CHECKLIST ERS – DOCUMENTO COMPLETO

Cada item contém um emoji ✅ indicando que a seção está presente e validada.

1. Identificação do Documento ✅

2. Resumo e Abstract ✅

3. Introdução ✅

4. Escopo do Sistema ✅

5. Minimundo ✅

6. Requisitos Funcionais ✅
- Medição de distância
- Alerta tátil
- Captura de imagem
- Reconhecimento de objetos
- Comunicação BLE
- TTS no app
- Logs, parâmetros, modo silêncio, exportação, auto-teste

7. Regras de Negócio ✅
- Priorização por distância
- Vibração por faixa
- Economia de energia- Fallback e armazenamento

8. Requisitos Não Funcionais ✅
- Latência
- Acurácia
- Autonomia
- Usabilidade
- Segurança
- Portabilidade
- Peso

9. Hardware e Arquitetura ✅
- Sensores ultrassônicos HC-SR04
- Atuador vibratório (vibracall)
- ESP32 / ESP32-CAM / ESP32-S3
- Bateria 3.7–5V com proteção
- Estrutura wearable (óculos/boné)
- Comunicação BLE e Wi-Fi opcional

- Módulos independentes (distância, visão, app)
- Requisitos de consumo, peso e autonomia

10. Casos de Uso ✅
- Detectar obstáculo
- Identificar objeto
- Emitir TTS
- Calibrar sistema- Exportar logs

11. Modelo Estrutural (Classes) ✅

12. Modelo de Interação (Sequência) ✅

13. Modelagem de Subsistemas ✅
- Sensoriamento tátil
- Visão computacional
- Aplicativo mobile

14. Glossário ✅

15. Referências ✅

Documento validado e organizado.

2. Documento de requisitos

2.1. Introdução

Mini-mundo: ambiente real e delimitado onde o PoC será validado — calçadas urbanas, áreas internas (salas, corredores), e trechos curtos de rua com fluxo moderado de pedestres. Cenários prioritários: obstáculos baixos (cadeiras, degraus), obstáculos suspensos (fios, galhos), pessoas em movimento, portas/aberturas. Condições: iluminação diurna e noturna urbana típica (iluminação pública), piso irregular e ruído sonoro ambiente moderado. Testes serão feitos com usuários com deficiência visual e/ou cegueira total, usando bengala como baseline.

Resumo do mini-mundo:

- Espaço físico: calçada pública e ambiente interno controlado (sala/corredor).
- Tipos de obstáculos: degraus, cadeiras, postes, fios, portas, pedestres/animais.
- Condições ambientais: iluminação variada, ruído ambiente, trânsito moderado.
- Uso esperado: caminhada normal (1–5 km/h), paradas comuns para inspeção.

2.2. Descrição do Propósito do Sistema

O sistema tem por finalidade **aumentar a percepção espacial** de pessoas com deficiência visual através de um wearable (óculos + pulseira vibratória + app), detectando perigos e comunicando alertas táteis e/ou sonoros em tempo real para prevenir colisões e incidentes. No PoC de 6 semanas o objetivo é demonstrar viabilidade técnica, usabilidade e métricas mínimas de desempenho (acurácia, latência, autonomia).

Objetivos práticos:

- Detectar obstáculos próximos e emitir vibração proporcional.
- Reconhecer 5 classes prioritárias com modelo leve (TFLite) e emitir TTS via app.
- Validar integração entre hardware e app com latência aceitável para navegação.

2.3. Descrição do Minimundo

Entidades do mini-mundo

- **Usuário** (pessoa com deficiência visual)
- **Óculos inteligente** (dispositivo embarcado)
- **Pulseira vibratória** (atuador tátil)
- **Smartphone Android** (app + TTS)
- **Obstáculo** (objetos físicos no ambiente)
- **Operador / Testador** (configura e observa testes)

2.4. Requisitos de Usuário

Requisitos Funcionais

Identificador	Descrição	Prioridade	Depende de
RF01	**Medição de Distância:** O sistema deve medir distância em frente ao usuário usando HC-SR04 e disponibilizar leitura atual a cada 200–500 ms.	Alta	-
RF02	**Alerta Tátil Proporcional:** Ao detectar distância abaixo de thresholds configuráveis, o dispositivo deve acionar o vibracall com intensidade proporcional (fraco/médio/alto) em ≤ 100 ms após leitura.	Alta	RF01

RF03	**Captura de Imagem:** O sistema deve capturar imagens a cada X segundos (configurável) e disponibilizar para inferência local ou envio ao app quando solicitado.	Alta	RF02
RF04	**Reconhecimento de Objetos (5 classes prioritárias):** O sistema deve inferir localmente (ou via app) as classes: fio/ramo, degrau, cadeira/obstáculo baixo, pessoa, porta. Para o PoC, acurácia mínima alvo: 70% em ambiente controlado.	Alta	RF02
RF05	**Comunicação com App (BLE):** Enviar eventos (tipo, distância, timestamp, confidence) via BLE para app Android e receber comandos de calibração.	Alta	RF04
RF06	**TTS no App:** O app deve reproduzir mensagem curta correspondente ao evento (ex.: “fio à frente, 1,2 metros”) em ≤ 400 ms após recebimento do evento	Média	RF05
RF07	**Registro/Logs:** Registrar localmente eventos com campos: timestamp, tipo, distância, confidence, bateria.	Alta	RF03
RF08	**Gestos/Botões de Controle:** Permitir ligar/desligar alertas via botão físico no wearable.	Médio	
RF09	**Fallback de processamento:** Caso a inferência local não alcance metas de latência ou acurácia, permitir offload da imagem para o smartphone para inferência.	Médio	
RF10	**Atualização de Parâmetros via App:** Permitir ajustes em thresholds de distância, sensibilidade de vibração, taxa de captura de imagem e volume de TTS pelo app.	Médio	
RF11	**Verificar dispositivos pelo App:** O aplicativo deve exibir o status da conexão BLE (Conectado/Desconectado), o nível de bateria restante do wearable (em porcentagem) e o status operacional dos sensores principais.	Médio	RF05

RF12	**Alerta de Bateria Baixa:** O sistema deve notificar ativamente o usuário (via vibração específica no wearable e alerta sonoro/TTS no app) quando o nível de bateria do wearable atingir um nível crítico (ex: 20%).	Alta	RF05, RF07
RF13	**Confirmação de Inicialização (Auto-Teste):** Ao ser ligado, o wearable deve executar um auto-teste (verificar conexão do HC-SR04 e da câmera) e emitir um sinal claro de "Sistema Pronto" (ex: uma vibração longa e duas curtas, ou um TTS "Sistema Ativo").	Alta	RF01, RF03, RF06
RF14	**Modo de Silêncio Temporário:** O usuário deve poder pausar (silenciar) temporariamente todos os alertas (táteis e sonoros) através de um comando no wearable (ex: um clique duplo no botão RF08) ou no App (via RF10).	Média	RF08, RF10
RF15	**Exportação de Logs de Teste:** O App deve possuir uma função (acessível pelo Operador/Testador) para exportar os logs de eventos (RF07) em um formato padrão (ex: CSV ou JSON) para análise de desempenho offline.	Média	RF05, RF07

Regras de Negócio

Identificador	Descrição	Prioridade	Depende de
RN01	Se a distância medida for ≤ 0.8 m → vibração alta + alerta TTS prioritário.	Alta	RF02

RN02	Se a distância estiver entre 0.8 m e 1.5 m → vibração média.	Alta	RF05
RN03	Se a distância > 1.5 m e detector de objeto identificar classe prioritária (fio/ramo) dentro do campo de visão → vibração fraca + TTS.	Média	RF05
RN04	Em caso de conflito (múltiplos eventos simultâneos), priorizar evento de menor distância; se distância similar, priorizar evento classificado como “perigo suspenso” (fio/ramo).		
RN05	odos os eventos registrados devem ter timestamp e tipo salvo localmente para posterior exportação.		
RN06	Se a bateria < 15% → notificar no app e reduzir frequência de captura de imagem para economizar energia.		
RN07	Comunicação BLE é padrão; se BLE indisponível e Wi-Fi disponível, usar Wi-Fi. Se ambos indisponíveis, armazenar eventos localmente (fallback).		

