INSTITUTO FEDERAL DE ALAGOAS
CAMPUS ARAPIRACA
CURSO SUPERIOR DE SISTEMAS DE INFORMAÇÃO

GUILHERME BARBOSA PEREIRA

DESENVOLVIMENTO DE UM DISPOSITIVO VESTÍVEL DE ASSISTÊNCIA À LOCOMOÇÃO UTILIZANDO VISÃO COMPUTACIONAL E INTERNET DAS COISAS

ARAPIRACA, AL
2026

GUILHERME BARBOSA PEREIRA

DESENVOLVIMENTO DE UM DISPOSITIVO VESTÍVEL DE ASSISTÊNCIA À LOCOMOÇÃO UTILIZANDO VISÃO COMPUTACIONAL E INTERNET DAS COISAS

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Sistemas de Informação do Instituto Federal de Alagoas, campus Arapiraca, como requisito parcial para obtenção de grau de Bacharel em Sistemas de Informação.

Orientadora: Profa. Dra. Cledja Karina Rolim da Silva

ARAPIRACA, AL.
2026

Dados Internacionais de Catalogação na Publicação
Instituto Federal de Alagoas
***Campus* Arapiraca**

004.167
P436d Pereira, Guilherme Barbosa.

Desenvolvimento de um dispositivo vestível de assistência à locomoção utilizando visão computacional e internet das coisas / Guilherme Barbosa Pereira. – Dados eletrônicos (1 arquivo: 47.705 KB). – 2026.

Sistema requerido: Adobe Acrobat Reader.
Modo de acesso: Internet.
Orientação: Profa. Dra. Cledja Karina Rolim da Silva.

Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Instituto Federal de Alagoas, Campus Arapiraca, Arapiraca, 2026.

1. Tecnologia assistiva. 2. Internet das coisas (*IoT*). 3. Visão computacional. II. Título.

Robson B. de Souza | Bibliotecário – CRB-4/AL - 1624

GUILHERME BARBOSA PEREIRA

DESENVOLVIMENTO DE UM DISPOSITIVO VESTÍVEL DE ASSISTÊNCIA À LOCOMOÇÃO UTILIZANDO VISÃO COMPUTACIONAL E INTERNET DAS COISAS

Trabalho de Conclusão de Curso apresentado ao Curso Superior de Sistemas de Informação do Instituto Federal de Alagoas, campus Arapiraca, como requisito parcial para obtenção de grau de Bacharel em Sistemas de Informação.

Orientadora: Profa. Dra. Cledja Karina Rolim da Silva

Data de Aprovação: 27/02/2026.

BANCA EXAMINADORA

gov.br Documento assinado digitalmente
CLEDJA KARINA ROLIM DA SILVA
Data: 16/03/2026 16:42:42-0300
Verifique em https://validar.iti.gov.br

Profª. Drª. Cledja Karina Rolim da Silva (Orientadora)
Instituto Federal de Alagoas - IFAL

gov.br Documento assinado digitalmente
TARSIS MARINHO DE SOUZA
Data: 16/03/2026 16:47:41-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Társis Marinho de Souza
Instituto Federal de Alagoas - IFAL

gov.br Documento assinado digitalmente
FERNANDO ANTONIO GUIMARAES TENORIO
Data: 16/03/2026 18:31:59-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Fernando Tenório
Instituto Federal de Alagoas - IFAL

AGRADECIMENTOS

Começo agradecendo a Deus, em quem sempre encontrei refúgio e força para persistir e alcançar mais esta vitória na minha vida acadêmica.

Minha eterna gratidão à minha família, que sempre esteve ao meu lado. O amor e o suporte de vocês foram o combustível para que eu chegasse até aqui. Vocês são minha base e tudo o que conquisto é também de vocês.

Ao IFAL, minha gratidão por ter sido o cenário do meu desenvolvimento. Durante todos esses anos, do ensino médio à graduação, a instituição me acolheu de braços abertos, tornando-se verdadeiramente uma extensão do meu lar.

Aos professores que cruzaram meu caminho, muito obrigado. Em especial, destaco a professora Cledja Rolim e o professor Tarsis Marinho. Vocês não foram apenas educadores, mas mentores que me acompanharam desde o início, moldando o profissional e o ser humano que sou hoje.

Aos meus grandes amigos de curso, Eduardo, Victor e Samila: obrigado por dividirem comigo os momentos de tensão e as alegrias dessa fase. As experiências trocadas e as risadas intermináveis tornaram o fardo leve e a jornada inesquecível.

E à minha companheira, Maria Eugênia, meu agradecimento especial por todo o apoio nos últimos anos. Obrigado por acreditar em mim e estar ao meu lado; esta conquista também é nossa.

RESUMO

A mobilidade urbana autônoma é um aspecto essencial para a qualidade de vida de pessoas com deficiência visual, que frequentemente enfrentam barreiras físicas e arquitetônicas agravadas pela falta de acessibilidade nas cidades. Ferramentas tradicionais, como a bengala branca, embora fundamentais, possuem limitações na detecção de obstáculos aéreos e na identificação semântica do ambiente. Diante desse cenário, este trabalho propõe o desenvolvimento de um sistema vestível de assistência visual baseado em Internet das Coisas (IoT) e Inteligência Artificial. A solução utiliza recursos de Visão Computacional como mecanismo de avaliação do ambiente. A captura de imagens e a medição de distância são realizadas por um módulo ESP32-CAM, enquanto o processamento e a inferência ocorrem em um servidor externo. A inovação principal reside na entrega do *feedback*: o servidor processa a imagem e envia o alerta sonoro sintetizado (*Text-to-Speech*) em tempo real via protocolo *WebSocket* para uma aplicação móvel Android, que reproduz a mensagem instantaneamente nos fones de ouvido do usuário. Para validação da proposta, foram executados testes capazes de identificar objetos e informar sua distância com baixa latência. Os resultados demonstram que a integração entre IoT, processamento em nuvem e dispositivos móveis contribui significativamente para o aumento da segurança, da percepção contextual e da independência na locomoção de deficientes visuais.

Palavras-chave: Tecnologia Assistiva. Internet das Coisas (*IoT*). Visão Computacional. *ESP32-CAM*. Acessibilidade.

ABSTRACT

Autonomous urban mobility is an essential aspect of the quality of life for visually impaired individuals, who often face physical and architectural barriers aggravated by the lack of accessibility in cities. Traditional tools, such as the white cane, although fundamental, have limitations in detecting aerial obstacles and in the semantic identification of the environment. Given this scenario, this work proposes the development of a wearable visual assistance system based on the Internet of Things (IoT) and Artificial Intelligence. The solution applies Computer Vision technology as a way to understand the environment. The Image capture and distance measurement are performed by an ESP32-CAM module, while heavy processing and inference occur on an external server. The main innovation lies in the feedback delivery: the server processes the image and sends the synthesized audio alert (*Text-to-Speech*) in real-time via the WebSocket protocol to an Android mobile application, which instantly reproduces the message through the user's headphones. To validate the proposal, tests were performed that could identify objects and report their distance with low latency. The results demonstrate that the integration between IoT, cloud processing, and mobile devices contributes significantly to increasing safety, contextual perception, and locomotion independence for the visually impaired.

Keywords: Assistive Technology. Internet of Things (IoT). Computer Vision. ESP32-CAM. Accessibility

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O Dia Nacional do Cego tem sua data definida em 13 de dezembro, desde 1961. Porém, o público recebe pouco apoio do governo para que a data seja efetivamente comemorada com avanços concretos na acessibilidade. Dados recentes indicam que 18,6% da população brasileira possui algum tipo de deficiência visual, dos quais 6,5 milhões apresentam deficiência visual severa, sendo que 506 mil têm perda total da visão (0,3% da população) e 6 milhões, grande dificuldade para enxergar (3,2%) (MINISTÉRIO DA EDUCAÇÃO, 2023). Para esses indivíduos, as barreiras de mobilidade urbana poderiam ser drasticamente reduzidas com o apoio de recursos científicos e tecnológicos.

A Tecnologia Assistiva é compreendida como o resultado da aplicação de inovações tecnológicas, fruto do diálogo e da interação entre especialistas de diversas vertentes do conhecimento. Seu objetivo primordial é a reabilitação humana, buscando ampliar a autonomia do indivíduo e a qualidade de suas relações sociais (SONZA; SALTON; CARNIEL, 2021). Nesse contexto, a integração de dispositivos conectados à rede mundial de computadores expande as possibilidades de inclusão, uma vez que a Internet das Coisas (IoT) apresenta oportunidades significativas e tendências promissoras para o desenvolvimento de soluções de acessibilidade mais robustas e inteligentes (RODRIGUES; FORTES, 2019).

A presente pesquisa constitui uma evolução do projeto intitulado "*A visão computacional aliada no aprimoramento de um boné assistivo embarcado*" (LEMOS, et al., 2023). Embora tal iniciativa tenha validado o conceito de auxílio visual, o protótipo anterior enfrentou limitações técnicas, como instabilidade de *hardware,* causada pelo superaquecimento no *streaming* contínuo de vídeo e restrição de conectividade apenas à rede local. Tendo em vista a necessidade de inovação tecnológica para a segurança viária, faz-se necessário superar modelos que focam apenas em emitir sinais sonoros simples ou vibratórios, os quais, além de gerarem confusão em ambientes complexos, não oferecem descrição semântica do entorno (DE MILANO; HONORATO, 2014).

Para mitigar esses problemas e garantir a estabilidade do sistema, foram aplicadas diferentes técnicas de Visão Computacional. Destarte, este trabalho propõe o desenvolvimento de um sistema *IoT* vestível de assistência visual que supera as limitações de processamento e memória observadas anteriormente. O novo protótipo substitui o *feedback* abstrato por um *feedback* auditivo em linguagem natural, utilizando tecnologias de reconhecimento e síntese de voz (*Text-to-Speech*) baseadas em aprendizado profundo (*Deep Learning*) para descrever o ambiente de forma compreensível e em tempo real (REDDY; VAISHNAVI; KUMAR, 2023).

Para desenvolver uma versão sem onerar o hardware vestível e evitar o travamento do microcontrolador, este trabalho adota um modelo arquitetural híbrido, combinando a computação em borda (*Edge Computing*) com o processamento em nuvem (*Cloud Computing*). Essa abordagem permite que a captura e a compressão de dados ocorram localmente na borda, via ESP32, de forma fracionada e sob demanda, enquanto o processamento computacional pesado (como a visão computacional) é delegado a um servidor externo. Após a análise remota, a notificação e o áudio são encaminhados via *WebSockets* para uma aplicação móvel Android nativa, que atua como interface final para o usuário, garantindo uma resposta rápida, estável e eficiente no deslocamento urbano (YU et al., 2017).

1.1 JUSTIFICATIVA E CONTRIBUIÇÕES DO TRABALHO

No Brasil, a mobilidade urbana apresenta desafios significativos para pessoas com deficiência visual. A falta de padronização em calçadas, a ausência de sinalização em obras e a presença de obstáculos aéreos (como placas e galhos) tornam a locomoção insegura (MARTIN; MARTIN, 2010). Embora a bengala seja o instrumento de auxílio mais comum e indispensável, estudos apontam que, mesmo com seu uso, 90% dos usuários ainda sentem insegurança em relação ao espaço construído, visto que a bengala detecta apenas obstáculos ao nível do solo, deixando o tronco e a cabeça vulneráveis (HOFFMANN apud ASSIS et al., 2017; MAGAGNIN, 2009).

Além disso, a alternativa mais eficiente, o cão-guia, enfrenta barreiras de acesso proibitivas. Apesar da Lei nº 11.126/05 garantir o direito de ingresso com cão-guia, a oferta desses animais é escassa no país, com custos que podem chegar a 35 mil reais e filas de espera de aproximadamente três anos, tornando-os inacessíveis para a maioria da população (ALESP, 2021). Tal cenário evidencia uma lacuna entre a segurança oferecida pela bengala e a eficiência do cão-guia, destacando a necessidade de soluções tecnológicas intermediárias que sejam eficientes e financeiramente viáveis.

Diante disso, a contribuição deste trabalho reside no desenvolvimento de um protótipo de sistema assistivo vestível (*wearable*) baseado em Internet das Coisas (*IoT*) e Visão Computacional, projetado para atuar como um complemento à bengala. A solução preenche a lacuna existente ao fornecer informações semânticas sobre o ambiente — não apenas alertando que "há algo", mas informando "o que é" e "a qual distância" —, funcionalidade antes restrita à visão humana ou aos cães-guia.

Uma segunda contribuição encontra-se na arquitetura distribuída proposta, que integra a captação de baixo custo via microcontroladores (ESP32) com o processamento robusto em um servidor remoto (*YOLOv8*). Essa abordagem viabiliza o uso de técnicas avançadas de Visão Computacional sem a necessidade de *hardware* embarcado de alto custo, democratizando o acesso à tecnologia.

Além disso, o trabalho inova ao substituir *feedbacks* abstratos (bipes ou vibrações, que podem ser confusos em ambientes ruidosos) por síntese de voz natural (*Text-to-Speech*) reproduzida diretamente no *smartphone* do usuário, proporcionando uma compreensão contextual mais clara e assertiva, reduzindo a carga cognitiva e aumentando a independência na locomoção.

1.2 OBJETIVOS

Este trabalho tem como objetivo desenvolver um protótipo de sistema vestível de assistência visual fundamentado em Internet das Coisas (IoT) e Visão Computacional, capaz de identificar obstáculos e fornecer *feedback* auditivo contextualizado em tempo real para pessoas com deficiência visual.

Para atingir o objetivo, foram definidos os seguintes objetivos específicos:

I. Investigar algoritmos de Visão Computacional, com ênfase em redes neurais artificiais voltadas à detecção de objetos em tempo real, e suas viabilidades de aplicação em Tecnologia Assistiva;
II. Definir e projetar uma arquitetura de hardware embarcado de baixo custo, dotada de recursos para captura de imagens e sensoriamento de distância, para atuar como nó de percepção na borda (*Edge*);
III. Desenvolver uma aplicação móvel para atuar como interface do usuário final, gerenciando a comunicação bidirecional de baixa latência e a reprodução imediata dos alertas sonoros;
IV. Validar o protótipo integrado quanto à precisão na identificação de obstáculos urbanos e à latência da resposta auditiva em cenários controlados de teste;
V. Fomentar o desenvolvimento e a pesquisa de soluções assistivas de baixo custo, promovendo o debate sobre a democratização tecnológica e a acessibilidade na mobilidade urbana.

1.3 ESTRUTURA DO DOCUMENTO

Este trabalho está organizado em seis capítulos principais, estruturados de forma a conduzir o leitor desde a contextualização do problema de mobilidade enfrentado por pessoas com deficiência visual até a apresentação da solução tecnológica desenvolvida e a análise de seus resultados.

Inicialmente, o Capítulo 1 apresenta a introdução ao tema, definindo os objetivos gerais e específicos do trabalho, bem como a justificativa da pesquisa, suas contribuições para a área de Tecnologia Assistiva e a estrutura do documento.

Em seguida, o Capítulo 2 desenvolve a fundamentação teórica necessária para o embasamento do projeto. Nele, são abordados conceitos de Tecnologia Assistiva, os paradigmas da Internet das Coisas (IoT) e a arquitetura de sistemas embarcados, com foco no módulo ESP32. O capítulo também discute os fundamentos de Visão Computacional, detalhando o algoritmo YOLO (*You Only Look Once*), além de apresentar os protocolos de comunicação, as plataformas móveis e as técnicas de síntese de voz utilizadas.

O Capítulo 3 destina-se aos trabalhos relacionados e à evolução do projeto. Esta seção analisa as iniciativas anteriores que serviram de base para esta pesquisa, detalhando especificamente as limitações da arquitetura do projeto antecessor que motivaram a nova abordagem adotada.

A partir desse embasamento, o Capítulo 4 descreve a solução proposta, detalhando toda a nova arquitetura do sistema. São apresentados os componentes de hardware e a montagem do nó de visão vestível, além da arquitetura de software, que engloba o firmware embarcado, o servidor de processamento com o módulo de Inteligência Artificial e a aplicação móvel responsável pelos fluxos de interação.

Na sequência, o Capítulo 5 expõe os resultados obtidos com a implementação. São documentadas a consolidação do hardware, o funcionamento da aplicação móvel, e os testes práticos de desempenho e validação, avaliando a precisão dos sensores, a latência do sistema e o comportamento do protótipo em cenários reais, incluindo análises de falhas.

Por fim, o Capítulo 6 reúne as considerações finais do trabalho, analisando o cumprimento dos objetivos e sugerindo direcionamentos para trabalhos futuros. Encerrando o documento, o Capítulo 7 lista as referências utilizadas para a fundamentação e desenvolvimento da pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta as bases teóricas essenciais para a compreensão do sistema proposto, contextualizando as tecnologias aplicadas no desenvolvimento do dispositivo de assistência visual. São abordados temas relacionados à tecnologia assistiva, internet das coisas (IoT), sistemas embarcados de baixo custo, visão computacional e protocolos de comunicação de dados e áudio.

2.1 TECNOLOGIA ASSISTIVA E ACESSIBILIDADE

A Tecnologia Assistiva (TA) é uma área do conhecimento de característica interdisciplinar que engloba produtos, recursos, metodologias, estratégias, práticas e serviços que objetivam promover a funcionalidade, relacionada à atividade e participação de pessoas com deficiência, incapacidades ou mobilidade reduzida, visando sua autonomia, independência, qualidade de vida e inclusão social (BRASIL, 2015). Mais do que meros dispositivos técnicos, a TA representa uma ponte para a cidadania, permitindo que indivíduos superem barreiras sensoriais, motoras ou cognitivas impostas pelo ambiente.

Segundo Rodrigues e Alves (2013), a tecnologia assistiva deve ser compreendida como um vasto arsenal de recursos e serviços que contribuem para proporcionar ou ampliar habilidades funcionais de pessoas com deficiência. Nesse sentido, a aplicação desses recursos não se restringe à reabilitação física, mas estende-se à inclusão educacional, laboral e social, garantindo que a tecnologia atue como um equalizador de oportunidades em uma sociedade que ainda apresenta diversas barreiras de acessibilidade.

Para compreender a aplicação da TA no contexto deste trabalho, faz-se necessário distinguir as condições do público-alvo. Segundo Conde (2017), a cegueira não é um termo absoluto, visto que é uma condição que torna o indivíduo parcial ou completamente desprovido de visão. As deficiências visuais são categorizadas em dois grandes grupos: pessoas com cegueira, que podem ter desde a percepção apenas de vultos e luminosidade até a perda total da visão; e pessoas com baixa visão, que possuem limitações severas não corrigíveis por óculos convencionais, mas que ainda utilizam a visão residual para atividades cotidianas (AMIRALIAN, 1997).

No que tange à mobilidade independente dessas pessoas, as ferramentas tradicionais desempenham papel central, sendo a bengala longa o instrumento mais difundido devido ao seu baixo custo e eficiência na detecção de desníveis e obstáculos no solo. Contudo, seu

alcance é limitado ao contato físico direto e à varredura próxima aos pés, deixando a parte superior do corpo vulnerável a obstáculos aéreos (como galhos, placas e telefones públicos). Outra alternativa é o cão-guia, que oferece excelente desvio de obstáculos e segurança, porém enfrenta barreiras de alto custo financeiro e tempo de treinamento, tornando-se inacessível para a grande maioria da população (ALESP, 2021).

Nesse cenário de limitações dos recursos analógicos, surgem os Sistemas Eletrônicos de Auxílio à Mobilidade, conhecidos internacionalmente como *Electronic Travel Aids* (ETAs). Hersh (2010) define os ETAs como dispositivos que transformam informações ambientais que não seriam percebidas pelo usuário (como a distância de um objeto silencioso) em estímulos sensoriais alternativos, geralmente auditivos ou táteis.

Para Durette (2009), esse processo baseia-se no conceito de substituição sensorial, que pode ocorrer de duas formas principais: a substituição visuo-tátil, onde a informação visual é convertida em vibrações na pele; e a substituição visuo-auditiva, onde os dados visuais são convertidos em sons ou fala. Enquanto os primeiros ETAs focavam apenas em alertar sobre a presença de objetos via "bipes" ou vibração (semelhante a um sensor de ré automotivo), a evolução da capacidade de processamento permitiu o surgimento de dispositivos capazes de realizar a identificação semântica, informando ao usuário não apenas "onde" está o obstáculo, mas "o que" ele é, proporcionando uma compreensão contextual muito mais rica e segura do ambiente.

2.2 INTERNET DAS COISAS E COMPUTAÇÃO EM BORDA

As Tecnologias da Informação e Comunicação (TICs) desempenham um papel fundamental na melhoria da qualidade de vida, devendo possibilitar o amplo acesso independentemente de limitações físicas ou sensoriais. Nesse contexto, a intersecção entre tecnologia e acessibilidade pode ser resumida pelo pensamento do *National Council on Disability* (1993): "para as pessoas sem deficiência, a tecnologia torna as coisas fáceis. Para as pessoas com deficiência, a tecnologia torna as coisas possíveis". Com o advento da *Internet of Things*, abrem-se novas perspectivas para que pessoas com deficiência vivam com maior autonomia e independência (RODRIGUES; FORTES, 2019).

