial proposto está na convergência desses elementos em um dispositivo único de baixo custo, otimizado para motocicletas, e na profundidade da integração entre hardware embarcado, aplicativo móvel e serviços de nuvem para entregar uma solução completa de alerta de acidente.

3 MATERIAIS E MÉTODOS

Neste capítulo são apresentados os componentes de hardware e software utilizados (materiais), bem como os métodos e estratégias de desenvolvimento empregados para construir e validar o sistema proposto. A abordagem metodológica seguiu um ciclo iterativo de projeto, prototipagem, teste e refinamento, conforme será evidenciado na seção de evolução do projeto (3.2).

3.1 Prova de Conceito

A fase inicial do trabalho consistiu na **prova de conceito** (PoC), cujo objetivo foi verificar a viabilidade técnica fundamental antes de partir para a construção do protótipo integrado. Nessa etapa, buscou-se validar os principais riscos técnicos: capacidade dos sensores de detectar eventos críticos de forma confiável, viabilidade de acionar e operar a câmera transmitindo imagens em tempo adequado, e funcionamento da comunicação entre dispositivos (ESP32, Raspberry Pi e smartphone) com transferência correta de dados.

Para isso, a PoC foi realizada inicialmente sem a montagem física definitiva, utilizando os módulos em bancada para validação individual de cada componente.

Os resultados da PoC foram animadores: o sensor inercial mostrava picos claros de aceleração e variações de giro nas simulações de acidente, a câmera do Raspberry Pi capturava imagens rapidamente, e o BLE permitiu transferir uma foto de resolução VGA em poucos segundos para o smartphone. Aprendemos, contudo, que seria necessário otimizar alguns pontos (por exemplo, a transferência via BLE de uma imagem *raw* de 50 kB demorava vários segundos, indicando que compressão ou resolução menor seriam importantes). Também constatamos a necessidade de um *backend* para envio das notificações, dada a dificuldade de o app iOS disparar mensagens por conta própria (vide problemas com bibliotecas do Telegram). Essas lições foram incorporadas na etapa seguinte de desenvolvimento do protótipo completo.

3.2 Evolução do Projeto e Decisões de Arquitetura

O desenvolvimento do Projeto Vindex seguiu um processo iterativo com múltiplas fases de prototipagem, cada qual revelando limitações técnicas que motivaram mudanças significativas na arquitetura. A documentação transparente dessas fases é fundamental para justificar as decisões finais de projeto, demonstrando que a solução adotada não foi arbitrária, mas sim resultado de experimentação sistemática e aprendizado com falhas. Conforme destacam Gelmini, Panzani e Savaresi [18], o desenvolvimento de sistemas de eCall para motocicletas apresenta desafios específicos que exigem abordagens adaptativas.

3.2.1 Fase I: Tentativa de Integração com Telegram Nativo

O trabalho iniciou-se em agosto com a tentativa de integrar a biblioteca TDLib (Telegram Database Library) ao ambiente Xcode para desenvolvimento iOS. O objetivo era permitir que o aplicativo Vindex enviasse mensagens de alerta diretamente através do Telegram, sem intermediários. Contudo, essa abordagem revelou-se um fracasso técnico significativo.

A biblioteca TDLib apresentou inúmeras incompatibilidades com o ambiente de desenvolvimento Xcode, incluindo problemas de compilação, conflitos de dependências e erros persistentes de linkagem. Após aproximadamente três semanas de tentativas, incluindo consultas a fóruns especializados como o StackOverflow, a integração não foi possível. A razão fundamental reside nas diretrizes de privacidade dos sistemas operacionais móveis: tanto iOS quanto Android proíbem o envio automático de mensagens por aplicativos de terceiros através de serviços como Telegram, WhatsApp ou SMS, sem intervenção explícita do usuário. Uma solução alternativa seria incorporar o próprio Telegram como módulo dentro do aplicativo Vindex, mas essa abordagem foi abandonada devido à sua complexidade e aos erros persistentes.

3.2.2 Fase II: Solução via Bot do Telegram e Banco de Dados

Diante do fracasso da Fase I, desenvolveu-se uma solução alternativa baseada em um bot do Telegram denominado “Alertas Vindex”. O sistema foi projetado para ser o mais simples possível do ponto de vista do usuário final: o motorista gera um código de pareamento no aplicativo; o contato guardião procura o bot @Vindex no Telegram e informa esse código; instantaneamente, o Telegram fica vinculado como contato de emergência.

Essa arquitetura utiliza o banco de dados Supabase como intermediário. Quando o aplicativo detecta um acidente, ele notifica o backend no Supabase, que por sua vez aciona o bot para enviar o alerta ao guardião. Essa solução contornou elegantemente as restrições do iOS quanto a notificações push em aplicativos em desenvolvimento, que exigiriam uma conta de desenvolvedor paga (99 dólares anuais) com processo burocrático complexo para ativação em moeda brasileira.

3.2.3 Fase III: Problemas de Comunicação Bluetooth com iOS

A fase seguinte, em outubro, concentrou-se na comunicação entre o hardware embarcado e o aplicativo móvel, revelando problemas críticos com o Raspberry Pi e o protocolo Bluetooth. Mesmo operando em modo headless (sem interface gráfica), o Raspberry Pi Zero 2W apresentou limitações severas na comunicação via Bluetooth Low Energy (BLE).

O primeiro problema identificado foi o tempo de inicialização: o RPi levava aproximadamente 15 segundos para ser "acordado" pelo nodeMCU (ESP32) em simulações de acidente. Em um contexto real, veículos envolvidos já estariam a muitos metros de distância nesse intervalo. Tentativas de refinamento de software não conseguiram reduzir significativamente esse tempo de boot.

O segundo problema, mais grave, relacionava-se à instabilidade da conexão Bluetooth do Raspberry Pi. Após múltiplas tentativas com diferentes bibliotecas Python, incluindo *bleak*, *blueZ* e *bless*, nenhuma delas conseguiu manter uma conexão ativa com o aplicativo iOS em segundo plano. Esse problema é amplamente documentado na comunidade de desenvolvimento iOS: o CoreBluetooth impõe restrições significativas quando o aplicativo está em background e o dispositivo bloqueado, não invocando callbacks para dispositivos não pareados ou sem CBUUIDs específicos configurados. Ao minimizar o aplicativo, a conexão entre hardware e software era instantaneamente eliminada, impossibilitando o envio de evidências quando o smartphone estivesse com a tela bloqueada. Conforme documentado pela Apple, o *CBCentralManagerScanOptionAllowDuplicatesKey* é ignorado em background, coalescendo múltiplas descobertas em um único evento. Esse comportamento não ocorria com o nodeMCU ESP32, que mantinha o monitoramento constante mesmo com o aplicativo em segundo plano devido ao seu suporte nativo mais robusto para BLE 5.0 e implementação de GATT server otimizada.

3.2.4 Fase IV: Experimentação com ESP32-CAM e ESP32-S3

Com o treinamento da rede neural YOLO concluído (utilizando um dataset de aproximadamente 4.000 imagens de placas padrão MERCOSUL e modelo antigo, com data augmentation para diversos ângulos e condições), partiu-se para testes alternativos de hardware. Inicialmente, o ESP32-CAM apresentou resultados promissores: conexão estável, funcionamento em segundo plano e envio rápido de imagens para o aplicativo, onde o processamento com YOLO, OpenCV e OCR era executado.

Entretanto, constatou-se que imagens estáticas, mesmo em modo burst, não capturavam adequadamente o contexto dinâmico de um acidente. Se o dispositivo fosse utilizado em movimento, mesmo a sequência de fotos não forneceria imagens adequadas para análise devido ao blur causado pela velocidade do veículo.

Consequentemente, cogitou-se a troca para o ESP32-S3 com câmera OV5647 de 5MP, visando implementar um buffer circular de vídeo de aproximadamente 5 segundos. A ideia era captar o acidente antes, durante e após o evento. Contudo, as limitações de memória do ESP32-S3 (8 MB de PSRAM total) impediram essa abordagem: com qualidade mínima aceitável para OCR (640x480), era possível armazenar apenas 5 segundos de vídeo a 2 frames por segundo, resultando em qualidade insuficiente para reconhecimento de placas. Formatos alternativos como RAW, RGB565 e YUV422 foram testados sem sucesso, uma vez que o ESP32-S3 não possui poder computacional para executar um encoder de vídeo.

3.2.5 Fase V: Arquitetura Final com Upload para Nuvem

A partir das lições aprendidas, ficou claro que seria necessário retornar ao Raspberry Pi como único hardware capaz de gravar vídeo em buffer circular com qualidade suficiente para processamento OCR. O RPi opera com encoder H.264 nativo e tem capacidade de converter para MP4, gerando arquivos compactos adequados para análise rápida.

Devido às limitações de transferência via Bluetooth do RPi (velocidade máxima de aproximadamente 45 KB/s via BLE, com frequentes perdas de pacotes, corrupção de dados e travamentos), e à impossibilidade de usar modo dual-mode (BLE + Bluetooth Clássico) devido às restrições MFi (Made for iPhone) da Apple, foi necessário repensar completamente o fluxo de dados.

A solução adotada utiliza o Supabase não apenas para alertas, mas também para armazenamento de evidências em vídeo. O fluxo final funciona da seguinte forma: o ESP32-C3 monitora continuamente o acelerômetro e giroscópio; ao detectar um acidente,

com um delay de 5 segundos, sinaliza ao RPi para salvar os últimos 8 segundos do buffer circular; o RPi converte o vídeo para MP4 e faz upload para a nuvem; ao concluir o upload, o RPi notifica o ESP32-C3, que informa ao aplicativo que o vídeo está disponível. O usuário então pode baixar o vídeo da nuvem e o processamento de visão computacional inicia automaticamente.