Requisitos Não Funcionais

Identificador	Descrição	Categoria	Escopo	Prioridade	Depende de
RNF01	Latência ponta-a-ponta (detecção → vibração/TTS) média ≤ 400 ms; vibração acionada em ≤ 100 ms a partir da leitura do microcontrolador.	Desempenho	Global	Alta	-
RNF02	Acurácia mínima por	Precisão	Global	Alta	-

	classe no PoC ≥ 70% em ambiente controlado; taxa de falso positivo ≤ 20%.				
RNF03	Autonomia mínima de prova ≥ 2 horas com ciclo de uso típico (captura periódica de imagem + vibração esporádica).	Energia	Autenticação	Alta	RF02
RNF04	Interface de calibração acessível no app com opções de volume, intensidade de vibração, e modo silencioso.	Usabilidade	Global	Médio	
	Disponibilidade do sistema durante sessão de teste ≥ 95% (tempo sem falhas).	Confiabilidade	Global		
	Dados de usuários e logs armazenados localmente devem ser acessíveis somente por aplicativo com autenticação local (senha/pin simples).	Segurança	Global		

	O firmware deve poder rodar em ESP32/ESP32-S3 e ter alternativa de execução parcial no smartphone.	Portabilidade	Global		
	Peso do conjunto (óculos + eletrônica) deve ser confortável para uso contínuo curto (meta ≲ 150–200 g, dependendo da estrutura).	Peso/Ergonomia	Global		

2.5. Casos de Teste

Id	Requisitos relacionados	Descrição	Pré-condições	Entradas	Fluxo	Resultados esperados	Pós-condições
CT01	RF01, NFR2	Medição HC-SR04 (Bench)	Sensor conectado ao ESP32, bancada com régua.	Leitura ±10 cm do valor real em 90% das tentativas.	Posicionar objeto a 0.5 m, 1.0 m, 1.8 m; ler valores.	Validar leitura correta do HC-SR04	
CT02	RF02	PWM e Intensidade de Vibração	Vibracall conectado e instrumentado (medir vibração qualitativa)	Três níveis perceptíveis e estáveis.	Enviar PWM para níveis fraco/médio/alto; observar motor.	Verificar intensidade proporcional ao PWM.	
CT03	RF3, RF5	Captura de Imagem (ESP32-CAM)		Imagem legível transferida com sucesso	Capturar imagem, salvar local, enviar via BLE.	Garantir snapshots e envio ao buffer.	
CT05		Testes de Integração					

3. Especificação de Requisitos

---

1. Introdução

1.1 Propósito do Documento

Este documento tem como objetivo especificar detalhadamente os requisitos funcionais e não funcionais, os modelos de caso de uso, os subsistemas e os diagramas de apoio do projeto **Óculos Inteligente de Baixo Custo para Auxílio à Locomoção de Pessoas com Deficiência Visual (LUMI)**.

O documento visa fornecer uma visão clara e completa para desenvolvedores, testadores e stakeholders, permitindo o correto entendimento, desenvolvimento e validação do sistema.

1.2 Escopo do Sistema

O sistema consiste em um **dispositivo wearable** (óculos com sensores ultrassônicos e câmera integrada) acoplado a uma **pulseira vibratória** e um **aplicativo móvel Android**.
O objetivo é auxiliar pessoas com deficiência visual a identificar obstáculos e perigos ambientais, emitindo **alertas táteis e sonoros em tempo real**.

Componentes principais:

- Módulo 1 – Sensoriamento de distância e vibração (HC-SR04 + motor vibratório).
- Módulo 2 – Câmera embarcada com IA leve (ESP32-CAM / ESP32-S3 + TensorFlow Lite).
- Módulo 3 – Aplicativo Android para calibração, TTS (text-to-speech), logs e controle.

2. Descrição Geral

2.1 Perspectiva do Produto

O produto é um **sistema embarcado integrado** que combina hardware e software.
O dispositivo será controlado por um microcontrolador ESP32, comunicando-se via BLE com um aplicativo Android.

O sistema deve operar de forma autônoma, com feedback vibratório proporcional à proximidade de obstáculos e alertas sonoros gerados no aplicativo.

2.2 Usuários e Stakeholders

Tipo de Usuário	Descrição	Nível de Acesso
Usuário Principal	Pessoa com deficiência visual que utiliza o dispositivo.	Uso direto do wearable e app.
Operador/Testador	Pessoa que auxilia em testes e calibrações.	Acesso técnico e coleta de logs.
Equipe de Desenvolvimento	Engenheiros e programadores.	Manutenção e atualização do sistema.
Orientador / Stakeholder Acadêmico	Avaliador do projeto.	Acesso a relatórios e métricas.

2.3 Restrições

- Tempo de resposta do sistema ≤ 400 ms.
- Acurácia de reconhecimento ≥ 70% (ambiente controlado).
- Autonomia mínima de 2 horas.
- Peso máximo do conjunto ≤ 200 g.

2.4 Suposições e Dependências

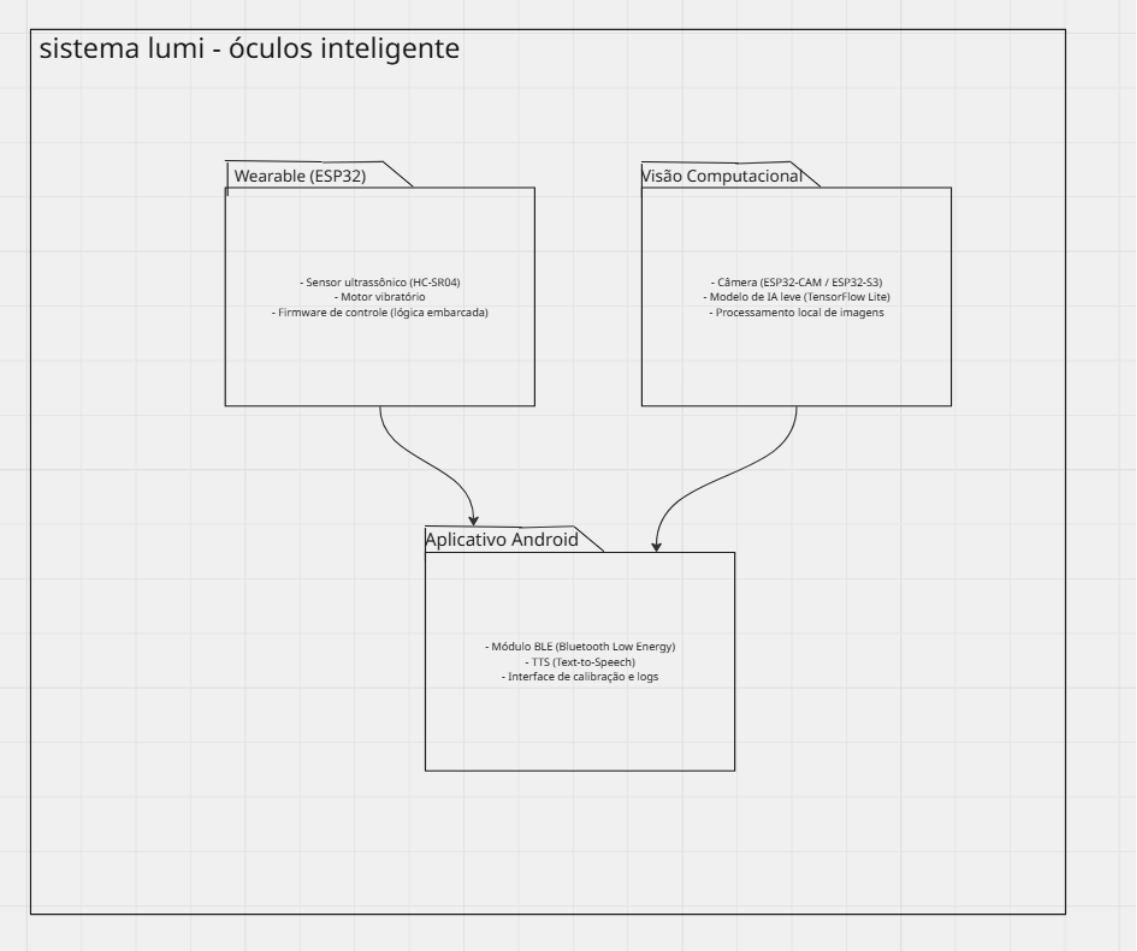
- Testes realizados em ambientes controlados (interno e externo leve).
- Smartphone Android com Bluetooth e TTS compatível.
- Alimentação via bateria recarregável de 3.7–5 V.

3. Modelagem de Subsistemas

3.1 Diagrama de Pacotes

Descrição:
O sistema é dividido em três subsistemas principais:

1. **Wearable** – Controle de sensores e atuadores (ESP32).
2. **Visão Computacional** – Captura e inferência de imagens.
3. **Aplicativo Android** – Gerenciamento de configuração, TTS e registro de logs.



(Imagem – Diagrama de Pacotes do Sistema)

3.2 Descrição dos Subsistemas

Subsistema	Descrição	Interfaces
Sensoriamento Tátil	Realiza medições de distância e aciona vibração conforme proximidade.	Sensor HC-SR04, motor vibratório, ESP32.
Visão Computacional	Reconhece objetos e classes prioritárias (fio, degrau, pessoa, porta, obstáculo).	Câmera ESP32-CAM / modelo TFLite.
Aplicativo Android	Exibe status, calibra sensores, reproduz TTS e armazena logs.	BLE, tela do usuário, TTS.

3.3 Diagrama Visual do Sistema

Esta seção apresenta a representação visual consolidada do fluxo de dados e comunicação entre os principais componentes do sistema "**Óculos Inteligente de Baixo Custo para Auxílio à Locomoção de Pessoas com Deficiência Visual**".

O Diagrama Visual do Sistema (Figura X) ilustra a interação entre os módulos embarcados (ESP32, ESP32-S3, sensores e atuadores), o aplicativo Android e o servidor intermediário. Ele demonstra o ciclo completo de detecção, processamento, transmissão e resposta tátil e sonora para o usuário final.