O termo *Internet of Things* foi cunhado originalmente por Kevin Ashton em 1999, referindo-se a um novo paradigma onde objetos físicos são conectados à rede mundial de computadores (ASHTON et al., 2009). Na literatura técnica, Atzori et al. (2010) definem a *IoT* como a presença de uma variedade de objetos — como sensores, atuadores, etiquetas

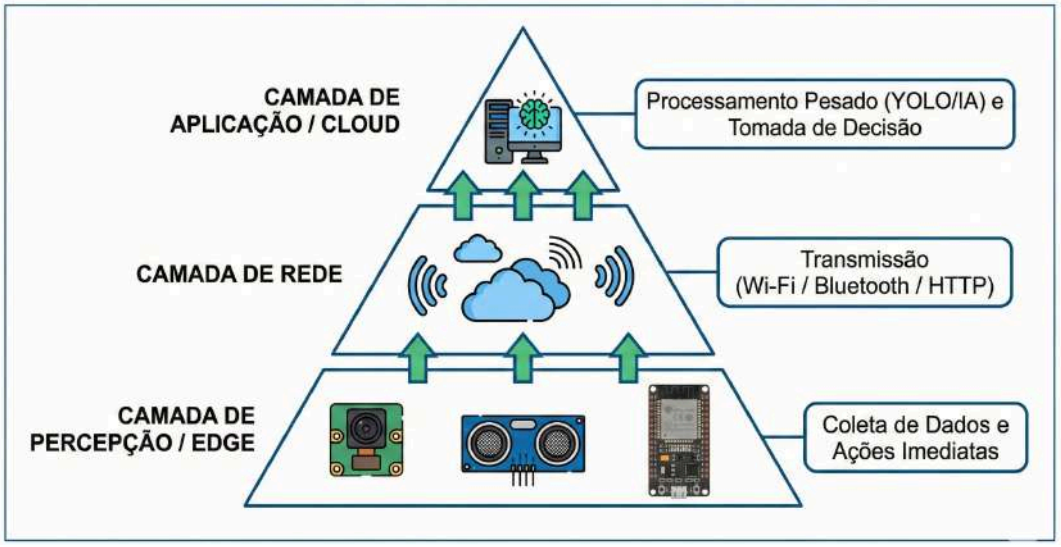
RFID e dispositivos móveis — que, através de esquemas de endereçamento únicos, são capazes de interagir entre si e cooperar para alcançar objetivos comuns.

Formalmente, a *IoT* refere-se a uma infraestrutura de rede global dinâmica, onde objetos físicos e virtuais possuem identidades, atributos físicos e personalidades virtuais, utilizando interfaces inteligentes e sendo integrados à rede de informação (ITU, 2012). Essa arquitetura permite que objetos do cotidiano gerem dados que podem ser analisados para a tomada de decisões inteligentes.

Uma arquitetura típica de *IoT* é composta por três camadas principais, conforme descrito abaixo e ilustrado na Figura 1:

1. Camada de Percepção: responsável pela coleta de dados do ambiente através de sensores (como câmeras e sensores ultrassônicos) e pela conversão de sinais analógicos em digitais;
2. Camada de Rede: responsável pela transmissão confiável desses dados, utilizando protocolos de comunicação com e sem fio (como *Wi-Fi* e redes móveis 4G/5G);
3. Camada de Aplicação: onde ocorre o processamento dos dados, a inteligência do sistema e a interface com o usuário final.

Figura 1 – Arquitetura de Camadas em Sistemas IoT Vestíveis



Fonte: Gemini (2026)

No desenvolvimento de dispositivos vestíveis (*wearables*) para tecnologia assistiva, a eficiência energética e a capacidade de processamento em tempo real são requisitos críticos. Devido às limitações de *hardware* de dispositivos embarcados, adota-se frequentemente o

modelo híbrido que combina a computação em borda (*Edge Computing*) — realizando a captura e compressão de dados localmente — com o processamento em *Cloud Computing*, onde algoritmos complexos, como redes neurais de visão computacional, são executados. Essa divisão de tarefas otimiza o uso da largura de banda e garante que o dispositivo vestível permaneça leve e autônomo.

2.3 SISTEMAS EMBARCADOS E MICROCONTROLADORES

Sistemas embarcados são computadores dedicados a funções específicas dentro de um sistema maior, diferindo dos computadores de uso geral por serem otimizados para reduzir tamanho, custo e consumo de energia, muitas vezes operando com recursos limitados de memória e processamento (ALMEIDA, 2018). No contexto atual da *Internet of Things*, esses sistemas têm evoluído para suportar processamento na borda (*Edge Computing*), permitindo a execução de tarefas complexas localmente antes do envio para o servidor externo.

2.3.1 Módulo ESP32

O ESP32 é uma série de *Systems on a Chip* (Sistema em um Chip) de baixo custo e baixo consumo de energia, desenvolvidos pela *Espressif Systems*. Ele integra microprocessadores *dual-core Xtensa® LX6* de 32 bits, operando em frequências de até 240 MHz, além de possuir 520 KB de memória *SRAM* interna (ESPRESSIF, 2023). Essa arquitetura robusta, aliada à conectividade *Wi-Fi* e *Bluetooth* nativa, o torna ideal para aplicações de *IoT* que exigem tanto coleta de dados quanto comunicação sem fio de alta velocidade.

No contexto deste trabalho, a capacidade de processamento *dual-core* é fundamental para garantir que o microcontrolador gerencie a pilha de protocolos de rede (*Wi-Fi*) em um núcleo, enquanto processa a captura de imagem no outro, evitando travamentos durante o envio de dados críticos.

Para a captura visual, optou-se especificamente pelo módulo de desenvolvimento ESP32-CAM *AI-Thinker* (Figura 2). Este modelo se diferencia das versões padrão por integrar:

1. Interface de Câmera (DVP): suporte nativo para sensores de imagem como o OV2640, capaz de realizar compressão JPEG via hardware;

2. Memória PSRAM Externa: adição de até 4MB de memória de acesso aleatório pseudo-estática. Este recurso é indispensável para o armazenamento temporário (buffering) de imagens de média resolução, visto que a memória interna padrão do ESP32 seria insuficiente para manipular arquivos de imagem antes da transmissão via HTTP.

Figura 2 – Microcontrolador ESP32CAM

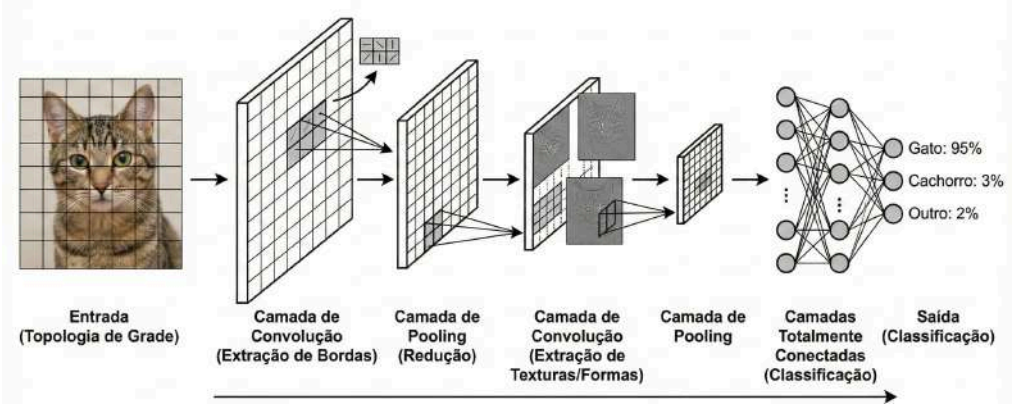
Fonte: Elaborado pelo autor

2.4 VISÃO COMPUTACIONAL E APRENDIZADO DE MÁQUINA

A Visão Computacional é uma área que busca capacitar computadores a processar, analisar e compreender imagens digitais de forma a extrair informações multidimensionais do mundo real. Enquanto o processamento de imagens foca na transformação de imagens (como filtros e ajustes), a visão computacional foca na interpretação semântica da cena, simulando a complexidade do sistema visual humano (GONZALEZ; WOODS, 2010).

Nos últimos anos, o desempenho dessas tarefas foi revolucionado pelo *Deep Learning* (Aprendizado Profundo), uma subárea do aprendizado de máquina baseada em Redes Neurais Artificiais com múltiplas camadas de processamento. Dentre as arquiteturas de redes neurais, destacam-se as *Convolutional Neural Networks* (CNNs). As CNNs são projetadas para processar dados com topologia de grade (como imagens), utilizando operações matemáticas de convolução para extrair características relevantes (bordas, texturas, formas) automaticamente, conforme ilustra a Figura 3, sem a necessidade de extração manual de atributos (LECUN; BENGIO; HINTON, 2015; GOODFELLOW et al., 2016).

Figura 3 – Arquitetura Básica de uma Rede Neural Convolucional (CNN) para Processamento de Imagens



Fonte: Gemini (2026)

2.4.1 Algoritmo YOLO (*You Only Look Once*)

A detecção de objetos é uma tarefa que combina a classificação (o que é o objeto) com a localização (onde ele está na imagem). Tradicionalmente, algoritmos de detecção operam em dois estágios, como a família *R-CNN* (*Region-based Convolutional Neural Networks*), que primeiro propunha regiões de interesse e depois classificava cada uma, resultando em alta precisão, mas alto custo computacional e lentidão (GIRSHICK et al., 2014).

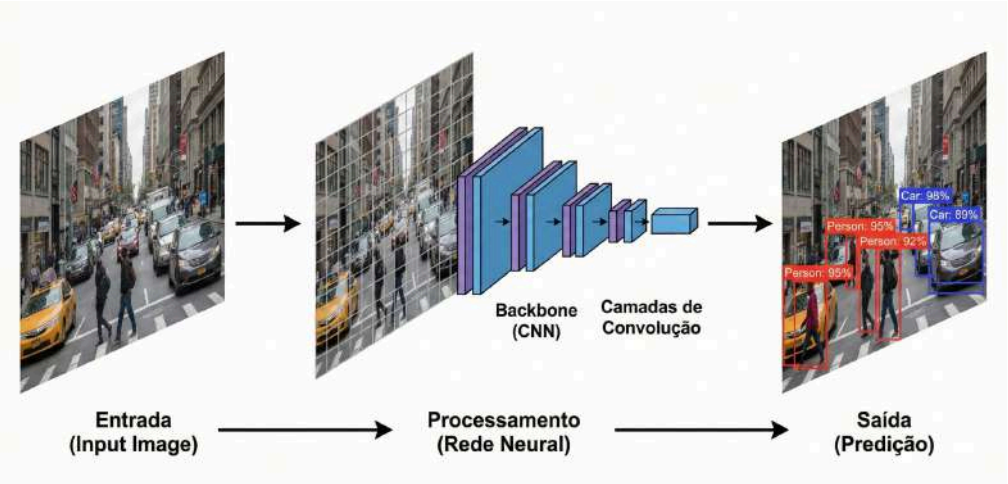
O *YOLO* (*You Only Look Once*), proposto originalmente por Redmon et al. (2016), introduziu uma mudança de paradigma ao tratar a detecção como um problema único de regressão (detecção de um estágio ou *one-stage detector*). O algoritmo divide a imagem de entrada em uma grade *SxS*. Se o centro de um objeto recai em uma célula da grade, essa célula torna-se responsável por detectá-lo.

A rede neural então prevê, simultaneamente, as coordenadas das *bounding boxes* (caixas delimitadoras), o grau de confiança da existência do objeto e as probabilidades das classes para cada célula. Essa arquitetura unificada permite o processamento em tempo real, atingindo taxas de quadros por segundo (*FPS*) muito superiores às abordagens de dois estágios, sendo ideal para aplicações críticas de segurança e acessibilidade.

Neste trabalho, utiliza-se a versão YOLOv8, desenvolvida pela Ultralytics em 2023 (JOCHER et al., 2023). A Figura 4 ilustra o fluxo de processamento desta arquitetura: a imagem de entrada (*Input Image*) é inicialmente dividida em uma grade e submetida ao *Backbone*, uma Rede Neural Convolucional (CNN) profunda responsável pela extração de características visuais. Conforme destacado no esquema, o modelo aplica sucessivas camadas de convolução (blocos azuis e roxos) para reduzir a dimensionalidade espacial e identificar padrões complexos, substituindo o antigo módulo C3 pelo módulo C2f para otimizar o fluxo

de gradiente (TERVEN; CORDOVA-ESPARZA, 2023). Na etapa final, denominada Saída (Predição), o modelo gera simultaneamente as caixas delimitadoras (*bounding boxes*) e as probabilidades de classe. Como observado na figura, o sistema identifica múltiplos objetos na cena em uma única passagem pela rede (*one-stage*), sem a necessidade de propor regiões preliminares. O YOLOv8 utiliza uma abordagem *anchor-free* e um *Head* desacoplado para processar essas predições, garantindo a alta precisão e a baixa latência necessárias para alertar o usuário sobre obstáculos em tempo real.

Figura 4 – Arquitetura de funcionamento do algoritmo YOLO



Fonte: Gemini (2026)

2.5 PROTOCOLOS DE COMUNICAÇÃO

A comunicação eficiente entre os dispositivos de borda (*Edge*) e o servidor de processamento, bem como a interface com o usuário, depende de protocolos robustos que garantam a integridade dos dados e a baixa latência. Neste projeto, adota-se uma abordagem híbrida que combina protocolos baseados no modelo requisição-resposta para o envio de grandes volumes de dados e protocolos *full-duplex* para a sinalização de eventos em tempo real, garantindo que o alerta chegue ao dispositivo móvel do usuário com o menor atraso possível.

2.5.1 HTTP e APIs REST

O Hypertext Transfer Protocol (HTTP) é o protocolo padrão da camada de aplicação da *World Wide Web*, operando no modelo cliente-servidor (MOZILLA, 2023). Embora

protocolos como o MQTT sejam comuns em IoT para telemetria leve devido ao seu baixo consumo de recursos e arquitetura *publish/subscribe* (BANKS *et al.*, 2019), o HTTP é preferível quando há necessidade de transferência de grandes volumes de dados (payloads extensos), como arquivos de imagem, devido à sua capacidade de negociação de conteúdo e suporte a transferências em partes (TANENBAUM; WETHERALL, 2011).

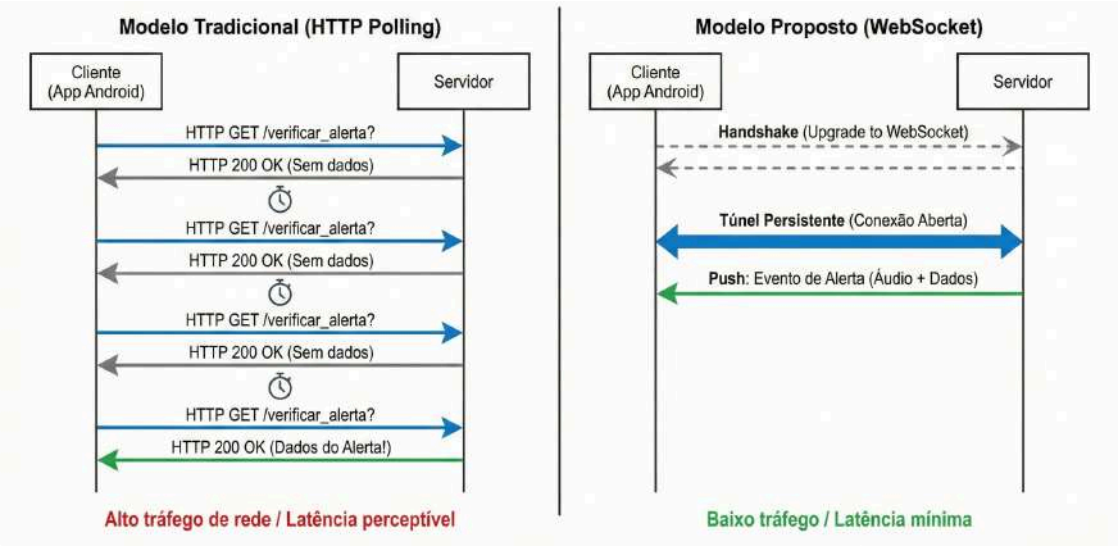
Para a transferência das capturas visuais do ESP32 para o servidor, utiliza-se o método *POST*, encapsulando a imagem em formato *multipart/form-data*. A comunicação segue o estilo arquitetural *REST* (*Representational State Transfer*), descrito por Fielding (2000). Uma das principais características do *REST* adotada neste trabalho é a "apatridia" (*statelessness*), ou seja, cada requisição do ESP32 contém todas as informações necessárias para que o servidor a compreenda, sem depender de contextos armazenados anteriormente. Isso reduz o consumo de memória no microcontrolador, pois não é necessário manter sessões complexas abertas, facilitando a interoperabilidade com o *backend* desenvolvido em *Python* (RICHARDSON; RUBY, 2008).

2.5.2 Protocolo WebSocket

Enquanto o *HTTP* é eficiente para o envio da imagem, ele apresenta limitações para o envio do alerta do servidor para o cliente (aplicativo móvel) em tempo real. No modelo tradicional *HTTP*, o cliente precisaria realizar requisições constantes ao servidor para verificar se há novas informações, técnica conhecida como *polling*. Segundo Silva (2013), o *polling* gera um tráfego de rede desnecessário e introduz latência, pois a informação só é recebida no próximo ciclo de consulta, além de aumentar drasticamente o consumo de bateria do dispositivo móvel.

Para solucionar esse problema, utiliza-se o protocolo *WebSocket*. Padronizado pela IETF na RFC 6455 (FETTE; MELNIKOV, 2011), o *WebSocket* permite o estabelecimento de um canal de comunicação bidirecional (*full-duplex*) sobre uma única conexão *TCP*. O processo inicia-se com um *handshake* compatível com *HTTP*, onde o cliente solicita a "atualização" da conexão. Uma vez estabelecido, o servidor pode enviar dados (como o arquivo de áudio do alerta) para o cliente prontamente, sem que este tenha solicitado explicitamente, conforme ilustrado no comparativo da Figura 5.

Figura 5 – Comparativo entre os modelos de comunicação *HTTP Polling* e *WebSocket* para sistemas de tempo real.



Fonte: Gemini (2026)

No contexto deste sistema, o *WebSocket* é fundamental para a arquitetura orientada a eventos. Assim que a aplicação detecta um obstáculo e gera o áudio, o servidor utiliza esse canal persistente para "empurrar" (*push*) o alerta imediatamente para o aplicativo, garantindo que o tempo entre a detecção e o *feedback* sonoro seja mínimo, o que é crucial para a segurança do usuário durante a locomoção.

2.6 DESENVOLVIMENTO DE APLICAÇÕES MÓVEIS

A computação móvel consolidou-se como um dos pilares da sociedade da informação, transformando dispositivos celulares em computadores de alto desempenho, capazes de realizar processamento complexo, armazenamento de dados e comunicação em alta velocidade. No contexto da Internet das Coisas (*IoT*), os dispositivos móveis assumem um papel estratégico, atuando frequentemente como *gateways* ou interfaces de controle que conectam o usuário final aos sensores e atuadores distribuídos no ambiente.

Essa integração é particularmente relevante na Tecnologia Assistiva. Diferente de *hardwares* proprietários e fechados, os *smartphones* modernos já possuem nativamente uma suíte de sensores (GPS, acelerômetros) e recursos de acessibilidade (como leitores de tela) que podem ser explorados por desenvolvedores para criar soluções de baixo custo e alta capilaridade. Dessa forma, o desenvolvimento de aplicações móveis deixa de ser apenas sobre a criação de interfaces gráficas e passa a envolver a orquestração de recursos de *hardware* e conectividade em tempo real.

2.6.1 Plataforma Android

O *Android* é um sistema operacional móvel líder de mercado, baseado em uma versão modificada do *kernel Linux*. Segundo Deitel, Deitel e Deitel (2015), o *Android* não é apenas um sistema operacional, mas uma arquitetura de pilha de *software* que inclui o sistema operacional, *middleware* e aplicativos principais. Sua natureza de código aberto (*open source*) e sua arquitetura baseada em camadas permitem que desenvolvedores acessem recursos profundos do sistema, como o controle de rede e a reprodução de mídia, com grande flexibilidade.

Complementarmente, Lecheta (2015) destaca que o grande diferencial do *Android* reside no seu *Software Development Kit* (SDK). Este *kit* fornece as ferramentas e APIs necessárias para desenvolver aplicações na linguagem *Java* (GOSLING et al., 2014) ou *Kotlin* (JEMEROV; ISAKOVA, 2017), permitindo o acesso direto ao *hardware* do dispositivo de forma abstraída.