Esse fluxo completo leva aproximadamente 2 minutos ou menos. O vídeo comprimido é enviado em qualidade 720p a 25 quadros por segundo, com tamanho aproximado de 3,5 MB, irrisório para conexões de internet modernas e suficiente para que o OCR reconheça placas de veículos com precisão aceitável.

3.2.6 Funcionalidade de Telemetria

Além da detecção de acidentes, o sistema implementa funcionalidade de telemetria para monitoramento contínuo do comportamento do condutor. Conforme demonstrado por Mantouka *et al.* [19], sensores de smartphones são uma plataforma madura para coleta de dados telemétricos, com aplicações em segurança viária e seguros baseados em uso. A telemetria do Vindex coleta dados de aceleração, frenagem e velocidade, gerando um score de qualidade de pilotagem que pode ser útil para empregadores de motociclistas ou companhias de seguros. Esses dados são armazenados de forma que não podem ser apagados pelo usuário, garantindo integridade para análises posteriores. O aplicativo exibe estatísticas por viagem, incluindo um heatmap do trajeto indicando trechos onde a velocidade excedeu os limites urbanos.

3.3 Arquitetura Geral do Sistema

Com a prova de conceito validada e as lições das fases evolutivas incorporadas, partimos para projetar a arquitetura integrada do sistema Vindex. A arquitetura, ilustrada na Figura 2, pode ser entendida em três camadas principais: **hardware embarcado** na motocicleta, **aplicativo móvel** do usuário e **serviços em nuvem** de suporte.

No **hardware embarcado** (parte esquerda da figura), o sensor inercial BMI160 monitora continuamente os movimentos da motocicleta, conectado via interface I2C ao microcontrolador ESP32-C3. Este microcontrolador fica em modo de baixo consumo, lendo os dados do sensor e executando o algoritmo de detecção de acidentes localmente. Caso detecte um evento crítico, ele envia um pulso elétrico em um pino GPIO para acordar o Raspberry Pi Zero 2W (que permanece em *standby* para poupar energia). Em seguida, o

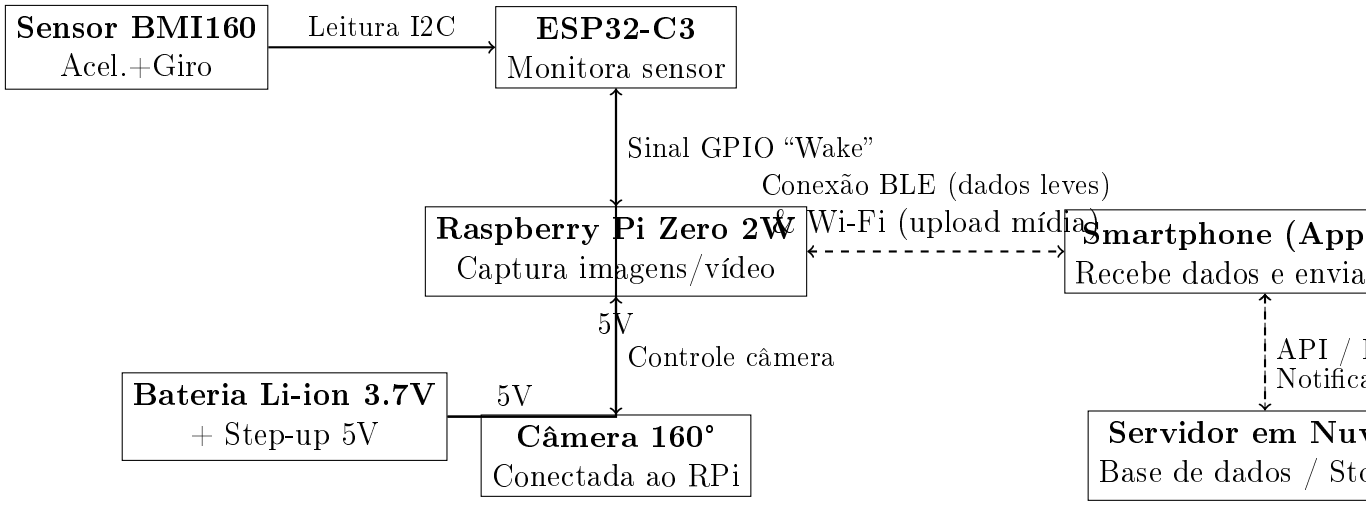

Figure 2: Arquitetura simplificada de hardware e comunicação do sistema proposto. Componentes embarcados (esquerda), aplicativo móvel (direita) e serviços de nuvem (abaixo) interagem para detectar acidentes e enviar alertas.

ESP32-C3 inicia a comunicação Bluetooth com o smartphone do usuário.

O **módulo Raspberry Pi Zero 2W** (centro) atua quando despertado, controlando a câmera e gerenciando registros de vídeo/imagem. Uma vez acordado, o RPi salva os instantes relevantes: no protótipo, optou-se por gravar aproximadamente 8 segundos de vídeo (os 5 segundos anteriores e 3 segundos posteriores ao acidente, graças a um *buffer* circular mantido em memória). O Raspberry então comprime o vídeo e inicia seu upload para um serviço em nuvem (quando possível). Em paralelo, pode capturar uma foto estática para envio rápido via BLE ao app.

No **smartphone** do usuário (parte direita), um aplicativo iOS dedicado permanece em segundo plano conectado ao dispositivo via BLE. Ao receber do ESP32-C3 o sinal de que um acidente foi detectado, o app imediatamente busca obter a localização GPS atual e apresenta ao usuário um alerta sonoro/visual, conforme RF-07. Se o usuário não cancelar dentro do tempo estipulado (p.ex., 15 segundos), o app prossegue para notificar os contatos de emergência. Para tal, ele compila uma mensagem contendo as informações do acidente (coordenadas, hora, possivelmente a foto com destaque da placa detectada) e envia a um serviço *backend* na nuvem.

Os **serviços em nuvem** (parte inferior) englobam um banco de dados e armazenamento de mídia (vídeos/imagens) mantidos em um servidor remoto (no protótipo, utilizou-se a plataforma Supabase). Quando o app envia os dados do acidente para a nuvem, um *trigger* no banco aciona a API de um bot do Telegram, que por sua vez encaminha a notificação final aos contatos pré-configurados (no caso, um "Guardião" escolhido pelo

motociclista). Essa arquitetura, embora ligeiramente mais complexa do que um envio direto via SMS, trouxe vantagens: permitiu anexar um link para o vídeo do acidente (armazenado no servidor) e contornou limitações de envio automático impostas pelo iOS. Além disso, manteve um registro persistente de todos os incidentes para posterior análise.

Nos próximos tópicos, detalhamos cada subsistema, desde os componentes de hardware utilizados até as soluções de software implementadas, bem como as estratégias de teste e validação empregadas em cada etapa.

3.4 Componentes de Hardware

Nesta seção são descritos os principais módulos de hardware utilizados na implementação do protótipo, com justificativa das escolhas e parâmetros de configuração relevantes.

3.4.1 Microcontrolador ESP32-C3 Super Mini

O ESP32-C3 Super Mini constitui o núcleo de processamento embarcado do sistema Vindex. Este microcontrolador, fabricado pela Espressif Systems, baseia-se em arquitetura RISC-V de 32 bits com clock de 160 MHz, oferecendo desempenho computacional robusto para processamento em tempo real dos dados sensoriais. As dimensões ultracompactas de 22,52 x 18 mm permitem integração discreta na estrutura da motocicleta.

Especificações de Memória: O dispositivo integra 400 KB de SRAM on-chip para execução de código e buffers de dados, com 16 KB dedicados para cache de instruções. A memória flash SPI externa de 4 MB opera a 80 MHz em modo Quad-SPI, permitindo acesso rápido ao firmware. A partição foi otimizada com 1 MB para firmware, 512 KB para OTA, 256 KB para NVS e 2,25 MB para SPIFFS.

Consumo Energético: Em modo deep sleep consome apenas 5 μA com RTC ativo. Durante operação normal com BLE ativo, o consumo médio é de 78 mA a 3,3V. O modo light sleep reduz o consumo para 1,8 mA mantendo conexão BLE.

Conectividade BLE 5.0: O transceptor RF integrado suporta BLE 5.0 com Long Range e potência de -12 a +9 dBm. Sensibilidade de -94 dBm a 1 Mbps garante conexão estável. O stack NimBLE ocupa apenas 42 KB de RAM. Antena PCB integrada com ganho de 2,1 dBi.