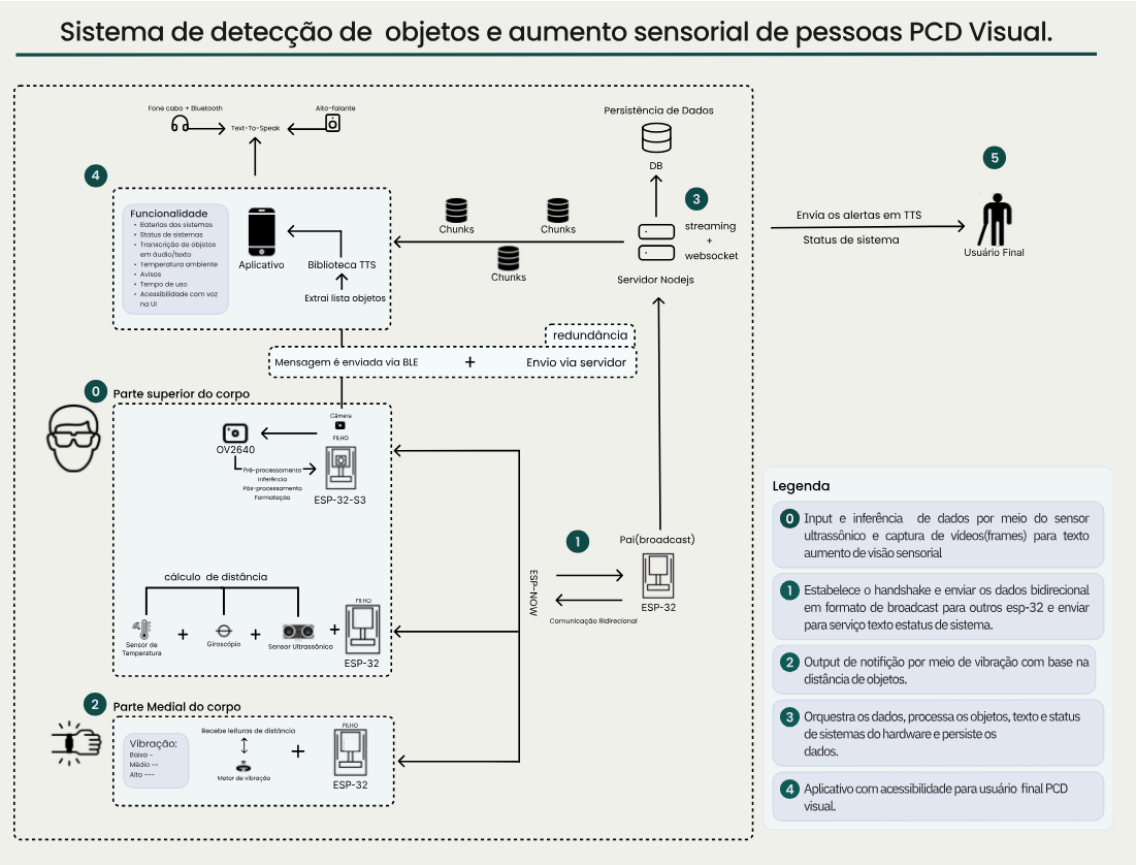
A arquitetura é organizada de forma hierárquica e redundante, permitindo que múltiplos **módulos ESP-32** atuem de forma cooperativa (modelo *pai–filho*) com comunicação bidirecional via BLE e Wi-Fi. O diagrama também evidencia o uso de streaming de vídeo, inferência local com pós-processamento, e envio de alertas TTS ao usuário.

Principais fluxos representados:

- **Entrada de dados: sensores ultrassônicos, câmera OV2640, giroscópio e sensores de temperatura;**
- **Processamento: pré-processamento, inferência e formatação dos dados;**
- **Comunicação: transmissão de informações entre ESPs e o servidor via WebSocket;**

- **Saída: feedback tátil (vibração) e sonoro (TTS);**
- **Persistência: armazenamento de logs e status de sistema no banco de dados;**

Interface: aplicativo Android com acessibilidade por voz e controle de calibração.



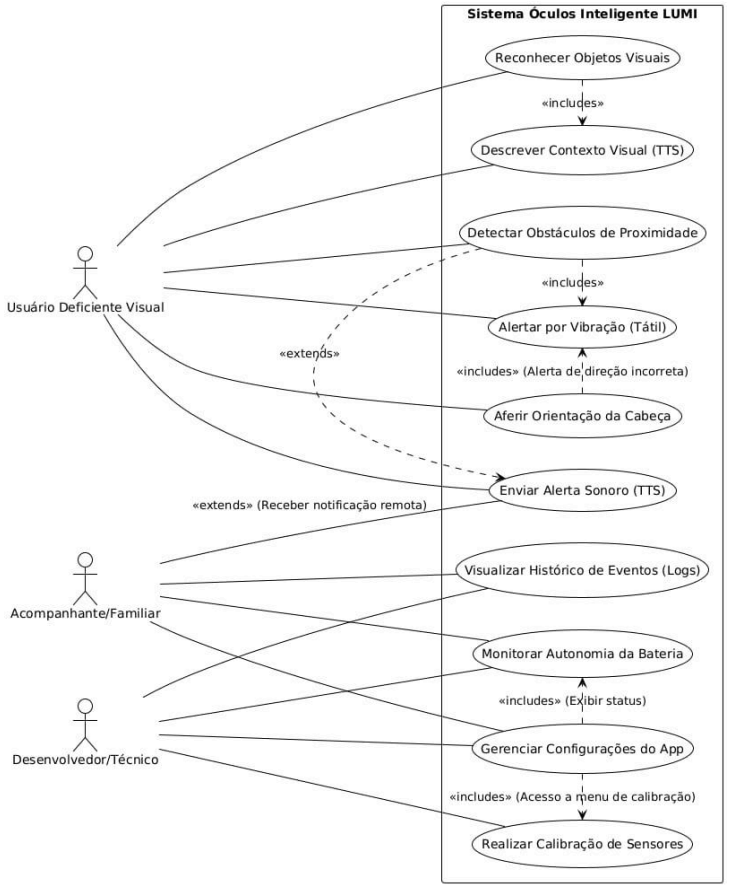
(Imagem – Diagrama de Visual)

4. Modelagem de Casos de Uso

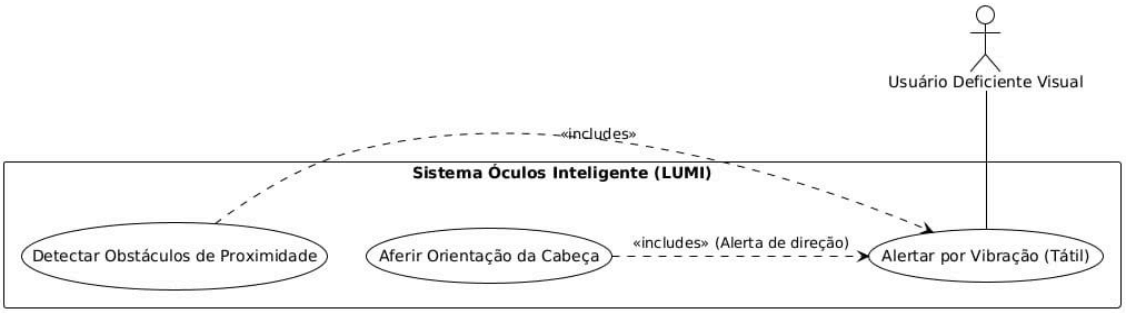
4.1 Atores

Ator	Descrição
Usuário	Utiliza o dispositivo durante a locomoção.
Aplicativo Android	Sistema auxiliar que fornece interface, TTS e calibração.
Operador/Testador	Auxilia no teste, calibração e coleta de dados.

4.2 Diagrama de Casos de Uso



(Imagem – Diagrama de Casos de Uso do Sistema)



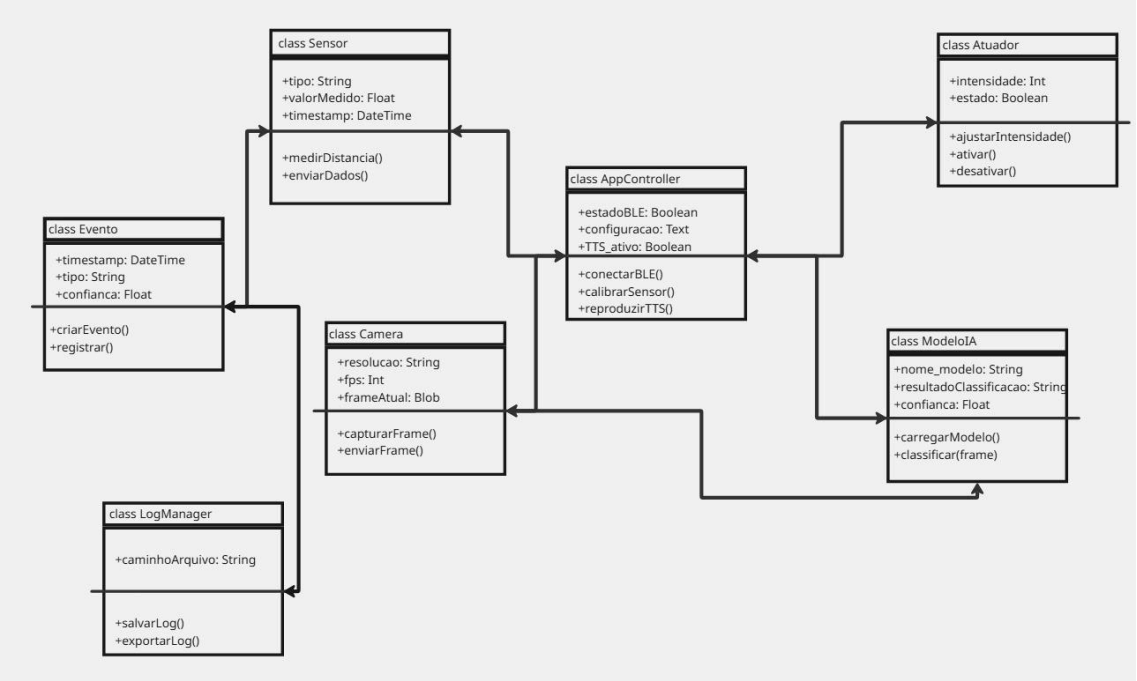
(Imagem – Diagrama de Casos de Uso do Sistema(específico))