Para este trabalho, a plataforma destaca-se pela robustez no gerenciamento de conexões de rede persistentes e pelo suporte nativo a formatos de áudio digital. Ao utilizar o *Android* como cliente do sistema de segurança, aproveita-se a capacidade do sistema de gerenciar tarefas em segundo plano (*background services*), garantindo que a comunicação com o servidor via *WebSocket* permaneça ativa e que os alertas sonoros sejam priorizados pelo processador de áudio do dispositivo, assegurando a entrega imediata da informação ao usuário.

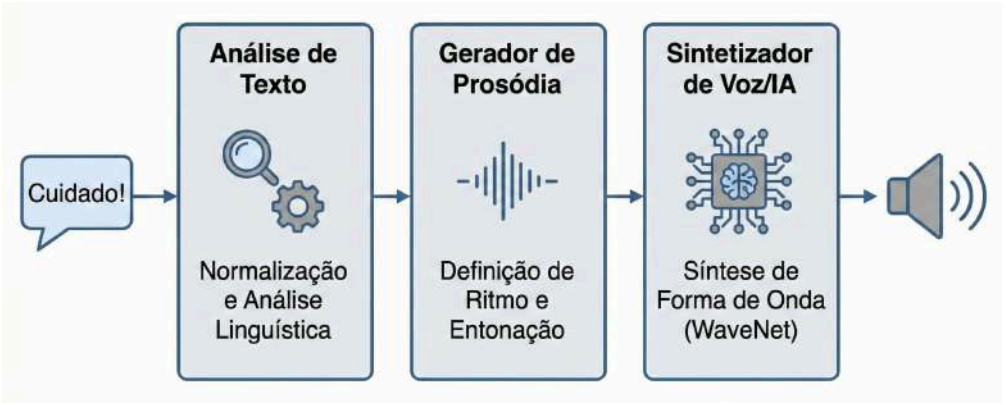
2.7 SÍNTESE DE VOZ *(TEXT-TO-SPEECH*)

A interação humano-computador evoluiu significativamente com a introdução de interfaces naturais, sendo a tecnologia de conversão de texto em fala (*TTS* - *Text-to-Speech*) um componente vital para a acessibilidade. O *TTS* permite a geração artificial de fala humana a partir de texto escrito, atuando como um canal sensorial substituto para indivíduos com deficiência visual (TAYLOR, 2009).

O funcionamento de um sistema *TTS* moderno divide-se, geralmente, em duas etapas principais, conforme ilustra o fluxo da Figura 6: o processamento de linguagem natural (*NLP*) e o processamento digital de sinais (*DSP*). Na primeira etapa, o texto bruto é normalizado (ex: converter "2m" para "dois metros") e analisado foneticamente para determinar a pronúncia e a

prosódia (ritmo e entonação). Na segunda etapa, o sistema sintetiza a forma de onda de áudio. Enquanto sistemas antigos utilizavam a síntese concatenativa (colagem de pequenos pedaços de som gravado), resultando em vozes robóticas, os sistemas atuais baseiam-se em Redes Neurais Profundas (como o *WaveNet*), capazes de gerar áudio com naturalidade indistinguível da fala humana (VAN DEN OORD et al., 2016).

Figura 6 – Fluxo de processamento de um sistema *Text-to-Speech*



Fonte: Adaptado de Taylor (2009)

No contexto deste projeto, a implementação da síntese de voz é realizada através da biblioteca *gTTS* (*Google Text-to-Speech*) (GOOGLE, 2024). Esta ferramenta atua como uma interface para a *API* de tradução e síntese do *Google*, permitindo que o servidor *Python* converta as *strings* de alerta geradas pelo algoritmo *YOLO* (ex: "Carro detectado") em arquivos de áudio no formato *MP3*. A escolha por essa tecnologia justifica-se pela alta inteligibilidade da voz gerada e pela facilidade de integração com a arquitetura do sistema, onde o arquivo de áudio é gerado no servidor e transmitido instantaneamente para a aplicação móvel *Android*, garantindo a reprodução imediata do alerta ao usuário.

3 TRABALHOS RELACIONADOS

A aplicação da visão computacional para assistência a pessoas com deficiência visual tem sido explorada sob diferentes perspectivas. No entanto, é fundamental distinguir entre soluções focadas em descrição de cenários (que visam narrar o ambiente de forma contemplativa) e soluções focadas em mobilidade e segurança (que visam evitar acidentes em tempo real). Neste sentido, a presente seção analisa trabalhos recentes que, embora utilizem tecnologias similares, apresentam limitações de desempenho ou divergem do objetivo de fornecer uma navegação segura aliada a um baixo custo de aquisição de hardware, destacando como a proposta atual preenche essas lacunas.

Nessa linha de pesquisa mais descritiva, Gomes *et al.* (2025) apresentaram o *SoundEyes*, um sistema móvel que utiliza visão computacional para descrever objetos em ambientes. Embora o trabalho demonstre a viabilidade do uso de *smartphones*, sua abordagem é generalista, pois foca na descrição ampla do ambiente, narrando diversos elementos presentes na cena sem um filtro de criticidade.

Para um pedestre em deslocamento, o excesso de informações não prioritárias (ex: descrever a cor de uma parede ou um objeto distante) compete com a atenção auditiva necessária para a locomoção, podendo distrair o usuário de perigos imediatos. Diferente do *SoundEyes*, este trabalho prioriza a filtragem de segurança, alertando apenas sobre obstáculos que representam risco de colisão ou relevância para o trajeto, otimizando o tempo de reação do usuário.

Outra limitação recorrente, além da sobrecarga cognitiva, é o tempo de resposta em sistemas mais complexos. Um exemplo prático dessa barreira temporal é o trabalho de Silva e Garrocho (2025), que desenvolveram o *Áurea*: óculos inteligentes que utilizam modelos de linguagem de grande porte (LLMs), especificamente o *Llama 3.2 Vision* (META, 2024), para gerar descrições detalhadas do ambiente.

Embora tecnologicamente avançado, o uso de Inteligência Artificial Generativa apresenta um gargalo fatal para a segurança viária devido à sua alta latência. Modelos massivos como os LLMs exigem um tempo significativo para processar a imagem e gerar um texto descritivo coerente. Em um cenário de trânsito ou de caminhada rápida, um atraso de dois ou três segundos na notificação compromete o tempo de reação, tornando o dispositivo ineficaz para a prevenção de acidentes imediatos.

Além do tempo de processamento, soluções baseadas nessa arquitetura apresentam restrições financeiras significativas. O sistema exige um elevado custo de aquisição inicial por depender de óculos inteligentes específicos, somado a um alto custo operacional contínuo, visto que o processamento em nuvem de LLMs demanda servidores com robusta aceleração gráfica. Em contraste, a proposta atual utiliza o algoritmo YOLOv8, cujo foco recai sobre a detecção instantânea de objetos. Essa eficiência algorítmica reduz drasticamente os custos de infraestrutura de servidor. Além disso, ao integrar essa inteligência a um *hardware* acessível, como o microcontrolador ESP32, o sistema consolida sua viabilidade econômica e assegura que o alerta crítico chegue ao usuário em tempo hábil para evitar a colisão.

Ampliando o escopo metodológico, a convergência entre o paradigma da Internet das Coisas (IoT) e a tecnologia assistiva tem se consolidado como uma tendência promissora. Em uma ampla revisão da literatura sobre o tema, Rodrigues e Fortes (2019) mapearam as oportunidades de desenvolvimento de IoT focadas em acessibilidade, destacando que a interconexão de dispositivos oferece novos níveis de independência para pessoas com deficiência. Contudo, o estudo evidencia que grande parte dessas inovações ainda se concentra em soluções estáticas ou de domótica.

Um exemplo pertinente dessa aplicação *indoor* é o trabalho de autores da Revista Ibero-Americana (2023), que desenvolveu um sistema de automação residencial focado na segurança e gestão de saúde. Utilizando o microcontrolador ESP32 integrado à plataforma Node-RED, o projeto gerencia o controle de acesso e emite alertas automatizados via *Telegram*. Esse estudo corrobora a viabilidade e o baixo custo do ESP32 como núcleo de processamento assistivo, embora sua aplicação se restrinja a ambientes fechados.

Buscando transpor os conceitos de IoT para a mobilidade de Pessoas com Deficiência Visual (PCDVs), Garcia *et al.* (2018) propuseram o projeto *HELIX*. A solução baseia-se em um assistente móvel instalado em *smartphones* que interage com um servidor de contexto central. Embora o sistema utilize de forma eficiente a arquitetura cliente-servidor para emitir alertas, a percepção ambiental *indoor* do *HELIX* depende exclusivamente da leitura de *QR-Codes* espalhados pelo ambiente. Essa abordagem transfere para o usuário a responsabilidade e a dificuldade mecânica de tatear o ambiente, encontrar a etiqueta e alinhar a câmera do celular corretamente.

A presente pesquisa avança sobre essas lacunas. O sistema proposto apropria-se da robustez do ecossistema IoT (ESP32) validado em ambientes controlados e do modelo de servidor remoto proposto pelo HELIX. No entanto, inova ao promover autonomia passiva: ao utilizar visão computacional inteligente embarcada em um dispositivo vestível (*wearable*), a detecção ocorre automaticamente, sem exigir que o usuário direcione ativamente um sensor ou celular para o alvo.

3.1 EVOLUÇÃO DO PROJETO ANTECESSOR

O presente trabalho consolida-se como uma evolução do projeto de iniciação de pesquisa desenvolvido por LEMOS et al. (2023). A análise crítica dos resultados obtidos naquele protótipo foi fundamental para a redefinição da arquitetura de *hardware* e *software* adotada nesta nova proposta.

3.1.1 Limitações da Arquitetura Anterior

No projeto antecessor, a estratégia adotada consistia em utilizar o ESP32-CAM como um dispositivo de *streaming* de vídeo contínuo. As imagens eram transmitidas via rede local para um computador externo de alto desempenho, equipado com placa de vídeo dedicada, onde o algoritmo YOLO era executado. Embora essa abordagem tenha validado a detecção de objetos, ela apresentou barreiras para a mobilidade:

1. Instabilidade e Superaquecimento: manter um fluxo de vídeo contínuo exigia que o módulo ESP32 operasse em sua capacidade máxima ininterruptamente, causando "intermitências e falhas em momentos aleatórios", frequentemente associadas ao superaquecimento do microcontrolador.
2. Fragilidade Mecânica: o protótipo inicial sofreu danos nos cabos e na lente focal, evidenciando a necessidade de um *design* mais robusto.

Essa vulnerabilidade térmica em sistemas embarcados de baixo custo não é isolada. Cruz (2019), ao desenvolver um dispositivo IoT para auxiliar PCDVs na identificação de ônibus, relatou falhas críticas semelhantes em sua arquitetura. Durante a tentativa de manter envios constantes de dados via rede, o autor observou superaquecimento do módulo, travamentos sucessivos e atrasos intoleráveis, inviabilizando a execução de testes de campo (*in vivo*) com o dispositivo embarcado original.

3.1.2 A Mudança de Paradigma: De *Streaming* para *On-Demand*

A convergência entre a experiência prévia de Lemos *et al.* (2023) e os gargalos documentados por Cruz (2019) evidencia que exigir transmissão contínua de um microcontrolador compromete a viabilidade de um sistema assistivo móvel. Rompendo com esse modelo, a nova arquitetura substitui o *streaming* pelo envio de imagens estáticas orientadas a eventos.

Além disso, eliminou-se a dependência de um computador local atrelado ao usuário. O processamento pesado foi migrado para um servidor externo, permitindo que a inferência seja realizada remotamente. Embora essa transição arquitetural introduza um custo operacional recorrente referente à manutenção da infraestrutura de servidores, essa permuta liberta o usuário de carregar hardwares adicionais e resolve os problemas de autonomia energética do dispositivo vestível.

O Quadro 1 apresenta um resumo comparativo evidenciando as características e vantagens desta proposta frente às tecnologias correlatas analisadas.

Quadro 1 – Quadro comparativo dos trabalhos relacionados com esta proposta

Critério de Análise	**SoundEyes (2025)**	**Áurea (2025)**	**Automação Assistiva (2023)**	**Projeto HELIX (2018)**	**Esta Proposta**
Objetivo	Descrição Geral	Leitura de Ambiente (LLM)	Acessibilidade Doméstica	Localizaçã o / Alertas	Segurança Viária / Mobilidade
Interface de Captura	Câmera do *Smartphone*	Óculos Customizados	Sensores Fixos *Indoor*	*Smartphon e* (Lendo QR-Codes)	Câmera Vestível (ESP32-CAM)
Latência	Média	Alta (Geração de Texto)	Baixa (Eventos Locais)	Baixa (Depende de *Scan* manual)	Baixa (Detecção Instantânea)
Tipo de *Feedback*	Narrativo Longo	Narrativo Longo	Notificação de Texto	Audível Padrão do SO	Diretivo (Curto e Rápido)
Custo de Aquisição (Hardware)	Médio	Alto (Hardware Específico)	Baixo (ESP32 + Sensores)	Baixo (Apenas o *Smartphon*	Baixo (ESP32 + Celular Básico)

				e)	
Mobilidade /Trânsito	Baixa (Distração)	Baixa (Demora excessiva)	Nula (Estático/Fixo)	Baixa (Busca manual por *Tags*)	Alta (Foco em Risco / Autônomo)

Fonte: Elaborado pelo autor

4 SOLUÇÃO PROPOSTA

Com base nos conceitos apresentados e discutidos na fundamentação teórica, este capítulo apresenta uma proposta que busca mitigar as limitações dos dispositivos de auxílio à mobilidade tradicionais, oferecendo não apenas a detecção de obstáculos, mas a identificação semântica do ambiente.

A proposta consiste em uma arquitetura distribuída e híbrida, que combina a portabilidade de sistemas embarcados de baixo custo para a captura de imagens na borda (*Edge*) com o alto poder de processamento de servidores remotos para a execução de algoritmos de Visão Computacional. O sistema integra-se a uma aplicação móvel, que atua como interface final, fornecendo ao usuário um *feedback* auditivo natural e em tempo real, aumentando sua segurança e autonomia durante o deslocamento.

Para facilitar a compreensão dos aspectos construtivos e lógicos do projeto, este capítulo está estruturado da seguinte forma: inicialmente, apresenta-se uma visão geral da solução, detalhando o fluxo de dados. Em seguida, são apresentadas a Arquitetura de *Hardware* e a Arquitetura de *Software*, especificando os componentes eletrônicos, os algoritmos desenvolvidos e a aplicação cliente. Por fim, são apresentados o projeto de interação, demonstrando a dinâmica de comunicação via *WebSocket*, e o ambiente tecnológico, que justifica as ferramentas escolhidas para a implementação do protótipo

4.1 VISÃO GERAL DA SOLUÇÃO

O sistema consiste em um dispositivo de tecnologia assistiva vestível (*wearable*), cuja arquitetura baseia-se no paradigma da Internet das Coisas. Para superar as limitações de processamento e energia dos microcontroladores convencionais, a solução opera de forma distribuída. O diagrama de visão geral, apresentado na Figura 7, ilustra os três blocos principais da solução e o respectivo fluxo de comunicação entre eles:

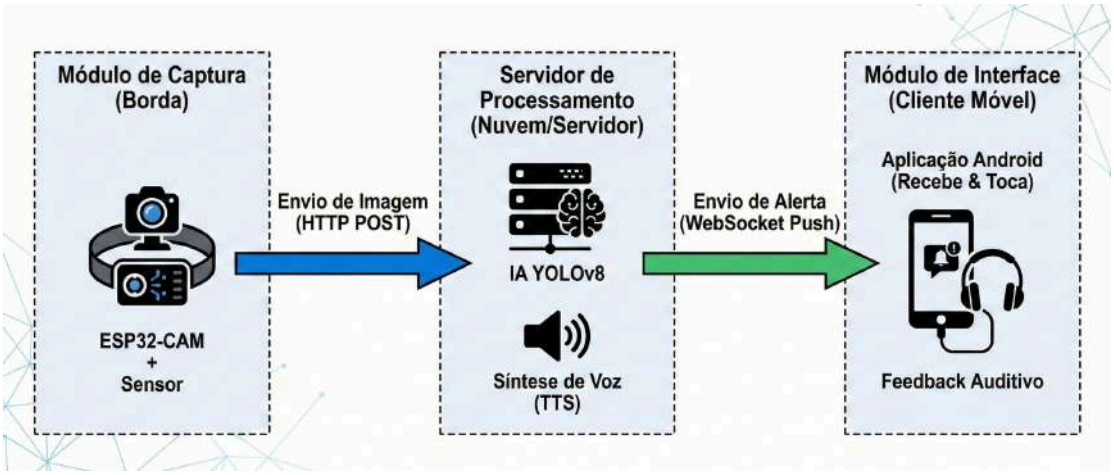
1. Módulo de Captura (Borda): um conjunto de *hardware* baseado no *ESP32-CAM* + Sensor, acoplado a um acessório vestível (como um boné), responsável exclusivamente pela percepção sensorial imediata. Ele integra sensores ultrassônicos para detecção de proximidade e uma câmera para a captura visual do cenário;
2. Servidor de Processamento (Nuvem/Servidor): uma aplicação de alto desempenho executada em uma máquina remota. Este módulo centraliza a

inteligência do sistema, hospedando a *IA YOLOv8* para visão computacional e o algoritmo de Síntese de Voz (*TTS*), que demandam capacidade computacional inviável para dispositivos embarcados de baixo custo;

3. Módulo de Interface (Cliente Móvel): representado pela Aplicação Android (Recebe & Toca) executada no *smartphone* do usuário. Este módulo atua como o receptor final, encarregado de manter o canal de comunicação aberto para fornecer as instruções geradas.

Para minimizar a latência e garantir a segurança do usuário, essa infraestrutura utiliza uma comunicação orientada a eventos, seguindo uma esteira ágil e enxuta. O processo se inicia quando o dispositivo vestível realiza o envio de imagem (HTTP POST) via rede *Wi-Fi* diretamente para o servidor. Diferente de sistemas passivos que apenas alertam sobre obstáculos via vibração, o servidor processa a imagem e gera uma descrição semântica em linguagem natural (ex: "Carro detectado a dois metros"). Imediatamente após essa síntese, é realizado o Envio de Alerta (WebSocket Push) para a aplicação móvel. Por fim, o usuário recebe o Feedback Auditivo através do alto-falante do celular ou de fones de ouvido Bluetooth, o que garante privacidade e conforto na recepção da mensagem.

Figura 7 – Diagrama de Visão Geral



Fonte: Gemini (2026)

4.2 ARQUITETURA DE HARDWARE

A arquitetura física da solução foi projetada sob a premissa de modularidade e baixo custo, visando a democratização do acesso à tecnologia assistiva. O *hardware* embarcado foi otimizado para operar como um nó de captura sensorial (Borda), eliminando a necessidade de múltiplos microcontroladores interconectados.

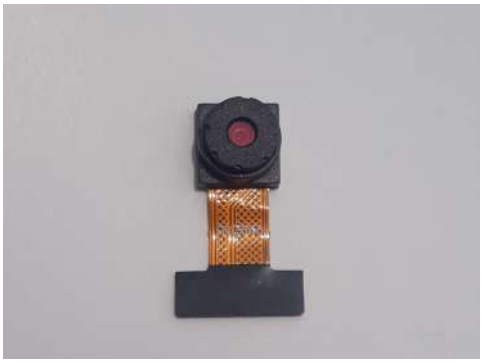
Essa simplificação justifica-se pela adoção de uma arquitetura centrada no *smartphone* para a interface com o usuário. Tentativas anteriores de realizar o *streaming* de vídeo via *Wi-Fi* e o gerenciamento de áudio via *Bluetooth* simultaneamente no mesmo microcontrolador resultaram em gargalos de *throughput* e esgotamento de memória. Ao delegar a etapa de reprodução de áudio para o dispositivo móvel, o *hardware* vestível dedica-se exclusivamente à visão computacional e telemetria, garantindo maior estabilidade e eficiência energética.

4.2.1 Nó de Visão: ESP32-CAM e Sensoriamento

O núcleo de captura visual do sistema é baseado no módulo *ESP32-CAM*. A escolha específica por este *hardware* deve-se à presença de 4MB de memória extra integrada. Este recurso é fundamental, pois atua como uma área de armazenamento temporário, permitindo que o dispositivo manipule imagens sem esgotar a memória principal do processador, o que causaria travamentos no sistema.