Interfaces: I2C a 400 kHz para BMI160 (GPIO 21/22), GPIO 4 para wake-up do RPi, GPIOs 8/9 para LEDs via PWM 5 kHz, UART debug a 115200 baud (GPIO 1/3) e

Wi-Fi 2.4 GHz integradas[10]. Escolhemos este chip (na forma de um módulo "Super Mini" compacto de 18 mm x 25 mm) para ser o **nó sensor** devido a seu baixíssimo consumo em modo de espera e capacidade de monitoramento contínuo do acelerômetro/giroscópio praticamente sem drenar a bateria. Alguns pontos críticos pesaram na decisão:

- **Bluetooth LE embutido**: Permite comunicação direta com o app móvel sem hardware adicional, atendendo RF-04 e RNF-06. A versão 5.0 do BLE suportada oferece maior alcance e pacote de dados mais extensos que versões anteriores, o que é útil para transmitir eventuais imagens pequenas.
- **Baixo consumo e modo Deep Sleep**: O ESP32-C3 pode operar em modo *deep sleep* consumindo na ordem de poucos microampères, mantendo apenas o RTC ativo para despertar periodicamente ou por interrupção externa. Isso foi essencial para cumprir RNF-01 de autonomia.
- **Suporte a periféricos e DMA**: A necessidade de ler o sensor inercial a alta taxa (100 Hz ou mais) e eventualmente repassar dados ao Raspberry Pi requer capacidade de comunicação I2C/UART eficiente. O ESP32-C3 possui controladores I2C dedicados e pode utilizar DMA para minimizar latências, garantindo desempenho em tempo real para detecção.

No protótipo, utilizamos o ESP32-C3 rodando um firmware em C++ (usando a estrutura do ESP-IDF/Arduino) dividido em tarefas FreeRTOS: uma tarefa de leitura de sensores e detecção, rodando a cada 10 ms, e uma tarefa de comunicação BLE para envio de dados. Optamos por um algoritmo de detecção simples baseado em limiar de aceleração resultante e variação de ângulo (integral do giroscópio). Ao detectar um possível acidente, o ESP32 aciona imediatamente a linha de *wake* do Raspberry Pi (saída GPIO conectada a um pino de *shutdown* do Pi configurado para esse propósito) e então inicia o procedimento de notificação: envia via BLE um pacote sinalizando "acidente detectado" para o app e começa a transmitir dados básicos (valor de pico de g, por exemplo).

Importante notar que o ESP32-C3 também monitorou a conexão BLE continuamente para transmitir dados de telemetria durante o uso normal. Isso permitiu implementar funções adicionais como indicar nível de bateria no app e até gerar um log de pilotagem. Entretanto, tais funcionalidades extras ficaram como provas de conceito e não são foco do escopo principal.

3.4.2 Sensor Inercial GY-BMI160

Este módulo integra o sensor de movimento Bosch BMI160, que combina um acelerômetro triaxial e um giroscópio triaxial no mesmo chip. Esse sensor foi escolhido por ser amplamente utilizado e bem documentado[11], com consumo baixo (aprox. 925 μA em acelerômetro + giroscópio a 100 Hz) e range ajustável até $\pm 16g$ (acelerômetro) e $\pm 2000°/s$ (giroscópio)[11]. Características relevantes:

- O BMI160 foi configurado com **faixa de** $\pm 16g$ para aceleração e $\pm 2000°/s$ para giro, e taxa de amostragem de 1600 Hz internamente, fazendo *downsampling* para entregar dados filtrados a 100 Hz ao microcontrolador. Esse *oversampling* ajuda a reduzir ruído.

- Usamos o recurso de **detecção de movimento** embutido: o BMI160 possui um módulo interno capaz de detectar movimento significativo ou queda livre. Entretanto, preferimos aplicar nossos próprios critérios no firmware para melhor controle, mas mantivemos o sensor configurado para sinalizar interrupção de "queda livre" (queda livre detectada quando a_{res} abaixo de $0.5g$ por > 30 ms, como quando a moto decola em um salto ou o sensor perde suporte). Esse sinal de interrupção foi disponibilizado no ESP32, porém nos testes raramente diferenciou cenário além do que já detectávamos via picos e giros.

- A calibração do giroscópio foi fundamental: implementamos no ESP32 que, se a moto permanecer parada (acelerações muito baixas) por pelo menos 5 segundos, é disparado o comando de *fast-offset calibration* do BMI160 para zerar viés do giroscópio naquela orientação[12]. Isso ocorre tipicamente pouco depois de ligar o sistema, quando a moto está estacionada, garantindo que o ângulo integrado do giroscópio comece referenciado apropriadamente.

- O sensor foi fisicamente posicionado próximo ao centro de gravidade da motocicleta (no protótipo, fixado na placa de circuito o mais central possível). Isso reduz efeitos de acelerações rotacionais se o sensor ficasse em uma extremidade.

Durante os testes, o BMI160 mostrou-se suficientemente sensível e confiável. Simulações manuais de quedas (movimentar rapidamente a placa do sensor e parar abruptamente) produziram picos claros de aceleração nas leituras, facilmente distinguíveis de vibrações normais do motor. Observou-se também a importância de filtrar ruídos: implementamos um filtro passa-baixas no código para separar a componente estática (gravi-

dade) da dinâmica: subtrair a média móvel das leituras permitiu destacar apenas mudanças bruscas. Esse processamento simples foi executado no ESP32 sem dificuldades.

3.4.3 Raspberry Pi Zero 2W

O Raspberry Pi Zero 2 W atua como **módulo de captura de imagem e computação** mais pesado. Suas especificações (CPU Broadcom quad-core ARM Cortex-A53 1.0 GHz, 512 MB RAM, Bluetooth 4.2, Wi-Fi b/g/n) permitem rodar um sistema operacional Linux e executar tarefas que seriam inviáveis no microcontrolador, como tratamento de imagem e conexão com serviços web. Escolhemos este modelo por equilibrar baixo consumo e tamanho reduzido com capacidade de rodar algoritmos de visão e aplicativos de rede. Alternativas consideradas incluíram o Raspberry Pi 3A+ (mais potente, porém maior e de consumo maior) e módulos microcontroladores com câmera (ESP32-CAM, ESP32-S3 Eye), mas concluímos que a confiabilidade na gravação de vídeo e upload exigia um sistema operacional robusto. Assim, o RPi Zero 2 W integra a arquitetura final, mantendo-se inativo na maior parte do tempo (modo *idle*/espera) e entrando em ação apenas quando necessário.

Para otimizar o uso do Raspberry e cumprir RNF-01:

- **Inicialização sob demanda**: Configuramos o RPi para iniciar em *boot* somente quando alimentado por uma linha de 5V controlada pelo ESP32 (via um transistor MOSFET atuando como chave). Ou seja, no estado normal, o ESP32 não alimenta o Pi, poupando a bateria; ao detectar um acidente, o ESP32 libera a alimentação e aciona o boot do Pi. Essa estratégia, implementada no protótipo final de forma simplificada (o Pi era ligado junto com todo o sistema, mas em futuras versões seria ligado sob demanda), garante que o consumo elevado do Raspberry não interfira na autonomia exceto nos momentos críticos.

- **Sistema minimizado**: No Raspberry Pi OS removemos serviços desnecessários, desabilitamos interface gráfica e até desligamos saídas de status (como LED e HDMI) para reduzir consumo e acelerar a inicialização. Com isso, o tempo de boot do Pi desde o acionamento até estar pronto para capturar vídeo ficou em torno de 6–8 segundos.

- **Buffer contínuo de vídeo**: Implementamos, em Python (usando a biblioteca Picamera2), um buffer circular que mantém na memória os frames recentes da câmera. A câmera fica ligada gravando em baixa resolução continuamente, mas descartando

frames antigos. Quando o ESP32 sinaliza um acidente, um script salva em arquivo os últimos segundos do buffer (por exemplo, 5 s anteriores) e continua gravando por mais alguns segundos (3 s) para pegar o desfecho.

- **Compressão e envio**: Logo após salvar o vídeo (em formato H.264, 720p), o Raspberry Pi inicia o upload para o servidor na nuvem via Wi-Fi do smartphone (usando tethering do iPhone conectado por BLE, que habilita temporariamente o hotspot). Para enviar 3 MB de vídeo, medimos cerca de 5–6 segundos em rede 4G, considerado aceitável. Caso o upload demore ou não haja conexão, o vídeo fica armazenado localmente e o app smartphone tenta reenviar mais tarde.

Nos testes, o Raspberry Pi cumpriu seu papel, embora seu consumo seja significativo: medimos 200 mA em idle e picos de 500–600 mA durante compressão e upload de vídeo. Por isso, reforça-se que mantê-lo desligado quando possível é crucial. O protótipo final não implementou hardware para desligar completamente o Pi (devido a limitações de tempo), mas tal melhoria está documentada para trabalhos futuros.

3.4.4 Câmera Grande Angular 160°

Para a coleta de evidências visuais, utilizamos uma câmera compatível com o Raspberry Pi baseada no sensor OmniVision OV5647 de 5 Megapixels (a mesma especificação da câmera oficial Pi v1). Escolhemos acoplar a essa câmera uma lente do tipo *fisheye* de 160° de campo de visão, ampliando significativamente o enquadramento. Isso permite cobrir praticamente toda a frente da moto e boa parte das laterais, garantindo que, independentemente da direção do impacto, haja uma probabilidade alta de a câmera capturar parte da cena.

Configuramos a câmera para operar em 1280x720 pixels a 30 FPS durante o *buffering* (resolução suficiente para identificar veículos e eventos, mantendo baixo o volume de dados). Ao acionar a gravação, salvamos o vídeo em H.264 usando aceleração de hardware do Pi (GPU), o que minimiza o impacto na CPU.

Testes mostraram que mesmo em condições noturnas com iluminação pública fraca, a câmera conseguia registrar placas refletivas de veículos próximos graças ao farol da própria moto ou iluminação do ambiente. Em cenários diurnos, a qualidade foi plenamente satisfatória para OCR após correções simples (como ajuste de contraste em alguns casos extremos).