4.3 Descrição dos Casos de Uso Principais

ID	Nome do Caso de Uso	Descrição	Atores	Requisitos Relacionados
CU01	Detectar Obstáculo	Mede a distância à frente e aciona vibração conforme proximidade.	Usuário	RF01, RF02
CU02	Reconhecer Objeto	Captura imagem e identifica classes prioritárias.	Usuário, App	RF03, RF04
CU03	Emitir Alerta TTS	O app anuncia o tipo e distância do obstáculo.	App	RF06
CU04	Calibrar Sistema	Ajusta thresholds de distância, vibração e volume.	Operador	RF10
CU05	Exportar Logs	Exporta registros de eventos para análise.	Operador	RF15

5. Modelo Estrutural

5.1 Diagrama de Classes (ou Entidade-Relacionamento)



(Imagem – Diagrama de Classes do Sistema)

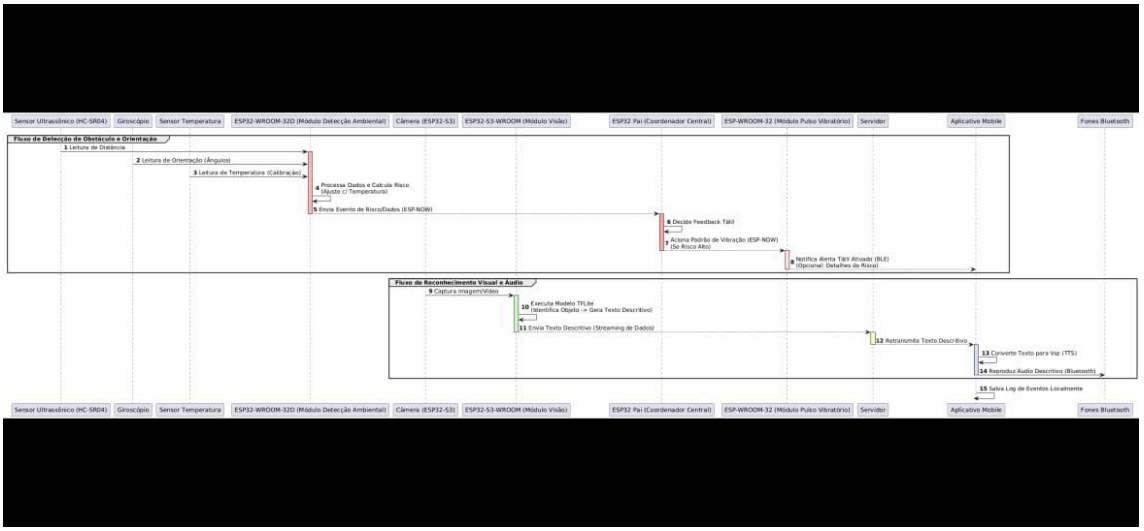
Descrição textual das principais classes:

- **Sensor**: armazena dados de medição de distância.
- **Atuador**: controla intensidade de vibração.
- **Câmera**: captura e envia frames para inferência.
- **ModeloIA**: executa classificação TFLite.
- **AppController**: gerencia BLE, calibração e TTS.
- **Eventos** registro de alertas (timestamp, tipo, confiança).

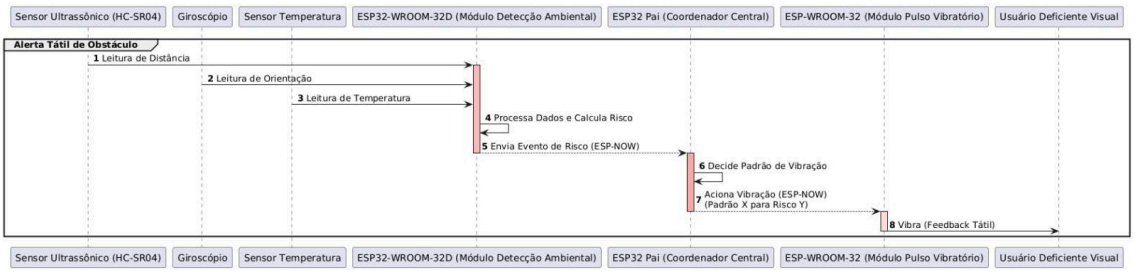
- **LogManager**: salva e exporta logs.

6. Modelo de Interação

6.1 Diagrama de Sequência – Fluxo de Detecção e Alerta



(Imagem – Diagrama de Sequência)



(Imagem – Diagrama de Sequência(específico))

Fluxo Descrito:

1. Sensor detecta obstáculo.
2. ESP32 processa a distância.
3. Se limiar ultrapassado, aciona vibração.
4. Envia evento ao App via BLE.
5. App converte evento em mensagem TTS.
6. Log é salvo localmente.

7. Requisitos Funcionais e Não Funcionais (Resumo)

7.1 Requisitos Funcionais

Principais requisitos já definidos no DR:

- RF01 – Medição de Distância
- RF02 – Alerta Tátil
- RF03 – Captura de Imagem
- RF04 – Reconhecimento de Objetos
- RF05 – Comunicação BLE
- RF06 – Alerta TTS
- RF07 – Registro de Logs
- RF10 – Ajuste de Parâmetros
- RF15 – Exportação de Logs

7.2 Requisitos Não Funcionais

- **Latência:** ≤ 400 ms
- **Acurácia:** ≥ 70%
- **Autonomia:** ≥ 2 horas
- **Peso:** ≤ 200 g
- **Usabilidade:** padrões de vibração distinguíveis
- **Segurança:** logs acessíveis apenas no app autenticado

8. Glossário

Termo	Definição
TTS (Text-to-Speech)	Recurso de voz do app que lê mensagens.
BLE (Bluetooth Low Energy)	Protocolo de comunicação entre o óculos e o app.
ESP32 / ESP32-CAM	Microcontroladores usados no protótipo.
TFLite (TensorFlow Lite)	Framework de IA leve para inferência embarcada.
PoC (Proof of Concept)	Prova de conceito – protótipo experimental de curta duração.
Vibracall	Atuador tátil de vibração usado no wearable.

10. Referências

- Documentação Técnica ESP32 (Espressif, 2024).
- TensorFlow Lite Micro Guide.
- Android BLE API Reference.
- Diretrizes de Acessibilidade W3C / WCAG 2.1.

MANUAL DA EQUIPE – PROJETO ÓCULOS INTELIGENTE

1. Objetivo do Manual

Este manual define **funções, responsabilidades, regras internas, fluxos de trabalho e procedimentos padrão** para a equipe do projeto “Óculos Inteligente de Baixo Custo”.
Serve como referência para manter **organização, alinhamento e consistência** durante todo o ciclo de desenvolvimento.

2. Estrutura da Equipe e Papéis

Product Owner – Fabio Brasileiro

- Define prioridades do backlog.
- Aprova critérios de aceitação.
- Alinha requisitos com stakeholders.

Scrum Master – Heloísa Cativo

- Garante que o time siga Scrum.
- Remove impedimentos.
- Facilita reuniões e boa comunicação.

Tech Lead / Arquiteto – José Filho

- Define arquitetura do sistema.
- Aprova decisões técnicas.
- Orienta devs sobre boas práticas.

Desenvolvedores Backend – Victor Fernandes, Victor Abreu, Guilherme Reis

- Implementam lógica embarcada e APIs.
- Integram sensores, IA e BLE.
- Criam testes de unidade.

Desenvolvedores Frontend/Android – Luis Oliveira, Eric Silva, João Pereira

- Desenvolvem app Android.
- Criam telas, acessibilidade e TTS.
- Integram BLE e calibradores do sistema.

QA / Testador – Mateus Miranda

- Cria planos e casos de testes.
- Valida módulos e sprint releases.
- Registra bugs e acompanha correções.

Config Manager – José Filho

- Cuida de versionamento (Git).
- Organiza branches e releases.
- Garante integridade do repositório.

CCB – Victor Abreu

- Aprova solicitações de mudança.
- Avalia impacto técnico e de cronograma.

3. Regras Internas da Equipe

3.1 Comunicação

- Canal principal: **WhatsApp + Discord**
- Reuniões:
    - **Daily:** 15 min
    - **Revisão de Sprint:** ao final de cada sprint
    - **Retrospectiva:** após review
- Chamadas urgentes: marcar @todos no Discord.