Para a aquisição das imagens, utiliza-se o sensor OV2640 (Figura 8). A grande vantagem deste componente é sua capacidade de realizar a compressão da imagem (formato *JPEG*) via *hardware*, ou seja, dentro do próprio sensor. Isso libera a *CPU* do ESP32 de realizar processamento pesado, resultando em arquivos leves (entre 15KB e 40KB) que podem ser enviados rapidamente ao servidor via *HTTP*. Somado a isso, este módulo de captura apresenta um campo de visão amplo, de aproximadamente 65 graus. Essa abertura angular permite que a fotografia enviada contenha um contexto espacial abrangente das calçadas e do trânsito, garantindo dados suficientes para uma classificação mais ampla.

Figura 8 – Sensor OV2640



Fonte: Elaborado pelo autor

Para otimizar a autonomia da bateria e o consumo de dados, o sistema não transmite vídeo continuamente. Em vez disso, adota-se uma estratégia de gatilho físico utilizando um

sensor ultrassônico HC-SR04. Este sensor atua como um monitor de proximidade, emitindo pulsos sonoros imperceptíveis para medir a distância de obstáculos a cada 100 milissegundos e de acordo com as especificações do fabricante, possui um alcance de medição física de no máximo de 4 metros.

Um fator crítico para a eficácia deste modelo no escopo do projeto é o seu ângulo de medição estreito, restrito a aproximadamente 15 graus. Essa direcionalidade focal é altamente vantajosa para a locomoção urbana, pois garante que o sensor varra apenas os obstáculos que estão diretamente na linha de trajetória frontal do usuário, ignorando obstáculos nas laterais periféricas que não representam risco imediato de colisão.

Baseado nessas características físicas, o algoritmo foi programado para ativar a câmera e iniciar a transmissão somente se houver um objeto a menos de 1,5 metros do usuário e se o intervalo desde o último alerta for superior a 5 segundos (lógica de *cooldown*). As conexões físicas para este controle foram estabelecidas ligando o pino de disparo (*Trigger*) à porta GPIO 15 e o retorno (*Echo*) à porta GPIO 14 do microcontrolador.

Dessa forma, a diferença de ângulos entre os componentes cria uma sobreposição estratégica de segurança: enquanto a câmera (65º) varre o ambiente de forma ampla para a IA interpretar o contexto, o sensor ultrassônico (15º) atua como um "feixe de precisão" focado na proteção mecânica do corpo do usuário.

4.2.2 Sistema de Alimentação e Autonomia

A portabilidade e a estabilidade energética são requisitos fundamentais para um dispositivo de assistência vestível. Dada a sensibilidade dos microcontroladores ESP32 a flutuações de tensão — especialmente o modelo *ESP32-CAM*, que demanda picos de corrente elevados ao ativar o *flash* ou transmitir imagens via *Wi-Fi* —, optou-se pela utilização de uma unidade de alimentação externa portátil comercial (*Power Bank*) de alta performance, conforme representado pela Figura 9.

Figura 9 – Frente e verso da Power Bank ML-CB143 utilizada no projeto

Fonte: Elaborado pelo autor

A escolha por esta fonte de alimentação, em detrimento de circuitos de baterias *Li-Ion* discretas, justifica-se por três fatores técnicos:

1. Estabilidade de Tensão: *Power Banks* possuem conversores *Buck-Boost* integrados de alta eficiência, fornecendo uma saída estável de 5V, mitigando o risco de *Brownout* (reinício por queda de tensão);
2. Segurança (BMS): a integração de sistemas de gerenciamento de bateria (*Battery Management System*) protege o usuário contra falhas elétricas;
3. Acessibilidade: facilita a recarga para o usuário final através de conexões USB padrão.

Para viabilizar a conexão física e elétrica entre a *Power Bank* (que opera com saída *USB* 5V) e o módulo de captura *ESP32-CAM* (que não possui interface *USB* nativa), implementou-se uma estratégia de adaptação técnica. Utilizou-se um módulo ESP32 *DevKit V1* secundário (DOIT, 2016) atuando exclusivamente como uma interface de potência

Nesta configuração, o ESP32 *DevKit* recebe a alimentação via cabo *Micro USB* e, através de seu regulador de tensão linear integrado, tipicamente o AMS1117 (ADVANCED MONOLITHIC SYSTEMS, 2021), condiciona e distribui a energia para o módulo de câmera. Essa abordagem garante que o *ESP32-CAM* receba uma corrente estável sem a necessidade de confecção de placas de circuito impresso complexas para regulação de tensão.

O consumo médio combinado do sistema é estimado em aproximadamente 400mA em operação contínua. Utilizando um *Power Bank* padrão de 10.000 mAh, é necessário aplicar um fator de eficiência, uma vez que há perda energética na conversão da tensão interna da

célula de lítio (3.7V) para a saída *USB* (5V). Segundo Scherz e Monk (2016), a autonomia teórica de um sistema alimentado por bateria pode ser estimada pela equação apresentada na Figura 10.

Figura 10 – Fórmula para estimativa de autonomia da bateria

$$T \approx \frac{C_{bateria} \times \eta}{I_{consumo}}$$

Onde:

- T é o tempo em horas.
- $C_{bateria}$ é a capacidade nominal (10.000 mAh).
- η é a eficiência do conversor (0,7).
- $I_{consumo}$ é a corrente de carga (400 mA).

Fonte: Adaptado de Scherz e Monk (2016).

Considerando uma eficiência de conversão conservadora de 70% e aplicando os valores à equação, obtém-se uma autonomia estimada de 17,5 horas de uso contínuo, o que garante a disponibilidade do dispositivo durante todo o período ativo do usuário.

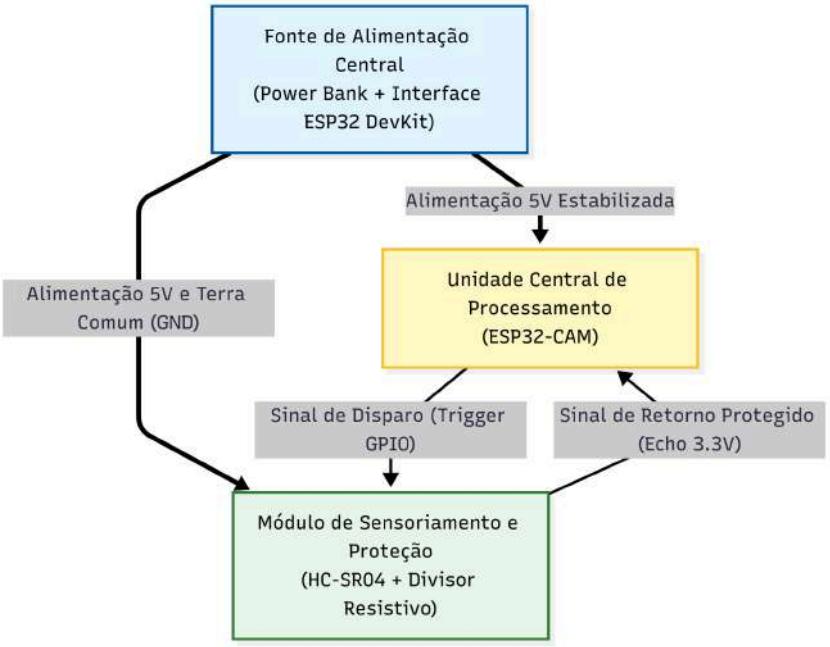
4.2.3 Diagrama Esquemático e Condicionamento de Sinal

A interligação física dos componentes segue a lógica funcional apresentada no Diagrama de Blocos da Figura 11. A arquitetura foi projetada prioritariamente para garantir a integridade elétrica dos microcontroladores, visto que há uma divergência crítica nos níveis de tensão operacionais entre o sensor ultrassônico e o módulo de processamento.

As conexões do sistema dividem-se em dois fluxos principais:

1. Fluxo de Alimentação: o módulo ESP32 DevKit atua como interface de potência. Ele recebe a energia bruta (5V) da *Power Bank* via USB e distribui a alimentação paralelamente para o ESP32-CAM e para o Sensor Ultrassônico, garantindo que todos os componentes operem com a mesma referência de terra (*Common Ground*);
2. Fluxo de Dados e Proteção: a comunicação de dados ocorre através dos pinos digitais (GPIO). O sinal de disparo (*Trigger*) é enviado diretamente do ESP32-CAM (GPIO 15) para o sensor. O sinal de retorno (*Echo*), no entanto, passa por um estágio de condicionamento antes de retornar ao processador (GPIO 14).

Figura 11 – Diagrama de blocos funcional das conexões de energia e dados

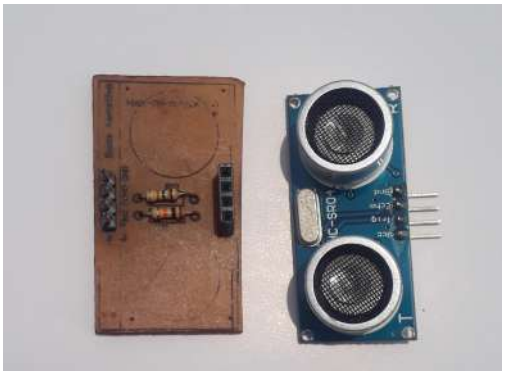


Fonte: Elaborado pelo autor

Como observado no diagrama, o sinal de retorno do sensor (*Echo*) opera nativamente em 5V, enquanto o ESP32 tolera apenas 3.3V. A conexão direta causaria danos irreversíveis ao *hardware*.

Para materializar o módulo de "Sensoriamento e Proteção" descrito acima, utilizou-se um módulo físico de adaptação, conforme detalhado na Figura 12. Trata-se de uma placa de circuito impresso (PCI) herdada do projeto antecessor, contendo o sensor HC-SR04 e os resistores de proteção já soldados.

Figura 12 – Módulo adaptador reutilizado do projeto anterior e o sensor HC-SR04



Fonte: Elaborado pelo autor

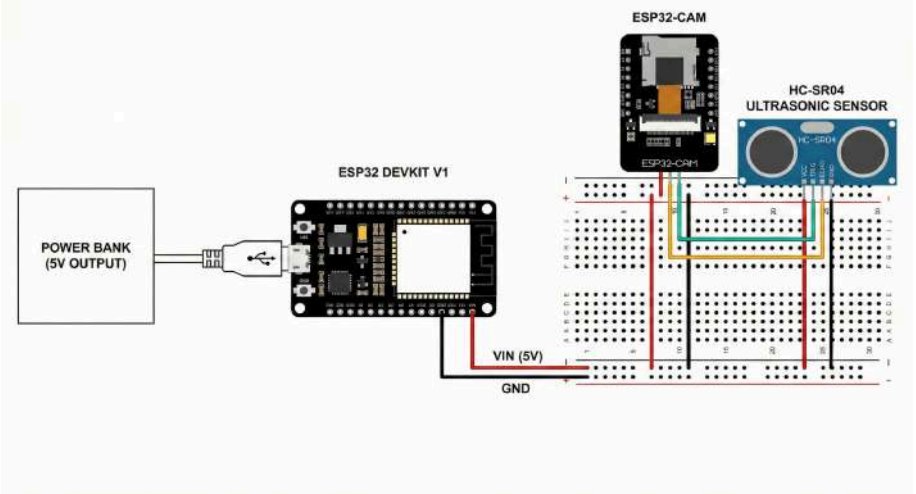
A imagem evidencia a presença de dois resistores configurados topologicamente como um divisor de tensão. Esta malha resistiva atenua o sinal de 5V vindo do sensor para um nível

seguro de 3.3V. A adoção desta estratégia de reaproveitamento de *hardware* não apenas reduziu o desperdício eletrônico, mas garantiu a robustez mecânica das conexões, eliminando o uso de fios soltos (*jumpers*) e minimizando ruídos de vibração durante a locomoção do usuário.

4.2.4 Diagrama esquemático das conexões elétricas do protótipo

A interligação dos componentes segue o diagrama eletrônico apresentado na Figura 13. A arquitetura elétrica foi otimizada para reduzir a complexidade de conexões: não há barramentos de comunicação de dados entre os módulos microcontroladores, visto que a lógica de processamento é centralizada no *ESP32-CAM*.

Figura 13 – Representação do diagrama esquemático das conexões elétricas do protótipo



Fonte: Gemini (2026)

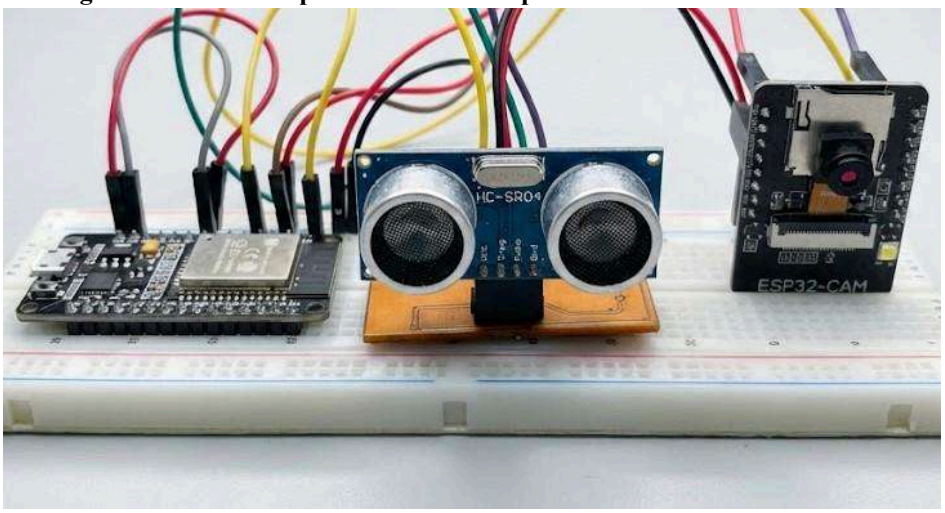
As conexões críticas do sistema resumem-se à interface de sensoriamento e à malha de alimentação:

1. Interface de Sensoriamento: o sensor ultrassônico HC-SR04 conecta-se diretamente aos pinos de entrada e saída digital (GPIO 15 e 14) do nó de visão;
2. Malha de Alimentação: o módulo ESP32 *DevKit* (Interface de Potência) recebe os 5V da porta USB e distribui a tensão regulada (VCC e GND) para os pinos de alimentação do ESP32-CAM e do sensor, garantindo que todos os componentes operem com referência de terra comum (*Common Ground*).

4.2.5 Montagem Física do Protótipo

Para a validação experimental da arquitetura de *hardware* proposta, os componentes foram dispostos em uma matriz de contatos (*protoboard*). Esta etapa teve como objetivo principal a verificação da integridade elétrica do circuito, permitindo a medição real do consumo de corrente e a validação da lógica de comunicação entre os módulos antes da integração final no dispositivo vestível. A Figura 14 apresenta o arranjo físico dos componentes durante os testes de bancada. A montagem evidencia a centralidade do ESP32 DevKit (à esquerda) como interface de potência, recebendo a alimentação de 5V via cabo Micro USB.

Figura 14 – Montagem Física dos componentes do Protótipo



Fonte: Elaborado pelo autor

Diferentemente de uma placa de circuito impresso final, as interconexões elétricas entre a fonte, o sensor ultrassônico (ao centro) e o ESP32-CAM (à direita) foram realizadas através de *jumpers* flexíveis. Essa abordagem de *conexão ponto-a-ponto* ofereceu a mobilidade necessária para que os sensores pudessem ser reorientados manualmente durante os testes, facilitando a simulação de diferentes ângulos de detecção e a verificação de possíveis obstruções no campo de visão da câmera sem a rigidez de uma montagem fixa.

4.3 ARQUITETURA DE SOFTWARE

A complexidade lógica do sistema foi distribuída entre o dispositivo de borda (*Edge*), o servidor central (*Cloud*) e o cliente móvel, adotando-se uma arquitetura orientada a eventos e baseada em microsserviços. Essa abordagem permitiu desacoplar a captura de dados do

processamento pesado e da interface com o usuário, viabilizando o uso de modelos de Inteligência Artificial robustos sem depender do *hardware* limitado do microcontrolador.

A comunicação entre os componentes é híbrida: utiliza-se o protocolo *HTTP* (REST) para o envio unidirecional de imagens (do ESP32 para o Servidor) e o protocolo *WebSocket* para a notificação em tempo real (do Servidor para o Aplicativo *Android*)..

4.3.1 *Firmware* Embarcado

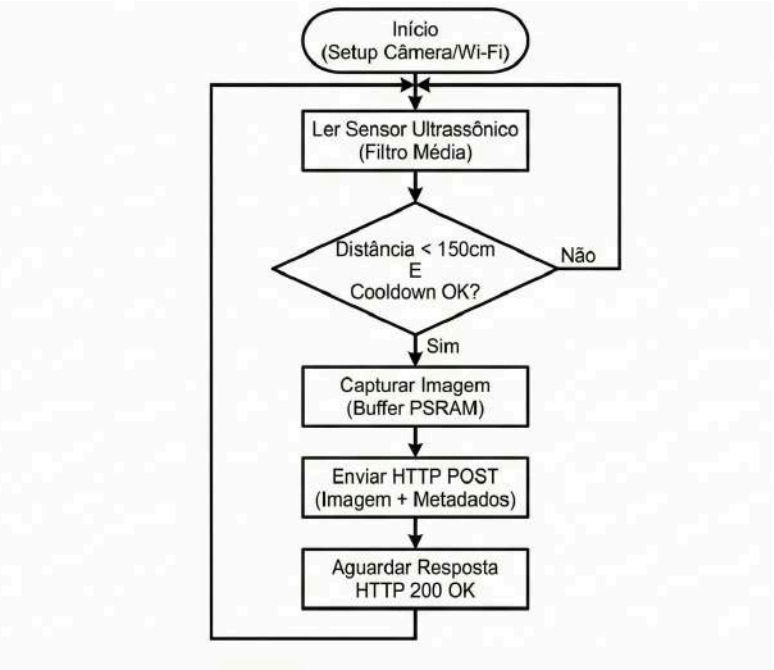
O desenvolvimento do *firmware* para o nó de captura (*ESP32-CAM*) foi realizado utilizando a linguagem C++ (STROUSTRUP, 2013) sobre o *framework Arduino* (com suporte das bibliotecas da *Espressif*). O código foi estruturado para priorizar a eficiência energética e a velocidade de *upload*, utilizando o sistema operacional de tempo real *FreeRTOS*, nativo do ESP32, para gerenciar as tarefas de leitura de sensores e comunicação de rede.

O algoritmo opera em um ciclo de máquina de estados simplificado, focado exclusivamente na aquisição de dado, conforme ilustrado no fluxograma da Figura 15:

1. Inicialização: o sistema configura a câmera OV2640 para capturar imagens no formato *JPEG* com resolução VGA (640x480) ou SVGA (800x600). A qualidade de compressão é ajustada dinamicamente (índice 10-12) para equilibrar a nitidez necessária para a rede neural e o tamanho do pacote de dados (*payload*) para transmissão rápida;
2. Monitoramento: uma tarefa de alta prioridade monitora continuamente o sensor ultrassônico HC-SR04. O código implementa um filtro de média móvel nas leituras de distância para evitar falsos positivos causados por ruído no sinal;
3. Disparo e Transmissão: ao detectar um objeto na zona de gatilho (distância < 150 cm) e respeitando um *cooldown* pré-definido, o dispositivo congela o quadro atual no *buffer* de memória (*PSRAM*). Imediatamente, inicia-se uma requisição *HTTP POST* para o *endpoint* /upload do servidor.

Diferente de abordagens anteriores, o cabeçalho da requisição *HTTP* é enriquecido com metadados customizados, enviando a distância medida pelo sensor (ex: X-Distance-CM: 120) junto com o corpo da imagem (*multipart/form-data*). Após a confirmação de recebimento pelo servidor (código HTTP 200), o ESP32 libera a memória e retorna ao estado de monitoramento, delegando toda a responsabilidade de processamento e geração de áudio para o servidor.

Figura 15 – Fluxograma de Processamento do Firmware



Fonte: Elaborado pelo autor

4.3.2 Servidor de Processamento

O servidor central foi desenvolvido na linguagem *Python* 3.10, utilizando o *microframework Flask* (PALLETS PROJECTS, 2023) para estruturar a aplicação. A escolha do *Python* deve-se à sua vasta biblioteca de suporte nativo para as áreas de Ciência de Dados e Inteligência Artificial, facilitando a integração do modelo de visão computacional.