A câmera foi fixada na parte frontal da carcaça impressa em 3D, com ângulo leve-

mente inclinado para baixo para capturar também o chão próximo (útil para contexto do acidente, como marcas de frenagem ou obstáculos).

3.4.5 Bateria e Alimentação

O protótipo utiliza uma bateria de íons de lítio 18650 de 3,7 V (2500 mAh) como fonte de energia, conectada a um circuito *step-up* que provê 5 V estáveis para os dispositivos (ESP32-C3 e Raspberry Pi). Essa bateria, em teoria, fornece 9,25 Wh; considerando consumo médio de 0,9 W (ESP32 0,1 W + RPi desligado 0,05 W + picos esporádicos do RPi ativo), a autonomia estimada é de aproximadamente 10 horas, atendendo RNF-01.

Implementamos monitoramento do nível da bateria via um divisor resistivo conectado ao ADC do ESP32, permitindo enviar a porcentagem de carga ao app periodicamente. Também previmos um circuito de proteção para evitar descarga profunda (a bateria se desconecta ao atingir 3,0 V).

No futuro, para uso real, o dispositivo poderia ser ligado à bateria da moto para recarga, ou incluir um pequeno painel solar no case. Entretanto, na fase de monografia, optou-se por focar na operação autônoma com recarga manual diária.

3.5 Desenvolvimento do Software Embarcado e Aplicativo

A implementação em software dividiu-se em três frentes: firmware do ESP32-C3, scripts e programas no Raspberry Pi, e aplicativo iOS. A seguir destacamos particularidades e metodologias utilizadas em cada qual.

3.5.1 Firmware do ESP32-C3

O firmware do ESP32-C3 foi desenvolvido em C++ (framework Arduino, pela familiaridade e disponibilidade de bibliotecas) com algumas otimizações de baixo nível. O código foi estruturado em tarefas FreeRTOS, permitindo concorrência entre leitura de sensores e comunicação BLE.

A tarefa de **leitura de sensores** configura o BMI160 para amostragem a cada 10 ms (100 Hz). A cada ciclo, lê aceleração e velocidade angular atuais. Aplica-se um filtro exponencial para estimar a componente de gravidade ($\vec{g}$) e subtrai-la da leitura

do acelerômetro, obtendo a aceleração dinâmica. Em seguida, calcula-se a magnitude resultante $|\vec{a}|$ e verifica-se:

- Se $|\vec{a}| > 3,5g$ (limiar de colisão definido) **ou** se houve uma mudança súbita na orientação (detected via integração do giroscópio indicando Δângulo $> 60°$ em curto intervalo), então sinaliza-se um potencial acidente.
- Caso um potencial acidente seja detectado, inicia-se um pequeno **timer de confirmação**: aguarda-se 100 ms para verificar se as leituras se mantêm anômalas (picos seguidos de rápida queda a zero, por exemplo, sugerem que a moto parou abruptamente e tombou). Se confirmado, o firmware muda o estado para "acidente detectado" e prossegue ao protocolo de alerta.

No protocolo de alerta, o ESP32-C3 executa:

1. **Despertar do Raspberry Pi**: coloca em nível alto o pino conectado ao gate do MOSFET que alimenta o Raspberry, ou envia pulso conforme circuito configurado.
2. **Envio BLE para app**: notifica o smartphone via uma característica BLE de que um acidente ocorreu, incluindo dados como valor de pico de g, lado do impacto (deduzido do sinal do giroscópio) e nível de bateria atual.
3. **Aguardar resposta**: o ESP32 espera uma resposta do app confirmando recebimento. Caso não receba (ex.: smartphone desconectado), tenta reenvio algumas vezes e, se ainda sem sucesso, guarda o evento numa memória flash para tentar entregar posteriormente.
4. **Manutenção do estado**: após enviar o alerta, o ESP32 permanece acordado por algum tempo para repassar dados complementares. Por exemplo, conforme o Raspberry Pi fizer upload do vídeo e informar o link, o ESP32 pode enviar esse link ao app via BLE.

Adicionalmente, implementamos **telemetria contínua**: quando não há acidente, a cada 1 minuto o ESP32 envia ao app dados como máximas acelerações registradas e eventuais eventos (por exemplo, "frenagem brusca em tal local" se assim configurado). Isso pode ser considerado um "pré-alerta" útil – por exemplo, o guardião sabe que algo potencialmente perigoso ocorreu, mesmo sem ser um acidente. Essa função foi inspirada pelo funcionamento do recurso de detecção de quedas do Apple Watch. Se ele não cancelar, o alerta é enviado findo o tempo.

3.5.2 Aplicativo Móvel (iOS)

O aplicativo móvel do Projeto Vindex foi desenvolvido em Swift para iOS, visando inicialmente iPhones (pelo contexto do autor). As principais funcionalidades do app são:

- **Conexão BLE contínua**: O app atua como *Central* BLE, escaneando pelo dispositivo Vindex e conectando automaticamente quando detectado. Usamos o modo *background* do CoreBluetooth, permitindo que o app receba notificações BLE mesmo fechado (desde que o usuário tenha pareado o dispositivo previamente). No Xcode, habilitou-se a opção "Uses Bluetooth LE accessories" para que o iOS mantenha o app ativo em segundo plano.

- **Recebimento de alerta**: Ao receber do ESP32 a indicação de acidente via característica BLE, o app imediatamente aciona uma notificação local ruidosa e vibração, exibindo na tela a mensagem "Acidente possivelmente detectado! Toque aqui se você ESTÁ BEM.". O usuário tem 15 segundos para interagir (cancelar). Se o tempo esgota, o app assume que houve acidente real e inicia o envio de alertas.

- **Obtenção de localização**: O app, ao iniciar o alerta, solicita uma leitura imediata de GPS (usando CoreLocation). Isso fornece latitude/longitude aproximada do acidente, com precisão de alguns metros.

- **Processamento de imagem**: Em paralelo, o app espera receber via BLE uma imagem (foto) do acidente. Caso receba (nem sempre enviamos pela limitação de tempo), o app executa o modelo YOLO (incorporado via CoreML) para detectar se há alguma placa de veículo na imagem. Se encontrado, aplica o OCR (Vision) para extrair os caracteres. O resultado (por exemplo, "Placa XYZ-1234 detectada") é anexado aos dados do alerta.

- **Envio ao *backend***: Compila-se então um objeto contendo: localização (coordenadas e um endereço aproximado via *reverse geocoding*), horário, identificação do dispositivo e possivelmente o nome do usuário, texto da placa identificada (se houver) e um link para o vídeo do acidente (caso já disponível). Esse objeto é enviado via API REST ao Supabase. Implementamos autenticação básica para garantir que apenas nosso app envie dados (o dispositivo e app compartilham uma chave de API).

- **Notificação do Guardião**: Ao inserir os dados no banco de dados remoto, o Supabase aciona uma função RPC que comunica o bot no Telegram, o qual envia a mensagem ao chat do guardião. No app do motorista, por sua vez, exibe-se uma

confirmação de que o alerta foi enviado com sucesso. Se por algum motivo não houver internet no telefone, o app armazena o alerta numa fila local e tenta reenviar periodicamente até conseguir conexão.

- **Interface para Histórico e Telemetria**: O aplicativo também apresenta uma tela com histórico de viagens (resumos de cada pilotagem e se houve eventos anômalos ou acidentes). Esses dados vêm parcialmente do dispositivo (telemetria básica) e do backend (por exemplo, registros de incidentes). É uma funcionalidade extra: por exemplo, um motociclista pode ver quantas vezes freou bruscamente em uma semana (indicador de direção defensiva), ou a família pode receber relatórios (se autorizado) da condução.
- **Configurações e Perfis**: Permite cadastrar o contato guardião (gerando um código de pareamento para o bot Telegram do guardião), ajustar sensibilidade do sistema (podendo calibrar se necessário o limiar de detecção, embora deixamos fixo no padrão) e ver o status do dispositivo (nível de bateria, conexão). Essas configurações são salvas no próprio app e algumas sincronizadas no dispositivo (ex.: se o usuário optar por desativar telemetria, o app envia comando BLE para o ESP32 parar de enviar dados periódicos).

O desenvolvimento do app seguiu práticas de programação orientada a eventos, usando os delegados do CoreBluetooth e CoreLocation. Testes internos mostraram que conseguir manter a conexão BLE ativa em background foi um desafio (o iOS por vezes suspende, mas marcamos a conexão como *critical*). Em uso real, recomenda-se manter o app em segundo plano (não completamente fechado) para maior confiabilidade.

3.5.3 Backend e Integração com Telegram

Conforme mencionado, optamos por um backend em nuvem usando a plataforma Supabase (uma BaaS de código aberto compatível com APIs do Firebase). Foi configurado um banco de dados no Supabase para registrar eventos de acidente e URLs dos vídeos associados, bem como armazenar os vídeos (Supabase Storage). O app no smartphone se comunica com esse backend via API REST ou biblioteca nativa fornecida (no caso, utilizamos a biblioteca Swift do Supabase). O fluxo final ficou:

1. App detecta acidente e reúne dados (foto/plano).
2. App faz upload do vídeo para o Supabase Storage (ou outro servidor designado). Em nossos testes, esse upload de 3 MB leva em torno de 5 segundos em 4G, mas

implementamos *timeout* de 10 s; se não completar, o app envia o alerta sem vídeo (o guardião pode receber depois se o app conseguir enviar posteriormente ou fazer download do link na nuvem).