3.2 Regras de Commit (Git)

Sempre seguir o padrão:

```
type: descrição curta

- o que foi feito
- motivo da mudança
```

Types permitidos:
`feat`, `fix`, `docs`, `test`, `refactor`, `perf`, `style`

Branch pattern:

- `main` → versão estável
- `develop` → versão de integração
- `feature/nome` → novas funções
- `hotfix/nome` → correções urgentes

3.3 Boas Práticas de Código

- Nome de variáveis claro e objetivo.
- Um commit por tarefa.
- Evitar código duplicado.
- Sempre documentar funções principais.
- Criar testes unitários onde possível.

4. Procedimentos Operacionais

4.1 Desenvolvimento de Hardware

- Validar sensores antes de integrar.
- Testar HC-SR04 em bancada com régua.
- Testar vibracall com PWM em três níveis.
- Testar câmera com snapshot antes da IA.
- Registrar consumo da bateria a cada iteração.

4.2 Desenvolvimento do Firmware (ESP32)

Fluxo padrão:

1. Implementar leitura de sensores.
2. Garantir tratamento de erros.
3. Enviar eventos via BLE.
4. Testar latência e estabilidade.

5. Integrar com app Android.

Checklist interno:

- Distância
- Vibração proporcional
- Captura de imagem
- Inferência TFLite
- BLE

4.3 Desenvolvimento do App Android

- Criar telas simples e acessíveis.
- Implementar TTS responsivo.
- Garantir sincronização BLE.
- Incluir calibração (thresholds e vibração).
- Feedback por voz e botões grandes.

4.4 Testes

QA deve validar:

- Funcionalidade dos sensores
- Reconhecimento de objetos
- Latência ponta-a-ponta
- Uso com 1–5 usuários reais
- Exportação de logs
- Estabilidade da conexão BLE

5. Organização do Repositório

```
/docs
   ERS/
   diagramas/

/hardware
   esp32/
   esquemáticos/

/app
   android app/

/firmware
   modules/
   ble/
   vision/

/tests
   unit/
   integration/
```

6. Solicitações de Mudança (CCB)

Qualquer alteração no sistema deve seguir:

1. Abrir issue: “Solicitação de Mudança”.
2. CCB avalia impacto.
3. Decisão registrada no GitHub.
4. Implementação inicia somente após aprovação.

7. Fluxo de Sprint (Scrum)

Duração: 7 dias

Dia 1 – Planejamento
Definição do backlog da sprint.

Dias 2–6 – Execução
Desenvolvimento, testes e integração.

Dia 7 – Review + Retrospectiva

8. Regras de Qualidade

- Latência ≤ 400 ms
- Acurácia ≥ 70%
- Autonomia ≥ 2 horas
- BLE estável
- Vibração perceptível
- Logs funcionando

9. Checklist Final da Sprint (interno)

- Código testado
- Testes feitos pelo QA
- Documentação atualizada

- Bugs corrigidos
- Branch mergeada corretamente
- Versão registrada

📌 10. Contatos e Papéis Rápidos

- **PO:** decisões de produto
- **SM:** processos ágeis
- **Tech Lead:** decisões técnicas
- **QA:** testes
- **Config Manager:** Git
- **CCB:** aprova mudanças

Guia Passo a Passo para o Desenvolvimento do Projeto "Óculos Inteligente de Baixo Custo"

Este documento apresenta um guia passo a passo detalhado para a execução do projeto "Óculos Inteligente de Baixo Custo para Auxílio à Locomoção de Pessoas com Deficiência Visual", conforme o **Modelo de Ciclo de Vida Incremental com Sprints Semanais** definido no Documento de Análise de Requisitos. O projeto tem uma duração total de 6 semanas e visa a entrega de um Protótipo de Prova de Conceito (PoC) funcional.

1. Estrutura de Governança e Papéis

A execução do projeto é baseada em uma estrutura de equipe ágil, com papéis e responsabilidades claramente definidos, conforme o organograma e a matriz de responsabilidades do documento de origem.

Papel	Responsável	Foco Principal
Product Owner (PO)	Fabio Brasileiro	Maximizar o valor do produto, gerenciar e priorizar o *Backlog* de funcionalidades.
Scrum Master (SM)	Heloisa Cativo	Garantir a adesão ao processo *Scrum*, remover impedimentos e facilitar as reuniões.
Tech Lead / Arquiteto	Jose Filho	Definir a arquitetura do sistema (Módulos 1, 2 e 3), garantir a integridade técnica e orientar os desenvolvedores.
Desenvolvedores (Devs)	Luis Oliveira, Eric Silva, João Pereira, Victor Fernandes, Victor Abreu, Guilherme Reis	Implementação do *firmware* embarcado, desenvolvimento do aplicativo Android e integração dos módulos.
QA/Testador	Mateus Miranda	Planejar e executar testes funcionais, de usabilidade e de desempenho (latência e acurácia).

2. Plano de Execução Incremental (6 Sprints)

O desenvolvimento é dividido em seis *sprints* semanais, cada uma focada na entrega de um incremento funcional do sistema. O objetivo é validar hipóteses de negócio e incorporar *feedback* de forma contínua.

Sprint 1: Módulo Tátil e Base de Hardware

Foco: Estabelecer a base de *hardware* e a funcionalidade primária de detecção de distância e *feedback* tátil.

Atividade	Requisitos Chave	Produto de Entrega
1.1. Implementação do Módulo 1	RF01 (Medição de Distância), RF02 (Alerta Tátil Proporcional)	Protótipo funcional do Módulo 1 (ESP32/Pico + HC-SR04 + Vibracall) capaz de medir distância e acionar vibração proporcional em ≤ 100 ms.
1.2. Auto-Teste de Inicialização	RF13 (Confirmação de Inicialização)	Rotina de *firmware* para auto-teste de sensores (HC-SR04) e emissão de sinal de "Sistema Pronto" (vibração longa/curta).
1.3. Monitoramento de Energia	RF12 (Alerta de Bateria Baixa)	Implementação do circuito de monitoramento de bateria e lógica de alerta crítico.

Sprint 2: Módulo de Visão e Coleta de Dados

Foco: Integrar o subsistema de visão computacional e a capacidade de registro de eventos (*logging*).

Atividade	Requisitos Chave	Produto de Entrega
2.1. Implementação do Módulo 2	RF03 (Captura de Imagem)	Módulo 2 (ESP32-CAM/S3) configurado para captura periódica de imagens.
2.2. Estrutura de Logs	RF07 (Registro/Logs)	Implementação da estrutura de dados para registro local de eventos (*timestamp*, tipo, distância, *confidence*).
2.3. Treinamento do Modelo	RF04 (Reconhecimento de Objetos)	Início do treinamento e otimização do modelo TFLite para as 5 classes prioritárias (fio/ramo, degrau, cadeira/obstáculo baixo, pessoa, porta).

Sprint 3: Comunicação e Aplicativo Mínimo

Foco: Estabelecer a comunicação sem fio entre o *wearable* e o *smartphone* e desenvolver a interface básica do usuário.

Atividade	Requisitos Chave	Produto de Entrega
3.1. Conexão BLE	RF05 (Comunicação com App - BLE)	Conexão BLE estável para envio de eventos e recebimento de comandos de calibração.

Atividade	Requisitos Chave	Produto de Entrega
3.2. Interface de Status	RF11 (Verificar dispositivos pelo App)	Aplicativo Android mínimo exibindo status de conexão, nível de bateria e status operacional dos sensores.
3.3. Testes de Latência	RNF01 (Latência)	Testes de ponta a ponta para garantir que a latência de comunicação esteja dentro do limite de 400 ms.

Sprint 4: Inteligência e Feedback Sonoro (TTS)

Foco: Integrar o modelo de reconhecimento e o sistema de alerta de voz, completando o ciclo de *feedback* multimodal.

Atividade	Requisitos Chave	Produto de Entrega
4.1. Integração do Reconhecimento	RF04 (Reconhecimento de Objetos)	Modelo TFLite integrado ao *firmware* (ou App, via *offload*) e funcional, com acurácia alvo de 70%.
4.2. Alerta de Voz (TTS)	RF06 (TTS no App)	Implementação da funcionalidade de *Text-to-Speech* (TTS) no App para reproduzir alertas verbais curtos.
4.3. Regras de Negócio	RN01, RN02, RN03, RN04	Implementação da lógica de priorização de alertas (distância vs. perigo suspenso) e intensidade de vibração.

Sprint 5: Usabilidade e Robustez

Foco: Refinar a experiência do usuário, adicionando controles de usabilidade e mecanismos de robustez.

Atividade	Requisitos Chave	Produto de Entrega
5.1. Controles de Usuário	RF08 (Gestos/Botões de Controle), RF14 (Modo de Silêncio Temporário)	Implementação de botão físico no *wearable* e funcionalidade no App para pausar temporariamente os alertas.
5.2. Interface de Calibração	RF10 (Atualização de Parâmetros via App), RNF04 (Interface de calibração)	Interface no App para ajustes de *thresholds* de distância, sensibilidade de vibração e volume de TTS.
5.3. Fallback de Processamento	RF09 (Fallback de processamento)	Lógica de *offload* da imagem para o *smartphone* para inferência, caso a latência local seja comprometida.

Sprint 6: Validação, Implantação e Entrega

Foco: Testes finais de campo, validação dos critérios de aceitação e preparação da entrega do PoC.