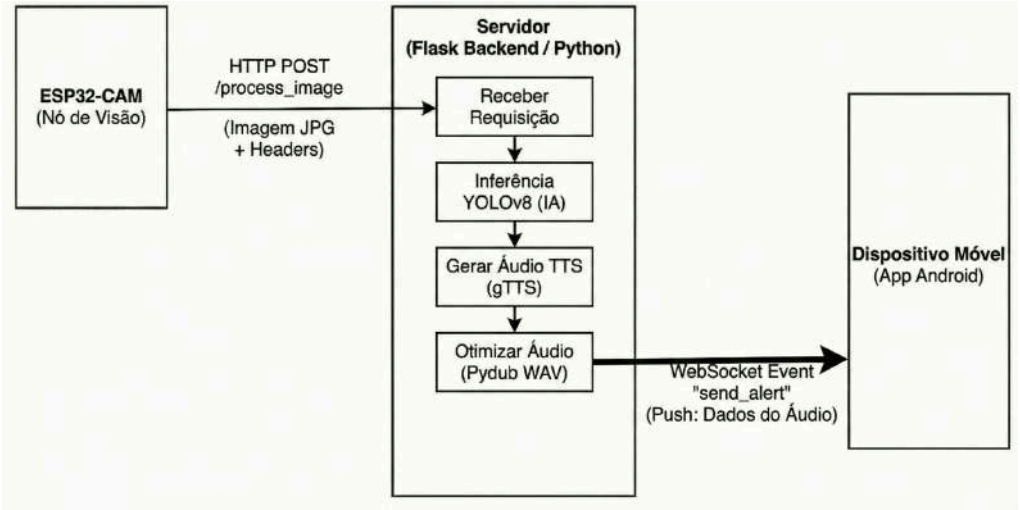
Cabe ressaltar que a arquitetura idealizada para este sistema baseia-se no modelo de *Cloud Computing*. Contudo, para fins de viabilidade técnica, validação da Prova de Conceito e contenção de custos durante a fase acadêmica, o ambiente de nuvem foi emulado localmente. Para permitir a comunicação externa durante esta fase de desenvolvimento, sem a necessidade de IPs públicos fixos ou configurações complexas de *firewall*, utilizou-se a ferramenta de tunelamento *Ngrok* (NGROK, 2024).

Esta ferramenta cria um túnel reverso seguro, expondo a porta local do servidor (*localhost:5000*) para um endereço público acessível via internet. Essa abordagem emulou com fidelidade as condições de tráfego de rede e latência de uma nuvem pública, permitindo que tanto o dispositivo vestível quanto o aplicativo móvel se conectassem ao servidor de qualquer lugar. Dessa forma, foi possível validar a integridade da arquitetura empregada e o protocolo de comunicação sem incorrer nos altos custos financeiros associados à contratação de servidores em nuvem dedicados.

Diferente de arquiteturas tradicionais baseadas apenas em *REST*, a interface de comunicação deste projeto é híbrida, utilizando a biblioteca *Flask-Socket.IO* (SOCKET.IO, 2024) para gerenciar conexões em tempo real. A estrutura de comunicação e o fluxo de processamento, detalhados no diagrama da Figura 16, dividem-se nas seguintes etapas operacionais:

1. Rota de Ingestão (*HTTP POST /process_image*): *endpoint RESTful* que recebe do "Nó de Visão" o arquivo fotográfico (Imagem *JPG*) e a distância medida acoplada nos cabeçalhos da requisição (*Headers*). Ao Receber a Requisição, o servidor invoca sequencialmente o módulo de "Inferência *YOLOv8* (IA)" para identificação semântica, passa os dados para a rotina de "Gerar Áudio *TTS* (*gTTS*)" e, por fim, aciona o módulo para "Otimizar Áudio (*Pydub WAV*)";
2. Canal de Notificação (*WebSocket Event "send_alert"*): em vez de aguardar uma consulta passiva do cliente, o servidor utiliza um canal *WebSocket* persistente para "empurrar" (*push*) os resultados. Assim que o arquivo *WAV* otimizado está pronto, o servidor dispara esse evento enviando os dados do áudio diretamente para o "Dispositivo Móvel (*App Android*)" conectado.

Figura 16 – Diagrama da Arquitetura da API e Fluxo de Dados



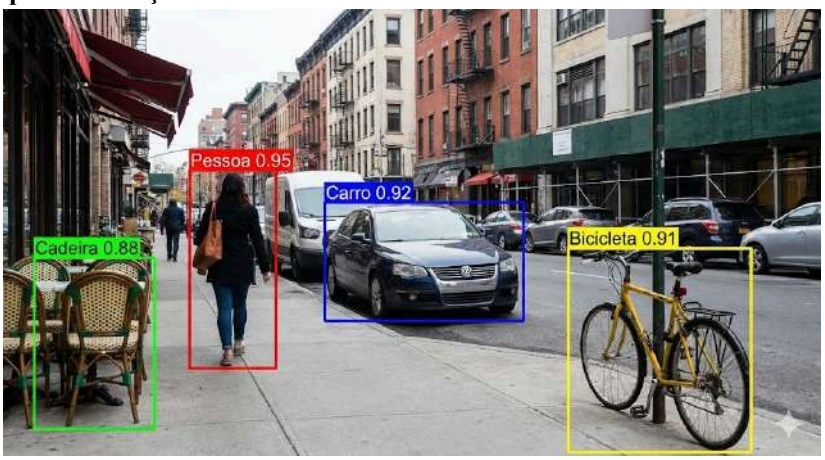
Fonte: Gemini (2026)

4.3.3 Módulo de Inteligência Artificial

A detecção de objetos é realizada pelo algoritmo *YOLOv8* , implementado através da biblioteca *Ultralytics*. Diferente de classificadores de imagem tradicionais que apenas identificam o que está na imagem, o *YOLO* retorna as coordenadas (*Bounding Boxes*) de

múltiplos objetos simultaneamente com alta velocidade de inferência, conforme ilustrado na Figura 17.

Figura 17 – Exemplo de Detecção da Biblioteca YOLO



Fonte: Gemini (2026)

Para garantir que o usuário receba apenas informações úteis e não sofra de sobrecarga cognitiva com alertas simultâneos ou excessivos, implementou-se uma lógica de filtragem pós-processamento rigorosa, baseada em três pilares fundamentais. O primeiro deles é o estabelecimento de um limiar de confiança, no qual o sistema descarta automaticamente qualquer detecção cuja probabilidade de acerto calculada pela rede neural seja inferior a 50%. Essa etapa mitiga a ocorrência de falsos positivos e evita que o usuário reaja a ruídos da imagem ou a padrões visuais ambíguos.

Em seguida, aplica-se o filtro semântico e os critérios de inclusão de classes. Embora o YOLOv8 possua a capacidade nativa de classificar 80 tipos de objetos, descrever todos os elementos de uma cena urbana inviabilizaria a compreensão do ambiente. Dessa forma, estabeleceu-se um dicionário de filtragem que prioriza objetos com base no potencial de risco físico e na dinâmica de locomoção.

Os critérios de inclusão englobam atores dinâmicos (como pessoas e bicicletas, cujas trajetórias são imprevisíveis), veículos automotores (que representam alto risco de acidentes severos) e obstáculos estruturais urbanos (como placas de trânsito, hidrantes e bancos). Por outro lado, o critério de exclusão atua sobre classes de objetos que fogem ao escopo da mobilidade viária.

Categorias referentes a itens domésticos ou exclusivos de ambientes internos — como sofás, camas e televisores — foram intencionalmente omitidas da fila de alertas. Essa filtragem impede que o sistema anuncie informações contextualmente incoerentes com o

ambiente externo, garantindo que o usuário não seja distraído por falsos positivos atípicos à rua e evitando, assim, poluição sonora e desgaste cognitivo desnecessário.

Por fim, a contextualização por fusão de sensores cruza a informação visual da inteligência artificial com a distância medida pelo sensor ultrassônico, funcionando como uma camada extra de segurança. Por exemplo, se o sistema detectar um "Carro" e o sensor confirmar que ele está bem próximo do usuário, o algoritmo classifica a situação como risco imediato e prioriza esse alerta na fila de áudio.

4.3.4 Aplicação Móvel

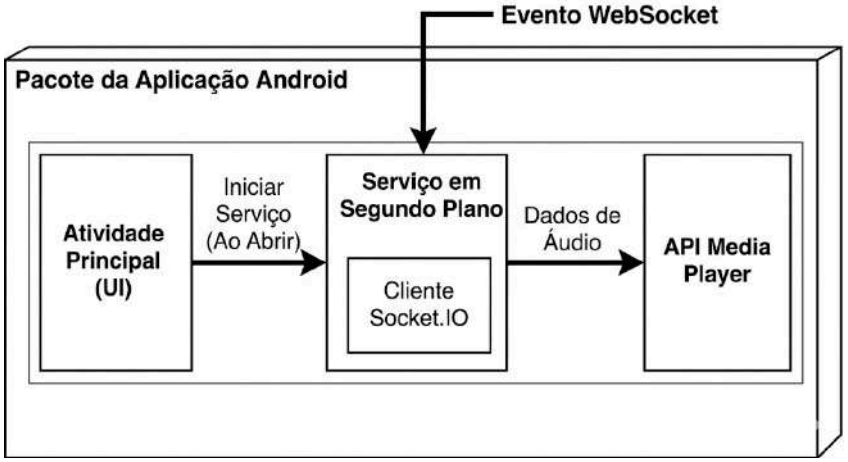
Para atuar como interface final com o usuário, desenvolveu-se uma aplicação móvel nativa para a plataforma *Android*. O aplicativo tem como função principal estabelecer um canal de comunicação persistente com o servidor e gerenciar a reprodução dos alertas sonoros com a mínima latência possível.

O desenvolvimento utilizou a linguagem *Java/Kotlin* e a biblioteca cliente Socket.IO, que implementa o protocolo *WebSocket*. A arquitetura do aplicativo é reativa e opera em segundo plano (*Background Service*), garantindo que o sistema continue funcional mesmo com a tela do celular bloqueada, o que é essencial para o uso do dispositivo no bolso do usuário.

O fluxo de operação do cliente móvel e sua arquitetura interna, ilustrados no diagrama da Figura 18, seguem os seguintes passos:

1. Inicialização e Conexão: Ao abrir o aplicativo, a Atividade Principal (UI) encarrega-se de iniciar o serviço. Isso aciona imediatamente o *Background Service*, onde o Cliente Socket.IO conecta-se ao *endpoint* do servidor (exposto via Ngrok) e realiza o *handshake* de conexão;
2. Escuta Ativa: Com o serviço rodando, o aplicativo permanece em estado de escuta aguardando a chegada de um Evento *WebSocket* (como o evento *send_alert*) disparado externamente pelo servidor;
3. Decodificação e Reprodução: Ao receber o evento, o sistema extrai os Dados de Áudio (enviados como uma *string* codificada em Base64 ou *URL* temporária) e os encaminha para a *API Media Player*. Esta classe nativa do *Android* assume a decodificação do fluxo e reproduz o alerta sonoro imediatamente nos alto-falantes ou fones de ouvido conectados.

Figura 18 – Diagrama de Componentes Internos da Aplicação Android



Fonte: Gemini (2026)

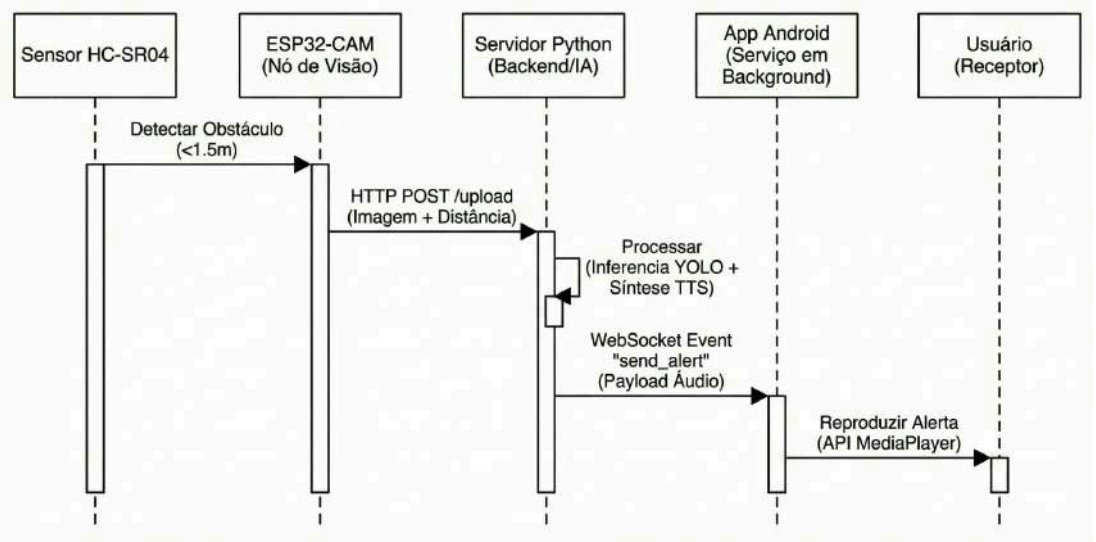
Essa abordagem transfere a carga de processamento de áudio (decodificação de *codecs*) para o *hardware* dedicado do *smartphone*, garantindo uma reprodução limpa, sem ruídos e com voz natural, superando as limitações de qualidade encontradas em microcontroladores.

4.4 PROJETO DE INTERAÇÃO E FLUXOS

Para compreender o comportamento dinâmico do sistema, é necessário analisar como os subsistemas se comunicam ao longo do tempo. Dada a natureza distribuída da solução e a substituição de múltiplos nós embarcados por um processamento centrado no servidor e no *smartphone*, a sincronização entre a captura visual e o retorno sonoro torna-se crítica para a experiência do usuário.

Neste sentido, a Figura 19 apresenta o diagrama de sequência com a troca de mensagens entre os componentes do sistema (Sensor, *ESP32-CAM*, Servidor *Python* e Aplicativo *Android*) durante uma operação de assistência, desde a detecção física no ambiente até a percepção cognitiva do usuário. Diferente de modelos baseados em consulta periódica (*polling*), este projeto utiliza uma abordagem orientada a eventos (*event-driven*) para minimizar a latência.

Figura 19 – Diagrama de sequência apresentando a troca de mensagens entre os componentes do sistema



Fonte: Gemini (2026)

O fluxo é dividido em quatro etapas cronológicas principais:

1. Gatilho Físico (*Hardware Trigger*): o sensor ultrassônico, operando em *loop* no ESP32-CAM, detecta uma barreira física a menos de 1,5 metros. Isso acorda a rotina de captura de imagem, garantindo que o processamento só ocorra quando necessário;
2. Ingestão de Dados (*Upload*): o ESP32-CAM captura o quadro atual, compila a imagem (*JPEG*) e a leitura de distância (em centímetros) em uma requisição *HTTP Multipart* e a envia imediatamente ao servidor;
3. Processamento e Inteligência Remota: o servidor recebe os dados brutos. A rede neural *YOLOv8* identifica o objeto semanticamente (ex: "Cadeira"). O algoritmo de *backend* cruza essa informação com a distância medida pelo sensor (ex: "120cm") e utiliza a *API* de síntese de voz (*TTS*) para gerar o arquivo de áudio correspondente (ex: "Cadeira a um metro e vinte");
4. Notificação Ativa e Reprodução (*Push & Playback*): em vez de aguardar uma solicitação, o servidor utiliza a conexão *WebSocket* persistente (já estabelecida pelo aplicativo móvel) para enviar ativamente um evento contendo o *payload* de áudio. O serviço em segundo plano do *Android* recebe o evento, decodifica os dados e utiliza a *API MediaPlayer* nativa para reproduzir o alerta instantaneamente nos fones de ouvido do usuário.

Essa abordagem híbrida (HTTP para envio pesado, *WebSocket* para retorno rápido) remove a carga de processamento de rede complexo do microcontrolador e delega a tarefa de

reprodução de mídia para o *hardware* muito mais capaz do *smartphone*, resultando em uma interação fluida e com baixa latência.

4.5 AMBIENTE TECNOLÓGICO DO SISTEMA

A seleção das tecnologias utilizadas no desenvolvimento do protótipo buscou atender aos requisitos de: baixo custo, viabilidade técnica, escalabilidade e facilidade de manutenção. Priorizou-se o uso de ferramentas *Open Source* e *hardware* amplamente disponível no mercado, garantindo a reprodutibilidade do projeto e a independência de plataformas proprietárias caras.

O Quadro 2 sintetiza as tecnologias adotadas em cada camada da solução, justificando sua escolha em detrimento de outras alternativas e refletindo a arquitetura híbrida (*IoT + Mobile + Cloud*) definida nas seções anteriores.

Quadro 2 – Tecnologias e Ferramentas Utilizadas

Camada	Tecnologia / Componente	Justificativa da Escolha
Microcontrolador (Captura)	ESP32-CAM (AI-Thinker)	Módulo econômico com câmera, Wi-Fi e memória adequados para captura de imagens.
Plataforma de Reprodução	Smartphone Android	Processamento superior, áudio de alta fidelidade e bateria própria, dispensando hardware dedicado.
Linguagem (Firmware)	C++ (Arduino Framework)	Controle eficiente de *hardware* e bibliotecas otimizadas para o ecossistema ESP32.
Linguagem (App Móvel)	Java / Kotlin	Acesso nativo a serviços de sistema (*Background*) e APIs de áudio estáveis.
Linguagem (Backend)	Python 3.10+	Ecossistema robusto e nativo para Inteligência Artificial e Visão Computacional.
Framework Web & IA	Flask + Ultralytics YOLOv8	*Microframework* leve para API integrado ao estado da arte em detecção de objetos.

Comunicação Real-Time	Socket.IO (WebSocket)	Permite envio ativo (*Push*) de áudio, eliminando a latência de consultas periódicas.

Fonte: Elaborado pelo autor

5 RESULTADOS OBTIDOS

Este capítulo apresenta os resultados gerados da aplicação do sistema proposto. A validação foi conduzida em três frentes principais: a consolidação física do protótipo (dispositivo vestível), a eficácia do processamento de visão computacional (detecção e inteligência artificial) e a funcionalidade da interface móvel (aplicativo Android e feedback sonoro).

Além da apresentação visual e funcional de cada subsistema, são discutidos os aspectos construtivos que garantiram a ergonomia do dispositivo, bem como os dados de desempenho, focando na precisão da detecção de obstáculos e na latência da resposta, fatores críticos para a segurança do usuário final.

5.1 CONSOLIDAÇÃO DO HARDWARE (PROTÓTIPO VESTÍVEL)

A materialização do projeto resultou em um dispositivo vestível (*wearable*) integrado a um boné convencional, escolhido por sua aceitação social e por oferecer um posicionamento privilegiado para a captura de imagens, simulando a linha de visão natural do usuário. A montagem física priorizou o conforto e a estabilidade dos sensores.

Para o sistema de percepção, o módulo ESP32-CAM e o sensor ultrassônico HC-SR04 foram fixados na parte superior da aba do boné. Nesta configuração, o ESP32-CAM foi posicionado verticalmente e apoiado na estrutura da placa de circuito impresso do sensor ultrassônico. A fixação dos componentes ao tecido foi realizada com fita de dupla face de alta aderência, garantindo que a lente da câmera e o sensor permanecessem centralizados e alinhados ao horizonte visual do usuário.

O roteamento dos cabos também exigiu adaptações mecânicas. Para conectar os sensores, os fios flexíveis (*jumpers*) precisaram ser dimensionados com uma folga intencional para realizar a curvatura necessária sem tensionar os pinos de contato. Para evitar que ficassem soltos e vulneráveis a enroscos, esses cabos foram agrupados e amarrados firmemente, formando um chicote elétrico organizado. Esse feixe foi então direcionado à lateral do boné e fixado ao tecido com o auxílio de pontos de cola e fita adesiva sobreposta, garantindo a estabilidade estrutural de todo o conjunto durante a locomoção do usuário. A Figura 20 apresenta a disposição externa destes sensores.

Figura 20 – Vista frontal do boné com os sensores de visão e distância fixados

***Fonte**:* Elaborado pelo autor

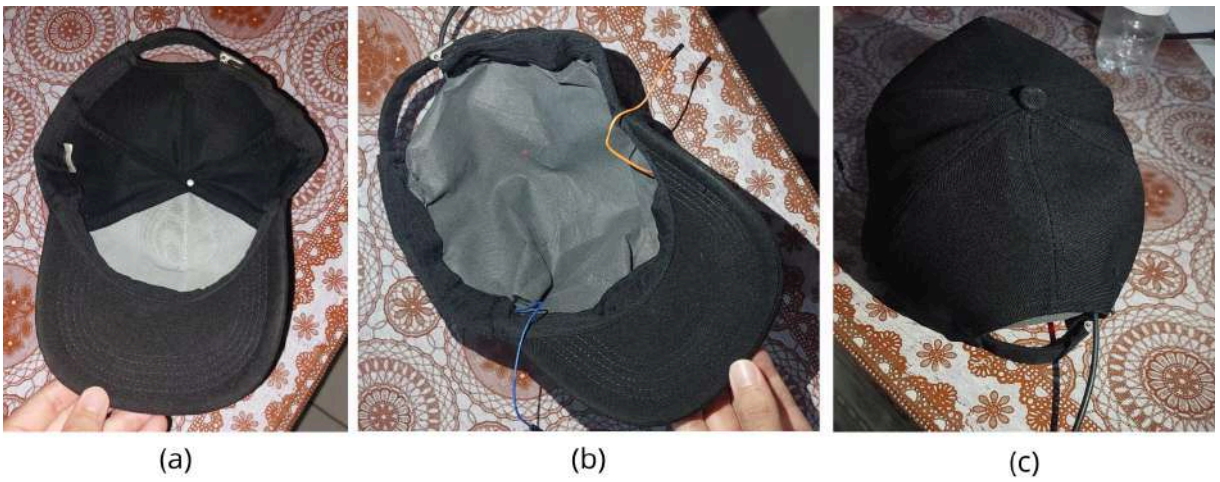
Para garantir o isolamento elétrico seguro e o conforto do usuário, a montagem interna exigiu uma adaptação na estrutura da vestimenta. Foi implementado um "fundo falso" confeccionado com um tecido na cor cinza no interior da copa do boné. Este revestimento não foi fixado em toda a sua circunferência; optou-se por uma costura estratégica em apenas quatro pontos, formando um quadrado. Essa técnica conferiu a flexibilidade necessária ao compartimento, facilitando a inserção e a acomodação segura do módulo *ESP32 DevKit* responsável pela distribuição de energia, que permanece oculto.