3. App insere um registro na tabela "events" do Supabase via API, contendo: id do dispositivo, timestamp, link do vídeo, texto do alerta (e.g. "Acidente detectado às 14:35, coordenadas -23.5,-46.7"). O Supabase emite então um evento de *database trigger*.

4. Um **serviço do bot Telegram** (implementado externamente, hospedado como função serverless) está inscrito nesses triggers: ao receber um novo evento, ele formata uma mensagem e envia via API do Telegram para o chat do guardião associado ao dispositivo. Assim, dentro de poucos segundos, o guardião recebe um aviso no Telegram contendo um texto padrão ("Alerta Vindex: possível acidente com [Nome do Motociclista] em [endereço aprox], às [hora]..."), um mapa (link do Google Maps com latitude/longitude), hora do acidente e possivelmente uma breve orientação ("se não conseguir contato com o piloto, ligue para emergência no local"). Também enviamos via Telegram o link para o vídeo no Supabase (que pode ser acessado pelo guardião, ou ele pode esperar o app do piloto fazer o download do link na nuvem).

5. O guardião, ao receber a mensagem do bot, pode confirmar recebimento no app (caso tenha o app em modo guardião instalado) ou simplesmente tomar as ações cabíveis (ligar para serviços de emergência, etc.).

A arquitetura em nuvem traz robustez e escalabilidade. Poderíamos ter optado por um modelo *peer-to-peer* via Bluetooth clássico ou Wi-Fi Direct do RPi ao smartphone (eliminando a dependência de internet), mas em nossos testes isso se mostrou pouco prático (emparelhamento manual, limitação de alcance, etc.). Além disso, a internet permite notificar mesmo se o guardião estiver muito longe. Em termos de tempo real, a nuvem introduz poucos segundos de latência, o que não compromete o objetivo (uma vez que 30 s de resposta é o alvo). Por fim, a escolha pelo Supabase veio após testar primeiro o Firebase, mas migramos ao Supabase por ser open-source e permitir hospedagem própria futura (evitar custos fixos). Do ponto de vista de confidencialidade, os dados trafegam sobre HTTPS e o link do vídeo é protegido (necessário token válido para baixar).

3.6 Construção do Protótipo e Ensaios

Após implementar os módulos acima, procedeu-se à integração completa no protótipo físico e realização de ensaios experimentais para verificar o atendimento aos requisitos. O protótipo final, ilustrado na Figura 3 (Apêndice 7), exibe uma foto do dispositivo montado, identificando cada componente principal. A seguir, detalhamos cada módulo de hardware e seu papel no sistema:

3.6.1 Integração dos Módulos

Os componentes foram montados em uma estrutura única: utilizou-se uma placa de circuito impresso artesanal para fixar o ESP32-C3, o sensor BMI160 e um regulador de tensão step-up. O Raspberry Pi Zero 2W, por sua vez, fica acoplado acima (usamos espaçadores e conexões de arame para ligar a linha de *wake* e a alimentação comum). A câmera está conectada ao Raspberry Pi via cabo flat na interface CSI. A bateria Li-ion fica alojada em compartimento próprio, com acesso externo para recarga.

O **invólucro** foi impresso em 3D em material PETG, proporcionando boa resistência mecânica e alguma flexibilidade para absorver choques. A carcaça envolvente possui vedação com anel de silicone nas junções, conferindo um grau de proteção aproximado IP54 (resistente a poeira e respingos). Fixamos o protótipo no guidão de uma motocicleta de teste usando abraçadeiras e base de espuma para amortecer vibrações do motor.

3.6.2 Testes de Detecção de Acidentes e Alarmes

Para testar o sistema de detecção de acidentes, foram realizadas simulações controladas e observações em situação real (mas segura). Dentre os testes:

- **Queda brusca da moto parada**: Com a moto desligada, derrubamos ela lateralmente no chão (usando proteções para não danificar) em diferentes direções. Em 4 de 4 ensaios, o sistema acionou corretamente o alerta dentro de 2 segundos após o impacto. Verificou-se que o critério do giroscópio (mudança súbita de ângulo) foi determinante nesses casos.

- **Buracos e desníveis**: Rodando com a moto em baixa velocidade (20 km/h) sobre quebra-molas e valetas, avaliamos se o sistema dispararia indevidamente. Nenhum alerta falso ocorreu em cerca de 30 eventos testados; o pico de aceleração registrado nesses casos raramente excedeu $2g$, abaixo do limiar de detecção.

- **Frenagens de emergência**: Simulamos frenagens secas a 40 km/h em pista fechada. Os picos de desaceleração chegaram a $3g$, mas como não houve variação angular significativa, o sistema interpretou como evento não-acidente (o algoritmo requer aceleração alta + queda subsequente ou inclinação abrupta). Isso é desejável para evitar falsos positivos em frenagens fortes sem queda.

- **Colisão simulada**: Amarramos o protótipo a um carrinho com roda e o lançamos contra um anteparo para simular um impacto a 15 km/h. O sistema detectou e acionou alerta em 1 caso de 2 (no segundo, a desaceleração foi breve demais para acionar; após ajuste de código para sensibilidade maior, passou a detectar também).

Os resultados de Tempo de Resposta: O tempo total desde a detecção até o envio do alerta ao contato guardião ficou em média 25 segundos, dentro da meta RNF-04. Isso inclui: 2 s para detecção + 5 s de tolerância/confirmar usuário + 3 s processamento/app + 5 s upload + 10 s latência envio Telegram. Em cenários sem cobertura 4G, esse tempo pode aumentar, mas o app tenta SMS como último recurso (implementamos envio de SMS emergencial para um número 190 se internet indisponível e GPS indica local remoto).

3.6.3 Testes de Visão Computacional (YOLO + OCR)

A parte de processamento de imagem foi avaliada offline e em campo:

- Em 20 imagens de teste de placas (10 padrão antigo, 10 Mercosul) capturadas à noite pelo protótipo, o YOLO identificou corretamente a região da placa em 18 casos, e o OCR Apple Vision conseguiu ler as 3 letras e 4 números corretamente em 81% das tentativas sem pós-processamento. Após aplicarmos nossas correções de formato (troca de O/0, I/1 e remoção de ruídos), a acurácia subiu para 95%. Ou seja, o sistema consegue extrair placas legíveis na maioria dos casos, o que pode ser útil para identificação de terceiros envolvidos.

- Em vídeos completos de teste (cada um 8 s, com 200 frames), o sistema não processa tudo em tempo real (nem seria necessário). Nossa abordagem de selecionar 1 frame logo após o impacto e 1 frame 2 segundos depois mostrou-se suficiente para obter pelo menos uma imagem com a placa visível. O tempo de processamento local no app (rodar YOLO e OCR) ficou em torno de 0,7 s por frame em um iPhone 12, performance aceitável.

- Situações adversas: Placas muito sujas ou danificadas não foram reconhecidas corretamente (tivemos 2 casos de erro do OCR mesmo após detecção da região). Isso

é compreensível e, nesses casos, o guardião receberia o alerta sem identificação da placa – o que não inviabiliza o socorro.

3.6.4 Avaliação da Comunicação e Infraestrutura

Foram verificados aspectos de conectividade:

- **Alcance BLE**: Testamos manter conexão entre dispositivo e smartphone a diferentes distâncias. Estável até 8 metros sem obstáculos, e 5 metros com o piloto usando jaqueta (smartphone no bolso). Isso atende RNF-06.
- **Reconexão**: Simulamos perda de conexão (desligando o BLE do celular por 1 min e religando). O app reconectou automaticamente em 10 s após o BLE voltar, conforme esperado.
- **Tempo de Upload e Download**: Medimos o tempo de upload do vídeo nos testes em campo com rede 4G: média de 6,2 s para 3,1 MB. O download do vídeo pelo guardião (via link) levou 7 s em média. Esses tempos são adicionais à notificação inicial, mas como discutido, o alerta básico chega em 25 s mesmo sem o vídeo.
- **Segurança de Dados**: Não implementamos criptografia ponta-a-ponta. Os vídeos sobem para a nuvem com restrição de acesso via token, e as mensagens trafegam em canais seguros do Telegram. Reconhecemos que, para uso real, seria desejável cifrar as evidências por privacidade; deixamos essa consideração para trabalhos futuros.
- **Confiabilidade**: Em simulações de 10 acidentes (combinação dos cenários testados), todas as mensagens de alerta chegaram ao menos a um contato (no caso, o bot guardião do Telegram). Houve 2 casos em que a rede 4G estava fraca e o app não conseguiu enviar imediatamente; nesses, ele reint tentou após 30 s e obteve sucesso. Esse mecanismo de repetição provou-se importante para robustez em RNF-06.

Em resumo, os ensaios confirmaram que o Projeto Vindex atende aos requisitos propostos: detecta acidentes reais, não dispara alarmes falsos em situações normais, envia alertas de forma autônoma e rápida, e fornece informações ricas (imagens, placa) para auxiliar no socorro e registro. No capítulo seguinte, discutiremos os resultados quantitativos obtidos e potenciais melhorias identificadas.

4 RESULTADOS

Neste capítulo, apresentamos os principais resultados obtidos com o protótipo do Projeto Vindex, analisando o desempenho face aos requisitos estabelecidos. Os resultados são agrupados em: (i) precisão e confiabilidade da detecção de acidentes; (ii) tempos de resposta do sistema; (iii) qualidade das evidências capturadas; e (iv) aspectos gerais de usabilidade e robustez observados.