Atividade	Requisitos Chave	Produto de Entrega
6.1. Testes de Campo com Usuários	RNF03 (Autonomia), RNF02 (Acurácia)	Relatório de testes de campo com 1-5 usuários, validando autonomia (≥ 2 horas) e acurácia em ambiente real.
6.2. Exportação de Logs	RF15 (Exportação de Logs de Teste)	Funcionalidade no App para exportar logs de eventos em formato CSV ou JSON para análise *offline*.
6.3. Implantação e Documentação	-	Versão Candidata (RC) finalizada e documentação técnica mínima do PoC.

3. Critérios de Aceitação do PoC

O sucesso do projeto será medido pela aderência aos seguintes critérios de aceitação, a serem validados na Sprint 6:

Critério	Meta	Requisito Relacionado
Acurácia	≥ 70% nas 5 classes prioritárias em ambiente controlado.	RF04, RNF02
Latência	Média de alerta (detecção → *feedback*) ≤ 400 ms.	RNF01
Autonomia	Duração mínima de prova ≥ 2 horas com ciclo de uso típico.	RNF03
Validação	Testes preliminares realizados com 1-5 usuários.	-
Funcionalidade	Integração estável e funcional dos Módulos 1, 2 e 3.	RF05, RF11

1. Plano de projeto

Projeto:

Óculos inteligente de baixo custo que detecta perigos ambientais e fornece alertas táteis e sonoros para melhorar a locomoção de pessoas com deficiência visual.

Registro de Alterações

Versão	Responsáveis	Data	Alterações
0.1	Equipe de Desenvolvimento	01/05/2025	Versão inicial do plano de projeto

1.1 Introdução

Projeto PoC de 6 semanas para desenvolve/r um wearable ou tecnologia vestível que combina sensores de distância, visão computacional e app móvel, aumentando a percepção espacial além da bengala tradicional.

1.2 Escopo do projeto

Projeto: Inclui: desenvolvimento de protótipo (Módulos 1 e 2), app Android mínimo, integração e testes de campo; exclui: produção em escala e certificações regulatórias.

Justificativa:

A bengala não detecta riscos acima do solo nem fornece informação contextual — um óculos assistivo acessível reduz riscos e amplia autonomia do usuário.

Produto:

Protótipo funcional integrado: óculos com sensor ultrassônico + câmera, pulseira vibratória, app Android para TTS, calibração e status de sistema, e documentação técnica mínima.

Descrição Preliminar do Sistema (breve):

Sistema modular composto por: Módulo 1 — sensores de distância e atuador vibratório (detecção local e alerta tátil); Módulo 2 — câmera com modelo TFLite para reconhecimento rápido de classes prioritárias; Módulo 3 — app Android para receber eventos, executar TTS, calibrar e registrar logs. Comunicação por BLE/Wi-Fi com fallback para processamento no celular quando necessário

1.3 Descrição Preliminar do Sistema

- Hardware disponível: ESP32 (com/câmera), ESP32-S3/ESP32-CAM, Raspberry Pi Pico, HC-SR04, motores de vibração, estrutura wearable, smartphone Android.
- Performance: latência ≤ 400 ms do evento ao alerta; autonomia mínima de prova ≥ 2 horas.
- Precisão: acurácia ≥ 70% nas 5 classes prioritárias em ambiente controlado.
- Comunicação: BLE obrigatório; Wi-Fi opcional (offload/telemetria).
- Energia: bateria 3.7 V até 5 V com circuito de carga e proteção.
- Usabilidade: padrões de vibração diferenciáveis, TTS claro e opções de volume/tempo.
- Testes: 1-5 usuários para validação de campo dentro das 6 semanas.

1.4. Modelo de Ciclo de Vida

Modelo: Incremental com sprints semanais.

Justificativa: Permite validar hipóteses de negócio (planos) e incorporar feedback dos primeiros usuários/parceiros rapidamente.

Atividades e papéis:

Etapa	Atividades Principais	Insumos	Produtos	Papéis Envolvidos
Levantamento	Pesquisas de mercado, entrevistas com salões	Questionários, benchmarks de concorrentes	Documento de requisitos e personas	Analista de Requisitos
Projeto	Modelagem de dados e arquitetura, UX/UI	Requisitos, personas	Protótipos, diagramas UML, wireframes	Arquiteto de Software, Designer

Implementação	Desenvolvimento de módulos(front-end, backend e APIs)	Backlog de funcionalidades	Módulos entregues por sprint	Desenvolvedores Full-stack
Testes	Testes funcionais, de usabilidade e end-to-end	Builds de sprint	Relatórios de bugs e recomendações	QA/Testador
Implantação	Deploy em staging e produção, configuração CI/CD	Versão candidata (RC)	Sistema em produção	DevOps, Gerente de Projeto
Manutenção	Correções, Otimizações e novos incrementos	Feedback de usuários, métricas	Atualizações e patches	Suporte técnico, Desenvolvedores

1.5. Estrutura da Equipe do Projeto

Organograma:

- Product Owner: Fabio Brasileiro
- Scrum Master: Heloisa Cativo
- Tech Lead / Arquiteto: Jose Filho
- Devs Backend (2–3): Victor Fernandes, Victor Abreu, Guilherme Reis
- Dev Frontend: Luis Oliveira, Eric Silva, Joao Pereira
- DBA: Fabio Brasileiro
- QA: Mateus Miranda
- Config Manager: Jose Filho
- CCB: Victor Abreu

Matriz de Responsabilidades (exemplo):

Atividade	GP	AR	DEV	QA	UX/UI	DEVOPS
Levantamento	R	R				
Projeto	C	R	C		R	
Implementação	A	C	R	C	C	
Testes	C		C	R		
Implantação	A		C	C		R
Manutenção	A	C	R	R		C

(R: Responsável, A: Aprovador, C: Consultado)

1.6. Definição de Medidas (KPIs)

Medida 1: Taxa de Entrega de Funcionalidades

- Nome: Funcionalidades entregues por sprint
- Definição: Número de funcionalidades concluídas em cada sprint
- Tipo: Derivada
- Entidade Medida: Sprint

- Atributo Medido: Entregas realizadas
- Escala: Quantitativa absoluta
- Unidade de Medida: Funcionalidades
- Fórmula: Total de funcionalidades concluídas / Total previstas na sprint
- Procedimento: Verificação dos tickets finalizados no Github
- Momento: Ao fim de cada sprint
- Responsável pela Medição: Scrum Master
- Procedimento de Análise: Comparação com metas da sprint
- Responsável pela Análise: Gerente de Projeto

Justificativa: Ajuda a avaliar o ritmo de desenvolvimento e a adequação ao planejamento.

CENTRO UNIVERSITÁRIO *FAMETRO*

ENGENHARIA DE SOFTWARE

Equipe DEV: Eric Silva, Fábio Brasileiro, Guilherme Fernandes, Heloísa Cativo, João Sousa Pereira, José Pinto Filho, Luis Oliveira, Mateus Miranda, Victor Fernandes, Victor Abreu.

Documento de análise de requisitos: **ÓCULOS INTELIGENTE DE BAIXO CUSTO PARA AUXÍLIO À LOCOMOÇÃO DE PESSOAS COM DEFICIÊNCIA VISUAL**

Manaus/AM

2025

Equipe DEV: Eric Silva, Fábio Brasileiro, Guilherme Fernandes, Heloísa Cativo, João Sousa Pereira, José Pinto Filho, Luis Oliveira, Mateus Miranda, Victor Fernandes, Victor Abreu.

Documento de análise de requisitos: ÓCULOS INTELIGENTE DE BAIXO CUSTO PARA AUXÍLIO À LOCOMOÇÃO DE PESSOAS COM DEFICIÊNCIA VISUAL

Trabalho do 6° período do curso de engenharia de software do Centro Universitário *FAMETRO*, Câmpus Manaus, como parte dos requisitos para obtenção da nota de **INOVATECH FEIRA DE TECNOLOGIA**.

Orientador: Prof. JEAN CARLOS FIGUEIREDO.

Manaus/AM

2025

RESUMO

Projeto de prova de conceito (PoC) para um óculos inteligente de baixo custo voltado à auxílio na locomoção de pessoas com deficiência visual. Em 6 semanas será desenvolvido um protótipo integrado composto por: (1) Módulo tátil de distância e vibração (HC-SR04 + vibracall controlado por ESP32/Pico), (2) Módulo de visão para reconhecimento rápido de objetos prioritários (ESP32-CAM / ESP32-S3 com modelo TFLite leve) e (3) App Android mínimo para TTS, calibração e logs. O sistema visa detectar perigos ambientais (fios/obstáculos suspensos, degraus, cadeiras, pessoas, portas) e transmitir alertas táteis ou sonoros ao usuário com baixa latência. Critérios de aceitação mínimos do PoC: acurácia ≥ 70% nas 5 classes prioritárias em ambiente controlado, latência média de alerta ≤ 400 ms, autonomia de prova ≥ 2 horas e testes com 1–5 usuários. O projeto prioriza componentes já disponíveis (ESP32, HC-SR04, vibracall, Raspberry Pi Pico, estrutura de óculos/boné) para reduzir tempo e custo.