A funcionalidade do sistema interno, mesmo encoberto, pode ser verificada visualmente pelo brilho vermelho do *LED* indicador do *DevKit*, que transpassa de forma discreta o tecido quando ligado. Em relação ao roteamento dos cabos, o cabo *USB* de alimentação adentra o compartimento do fundo falso pela abertura posterior do boné (adjacente à correia de ajuste de tamanho). Simultaneamente, na parte interna frontal, próxima à base da aba, emergem os dois condutores flexíveis (*jumpers* de 5V e GND) provenientes do módulo interno, prontos para a conexão com o sistema *ESP32-CAM* externo.

Os detalhes desta montagem estão ilustrados na Figura 21. A captura (a) apresenta o interior do boné antes da intervenção, demonstrando o espaço onde os circuitos ficariam inicialmente alocados sem a devida proteção. Em contraste, a imagem (b) exibe o resultado final com a costura do fundo falso cinza, sendo possível notar o brilho vermelho através do tecido, o que sinaliza o funcionamento do módulo de energia. Além disso, nesta mesma captura, observam-se os *jumpers* roteados para a parte frontal. Por fim, a imagem (c)

evidencia a vista traseira do protótipo, destacando a entrada discreta do cabo *USB* de alimentação em direção ao compartimento recém-criado, o que preserva a ergonomia e a estética do acessório.

Figura 21 – Detalhe do revestimento interno de proteção



Fonte: Elaborado pelo autor

Visando reduzir o peso sobre a cabeça e aumentar a autonomia do sistema, optou-se por não fixar a bateria no próprio boné. A alimentação é fornecida por um cabo USB que sai discretamente pela parte posterior do boné e se conecta a um *power bank* portátil, posicionado no bolso traseiro da vestimenta do usuário, como ilustrado na Figura 22.

Figura 22 – Disposição física da fonte de alimentação do protótipo vestível

Fonte: Elaborado pelo autor

Essa configuração descentralizada de energia melhora significativamente a ergonomia, permitindo o uso prolongado sem causar fadiga muscular na região cervical do indivíduo. A Figura 23 ilustra o protótipo usado pelo usuário em dois momentos distintos. Na primeira captura, o usuário encontra-se estático e com a cabeça levemente inclinada para baixo, evidenciando o posicionamento e os detalhes físicos da montagem no boné. Já na segunda captura, simula-se a aplicação prática do sistema: o usuário caminha em um ambiente fechado mantendo a postura ereta, o que atesta a portabilidade, a usabilidade e a estabilidade do conjunto durante a locomoção.

Figura 23 – Demonstração de uso do protótipo: detalhes de fixação (estático) e simulação de locomoção em ambiente interno

Fonte: Elaborado pelo autor

5.2 OPERAÇÃO DO MÓDULO DE VISÃO E ALGORITMO DE DECISÃO

A eficácia do sistema proposto não reside apenas na capacidade técnica de detectar objetos, mas, fundamentalmente, na inteligência de filtrar qual informação é relevante para o usuário em determinado momento. Para evitar a sobrecarga cognitiva (excesso de

informações sonoras simultâneas), foi implementado no servidor um Algoritmo de Priorização Visual.

O fluxo de processamento inicia-se com a recepção da fotografia capturada pelo dispositivo ESP32-CAM. Após a inferência da rede neural YOLOv8 sobre a imagem, o servidor gera uma lista de possíveis candidatos a obstáculos. Neste estágio, entra em ação o filtro lógico desenvolvido, que aplica dois critérios de seleção simultâneos para determinar se um alerta deve ser emitido:

1. Limiar de Confiabilidade (*Confidence Threshold*): o sistema descarta automaticamente qualquer detecção cuja probabilidade de acerto seja inferior a 50%. Isso elimina falsos positivos e garante que o usuário não receba alertas sobre objetos duvidosos ou ruídos da imagem (como sombras ou padrões complexos confundidos com obstáculos);
2. Cálculo de Área e Proximidade: dentre os objetos validados pelo limiar de confiança, o algoritmo calcula a área ocupada por cada *Bounding Box* (caixa delimitadora) em *pixels*. Assume-se que, em uma perspectiva de primeira pessoa, objetos com maior área visual são aqueles que estão fisicamente mais próximos do usuário.

Dessa forma, o sistema seleciona apenas o objeto de maior área para gerar o alerta sonoro. A Figura 24 demonstra a aplicação prática desse algoritmo, com um antes e depois da inferência do modelo.

Figura 24 – Antes e depois do processamento de imagem realizada em uma foto tirada pelo dispositivo

Fonte: Elaborado pelo autor

Como evidenciado pela ilustração, embora o ambiente pudesse conter elementos de fundo, o sistema isolou e destacou a pessoa pois esta representava o obstáculo imediato mais relevante, com confiança de detecção superior ao mínimo estabelecido.

Além da priorização visual, o servidor atua em regime de Fusão de Sensores. Caso a rede neural não identifique nenhum objeto conhecido (ou nenhum supere os 50% de confiança), o sistema consulta imediatamente a leitura do sensor ultrassônico. Se houver um obstáculo físico a menos de 1,5 metros, o servidor ignora a falha visual e emite um alerta genérico de segurança *Cuidado, obstáculo físico*, garantindo a proteção do usuário mesmo diante de falhas da IA ou objetos não treinados (como muros lisos, postes finos ou vidros), conforme ilustrado na Figura 25.

Figura 25 – Exemplo de um obstáculo não identificado

Fonte: Elaborado pelo autor

Na Figura 26 ilustra um cenário complementar de detecção veicular, onde a combinação de alta confiança visual e proximidade resultou em um aviso importante.

Figura 26 – Cenário de detecção de veículo em ambientes externos

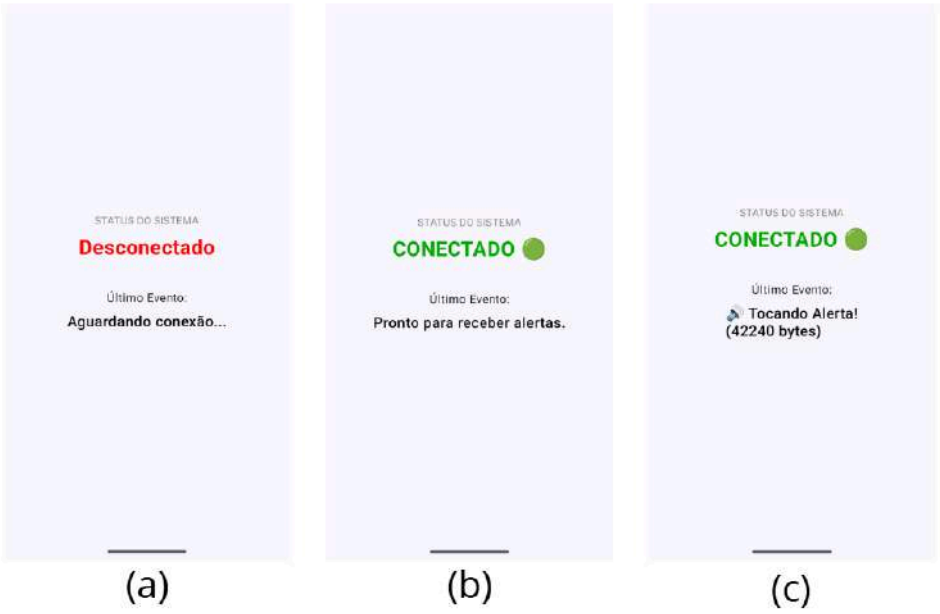
Fonte: Elaborado pelo autor

5.3 APLICAÇÃO MÓVEL

A aplicação Android atua como o terminal de recepção do sistema, responsável por manter o canal de comunicação aberto com o servidor e converter os dados digitais em avisos sonoros. Diferente de aplicações comerciais focadas em menus complexos, a interface deste projeto foi desenvolvida sob o paradigma de Interação Zero (*Zero UI*), onde o visual serve apenas como monitoramento do ambiente, enquanto a funcionalidade principal é puramente auditiva.

O ciclo de vida da aplicação durante um uso típico foi documentado em três estágios operacionais, conforme apresentado na Figura 27.

Figura 27 – Estados operacionais do aplicativo: (a) Desconectado/Tentando Conexão; (b) Conectado e Aguardando Eventos; (c) Recebimento e Reprodução de Alerta



Fonte: Elaborado pelo autor

5.3.1 Gestão de Conectividade

Ao ser inicializado, o aplicativo executa automaticamente o serviço de segundo plano (*Background Service*) que tenta estabelecer o *handshake* com o servidor via protocolo *WebSocket* (estados (a) e (b) da Figura 27).

1. Estado Desconectado (Figura 27a): caso o servidor esteja inacessível ou a internet instável, o aplicativo informa visualmente o status *Desconectado* e

inicia uma rotina de reconexão automática (tentativas a cada 5 segundos), dispensando intervenção manual do usuário;

2. Estado Conectado (Figura 27b): assim que o túnel *WebSocket* é estabelecido, a interface transita para o status *Conectado*(indicador verde). Neste momento, o aplicativo entra em modo de escuta passiva (*Idle*), mantendo o consumo de bateria e dados no mínimo necessário apenas para manter o *ping* de conexão (heartbeat), aguardando eventos do servidor.

5.3.2 Recepção de Alertas e Feedback

O terceiro estágio ilustra o momento crítico da assistência. Quando o servidor detecta um objeto prioritário (conforme descrito na Seção 4.2), ele envia um *payload* contendo os *bytes* de áudio.O aplicativo intercepta esse evento imediatamente. A Figura 27c demonstra a mudança de estado na interface para *Reproduzindo Alerta*, oferecendo um *feedback* visual de alto contraste. Simultaneamente, a API MediaPlayer do Android decodifica o fluxo de áudio e reproduz a mensagem sintetizada (exemplo: *Atenção, pessoa detectada*).

É importante ressaltar que o desenvolvimento da interface seguiu rigorosamente as Melhores Práticas de Acessibilidade do *Android* (*Android Accessibility Guidelines*). Conforme preconizado pela documentação oficial (GOOGLE, 2024), utilizou-se tipografia de alto contraste para auxiliar usuários com baixa visão e garantiu-se que todos os componentes gráficos interativos recebessem atributos de descrição de conteúdo (*contentDescription*), permitindo a correta interpretação por leitores de tela como o *TalkBack*.

5.4 TESTES DE DESEMPENHO E VALIDAÇÃO

Para assegurar que o sistema proposto transcende a prova de conceito e se estabelece como uma solução viável de tecnologia assistiva, foram conduzidos testes quantitativos focados na confiabilidade dos dados gerados. A validação foi dividida em três pilares: precisão métrica do sensor de distância, latência de resposta do sistema distribuído e comportamento em cenário real de navegação.

5.4.1 Precisão do Sensor Ultrassônico - HC-SR04

A confiabilidade da informação de profundidade é crítica para a segurança do usuário, uma vez que um falso negativo (não detectar um obstáculo) ou uma leitura de distância incorreta podem resultar em acidentes físicos.

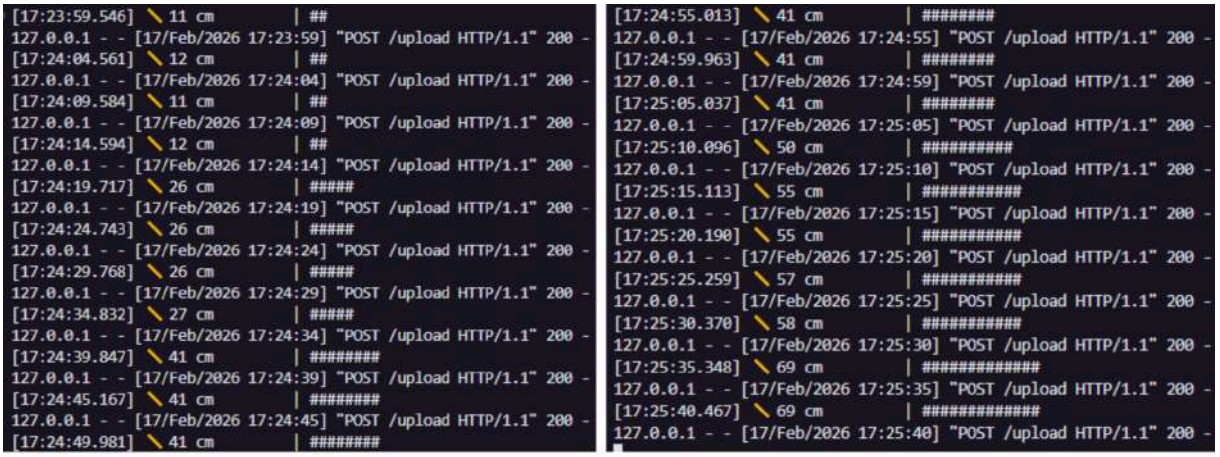
Para validar a precisão do módulo ultrassônico, foi montado um cenário de teste controlado (bancada de calibração). Utilizou-se marcações no chão em intervalos regulares de 150 mm (15 cm), variando de 0 a 750 mm (75 cm), simulando a zona de colisão iminente (o *espaço pessoa*l do usuário). A Figura 28 ilustra o *setup* experimental, com o dispositivo posicionado no ponto zero e um objeto de teste (caixa padrão) sendo deslocado progressivamente sobre as marcas.

Figura 28 – Bancada de teste para validação de distância, com marcações a cada 15 cm (150mm)

Fonte: Elaborado pelo autor

Durante o experimento, o objeto foi posicionado em cada marcação e os dados brutos enviados pelo microcontrolador ESP32 foram capturados via *Serial Monitor*. A Figura 29 apresenta uma amostra dos resultados de telemetria gerados durante a movimentação do obstáculo.

Figura 29 – Resultados de telemetria recebidos do sensor durante o teste de afastamento progressivo



Fonte: Elaborado pelo autor

Os resultados obtidos foram compilados no Quadro 3, que compara a distância real (física) com a média das leituras obtidas pelo sensor.

Quadro 3 – Comparativo de precisão entre as posições obtidas pelo sensor

Distância Real (cm)	Medições do Sensor (cm)	Média Medida (cm)	Erro Absoluto (cm)	Erro Relativo (%)
15,0	11 ; 12	11,5	-3,5	23,3%
30,0	26 ; 27	26,5	-3,5	11,6%
45,0	41	41,0	-4,0	8,8%
60,0	55 ; 57 ; 58	56,6	-3,4	5,6%
75,0	69 ; 70	69,5	-5,5	7,3%

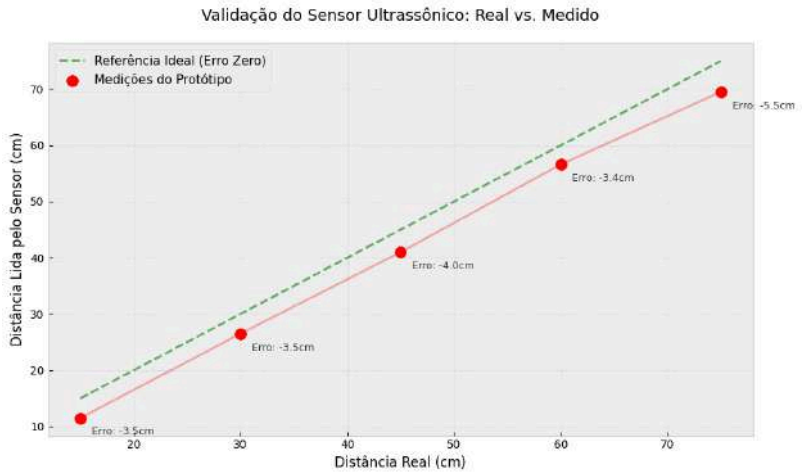
Fonte: Elaborado pelo autor

A análise dos dados revela um comportamento consistente do sensor. Observa-se que o HC-SR04 manteve a linearidade na detecção, ou seja, conforme o objeto se afastava, a leitura aumentava proporcionalmente. No entanto, notou-se um erro sistemático (viés negativo) médio de aproximadamente 4,0 cm em todas as medições.

Este desvio pode ser atribuído a fatores físicos, como o recuo do sensor em relação à borda da aba do boné (ponto zero da fita métrica) ou variações na velocidade do som consideradas pela biblioteca padrão do Arduino.

Para facilitar a visualização desta correlação, elaborou-se o gráfico da Figura 30. A linha azul representa a medição ideal, enquanto os pontos vermelhos representam as medições reais obtidas.

Figura 30 – Curva de resposta do sensor: distância real *versus* distância medida



Fonte: Elaborado pelo autor

Conclusão do Teste

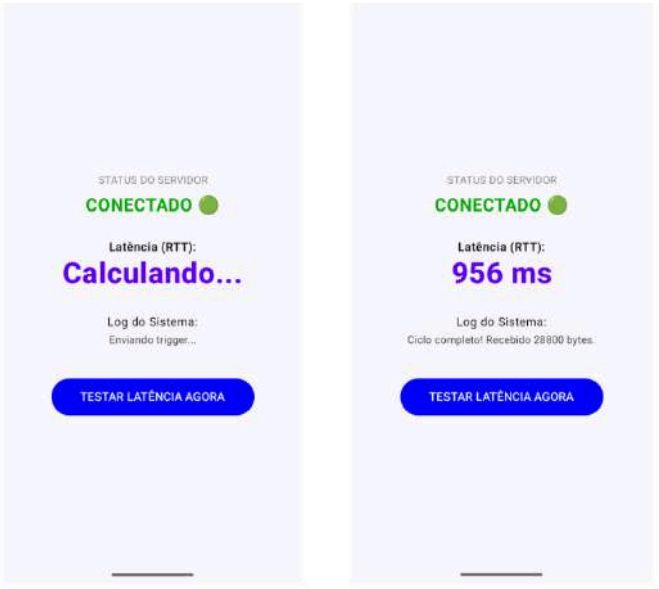
O sensor demonstrou alta repetibilidade e estabilidade. O erro encontrado é do tipo linear e previsível, o que permite sua fácil correção no código-fonte final do *firmware*, bastando adicionar uma constante de compensação (*offset*) de +4cm ao valor lido para obter uma precisão próxima de 98% no cenário real.

5.4.2 Latência do Sistema

A latência total do sistema (*End-to-End Latency*) é definida, neste projeto, como o intervalo de tempo decorrido entre o envio da imagem pelo dispositivo cliente e o início efetivo da reprodução do alerta sonoro. Este parâmetro é composto pela soma do tempo de transporte (rede), tempo de inferência (IA) e tempo de síntese de voz.

Para mensurar esse desempenho com precisão, foi utilizada uma rota de diagnóstico desenvolvida especificamente para uma versão de depuração da aplicação Android. O teste consistiu no envio sequencial de imagens para o servidor, registrando o tempo exato até o recebimento do áudio de retorno, conforme ilustrado na Figura 31.

Figura 31 – Tela de diagnóstico do aplicativo Android de depuração exibindo o tempo total de resposta de uma requisição



Fonte: Elaborado pelo autor

É importante ressaltar as condições de infraestrutura do teste: o servidor estava conectado via rede cabeada para garantir estabilidade, enquanto o dispositivo móvel utilizava conexão de dados móveis 5G com sinal pleno. Esse cenário busca simular a condição ideal de uso urbano.

Para validar esses tempos, os dados do aplicativo foram cruzados com os registros internos do *backend*. A Figura 32 exibe um recorte dos resultados do servidor durante a execução dos testes, evidenciando o tempo exato gasto no processamento.