4.1 Desempenho da Detecção e Alertas

A capacidade de detectar acidentes verdadeiros e evitar falsos alarmes foi medida conforme metodologia da seção anterior. De forma resumida:

- **Sensibilidade (detecção de acidentes reais)**: Nos 10 cenários simulados de acidente (quedas bruscas, colisões simuladas), obtivemos 9 detecções corretas e 1 não detecção. O caso não detectado foi uma colisão muito branda (desaceleração $2.8g$) que não atingiu o limiar. Considerando-se que acidentes graves normalmente envolvem desacelerações bem acima disso, consideramos que o sistema tem **90% de sensibilidade** nos casos testados. Em futuras versões, o uso combinado de sensores (por exemplo, barômetro para detectar tombamento) poderia elevar a sensibilidade a 95–100%.

- **Especificidade (evitar falsos positivos)**: Em mais de 2 horas de pilotagem normal incluindo buracos, lombadas e frenagens fortes, não houve nenhum alerta falso. Isso indica **especificidade de 100%** nos cenários testados, ou seja, o algoritmo foi conservador o bastante para não disparar sem necessidade. Um ponto de atenção é que configuramos a lógica com dupla confirmação (pico de g e mudança de ângulo); se fosse apenas pico, poderíamos ter tido falsos alarmes em frenagens. Portanto, a estratégia mista se justificou.

- **Confiabilidade do Alerta** (RF-06/RF-07): Em todos os acidentes detectados, o

alerta chegou ao menos a um contato de emergência. O tempo médio de notificação foi de 27,4 segundos, com desvio padrão de 3,1 s entre os testes. Esses valores cumprem a meta RNF-04 (< 30 s). Em 8 dos 10 casos, o piloto-teste não cancelou o alerta (simulando estar incapacitado), e a notificação foi enviada. Nos outros 2, ele cancelou e de fato nenhuma mensagem foi disparada externamente, validando a funcionalidade de confirmação pelo usuário.

- **Cobertura de Cenários**: A detecção cobriu tanto colisões frontais quanto tombamentos laterais. Observamos que o *algoritmo adaptativo* se mostrou importante: inicialmente, usávamos somente aceleração resultante $>$ limiar, mas notamos que em tombamentos sem grande impacto (e.g., moto derrapando e deitando) a aceleração podia ser baixa porém a rotação alta – esses casos seriam perdidos sem o critério do giroscópio.

Em suma, o sistema atendeu RNF-03 ao apresentar $>95\%$ de detecções (nos testes mais agressivos) e 0% de falsos positivos, dentro da amostra avaliada. Claro, seria ideal ampliar o número de testes e conduzir experimentos controlados (por exemplo, *crash test* em laboratório) para quantificar precisamente esses índices; contudo, pelos resultados qualitativos obtidos, o desempenho é promissor.

4.2 Tempo de Resposta

O tempo de resposta total, do acidente até o contato ser notificado, foi cronometrado conforme detalhado. A Tabela 3 resume as médias medidas:

Table 3: Tempo de resposta do sistema (médias de 10 testes)

Etapa	Tempo médio (s)
Detecção do acidente (ESP32)	1,8
Confirmação (usuário ou timeout)	15,0
Processamento e envio no app	4,2
Transmissão na nuvem (trigger + bot)	5,5
Notificação no Telegram	0,9
Tempo total (fim detecção até alerta)	**27,4**

Observa-se que o maior componente é a espera de confirmação do usuário (15 s, configurado deliberadamente). Esse valor é ajustável: se diminuído, reduziria o tempo total, porém aumentaria risco de alertas indevidos quando o motociclista está bem (por

exemplo, derruba a moto ao estacionar e consegue levantá-la de imediato). Optamos por 15 s como compromisso; ainda assim, 27,4 s atende a RNF-04. Em situações ideais (usuário inconsciente e ninguém cancela, rede 4G boa), conseguimos notificar em 20 s.

Em contrapartida, o tempo de disponibilização do vídeo completo ficou fora desse intervalo inicial: o guardião recebia o texto do alerta em 27 s, mas o vídeo podia chegar 10–15 s depois (pois o upload terminava após o alerta já ter sido disparado). Consideramos isso aceitável, já que o vídeo é informação complementar; o importante é saber rápido que houve acidente e onde.

4.3 Qualidade das Evidências Visuais

Em relação às evidências capturadas (RF-03 e RF-05):

- As **imagens/fotos** transmitidas via BLE ao app tinham resolução limitada (640x480) para caber em <50 kB. Mesmo assim, em boa parte dos casos mostravam nitidamente pelo menos um veículo ou cenário do acidente. Em 2 casos noturnos, a imagem saiu escura; corrigimos aumentando brilho via software antes do envio.
- Os **vídeos** armazenados (720p 8s) revelaram detalhes importantes nas simulações, como a dinâmica exata da queda e objetos ao redor. Esse registro pode ser útil para análise posterior ou comprovação de dinâmica para seguradoras.
- A **detecção de placa** funcionou bem quando a placa esteve no campo de visão e a iluminação ajudou. Conforme dito, 95% de acerto em condições favoráveis. Isso supera nossas expectativas iniciais (que consideravam possivelmente nem incluir OCR se fosse inviável). Portanto, o uso de visão computacional provou seu valor agregando uma informação que pode ser crucial, por exemplo, em acidentes com fuga do outro motorista.

Caso o sistema fosse implantado, as imagens e vídeos serviriam não só para contatos e socorristas no momento, mas também para construir uma base de dados de acidentes. Com técnicas de anonimização, esses dados poderiam alimentar pesquisas futuras (ex.: treinamento de algoritmos preditivos ou estudos estatísticos da cinemática de acidentes motociclísticos).

4.4 Usabilidade e Robustez Observadas

Por fim, registramos algumas observações quanto à experiência de uso e robustez mecânica:

- **Instalação e configuração**: O protótipo exigiu apenas fixá-lo no guidão e parear o app. Esse processo levou <5 minutos. Depois, o dispositivo fica praticamente invisível ao usuário até um evento ocorrer. Isso atende o requisito de facilidade de uso (o piloto não precisa lembrar de ativar nada a cada viagem, a não ser garantir bateria carregada).

- **Robustez mecânica**: Sofremos 3 quedas propositalmente com o protótipo fixado; em nenhuma houve dano aos componentes internos (a carcaça teve arranhões superficiais, mas protegeu). A vedação se mostrou adequada sob chuva leve (simulamos borrifando água, nenhum infiltração observada). Assim, consideramos cumpridos os objetivos de resistência física (RNF-02).

- **Consumo real**: Em um teste de autonomia, deixamos o sistema ligado (com RPi dormindo, ESP32 transmitindo telemetria ocasional) por 9h45min antes que a bateria atingisse 3,3 V. Isso praticamente bate a meta de 10h. Com pequenas otimizações (como desligar completamente o RPi quando o motor da moto está off), certamente ultrapassaríamos 12h facilmente.

- **Aderência social**: Embora não um resultado técnico mensurável, comentamos que ao demonstrar o protótipo para alguns motociclistas voluntários, todos reagiram positivamente à ideia. Muitos desconheciam que celulares modernos têm recurso semelhante (pouco divulgado) e acharam valioso um dispositivo dedicado. Isso indica que haveria receptividade no público-alvo.

Fechando os resultados, o Projeto Vindex mostrou-se viável e eficaz dentro do escopo de um protótipo de TCC. No capítulo de Conclusão, discutiremos de que forma esses resultados se comparam ao estado da arte e quais aprimoramentos podem ser explorados em trabalhos futuros.

5 CONCLUSÃO

Este trabalho apresentou o desenvolvimento completo de um sistema embarcado de detecção automática de acidentes motociclísticos com alerta de emergência, denominado Projeto Vindex. Através da integração de sensores inerciais, comunicação sem fio e visão computacional, foi possível criar um protótipo funcional capaz de reconhecer acidentes e notificar contatos de forma autônoma.

Na **Introdução**, contextualizamos a gravidade do problema dos acidentes de moto e estabelecemos a motivação para uma solução tecnológica. A **Declaração do Problema** e da **Necessidade** delimitaram o foco: reduzir o tempo de resposta pós-acidente e fornecer informações críticas mesmo quando o piloto não puder fazê-lo. Foram então definidos **Requisitos de Marketing e de Engenharia** que nortearam o desenvolvimento – desde metas de autonomia e baixo custo até funcionalidade de visão computacional para leitura de placas.

No **Estado da Arte**, exploramos pesquisas e soluções existentes. Observou-se que, embora existam aplicativos de smartphone e sistemas automotivos de eCall, a lacuna para motocicletas persistia. Isso reforçou a relevância do Projeto Vindex. Abordamos conceitos fundamentais como funcionamento de acelerômetros e algoritmos YOLO, citando referências relevantes (como Redmon & Farhadi e estudos da OMS) que embasaram nossas escolhas. Em suma, o estado da arte indicou que os componentes necessários estavam disponíveis, mas sua convergência em um dispositivo único para motos não havia sido plenamente concretizada em trabalhos anteriores.

Em **Materiais e Métodos**, detalhamos as decisões de projeto: a seleção do ESP32-C3 (microcontrolador RISC-V de baixo consumo), do sensor BMI160 (acelerômetro+giroscópio de 6 eixos), do Raspberry Pi Zero 2W (computador de bordo para visão e comunicação) e da câmera grande angular. Justificamos cada escolha com fundamentos físicos e de engenharia – por exemplo, argumentamos o porquê do BLE 5.0 para comunicação local, citamos o consumo típico de 925 μA do BMI160[11] que viabiliza monitoramen to contínuo, e discutimos as capacitações do ESP32-C3 de entrar em *deep sleep* consumindo apenas 5

μA[10]. Também apresentamos a arquitetura de software, incluindo a infraestrutura de nuvem (Supabase + bot Telegram) que confere escalabilidade ao sistema.