Palavras-chave: …

ABSTRACT

This project proposes a six-week Proof of Concept (PoC) for the development of a low-cost smart glasses system focused on assisting the autonomous locomotion of visually impaired individuals. The system is designed to detect common environmental hazards (e.g., suspended obstacles, steps, people, doors) and provide feedback to the user with minimal latency. The prototype's architecture integrates three main modules: (1) a haptic distance detection module, comprising an HC-SR04 ultrasonic sensor and vibracall actuators, controlled by a microcontroller (ESP32/Pico); (2) an embedded computer vision module (ESP32-CAM/S3) running an optimized TFLite model for rapid recognition of priority objects; and (3) a minimal Android application for managing audio alerts (TTS), calibration, and log collection. Acceptance criteria for the PoC include a recognition accuracy of $\geq$ 70% for priority classes in a controlled environment, an average alert latency of $\leq$ 400 ms, a test battery life of $\geq$ 2 hours, and preliminary validation with 1-5 users. The project prioritizes the use of readily available hardware components to ensure rapid prototyping and low-cost viability..

Keywords: .

SUMÁRIO

INTRODUÇÃO

A mobilidade autônoma de pessoas com deficiência visual ainda depende majoritariamente de ferramentas táteis (bengala) e orientação tátil/sonora humana. Embora eficazes, essas soluções apresentam limitações: alcance físico restrito, incapacidade de detectar riscos acima do solo (fios, ramos, sinalizações suspensas) e falta de informação contextual antecipada. Dispositivos comerciais mais avançados costumam ser caros ou não combinam adequadamente sensores de distância, visão computacional e feedback tátil em uma solução compacta e acessível.

Este projeto propõe desenvolver um **óculos inteligente de baixo custo** como solução complementar à bengala, combinando sensores ultrassônicos para detecção de distância, câmera para reconhecimento de objetos relevantes e mecanismos de entrega de alerta (vibração e áudio). A arquitetura modular — Módulo 1 (distância + vibração), Módulo 2 (câmera + reconhecimento), Módulo 3 (app móvel) — permite desenvolvimento incremental e validação contínua ao longo de 6 semanas. O objetivo do PoC é entregar um protótipo funcional que comprove viabilidade técnica, usabilidade e ganho de percepção espacial para o usuário, utilizando majoritariamente hardware já disponível na equipe para acelerar a execução e minimizar custos.

1. Plano de projeto

Projeto:

Óculos inteligente de baixo custo que detecta perigos ambientais e fornece alertas táteis e sonoros para melhorar a locomoção de pessoas com deficiência visual.

Registro de Alterações

Versão	Responsáveis	Data	Alterações
0.1	Equipe de Desenvolvimento	01/05/2025	Versão inicial do plano de projeto

1.1 Introdução

Projeto PoC de 6 semanas para desenvolve/r um wearable ou tecnologia vestível que combina sensores de distância, visão computacional e app móvel, aumentando a percepção espacial além da bengala tradicional.

1.2 Escopo do projeto

Projeto: Inclui: desenvolvimento de protótipo (Módulos 1 e 2), app Android mínimo, integração e testes de campo; exclui: produção em escala e certificações regulatórias.

Justificativa:

A bengala não detecta riscos acima do solo nem fornece informação contextual — um óculos assistivo acessível reduz riscos e amplia autonomia do usuário.

Produto:

Protótipo funcional integrado: óculos com sensor ultrassônico + câmera, pulseira vibratória, app Android para TTS, calibração e status de sistema, e documentação técnica mínima.

Descrição Preliminar do Sistema (breve):

Sistema modular composto por: Módulo 1 — sensores de distância e atuador vibratório (detecção local e alerta tátil); Módulo 2 — câmera com modelo TFLite para reconhecimento rápido de classes prioritárias; Módulo 3 — app Android para receber eventos, executar TTS, calibrar e registrar logs. Comunicação por BLE/Wi-Fi com fallback para processamento no celular quando necessário

1.3 Descrição Preliminar do Sistema

- Hardware disponível: ESP32 (com/câmera), ESP32-S3/ESP32-CAM, Raspberry Pi Pico, HC-SR04, motores de vibração, estrutura wearable, smartphone Android.
- Performance: latência ≤ 400 ms do evento ao alerta; autonomia mínima de prova ≥ 2 horas.
- Precisão: acurácia ≥ 70% nas 5 classes prioritárias em ambiente controlado.
- Comunicação: BLE obrigatório; Wi-Fi opcional (offload/telemetria).
- Energia: bateria 3.7 V até 5 V com circuito de carga e proteção.
- Usabilidade: padrões de vibração diferenciáveis, TTS claro e opções de volume/tempo.
- Testes: 1-5 usuários para validação de campo dentro das 6 semanas.

1.4. Modelo de Ciclo de Vida

Modelo: Incremental com sprints semanais.

Justificativa: Permite validar hipóteses de negócio (planos) e incorporar feedback dos primeiros usuários/parceiros rapidamente.

Atividades e papéis:

Etapa	Atividades Principais	Insumos	Produtos	Papéis Envolvidos
Levantamento	Pesquisas de mercado, entrevistas com salões	Questionários, benchmarks de concorrentes	Documento de requisitos e personas	Analista de Requisitos
Projeto	Modelagem de dados e arquitetura, UX/UI	Requisitos, personas	Protótipos, diagramas UML, wireframes	Arquiteto de Software, Designer

Implementação	Desenvolvimento de módulos(front-end, backend e APIs)	Backlog de funcionalidades	Módulos entregues por sprint	Desenvolvedores Full-stack
Testes	Testes funcionais, de usabilidade e end-to-end	Builds de sprint	Relatórios de bugs e recomendações	QA/Testador
Implantação	Deploy em staging e produção, configuração CI/CD	Versão candidata (RC)	Sistema em produção	DevOps, Gerente de Projeto
Manutenção	Correções, Otimizações e novos incrementos	Feedback de usuários, métricas	Atualizações e patches	Suporte técnico, Desenvolvedores

1.5. Estrutura da Equipe do Projeto

Organograma:

- Product Owner: Fabio Brasileiro
- Scrum Master: Heloisa Cativo
- Tech Lead / Arquiteto: Jose Filho
- Devs Backend (2–3): Victor Fernandes, Victor Abreu, Guilherme Reis
- Dev Frontend: Luis Oliveira, Eric Silva, Joao Pereira
- DBA: Fabio Brasileiro
- QA: Mateus Miranda
- Config Manager: Jose Filho
- CCB: Victor Abreu

Matriz de Responsabilidades (exemplo):

Atividade	GP	AR	DEV	QA	UX/UI	DEVOPS
Levantamento	R	R				
Projeto	C	R	C		R	
Implementação	A	C	R	C	C	
Testes	C		C	R		
Implantação	A		C	C		R
Manutenção	A	C	R	R		C

(R: Responsável, A: Aprovador, C: Consultado)

1.6. Definição de Medidas (KPIs)

Medida 1: Taxa de Entrega de Funcionalidades

- Nome: Funcionalidades entregues por sprint
- Definição: Número de funcionalidades concluídas em cada sprint
- Tipo: Derivada
- Entidade Medida: Sprint

- Atributo Medido: Entregas realizadas
- Escala: Quantitativa absoluta
- Unidade de Medida: Funcionalidades
- Fórmula: Total de funcionalidades concluídas / Total previstas na sprint
- Procedimento: Verificação dos tickets finalizados no Github
- Momento: Ao fim de cada sprint
- Responsável pela Medição: Scrum Master
- Procedimento de Análise: Comparação com metas da sprint
- Responsável pela Análise: Gerente de Projeto

Justificativa: Ajuda a avaliar o ritmo de desenvolvimento e a adequação ao planejamento.

2. Documento de requisitos

2.1. Introdução

Mini-mundo: ambiente real e delimitado onde o PoC será validado — calçadas urbanas, áreas internas (salas, corredores), e trechos curtos de rua com fluxo moderado de pedestres. Cenários prioritários: obstáculos baixos (cadeiras, degraus), obstáculos suspensos (fios, galhos), pessoas em movimento, portas/aberturas. Condições: iluminação diurna e noturna urbana típica (iluminação pública), piso irregular e ruído sonoro ambiente moderado. Testes serão feitos com usuários com deficiência visual e/ou cegueira total, usando bengala como baseline.

Resumo do mini-mundo:

- Espaço físico: calçada pública e ambiente interno controlado (sala/corredor).
- Tipos de obstáculos: degraus, cadeiras, postes, fios, portas, pedestres/animais.
- Condições ambientais: iluminação variada, ruído ambiente, trânsito moderado.
- Uso esperado: caminhada normal (1–5 km/h), paradas comuns para inspeção.

2.2. Descrição do Propósito do Sistema

O sistema tem por finalidade aumentar a percepção espacial de pessoas com deficiência visual através de um wearable (óculos + pulseira vibratória + app), detectando perigos e comunicando alertas táteis e/ou sonoros em tempo real para prevenir colisões e incidentes. No PoC de 6 semanas o objetivo é demonstrar viabilidade técnica, usabilidade e métricas mínimas de desempenho (acurácia, latência, autonomia).

Objetivos práticos:

- Detectar obstáculos próximos e emitir vibração proporcional.
- Reconhecer 5 classes prioritárias com modelo leve (TFLite) e emitir TTS via app.
- Validar integração entre hardware e app com latência aceitável para navegação.