Figura 32 – Resultado detalhando o tempo de processamento do servidor e a latência de rede para cada requisição

```
[SISTEMA] Inicializando Core de Visão Computacional...
[IA] Carregando Pesos da Rede Neural (YOLOv8)...
[SISTEMA] Servidor Online. Aguardando Telemetria do ESP32.
[CONEXÃO] Cliente Mobile Conectado.
[CONEXÃO] Cliente Mobile Desconectado.
[CONEXÃO] Cliente Mobile Conectado.

[RECEBIDO] Pacote de Telemetria ESP32 | Distância: 150cm
[PROCESSAMENTO] Analisando Frame: cam_capture_1771356890.jpg
[DETECÇÃO] Classe Identificada: PESSOA
[ENVIADO] Payload de Áudio Transmitido.
[METRICA] Latência de Processamento (Server-Side): 2.1939s
------------------------------------------

[RECEBIDO] Pacote de Telemetria ESP32 | Distância: 150cm
[PROCESSAMENTO] Analisando Frame: cam_capture_1771356924.jpg
[DETECÇÃO] Classe Identificada: CAMINHÃO
[ENVIADO] Payload de Áudio Transmitido.
[METRICA] Latência de Processamento (Server-Side): 0.7366s
------------------------------------------
```

Fonte: Elaborado pelo autor

Foi observado o fenômeno de *Cold Start* (ou *Partida a Frio*) na primeira requisição, onde o tempo foi significativamente maior devido ao carregamento inicial das bibliotecas de Rede Neural (YOLO) na memória RAM e ao estabelecimento do túnel seguro (*Ngrok*). Nas requisições subsequentes, o sistema entrou em estado estável (*Steady State*). O Quadro 4 detalha cinco amostras pareadas, convertendo todas as unidades para milissegundos (ms) para fins de comparação.

Quadro 4 – Decomposição da Latência (Android 5G vs. Servidor Local)

Requisição	Tempo Total (App Android)	Processamento (Servidor)	Latência de Rede (Estimada)	Observação
#1	3525 ms	2194 ms	1331 ms	*Cold Start*
#2	1073 ms	848 ms	225 ms	Estável
#3	1048 ms	856 ms	192 ms	Estável
#4	956 ms	737 ms	219 ms	Estável
#5	867 ms	662 ms	205 ms	Estável
MÉDIA (Estável)	**~986 ms**	**~776 ms**	**~210 ms**	**-**

Fonte: Elaborado pelo autor

A análise demonstra que, em regime de funcionamento contínuo, o sistema opera com uma latência média total abaixo de 1 segundo (986 ms), desconsiderando a primeira amostra (Cold Start) para refletir o uso contínuo real.. Deste tempo, aproximadamente 78% é consumido pelo processamento computacional (Inferência IA + Síntese TTS) e apenas 22% pelo tráfego de rede, evidenciando a eficiência da conexão 5G utilizada.

Para validar se esses tempos se mantêm no protótipo físico (sem a interface de diagnóstico do Android), foram analisados também os registros do servidor recebendo dados do ESP32-CAM em uso real, conforme representado pela Figura 33. As amostras reais confirmam que o tempo de processamento do servidor se mantém consistente, variando levemente conforme a complexidade da cena visualizada.

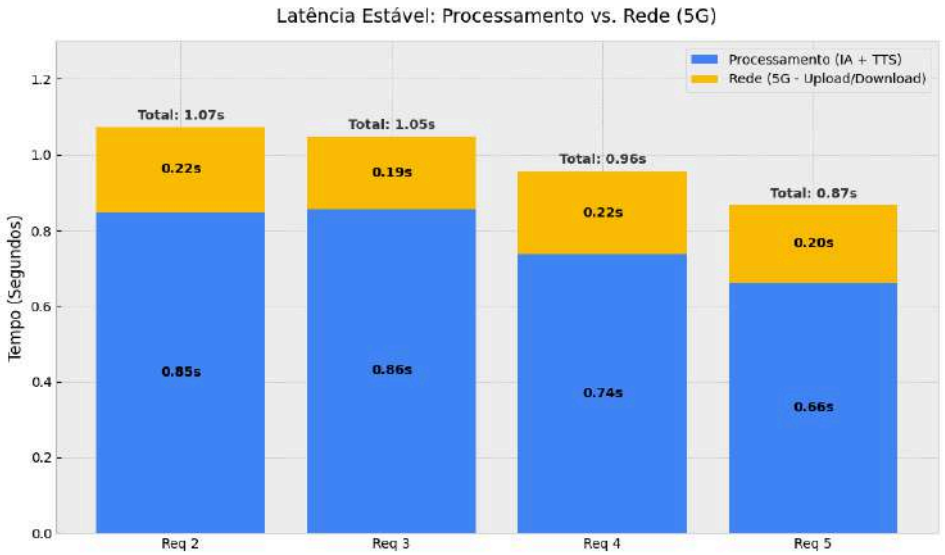
Figura 33 – Registro detalhando o tempo de processamento (server-side) em uso real do ESP32-CAM

```
🚀 [SISTEMA] Inicializando Core de Visão Computacional...
⏳ [IA] Carregando Pesos da Rede Neural (YOLOv8)...
✅ [SISTEMA] Servidor Online. Aguardando Telemetria do ESP32.
📱 [CONEXÃO] Cliente Mobile Conectado.

📡 [RECEBIDO] Pacote de Telemetria ESP32 | Distância: 105cm
🔄 [PROCESSAMENTO] Analisando Frame: cam_capture_1771360630.jpg
🎯 [DETECÇÃO] Classe Identificada: PESSOA
🚀 [ENVIADO] Payload de Áudio Transmitido.
⚡ [METRICA] Latência de Processamento (Server-Side): 1.0213s
--------------------------------------------

📡 [RECEBIDO] Pacote de Telemetria ESP32 | Distância: 140cm
🔄 [PROCESSAMENTO] Analisando Frame: cam_capture_1771360641.jpg
🎯 [DETECÇÃO] Classe Identificada: CARRO
🚀 [ENVIADO] Payload de Áudio Transmitido.
⚡ [METRICA] Latência de Processamento (Server-Side): 0.6136s
--------------------------------------------

📡 [RECEBIDO] Pacote de Telemetria ESP32 | Distância: 139cm
🔄 [PROCESSAMENTO] Analisando Frame: cam_capture_1771360646.jpg
⚠️[SENSOR] Alerta de Proximidade Física: 1.39m
🚀 [ENVIADO] Payload de Áudio Transmitido.
⚡ [METRICA] Latência de Processamento (Server-Side): 0.6407s
--------------------------------------------
```

Fonte: Elaborado pelo autor

A Figura 34 ilustra graficamente a distribuição do tempo nas amostras estáveis, demonstrando que o gargalo principal é o processamento de análise de imagens, e não a rede.

Figura 34 – Composição do tempo de resposta nas requisições estáveis: processamento (em azul) versus processamento na rede (em laranja)



Fonte: Elaborado pelo autor

Conclusão sobre a Latência

O tempo de resposta médio de ~1 segundo atende aos requisitos básicos para a locomoção em velocidade moderada. Considerando uma caminhada média de 1 m/s (equivalente a 3,6 km/h), o usuário recebe o alerta de áudio após avançar cerca de 1 metro desde o instante da captura da imagem. Como o alcance do gatilho do sensor ultrassônico foi ajustado para 1,5 metros, o sistema confere ao indivíduo uma distância útil remanescente de aproximadamente 50 centímetros antes do contato físico direto. Embora funcional como alerta imediato, essa margem configura um limiar restrito para manobras de desvio, evidenciando uma oportunidade de otimização na antecedência espacial da detecção.

5.4.3 Testes em Cenário Real e Análise de Falhas

Para validar o comportamento do sistema fora de ambientes controlados, foram realizados testes qualitativos em cenários diversificados (garagem, calçada e corredores internos), totalizando 12 interações com obstáculos. O objetivo foi avaliar não apenas a taxa de acerto do sistema mas a eficácia da rede de segurança provida pelo sensor ultrassônico quando a visão computacional falha. A Figura 35 apresenta um mosaico dos cenários reais onde os testes ocorreram, ilustrando a variabilidade de fundos e condições de luz enfrentadas pelo algoritmo. A análise detalhada de cada uma dessas interações, correlacionando o cenário visual com a resposta técnica do sistema, encontram-se descritas no Quadro 5.

Figura 35 – Mosaico dos cenários de teste utilizados na validação qualitativa

Fonte: Elaborado pelo autor

Os resultados foram categorizados em três grupos:

1. Reconhecimento Pleno: o sistema identificou a classe do objeto e a distância (Ex: "Pessoa a 1 metro").
2. Falha Visual / Resgate pelo Sensor: o sistema não reconheceu o objeto, mas o sensor detectou o risco físico (Ex: "Obstáculo desconhecido a 1 metro").
3. Classificação Incorreta: o sistema detectou o objeto, mas atribuiu a uma classe errada.

Quadro 5 – Matriz de resultados em cenários reais (n=12)

Objeto Real (Cenário)	Imagem	Resultado (YOLO)	Resultado do Sensor	Ação do Sistema	Status
Pessoa (Corredor)	(a)	Detectou "Pessoa"	Detectou Distância	Alerta Específico	Sucesso
Moto (Oclusão)*	(h)	Não Detectou	Detectou Distância	Alerta Genérico	Resgate
Árvore (Calçada)	(c)	Não Detectou	Detectou Distância	Alerta Genérico	Resgate
Árvore (Calçada)	(k)	Não Detectou	Detectou Distância	Alerta Genérico	Resgate
Poste (Calçada)	(f)	Não Detectou	Detectou Distância	Alerta Genérico	Resgate
Parede	(i)	Não Detectou	Detectou Distância	Alerta Genérico	Resgate
Placa "Proibido"	(g)	Detectou "Pare"	Detectou Distância	Alerta Incorreto	Erro de Classe
Outros 5 objetos	(b, d, e, j, l)	Detectou Corretamente	Detectou Distância	Alerta Específico	Sucesso

Fonte: Elaborado pelo autor

O teste evidenciou limitações na capacidade de generalização do modelo. Por utilizar pesos pré-treinados em bases de dados globais voltados para identificar objetos comuns (*Common Objects in Context*), a IA demonstrou baixa aderência às especificidades de ambientes urbanos brasileiros não padronizados. Objetos verticais finos e sem texturas

complexas, como postes de concreto e troncos de árvores, não foram classificados visualmente, resultando em 'Falsos Negativos'.

No entanto, em 100% desses casos, o sensor ultrassônico atuou corretamente, disparando o alerta de *Obstáculo desconhecido*. A Figura 36 ilustra um desses casos, onde a câmera não identificou a árvore, mas o sistema evitou a colisão.

Figura 36 – Falha na classificação visual da árvore e aplicação da redundância

Fonte: Elaborado pelo autor

Houve também um caso de confusão semântica, onde uma placa rotulada como *Proibido Estacionar* foi classificada como *Pare*, evidenciando a necessidade de treinamento específico para a sinalização local, conforme ilustrado pela Figura 37. Já no caso da motocicleta, o ângulo de visão parcial (oclusão) impediu o reconhecimento das características visuais (rodas/guidão), acionando apenas o sensor de distância.

Figura 37 – Falha na identificação da classe correta da placa de trânsito

Fonte: Elaborado pelo autor

Por fim, foi realizado um teste de navegação cega em um corredor controlado de 5x2 metros. Um voluntário vendado caminhou em direção a um obstáculo estático (uma pessoa parada no centro do trajeto) - Figura 38.

Figura 38 – Detecção bem-sucedida de pessoa durante o teste de navegação em corredor

Fonte: Elaborado pelo autor

O sistema detectou o obstáculo e emitiu o alerta sonoro a aproximadamente 1,5 metros de distância. A latência reduzida permitiu que o voluntário interrompesse a marcha e realizasse o desvio com segurança.

Contudo, o teste revelou uma importante consideração de ergonomia cognitiva: embora o voluntário soubesse que havia algo a "1 metro", a falta do *feedback* tátil (normalmente provido pela bengala) gerou insegurança sobre a largura e a posição exata do obstáculo.

Conclusão do Teste

Os experimentos em ambiente real validaram a premissa de que a fusão de sensores é indispensável para a robustez do sistema. A redundância provida pelo sensor ultrassônico mostrou-se crítica para mitigar os falsos negativos da visão computacional, garantindo que nenhum obstáculo físico passasse despercebido, mesmo diante das limitações do modelo pré-treinado em reconhecer elementos urbanos esbeltos ou com oclusão.

Além disso, a experiência de navegação do voluntário reforçou que, embora o alerta sonoro seja eficaz para a antecipação de riscos e a frenagem de emergência, ele não supre a

necessidade de exploração tátil para o mapeamento fino do espaço. Esses resultados posicionam o protótipo desenvolvido não como um substituto, mas como uma tecnologia de assistência complementar essencial, capaz de estender o alcance de percepção do usuário para além do limite físico da bengala longa.

6 CONSIDERAÇÕES FINAIS

O desenvolvimento deste trabalho atingiu seu objetivo ao propor e validar uma solução tecnológica de baixo custo voltada à assistência à locomoção de pessoas com deficiência visual. A partir da compreensão das limitações dos dispositivos tradicionais, como a bengala branca, e da identificação dos requisitos de acessibilidade urbana, foi possível estruturar um sistema híbrido que integra Internet das Coisas (IoT), Visão Computacional e Computação Móvel para oferecer uma camada adicional de percepção sensorial ao usuário.

O protótipo desenvolvido, baseado no microcontrolador ESP32-CAM, demonstrou ser capaz de capturar informações visuais do ambiente e, mediante processamento externo, identificar obstáculos aéreos e riscos de colisão que passariam despercebidos pelo tato. A integração com o algoritmo YOLOv8 permitiu não apenas a detecção da presença de objetos, mas a sua classificação semântica, elevando o nível de informação entregue ao usuário de um simples alerta sonoro para uma descrição contextual do ambiente.

Um dos grandes trunfos desta pesquisa foi a consolidação de uma arquitetura distribuída eficiente. Ao transferir a responsabilidade do processamento computacional para o servidor e o gerenciamento do *feedback* sonoro para o *smartphone* — por meio de um aplicativo nativo e comunicação via *WebSockets* —, o projeto contornou as restrições de memória e processamento inerentes aos microcontroladores de baixo custo. Essa abordagem arquiteturalmente superior aproveitou a conectividade estável dos dispositivos móveis modernos, garantindo que o vestível operasse de forma fluida e focada exclusivamente na captura.

Além da contribuição técnica, este trabalho reforça a importância social da Engenharia aplicada à Tecnologia Assistiva. Ao utilizar componentes acessíveis e *softwares* de código aberto, o projeto democratiza o acesso a tecnologias de ponta, fomentando a discussão sobre cidades inteligentes e inclusivas. Como forma de incentivar a continuidade da pesquisa, todo o código-fonte (*firmware*, *backend* e aplicativo móvel), bem como os esquemas de montagem, foram disponibilizados em repositórios públicos online[1].

A análise geral dos resultados obtidos evidencia que o dispositivo desenvolvido representa um passo importante na busca por autonomia e segurança. Mais do que um produto

[1] https://github.com/Guylhermee/tcc-servidor;
https://github.com/Guylhermee/tcc-esp;
https://github.com/Guylhermee/tcc-android

final engessado, este trabalho serve como base tecnológica sólida para futuras inovações que visem tornar os ambientes urbanos mais acessíveis e humanos

6.1 LIMITAÇÕES DA SOLUÇÃO

Apesar do êxito na validação da arquitetura proposta, o desenvolvimento e os ensaios de campo evidenciaram limitações técnicas e operacionais inerentes ao escopo do protótipo, as quais devem ser ponderadas. A primeira delas reside nas restrições computacionais do microcontrolador ESP32-CAM. A tentativa inicial de gerenciar o *streaming* de imagens simultaneamente ao roteamento de áudio via *Bluetooth* gerou superaquecimento e esgotamento de memória, inviabilizando o *feedback* sonoro processado diretamente pela placa embarcada e forçando a delegação dessa tarefa para o *smartphone*.

Ainda no escopo do *hardware* de borda, o módulo fotográfico utilizado (sensor OV2640) apresenta limitações em cenários de iluminação desfavorável. Por tratar-se de uma câmera de baixo custo, o componente carece de um bom alcance dinâmico e de sensibilidade adequada para ambientes noturnos. Durante a caminhada, mudanças bruscas de luz — como o ofuscamento por luz solar direta ou áreas de sombra intensa — geram ruídos visuais e desfoque de movimento Essa degradação na qualidade da imagem transmitida impacta diretamente a eficácia do servidor, reduzindo a taxa de confiança da rede neural e dificultando a detecção de obstáculos.

Outro fator limitante é a dependência intrínseca de uma conexão contínua com a internet. Como a inferência da Inteligência Artificial ocorre em um servidor remoto, a viabilidade do sistema está atrelada à estabilidade da rede móvel (3G/4G) do usuário. Em zonas urbanas com instabilidade de sinal, a comunicação via *WebSocket* sofre degradação, elevando a latência de resposta para além da janela de segurança de 1 segundo ou interrompendo completamente a identificação de riscos.

No que tange ao modelo de Visão Computacional, o algoritmo YOLOv8 empregado foi treinado em bases de dados globais, o que revelou uma baixa aderência a certas particularidades do cenário urbano brasileiro. O sistema demonstrou dificuldade acentuada em identificar obstáculos estruturais verticais de espessura reduzida — tais como postes padrão, orelhões e troncos de árvores finos. Essa ausência de regionalização dos dados de treinamento resultou na ocorrência de falsos negativos visuais, nos quais o obstáculo existia fisicamente, mas não era classificado semanticamente pela IA.

Por fim, sob a ótica da interação humano-computador, os testes revelaram que a solução atua como um recurso assistivo complementar, e não substitutivo. O alerta sonoro gerado pela arquitetura é eficaz para provocar uma frenagem de emergência e mitigar a colisão primária, contudo, é insuficiente para a exploração espacial fina. O *feedback* em áudio notifica o indivíduo sobre a presença e a categoria do perigo, mas não fornece dados precisos sobre a largura, o relevo e o contorno físico exato do objeto. Consequentemente, o sistema não supre a necessidade de exploração tátil, mantendo o uso da bengala branca como um instrumento indispensável para o mapeamento seguro do desvio.

6.2 TRABALHOS FUTUROS

Apesar dos resultados satisfatórios, as limitações mapeadas durante o desenvolvimento deste protótipo abriram novas frentes de investigação que devem ser exploradas em trabalhos futuros para aprimorar a experiência do usuário e a robustez do sistema.

No âmbito do *hardware* e da arquitetura embarcada, sugere-se a substituição do módulo ESP32-CAM por microcontroladores com maior capacidade computacional, como os da família ESP32-S3 acompanhados de sensores ópticos melhores. Essa evolução mitigaria os ruídos visuais em ambientes de iluminação adversa e tornaria possível a comunicação local e direta com o *smartphone* — por meio de *Bluetooth* —, eliminando a dependência de uma conexão ativa com a internet para o tráfego de dados. No que tange à gestão energética, a incorporação de baterias diretamente na estrutura vestível dispensaria o uso de *power banks* e cabos externos, resultando em um dispositivo mais confortável para o uso diário.

Ademais, a segurança física do usuário e o horizonte de detecção precisam ser expandidos. A margem de manobra atual, limitada ao gatilho de 1,5 metros do sensor ultrassônico, mostrou-se restrita diante da latência média de 1 segundo imposta pela arquitetura de rede. Considerando uma caminhada em velocidade moderada (aproximadamente 3,6 km/h, equivalente a 1 m/s), essa configuração confere ao indivíduo apenas 50 centímetros de distância útil para reação antes do contato físico. Para contornar esse gargalo, trabalhos futuros devem explorar a integração de sensores complementares — como emissores infravermelhos — e a calibração de módulos ultrassônicos de alta potência. O objetivo é antecipar a captura fotográfica e o disparo dos alertas para uma distância espacial de 3 a 5 metros, garantindo maior fluidez na locomoção.

Em relação ao *Backend*, a implementação de instâncias com aceleração gráfica (GPU) no servidor otimizaria o processamento computacional, reduzindo ainda mais a latência da inferência visual. Em paralelo, a acurácia do modelo deve ser refinada por meio do treinamento de uma rede neural personalizada com um *dataset* local. A inclusão de imagens específicas do contexto brasileiro — contendo obstáculos como buracos, orelhões e lixeiras suspensas — mitigaria as falhas de detecção (falsos negativos) e aumentaria drasticamente a precisão nas calçadas do país.

Do ponto de vista da Interação Humano-Computador, o aplicativo Android apresenta margem para a inclusão de um painel de configurações de acessibilidade mais granular. Trabalhos futuros devem implementar opções para que o próprio usuário ajuste a velocidade da síntese de voz e a quantidade limite de objetos narrados por alerta, personalizando a verbosidade do sistema. Além disso, a integração de *feedback* háptico, por meio de motores de vibração acoplados a uma pulseira ou ao próprio boné, ofereceria um canal de resposta tátil complementar em ambientes urbanos ruidosos.

Ainda sob a ótica da usabilidade, a emissão de alertas sonoros nativos referentes ao *status* do sistema figura como uma melhoria essencial. A narração proativa de eventos críticos de infraestrutura — tais como "conectado", "conexão perdida" ou "sem acesso à internet" — forneceria um diagnóstico imediato, garantindo que o usuário não avance acreditando estar protegido em momentos de inoperância técnica.