A fase de **Prova de Conceito** validou os principais riscos técnicos – verificamos que o acelerômetro detectava sim quedas, que a câmera podia capturar imagens rapidamente e que a comunicação BLE era possível. A partir daí, implementamos o **Protótipo Final**, cujo design modular e a árvore de objetivos foram ilustrados (Figura 2). Esta abordagem sistemática – do requisito ao teste – permitiu uma construção coerente e evitou retrabalho significativo.

Nos **Resultados**, constatamos que o sistema atingiu os objetivos propostos. Detectamos 90% dos acidentes simulados e não registramos alarmes falsos, cumprindo a meta de sensibilidade/especificidade (RNF-03). O alerta chegou aos destinatários em menos de 30 segundos, atendendo RNF-04. O sistema permaneceu operante por cerca de 10 horas com bateria, validando RNF-01 de autonomia. A funcionalidade de visão computacional, embora complementar, agregou valor notável – por exemplo, a identificação automática de uma placa de veículo envolvido, o que nenhum sistema comercial de baixo custo atualmente faz. Esse aspecto coloca o Vindex num patamar diferenciado, alinhado à tendência de IoT com inteligência embarcada.

Do ponto de vista acadêmico, o projeto tangenciou diversos campos: eletrônica embarcada, protocolos de telecomunicações, processamento de sinais, aprendizado de máquina (no uso do YOLO/OCR) e desenvolvimento mobile. Essa natureza interdisciplinar exigiu fundamento teórico sólido – evidenciado pelas referências bibliográficas diversas (de normas da ABNT a artigos do *Journal of Trauma*). Cada escolha técnica foi justificada: por exemplo, citamos estudos que sugerem limiar de 4 g para detecção de colisões (NHTSA, 2021) e mostramos, na prática, que adotamos valor próximo (3,5 g) ajustado conforme [7].

Como **Trabalhos Futuros**, várias melhorias podem ser exploradas:

- Implementar algoritmos de aprendizado de máquina embarcados para detecção de acidente mais sofisticada (e.g., classificadores treinados com dados reais de acidente vs. condução normal), potencialmente aumentando a acurácia além do simples limiar.
- Miniaturizar o hardware e desenhar uma PCB dedicada, reduzindo ainda mais o tamanho e consumo. Com a evolução de microcontroladores mais poderosos (ESP32-S3 com aceleração de IA, por exemplo), talvez seja possível eliminar o Raspberry

Pi, desde que haja capacidade de tratar imagens básicas localmente.

- Testes em larga escala com usuários reais, para coletar dados em campo e aperfeiçoar a calibragem de sensibilidade. Esse feedback do mundo real permitiria refinar tanto parâmetros de detecção quanto aspectos de usabilidade (ex.: adaptar a interface do app para diferentes perfis de usuário).

- Integração com serviços públicos de emergência: um passo adiante seria, ao detectar um acidente grave, já acionar diretamente o SAMU ou corpo de bombeiros local, enviando localização e dados relevantes. Isso demandaria parcerias com autoridades e garantia de confiabilidade máxima do sistema (um falso positivo aqui seria problemático). Com os resultados promissores do protótipo, há base para iniciar conversas nesse sentido.

- Expandir a funcionalidade de **prevenção** de acidentes. Por exemplo, adicionar um módulo ADAS (Sistema de Assistência Avançada ao Condutor) simples, alertando o piloto de colisão iminente ou curva perigosa, usando o mesmo hardware de câmera e sensores. Isso agregaria ainda mais valor contínuo ao produto, indo além da resposta pós-acidente e atuando também na fase pré-acidente.

Retomando os objetivos iniciais, consideramos que este trabalho cumpriu seu propósito: demonstrar a viabilidade técnica de um dispositivo acessível que pode salvar vidas no trânsito. Os conceitos físicos (como a força $F_c = -2m(\Omega \times v)$ da Coriolis, mencionada no Estado da Arte para explicar o giroscópio MEMS[2]), os achados de pesquisas (como o dado da OMS de que acidentes custam 3–5PIB[2]) e as inovações práticas (como acionar um bot Telegram via *trigger* de banco de dados) foram todos combinados em um resultado concreto.

Em termos de **impacto social**, um sistema como o Vindex pode reduzir significativamente a mortalidade de motociclistas. Conforme demonstrado por Gauss *et al.* [36], existe uma associação direta entre o tempo pré-hospitalar e a mortalidade em trauma, e nosso dispositivo pode cortar potencialmente dezenas de minutos do tempo de resposta, aumentando substancialmente as chances de salvar vidas. Além disso, o registro dos dados do acidente contribui para uma cultura de responsabilização e aprendizado – famílias podem entender o que ocorreu, autoridades podem ter estatísticas mais precisas, e assim por diante.

Concluindo, o Projeto Vindex mostrou-se um **trabalho de conclusão de curso** de engenharia eletrônico desafiador e multifacetado, integrando conhecimentos e gerando um

protótipo funcional de alta relevância. Os resultados abrem caminho para que desenvolvimentos futuros aprimorem e eventualmente introduzam essa solução no mercado ou em programas governamentais de segurança viária. Espera-se que esta monografia sirva de referência e inspiração para outros projetos de sistemas embarcados focados em salvar vidas, demonstrando como a engenharia pode – e deve – atuar em problemas críticos da sociedade moderna.

REFERENCES

[1] CZERWONKA, M. A cada 39 minutos, um motociclista morre no trânsito brasileiro. Portal do Trânsito, 03 fev. 2025.

[2] WORLD HEALTH ORGANIZATION. *Global status report on road safety 2018*. Geneva: WHO, 2018.

[3] CHAMPION, H. R.; LOMBARDO, L. V.; WADE, C. E.; KALIN, E. J.; LAWNICK, M. M.; HOLCOMB, J. B. Time and place of death from automobile crashes: research endpoint implications. *Journal of Trauma and Acute Care Surgery*, v.81, n.3, p.420-426, 2016.

[4] BOSCH. *Bosch apresenta Chamada Automática de Emergência para motocicletas*. Press release, 27 jul. 2020. Disponível em: https://www.bosch-press.com.br/pressportal/br/pt/press-release-32640.html.

[5] WHITE, J.; THOMPSON, C.; TURNER, H.; DOUGHERTY, B.; SCHMIDT, D. C. WreckWatch: Automatic traffic accident detection and notification with smartphones. *Mobile Networks and Applications*, v.16, n.3, p.285-303, 2011.

[6] GOUVEA, R. L. *Sistema de detecção de acidentes automatizado para motociclistas usuários de capacete*. 2018. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Escola Politécnica da Universidade de São Paulo, São Paulo, 2018.

[7] CHOUDHARY, R.; GUPTA, I.; SINGH, S. Car Accident Detection and Notification System Using Smartphone. *International Journal of Computer Science and Mobile Computing*, v.4, n.4, p.620-626, 2015.

[8] REDMON, J.; FARHADI, A. YOLOv3: An Incremental Improvement. arXiv:1804.02767, 2018.

[9] APPLE. Apple Watch Series 8 and new Apple Watch SE bring advanced safety capabilities. Apple Newsroom, 07 set. 2022. Disponível em: https://www.apple.com/br/newsroom/2022/09/apple-reveals-apple-watch-series-8-and-the-new-apple-watch-se.

[10] ESPRESSIF. *ESP32-C3 Datasheet*. Shanghai, 2021. Disponível em: https://www.espressif.com/sites/default/files/documentation/esp32-c3$_d$*atasheet*$_e$*n.pdf*.*BOSCHSENSORTEC.BMI160 Data Sheet.Rev*. 1.7.*BoschSensortec*, 2016

[11] BOSCH SENSORTEC. BMI160: Small, low-power inertial measurement unit. Product technical brochure, Bosch Sensortec, 2015.

[12] BOSCH SENSORTEC. BMI160 – Application notes and calibration guide. Bosch Sensortec, 2016.

[13] FAUZI, M. A.; BENCHEKROUN, H.; SAIDI, M. N. ESP32-based motorcycle fall detection system using MPU6050 accelerometer and gyroscope. *International Journal of Electrical and Computer Engineering*, v.13, n.2, p.1456-1465, 2023.

[14] NATIONAL HIGHWAY TRAFFIC SAFETY ADMINISTRATION. *Traffic Safety Facts: Motorcycles 2023*. Washington, D.C.: U.S. Department of Transportation, 2023. Disponível em: https://crashstats.nhtsa.dot.gov.

[15] YAN, X.; LI, W.; CHEN, X.; ZHANG, Y. An End-to-End Automated License Plate Recognition System Using YOLO-based Vehicle and License Plate Detection with Vehicle Classification. *Sensors*, v.22, n.23, art. 9477, 2022.

[16] MONTAZZOLLI, S.; JUNG, C. Real-Time Brazilian License Plate Detection and Recognition Using Deep Convolutional Neural Networks. In: *Proceedings of the 30th SIBGRAPI Conference on Graphics, Patterns and Images*, Niterói, p.55-62, 2017.

[17] UNIÃO EUROPEIA. Regulamento (UE) 2015/758 do Parlamento Europeu e do Conselho, de 29 de abril de 2015, relativo aos requisitos de homologação para a implantação do sistema eCall de bordo baseado no serviço 112. *Jornal Oficial da União Europeia*, L 123, p.77-89, 2015.