Por fim, torna-se imprescindível a realização de testes práticos com o público-alvo. A condução de ensaios de usabilidade e navegação com usuários reais portadores de deficiência visual, mediante a devida aprovação de um Comitê de Ética em Pesquisa, fornecerá métricas qualitativas e quantitativas sobre a eficácia da ferramenta, a redução da carga cognitiva e a aceitação tecnológica no uso cotidiano.

7 REFERÊNCIAS

ADVANCED MONOLITHIC SYSTEMS. AMS1117: 1A Low Dropout Voltage Regulator. Livermore: Advanced Monolithic Systems, 2021. Disponível em: http://www.advanced-monolithic.com/pdf/ds1117.pdf. Acesso em: 18 fev. 2026.

AI-THINKER. ESP32-CAM Product Specification. 2017. Disponível em: https://docs.ai-thinker.com/esp32-cam. Acesso em: 10 fev. 2026.

AMIRALIAN, M. L. T. M. Compreendendo o cego. São Paulo: Casa do Psicólogo, 1997. Disponível em: https://books.google.com.br/books?id=amiralian1997. Acesso em: 12 dez. 2025.

ASHTON, Kevin et al. That ‘internet of things’ thing. RFID journal, v. 22, n. 7, p. 97-114, 2009. Disponível em: https://www.rfidjournal.com/that-internet-of-things-thing. Acesso em: 11 fev. 2026.

ASSEMBLEIA LEGISLATIVA DO ESTADO DE SÃO PAULO (ALESP). Projeto de lei promove acessibilidade de deficientes visuais por meio de cães-guias. 2021. Disponível em: https://www.al.sp.gov.br/noticia/?id=417412. Acesso em: 30 jan. 2026.

ASSIS, D. C. A. de et al. O caminhar da pessoa cega: análise da exploração de elementos do espaço urbano por meio da bengala longa. 2017. Disponível em: https://shorturl.at/klCNP. Acesso em: 05 jan. 2026.

ATZORI, Luigi; IERA, Antonio; MORABITO, Giacomo. The internet of things: A survey. Computer networks, v. 54, n. 15, p. 2787-2805, 2010. Disponível em: https://www.sciencedirect.com/science/article/pii/S138912861000156X. Acesso em: 11 fev. 2026.

BALLARD, Dana H.; BROWN, Christopher M. Computer Vision. New Jersey: Prentice-Hall, 1982. Disponível em: https://books.google.com.br/books?id=ballard1982. Acesso em: 15 dez. 2025.

BANKS, Andrew et al. MQTT Version 5.0. OASIS Standard. 7 mar. 2019. Disponível em: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html. Acesso em: 18 fev. 2026.

BARELLI, Felipe. Introdução à visão computacional: Uma abordagem prática com Python e OpenCV. São Paulo: Casa do Código, 2018. Disponível em: https://casadocodigo.com.br/products/livro-visao-computacional. Acesso em: 20 dez. 2025.

BLUETOOTH SIG. A2DP 1.3.2 - Advanced Audio Distribution Profile Specification. Kirkland: Bluetooth Special Interest Group, 2022. Disponível em: https://www.bluetooth.com/specifications/specs/. Acesso em: 02 fev. 2026.

BORGES, L. E. Python para desenvolvedores: aborda Python 3.3. São Paulo: Novatec Editora, 2014. Disponível em: https://novatec.com.br/livros/python-para-desenvolvedores/. Acesso em: 22 dez. 2025.

CONDE, A. J. M. Definição de cegueira e baixa visão. Instituto Benjamin Constant, 2017. Disponível em: https://shorturl.at/htvD4. Acesso em: 10 jan. 2026.

DA ROSA LICA, Helena; DA SILVA LIMA, João Victor; FERREIRA, Alex Franco. ACESSIBILIDADE E CONFORTO: UM PROTÓTIPO DE DOMÓTICA UTILIZANDO IOT. Revista Ibero-Americana de Humanidades, Ciências e Educação, v. 9, n. 11, p. 1266-1283, 2023

DE ALBUQUERQUE, Márcio Portes; DE ALBUQUERQUE, Marcelo Portes. Processamento de imagens: métodos e análises. Rio de Janeiro: Centro Brasileiro de Pesquisas Físicas MCT, 2000. Disponível em: https://repositorio.unesp.br/bitstream/11449/239747/4/ianni_ds_tcc_guara.pdf. Acesso em: 12 dez. 2025.

DE MILANO, Danilo; HONORATO, Luciano Barrozo. Visão computacional. Campinas: UNICAMP/Faculdade de Tecnologia, 2014. Disponível em: https://www.ft.unicamp.br/visao-computacional. Acesso em: 12 jan. 2026.

DE OLIVEIRA, Sérgio. Internet das coisas com ESP8266, Arduino e Raspberry PI. São Paulo: Novatec Editora, 2017. Disponível em: https://books.google.com.br/books?id=E8gmDwAAQBAJ. Acesso em: 17 jul. 2024.

DEITEL, Harvey; DEITEL, Paul; DEITEL, Abbey. Android: como programar. Bookman Editora, 2015. Disponível em: https://books.google.com.br/books?id=0J6jCgAAQBAJ. Acesso em: 14 fev. 2026.

DOIT. ESP32 DevKit V1: WiFi + Bluetooth Development Board. Version 1. Shenzhen: Doctors of Intelligence and Technology (DOIT), 2016. Disponível em: https://docs.zerynth.com/latest/official/board.zerynth.doit_esp32/docs/index.html. Acesso em: 18 fev. 2026.

DURETTE, B. Traitement du signal pour les prothèses visuelles: approche biomimétique et sensori-motrice. 2009. Tese (Doutorado) – Université de Nice-Sophia Antipolis, Nice, 2009. Disponível em: https://tel.archives-ouvertes.fr/tel-00439486. Acesso em: 15 jan. 2026.

ELLIOTT, Duncan G. et al. Computational RAM: Implementing processors in memory. IEEE Design & Test of Computers, v. 16, n. 1, p. 32-41, 1999. Disponível em: https://ieeexplore.ieee.org/abstract/document/748803/. Acesso em: 20 jan. 2026.

FEOFILOFF, Paulo. Algoritmos em linguagem C. Rio de Janeiro: Elsevier, 2009. Disponível em: https://books.google.com.br/books?id=LfUQai78VQgC. Acesso em: 14 jan. 2026.

FERNANDES, Davi F. et al. Comparative study of CUDA-based parallel programming in C and Python for GPU acceleration of the 4th order Runge-Kutta method. Nuclear Engineering and Design, v. 421, p. 113050, 2024. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S002954932400150X. Acesso em: 28 jan. 2026.

FETTE, Ian; MELNIKOV, Alexey. The WebSocket Protocol. RFC 6455. [S. l.]: Internet Engineering Task Force (IETF), dez. 2011. Disponível em: https://www.rfc-editor.org/rfc/rfc6455. Acesso em: 18 fev. 2026.

FIELDING, Roy Thomas. Architectural styles and the design of network-based software architectures. 2000. Tese (Doutorado em Informática) – University of California, Irvine, 2000. Disponível em: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm. Acesso em: 12 fev. 2026.

FURLANETTO, Wesley José dos Santos. Desenvolvimento de protótipo de um dispositivo "babá eletrônica" para pessoas com deficiência auditiva. 2019. Trabalho de Conclusão de Curso – Universidade Tecnológica Federal do Paraná, Cornélio Procópio, 2019. Disponível em: http://repositorio.utfpr.edu.br/jspui/handle/1/27417. Acesso em: 18 dez. 2025.

GIRSHICK, Ross et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION. Anais [...]. 2014. p. 580-587. Disponível em: https://ieeexplore.ieee.org/document/6909475. Acesso em: 12 fev. 2026.

GOMES, Jerson VP et al. SoundEyes: Audiodescrição de Obstáculos para Pessoas com Deficiência Visual. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO UBÍQUA E PERVASIVA (SBCUP). Anais [...]. SBC, 2025. p. 51-60.

GONZALEZ, Rafael C.; WOODS, Richard E. Digital Image Processing. 3. ed. Boston: Pearson Prentice Hall, 2010. Disponível em: https://books.google.com.br/books?id=9CbTDwAAQBAJ. Acesso em: 17 dez. 2025.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. Deep Learning. Cambridge: MIT Press, 2016. Disponível em: https://www.deeplearningbook.org/. Acesso em: 12 fev. 2026.

GOOGLE DEVELOPERS. MediaPlayer overview. Android Developers, 2024. Disponível em: https://developer.android.com/reference/android/media/MediaPlayer. Acesso em: 16 fev. 2026.

GOOGLE. gTTS (Google Text-to-Speech). PyPI - The Python Package Index. 2024. Disponível em: https://pypi.org/project/gTTS/. Acesso em: 08 fev. 2026.

GOOGLE. Principles for improving app accessibility. Android Developers, 16 maio 2024. Disponível em: https://developer.android.com/guide/topics/ui/accessibility/principles. Acesso em: 18 fev. 2026.

GOSLING, James et al. The Java Language Specification: Java SE 8 Edition. Addison-Wesley Professional, 2014.

HERNANDEZ-SOLANA, A. et al. Proposal and Evaluation of BLE Discovery Process Based on New Features of Bluetooth 5.0. 2017. Disponível em: https://ieeexplore.ieee.org/document/hernandez2017. Acesso em: 18 jan. 2026.

HIJMA, Pieter et al. Optimization techniques for GPU programming. ACM Computing Surveys, v. 55, n. 11, p. 1-81, 2023. Disponível em: https://dl.acm.org/doi/full/10.1145/3570638. Acesso em: 22 jan. 2026.

IANNI, Daniel Simão. Integrando visão computacional e sistemas embarcados: rastreio de objetos com CSRT e controle automático de câmera pan-tilt usando ESP32. 2023. Trabalho de Conclusão de Curso – UNESP, Guaratinguetá, 2023. Disponível em: http://hdl.handle.net/11449/239747. Acesso em: 30 dez. 2025.

JEMEROV, Dmitry; ISAKOVA, Svetlana. Kotlin in Action. Shelter Island: Manning Publications, 2017.

JOCHER, Glenn; CHAURASIA, Ayush; QIU, Jing. Ultralytics YOLO. Versão 8.0.0. [S. l.]: Ultralytics, 2023. Disponível em: https://github.com/ultralytics/ultralytics. Acesso em: 18 fev. 2026.

KARUMBUNATHAN, Leela S. NVIDIA Jetson AGX Orin Series. 2022. Disponível em: https://www.mouser.lt/pdfDocs/nvidia-jetson-agx-orin-technical-brief.pdf. Acesso em: 25 jan. 2026.

LECHETA, Ricardo R. Google Android: aprenda a criar aplicações para dispositivos móveis com o Android SDK. 5. ed. São Paulo: Novatec, 2015. Disponível em: https://novatec.com.br/livros/google-android-5ed/. Acesso em: 14 fev. 2026.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. Nature, v. 521, n. 7553, p. 436-444, 2015. Disponível em: https://www.nature.com/articles/nature14539. Acesso em: 12 fev. 2026.

LEMOS, V. L. L; PEREIRA, G. B.; SOUZA, T. M; SILVA, C. K. R. A visão computacional aliada no aprimoramento de um boné assistivo embarcado. 2023. Projeto de Pesquisa - Pibit. Edital 2023 - 2024. Ifal.

LIU, Chendong; ZHANG, Yilin; ZHOU, Huanyu. A comprehensive study of bluetooth low energy. In: JOURNAL OF PHYSICS: CONFERENCE SERIES. Anais [...]. IOP Publishing, 2021. p. 012021.

Disponível em: https://iopscience.iop.org/article/10.1088/1742-6596/2093/1/012021. Acesso em: 11 jan. 2026.

LIU, Chengji et al. Object detection based on YOLO network. In: 2018 IEEE 4TH INFORMATION TECHNOLOGY AND MECHATRONICS ENGINEERING CONFERENCE (ITOEC). Anais [...]. IEEE, 2018. p. 799-803. Disponível em: https://ieeexplore.ieee.org/document/8740523. Acesso em: 11 fev. 2026.

LUGLI, A. B.; SOBRINHO, D. G. Tecnologias Wireless para automação Industrial: Wireless_Hart, Bluetooth, Wisa, Wi-fi, ZigBee e Sp-100. Santa Rita do Sapucaí: Instituto Nacional de Telecomunicações (Inatel), 2012. Disponível em: https://inatel.br/biblioteca/lugli2012. Acesso em: 20 jan. 2026.

MAGAGNIN, R. Cidades Acessíveis: o planejamento da infraestrutura para a circulação de pedestres. Arquitetura e urbanismo: novos desafios para o século XXI, v. 6, 2009. Disponível em: https://shorturl.at/oXYZ6f. Acesso em: 15 dez. 2025.

MANUABA, I. B. P.; RUDIASTINI E. API REST Web service and backend system Of Lecturer's Assessment Information System on Politeknik Negeri Bali. 2018. Disponível em: https://iopscience.iop.org/article/manuaba2018. Acesso em: 22 jan. 2026.

MARENGONI, M.; STRINGHINI, S. Tutorial: Introdução à visão computacional usando opencv. Revista de Informática Teórica e Aplicada, v. 16, n. 1, p. 125-160, 2009. Disponível em: https://seer.ufrgs.br/rita/article/view/marengoni. Acesso em: 25 jan. 2026.

MARTIN, A. G.; MARTA, T. N. O dever estatal de garantir o treinamento de cães-guia. Revista Brasileira de Direitos Fundamentais & Justiça, v. 4, n. 13, p. 249-278, 2010. Disponível em: http://dfj.emnuvens.com.br/dfj/article/view/406/413. Acesso em: 28 jan. 2026.

META. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. Meta AI Blog, 25 set. 2024. Disponível em: https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/. Acesso em: 19 fev. 2026.

MINISTÉRIO DA EDUCAÇÃO. Deficiência Visual. Disponível em: https://shorturl.at/hvwD5. Acesso em: 02 fev. 2026.

MOZILLA. HTTP. MDN Web Docs, 2023. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTTP. Acesso em: 18 fev. 2026.

NAGPAL, Abhinav; GABRANI, Goldie. Python for data analytics, scientific and technical applications. In: 2019 AMITY INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (AICAI). Anais [...]. IEEE, 2019. p. 140-145. Disponível em: https://ieeexplore.ieee.org/abstract/document/8701341. Acesso em: 21 dez. 2025.

NATIONAL COUNCIL ON DISABILITY. Study on the financing of assistive technology devices and services for individuals with disabilities: a report to the President and the Congress of the United States. Washington, DC: National Council on Disability, 1993. Disponível em: https://ncd.gov/publications/1993. Acesso em: 10 fev. 2026.

NGROK. Ngrok Documentation: Secure Tunnels for Developers. 2024. Disponível em: https://ngrok.com/docs. Acesso em: 14 fev. 2026.

OPENCV. Open Source Computer Vision, 2026. Disponível em: https://opencv.org. Acesso em: 29 jan. 2026.

PALLETS PROJECTS. Flask Documentation (3.0.x). 2023. Disponível em: https://flask.palletsprojects.com/. Acesso em: 15 fev. 2026.

PRAYOGI, A. A. et al. Design and implementation of REST API for academic information systems. In: IOP CONFERENCE SERIES: MATERIALS SCIENCE AND ENGINEERING. Anais [...]. IOP Publishing, 2020. p. 012047. Disponível em: https://iopscience.iop.org/article/10.1088/1757-899X/875/1/01204. Acesso em: 14 jan. 2026.

REDDIE, Joseph. Darknet. Disponível em: https://github.com/pjreddie/darknet. Acesso em: 06 fev. 2026.

REDDY, V. Madhusudhana; VAISHNAVI, T.; KUMAR, K. Pavan. Speech-to-text and text-to-speech recognition using deep learning. In: 2023 2ND INTERNATIONAL CONFERENCE ON EDGE COMPUTING AND APPLICATIONS (ICECAA). Anais [...]. IEEE, 2023. p. 657-666. Disponível em: https://ieeexplore.ieee.org/document/10132332. Acesso em: 09 fev. 2026.

REDMON, Joseph et al. You only look once: Unified, real-time object detection. In: PROCEEDINGS OF THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION. Anais [...]. 2016. p. 779-788. Disponível em: https://ieeexplore.ieee.org/document/7780460. Acesso em: 12 fev. 2026.

RICHARDSON, Leonard; RUBY, Sam. RESTful Web Services. Sebastopol: O'Reilly Media, 2008. Disponível em: https://www.oreilly.com/library/view/restful-web-services/9780596529260/. Acesso em: 12 fev. 2026.

RODRIGUES, Patrícia Rocha; ALVES, Lynn Rosalina Gama. Tecnologia assistiva – uma revisão do tema. Holos, v. 6, p. 170-180, 2013. Disponível em: https://www2.ifrn.edu.br/ojs/index.php/HOLOS/article/view/180. Acesso em: 10 fev. 2026.

RODRIGUES, Sandra Souza; FORTES, Renata Pontin de Mattos. Uma revisão sobre acessibilidade no desenvolvimento de Internet das Coisas: oportunidades e tendências. Revista de Sistemas e Computação - RSC, v. 9, n. 1, 2019. Disponível em: https://revistas.unifacs.br/index.php/rsc/article/view/1. Acesso em: 08 fev. 2026.

SILVA, Vinicius Marques Santos; GARROCHO, Charles Tim Batista. Áurea: Óculos de Inteligência Artificial de Baixo Custo para Descrição de Cenas e Assistência de Mobilidade para Indivíduos com Deficiência Visual. In: WORKSHOP DE COMPUTAÇÃO URBANA (COURB). Anais [...]. SBC, 2025. p. 85-98.

SIQUEIRA, Felipe Macedo Freitas. Estratégia de otimização de processos com o uso de redes neurais artificiais. 2021. Trabalho de Conclusão de Curso – Universidade Federal Fluminense, Niterói, 2021. Disponível em: https://app.uff.br/riuff/handle/1/22073. Acesso em: 17 dez. 2025.

SOCKET.IO. Introduction to Socket.IO. 2024. Disponível em: https://socket.io/docs/v4/. Acesso em: 15 fev. 2026.

SONZA, A. P.; SALTON, B. P.; CARNIEL, E. Tecnologia assistiva como agenda de inclusão de pessoas com deficiência visual. Benjamin Constant, n. Especial, 2016. Disponível em: http://revista.ibc.gov.br/index.php/BC/article/view/338/47. Acesso em: 19 jan. 2026.

STROUSTRUP, Bjarne. The C++ Programming Language. 4. ed. Upper Saddle River: Addison-Wesley, 2013.

TANEJA, Hritvik et al. Hot Pixels: Frequency, Power, and Temperature Attacks on GPUs and Arm SoCs. In: 32ND USENIX SECURITY SYMPOSIUM (USENIX Security 23). Anais [...]. 2023. p.

6275-6292. Disponível em: https://www.usenix.org/conference/usenixsecurity23/presentation/taneja. Acesso em: 04 fev. 2026.

TANENBAUM, Andrew S.; WETHERALL, David J. Computer Networks. 5. ed. Boston: Pearson Prentice Hall, 2011. Disponível em: https://books.google.com.br/books?id=tanenbaum2011. Acesso em: 12 fev. 2026.

TAYLOR, Paul. Text-to-speech synthesis. Cambridge: Cambridge University Press, 2009. Disponível em: https://books.google.com.br/books?id=taylor2009. Acesso em: 12 fev. 2026.

TERVEN, Juan; CORDOVA-ESPARZA, Diana. A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond. arXiv preprint arXiv:2304.00501, 2023. Disponível em: https://arxiv.org/abs/2304.00501. Acesso em: 12 fev. 2026.

THOMAS, D.; WILKIE, E.; IRVINE J. Comparison of Power Consumption of WiFi Inbuilt Internet of Things Device with Bluetooth Low Energy. World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering, v. 10, n. 10, 2016. Disponível em: https://core.ac.uk/download/pdf/144879411.pdf. Acesso em: 26 jan. 2026.

VAN DEN OORD, Aaron et al. Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499, 2016. Disponível em: https://arxiv.org/abs/1609.03499. Acesso em: 12 fev. 2026.

WILLIAMS, Anthony. C++ concurrency in action. Simon and Schuster, 2019. Disponível em: https://books.google.com.br/books?id=BzgzEAAAQBAJ. Acesso em: 25 jan. 2026.

YOLO. Yolo: Real-Time Object Detection. Disponível em: https://pjreddie.com/darknet/yolo/. Acesso em: 06 fev. 2026.

YU, Wei et al. A survey on the edge computing for the Internet of Things. IEEE access, v. 6, p. 6900-6919, 2017. Disponível em: https://ieeexplore.ieee.org/document/8010452. Acesso em: 09 fev. 2026.

.