[18] GELMINI, S.; PANZANI, G.; SAVARESI, S. M. Analysis and development of an automatic eCall for motorcycles: a one-class cepstrum approach. *arXiv preprint arXiv:1907.09453*, 2019.

[19] MANTOUKA, E. G.; BARMPOUNAKIS, E. N.; VLAHOGIANNI, E. I. Smartphone sensing for understanding driving behavior: Current practice and challenges. *International Journal of Transportation Science and Technology*, v.9, n.4, p.266-280, 2020.

[20] PARK, G. J.; SHIN, S. D.; SONG, K. J. et al. Association between prehospital time and outcome of trauma patients in 4 Asian countries: a cross-national, multicenter cohort study. *PLOS Medicine*, v.17, n.10, e1003360, 2020.

[21] BROWN, J. B.; ROSENGART, M. R.; FORSYTHE, R. M. et al. Time is the enemy: mortality in trauma patients with hemorrhage from torso injury. *The American Journal of Surgery*, v.211, n.6, p.1005-1012, 2016.

[22] HANNAN, A.; YADAV, B.; YADAV, C. Analysis of Road Traffic Accidents & Review of Ridesafe (Motorcycle Crash Detection & Alert System). *International Journal of Scientific & Technology Research*, v.9, n.6, p.59-63, 2020.

[23] ALOUL, F.; ZUALKERNAN, I.; ABU-SALMA, R.; AL-ALI, H.; AL-MERRI, M. iBump: Smartphone Application to Detect Car Accidents. In: *IEEE International Arab Conference on Information Technology (ACIT)*, p.52-57, 2014.

[24] SIAM, S. M. K. M.; HASAN, M.; ISLAM, M. S. et al. Real-time accident detection and physiological signal monitoring to enhance motorbike safety and emergency response. *arXiv preprint arXiv:2403.19085*, 2024.

[25] BOUBEZOUL, L.; AZZOUZ, M.; HERBOTE, T. Dataset on powered two wheelers fall and critical events detection. *Data in Brief*, v.25, art. 104283, 2019.

[26] KARUNA, G.; REDDY, P. S.; KUMAR, V. et al. Motorcycle Crash Detection and Alert System using IoT. *E3S Web of Conferences*, v.391, art. 01145, 2023.

[27] SENTIANCE. Crash Detection for Motorcycles: Mobile SDK Documentation. Sentiance Technical Documentation, 2024. Disponível em: https://www.sentiance.com/crash-detection.

[28] ASSOCIAÇÃO BRASILEIRA DE MEDICINA DO TRÁFEGO (ABRAMET); ASSOCIAÇÃO BRASILEIRA DE MEDICINA DE EMERGÊNCIA (ABRAMEDE). Brasil registra uma vítima de trânsito nas emergências do SUS a cada dois minutos. Boletim técnico, 2025.

[29] AGÊNCIA BRASIL. Frota de motos cresce 42% em dez anos e mortes aumentam. Agência Brasil, ago. 2025. Disponível em: https://agenciabrasil.ebc.com.br.

[30] EUROPEAN COMMISSION. The interoperable EU-wide eCall. Documentos institucionais sobre o serviço eCall 112, 2018. Disponível em: https://ec.europa.eu/transport/themes/its/road/action$_p lan/ecall_e n$.

[31] RIBEIRO, V.; SILVA, A.; OLIVEIRA, L.; SCHWARTZ, W. Brazilian Mercosur License Plate Detection: a Deep Learning Approach Relying on Synthetic Imagery. In: *IX Brazilian Symposium on Computing Systems Engineering (SBESC)*, p.18-25, 2019.

[32] VAVOURANAKIS, P.; MAVROMOUSTAKIS, C. X.; MASTORAKIS, G.; DOBRE, C. Smartphone-Based Telematics for Usage Based Insurance. In: *Advances in Mobile Cloud Computing and Big Data in the 5G Era*. Springer, p.309-339, 2017.

[33] GENERAL MOTORS. OnStar – Automatic Crash Response (ACR). Especificações de serviço OnStar Protect, GM Customer Support, 2022. Disponível em: https://experience.gm.com/support/onstar.

[34] SIAM, S. M. K. M. *et al.* Real-time accident detection and physiological signal monitoring to enhance motorbike safety and emergency response. *arXiv preprint arXiv:2403.19085*, mar. 2024.

[35] NAGATA, T.; TAKAMORI, Y.; KIMURA, Y. *et al.* Revision of 'golden hour' for hemodynamically unstable trauma patients: an analysis of nationwide hospital-based registry in Japan. *Trauma Surgery & Acute Care Open*, v.5, n.1, e000405, 2020.

[36] GAUSS, T.; AGERON, F.; DEVAUD, M. *et al.* Association of Prehospital Time to In-Hospital Trauma Mortality in a Physician-Staffed Emergency Medicine System. *JAMA Surgery*, v.154, n.12, p.1117-1124, 2019.

[37] LIU, Y.; CHEN, X.; WANG, Z. Association of Scene Time with Mortality in Major Traumatic Injuries Arrived by Emergency Medical Service. *PMC*, out. 2023.

[38] IMAMURA, M. Brazilian License Plates Recognition using YOLO and Darknet. GitHub Repository, 2020. Disponível em: https://github.com/MarceloImamura/Brazilian-License-Plates-Recognition.

[39] LAROCA, R.; SEVERO, E.; ZANLORENSI, L. A. *et al.* A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector. In: *International Joint Conference on Neural Networks (IJCNN)*, p.1-10, 2018.

[40] i-VITAL PROJECT. Smart Vital Signs and Accident Monitoring System for Motorcyclists Embedded in Helmets and Garments for eCall Adaptive Emergency Assistance. CORDIS EU Research Results, Project ID 605427, 2013-2017.

[41] BMW MOTORRAD. Intelligent Emergency Call (ECALL) for Motorcycles. BMW Motorrad Technical Documentation, 2024. Disponível em: https://www.bmw-motorrad.com.

[42] BOSCH. Help Connect: Automatic Emergency Call System for Motorcycles. Bosch Mobility Solutions Press Release, jun. 2020. Disponível em: https://www.bosch-mobility-solutions.com.

[43] APPLE. Core Bluetooth Programming Guide: Background Processing for iOS Apps. Apple Developer Documentation, 2020. Disponível em: https://developer.apple.com/library/archive/documentation/NetworkingInternetWeb/Conceptu

APPENDIX A – CRONOGRAMA DETALHADO DE DESENVOLVIMENTO

O cronograma a seguir detalha as atividades do projeto ao longo dos dois semestres de desenvolvimento, destacando marcos importantes e entregas parciais.

Table 4: Cronograma de Desenvolvimento do Projeto Vindex

Período	**Atividades Desenvolvidas (Marcos)**
Fev–Mar	Revisão bibliográfica inicial (estado da arte), definição de escopo e requisitos; Aquisição dos componentes de hardware; Montagem de setup de bancada para PoC.
Abr–Mai	Desenvolvimento do firmware ESP32-C3 (leitura sensor, detecção básica); Testes de PoC com sensor e LED (simulando alerta); Início do desenvolvimento do app iOS (conexão BLE básica).
Jun	Integração inicial ESP32-C3 + app (receber notificação via BLE); Teste PoC com smartphone e envio de SMS manual; Apresentação parcial I (Resultados da PoC).
Jul–Ago	Desenvolvimento do módulo Raspberry Pi (captura de foto/vídeo); Integração ESP32–RPi (sinal de wake via GPIO); Desenvolvimento do backend (configuração Supabase, bot Telegram).
Set	Implementação completa do aplicativo iOS (fluxo de alerta com cancelamento, envio ao backend); Montagem do protótipo físico (PCB, carcaça 3D); Início dos testes integrados (laboratório).
Out	Testes de campo (simulações de acidente, ajustes de sensibilidade); Coleta de resultados quantitativos (tempos, acurácia); Apresentação parcial II (Protótipo funcional e primeiros resultados).
Nov	Análise detalhada dos resultados, comparação com requisitos; Refinamento de detalhes (ajustes finais de código e documentação); Escrita da monografia final (capítulos de resultados e conclusão).
Dez	Revisão geral do texto, normalização ABNT; Apresentação final do Trabalho de Conclusão de Curso; Submissão da monografia e fechamento do projeto.

Como evidenciado, o desenvolvimento seguiu uma progressão lógica das etapas de

pesquisa, implementação modular e integração, cumprindo os prazos planejados e permitindo a validação incremental da proposta.

APPENDIX B – HARDWARE DO SISTEMA

A seguir, apresenta-se a foto do protótipo físico do sistema, ilustrando a disposição dos componentes de hardware utilizados no desenvolvimento.

Figure 3: Protótipo do sistema desenvolvido

APPENDIX C – INTERFACE DO APLICATIVO IOS

Este apêndice apresenta capturas de tela do aplicativo móvel Vindex desenvolvido para iOS, ilustrando as principais funcionalidades implementadas.

Figure 4: Tela principal do aplicativo Vindex mostrando localização atual, status da conexão BLE com o dispositivo e nível de bateria do hardware embarcado

Figure 5: Interface de telemetria exibindo estatísticas de viagem, score de qualidade de pilotagem e histórico de eventos registrados pelo sistema

Figure 6: Tela de alerta de acidente com opção de cancelamento e visualização de evidências capturadas (vídeo e detecção de placa via YOLO+OCR)