2.3. Descrição do Minimundo

Entidades do mini-mundo

- **Usuário** (pessoa com deficiência visual)
- **Óculos inteligente** (dispositivo embarcado)
- **Pulseira vibratória** (atuador tátil)
- **Smartphone Android** (app + TTS)
- **Obstáculo** (objetos físicos no ambiente)
- **Operador / Testador** (configura e observa testes)

2.4. Requisitos de Usuário

Requisitos Funcionais

Identificador	Descrição	Prioridade	Depende de
RF01	Medição de Distância: O sistema deve medir distância em frente ao usuário usando HC-SR04 e disponibilizar leitura atual a cada 200–500 ms.	Alta	-
RF02	Alerta Tátil Proporcional: Ao detectar distância abaixo de thresholds configuráveis, o dispositivo deve acionar o vibracall com intensidade proporcional (fraco/médio/alto) em ≤ 100 ms após leitura.	Alta	RF01

RF03	Captura de Imagem: O sistema deve capturar imagens a cada X segundos (configurável) e disponibilizar para inferência local ou envio ao app quando solicitado.	Alta	RF02
RF04	Reconhecimento de Objetos (5 classes prioritárias): O sistema deve inferir localmente (ou via app) as classes: fio/ramo, degrau, cadeira/obstáculo baixo, pessoa, porta. Para o PoC, acurácia mínima alvo: 70% em ambiente controlado.	Alta	RF02
RF05	Comunicação com App (BLE): Enviar eventos (tipo, distância, timestamp, confidence) via BLE para app Android e receber comandos de calibração.	Alta	RF04
RF06	TTS no App: O app deve reproduzir mensagem curta correspondente ao evento (ex.: “fio à frente, 1,2 metros”) em ≤ 400 ms após recebimento do evento	Média	RF05
RF07	Registro/Logs: Registrar localmente eventos com campos: timestamp, tipo, distância, confidence, bateria.	Alta	RF03
RF08	Gestos/Botões de Controle: Permitir ligar/desligar alertas via botão físico no wearable.	Médio	
RF09	Fallback de processamento: Caso a inferência local não alcance metas de latência ou acurácia, permitir offload da imagem para o smartphone para inferência.	Médio	
RF10	Atualização de Parâmetros via App: Permitir ajustes em thresholds de distância, sensibilidade de vibração, taxa de captura de imagem e volume de TTS pelo app.	Médio	
RF11	Verificar dispositivos pelo App: O aplicativo deve exibir o status da conexão BLE (Conectado/Desconectado), o nível de bateria restante do wearable (em porcentagem) e o status operacional dos sensores principais.	Médio	RF05

RF12	Alerta de Bateria Baixa: O sistema deve notificar ativamente o usuário (via vibração específica no wearable e alerta sonoro/TTS no app) quando o nível de bateria do wearable atingir um nível crítico (ex: 20%).	Alta	RF05, RF07
RF13	Confirmação de Inicialização (Auto-Teste): Ao ser ligado, o wearable deve executar um auto-teste (verificar conexão do HC-SR04 e da câmera) e emitir um sinal claro de "Sistema Pronto" (ex: uma vibração longa e duas curtas, ou um TTS "Sistema Ativo").	Alta	RF01, RF03, RF06
RF14	Modo de Silêncio Temporário: O usuário deve poder pausar (silenciar) temporariamente todos os alertas (táteis e sonoros) através de um comando no wearable (ex: um clique duplo no botão RF08) ou no App (via RF10).	Média	RF08, RF10
RF15	Exportação de Logs de Teste: O App deve possuir uma função (acessível pelo Operador/Testador) para exportar os logs de eventos (RF07) em um formato padrão (ex: CSV ou JSON) para análise de desempenho offline.	Média	RF05, RF07

Regras de Negócio

Identificador	Descrição	Prioridade	Depende de
RN01	Se a distância medida for ≤ 0.8 m → vibração alta + alerta TTS prioritário.	Alta	RF02

RN02	Se a distância estiver entre 0.8 m e 1.5 m → vibração média.	Alta	RF05
RN03	Se a distância > 1.5 m e detector de objeto identificar classe prioritária (fio/ramo) dentro do campo de visão → vibração fraca + TTS.	Média	RF05
RN04	Em caso de conflito (múltiplos eventos simultâneos), priorizar evento de menor distância; se distância similar, priorizar evento classificado como “perigo suspenso” (fio/ramo).		
RN05	odos os eventos registrados devem ter timestamp e tipo salvo localmente para posterior exportação.		
RN06	Se a bateria < 15% → notificar no app e reduzir frequência de captura de imagem para economizar energia.		
RN07	Comunicação BLE é padrão; se BLE indisponível e Wi-Fi disponível, usar Wi-Fi. Se ambos indisponíveis, armazenar eventos localmente (fallback).		

Requisitos Não Funcionais

Identificador	Descrição	Categoria	Escopo	Prioridade	Depende de
RNF01	Latência ponta-a-ponta (detecção → vibração/TTS) média ≤ 400 ms; vibração acionada em ≤ 100 ms a partir da leitura do microcontrolador.	Desempenho	Global	Alta	-
RNF02	Acurácia mínima por	Precisão	Global	Alta	-

	classe no PoC ≥ 70% em ambiente controlado; taxa de falso positivo ≤ 20%.				
RNF03	Autonomia mínima de prova ≥ 2 horas com ciclo de uso típico (captura periódica de imagem + vibração esporádica).	Energia	Autentic ação	Alta	RF02
RNF04	Interface de calibração acessível no app com opções de volume, intensidade de vibração, e modo silencioso.	Usabilidade	Global	Médio	
	Disponibilidade do sistema durante sessão de teste ≥ 95% (tempo sem falhas).	Confiabilida de	Global		
	Dados de usuários e logs armazenados localmente devem ser acessíveis somente por aplicativo com autenticação local (senha/pin simples).	Segurança	Global		

	O firmware deve poder rodar em ESP32/ESP32-S3 e ter alternativa de execução parcial no smartphone.	Portabilidade	Global		
	Peso do conjunto (óculos + eletrônica) deve ser confortável para uso contínuo curto (meta ≤ 150–200 g, dependendo da estrutura).	Peso/Ergonomia	Global		

2.5. Casos de Teste

Id	Requisitos relacionados	Descrição	Pré-condições	Entradas	Fluxo	Resultados esperados	Pós-condições
CT01	RF01, NFR2	Medição HC-SR04 (Bench)	Sensor conectado ao ESP32, bancada com régua.	Leitura ±10 cm do valor real em 90% das tentativas.	Posicionar objeto a 0.5 m, 1.0 m, 1.8 m; ler valores.	Validar leitura correta do HC-SR04	
CT02	RF02	PWM e Intensidade de Vibração	Vibracall conectado e instrumentado (medir vibração qualitativa)	Três níveis perceptíveis e estáveis.	Enviar PWM para níveis fraco/médio/alto; observar motor.	Verificar intensidade proporcional ao PWM.	
CT03	RF3, RF5	Captura de Imagem (ESP32-CAM)		Imagem legível transferida com sucesso	Capturar imagem, salvar local, enviar via BLE.	Garantir snapshots e envio ao buffer.	
CT05		Testes de Integração					

3. Especificação de Requisitos

1. Introdução

1.1 Propósito do Documento

Este documento tem como objetivo especificar detalhadamente os requisitos funcionais e não funcionais, os modelos de caso de uso, os subsistemas e os diagramas de apoio do projeto **Óculos Inteligente de Baixo Custo para Auxílio à Locomoção de Pessoas com Deficiência Visual (LUMI)**.

O documento visa fornecer uma visão clara e completa para desenvolvedores, testadores e stakeholders, permitindo o correto entendimento, desenvolvimento e validação do sistema.

1.2 Escopo do Sistema

O sistema consiste em um **dispositivo wearable** (óculos com sensores ultrassônicos e câmera integrada) acoplado a uma **pulseira vibratória** e um **aplicativo móvel Android**.
O objetivo é auxiliar pessoas com deficiência visual a identificar obstáculos e perigos ambientais, emitindo **alertas táteis e sonoros em tempo real**.

Componentes principais:

- Módulo 1 – Sensoriamento de distância e vibração (HC-SR04 + motor vibratório).
- Módulo 2 – Câmera embarcada com IA leve (ESP32-CAM / ESP32-S3 + TensorFlow Lite).
- Módulo 3 – Aplicativo Android para calibração, TTS (text-to-speech), logs e controle.

2. Descrição Geral

2.1 Perspectiva do Produto

O produto é um **sistema embarcado integrado** que combina hardware e software. O dispositivo será controlado por um microcontrolador ESP32, comunicando-se via BLE com um aplicativo Android.

O sistema deve operar de forma autônoma, com feedback vibratório proporcional à proximidade de obstáculos e alertas sonoros gerados no aplicativo.

2.2 Usuários e Stakeholders

Tipo de Usuário	Descrição	Nível de Acesso
Usuário Principal	Pessoa com deficiência visual que utiliza o dispositivo.	Uso direto do wearable e app.
Operador/Testador	Pessoa que auxilia em testes e calibrações.	Acesso técnico e coleta de logs.
Equipe de Desenvolvimento	Engenheiros e programadores.	Manutenção e atualização do sistema.
Orientador / Stakeholder Acadêmico	Avaliador do projeto.	Acesso a relatórios e métricas.

2.3 Restrições

- Tempo de resposta do sistema ≤ 400 ms.
- Acurácia de reconhecimento ≥ 70% (ambiente controlado).
- Autonomia mínima de 2 horas.
- Peso máximo do conjunto ≤ 200 g.

2.4 Suposições e Dependências

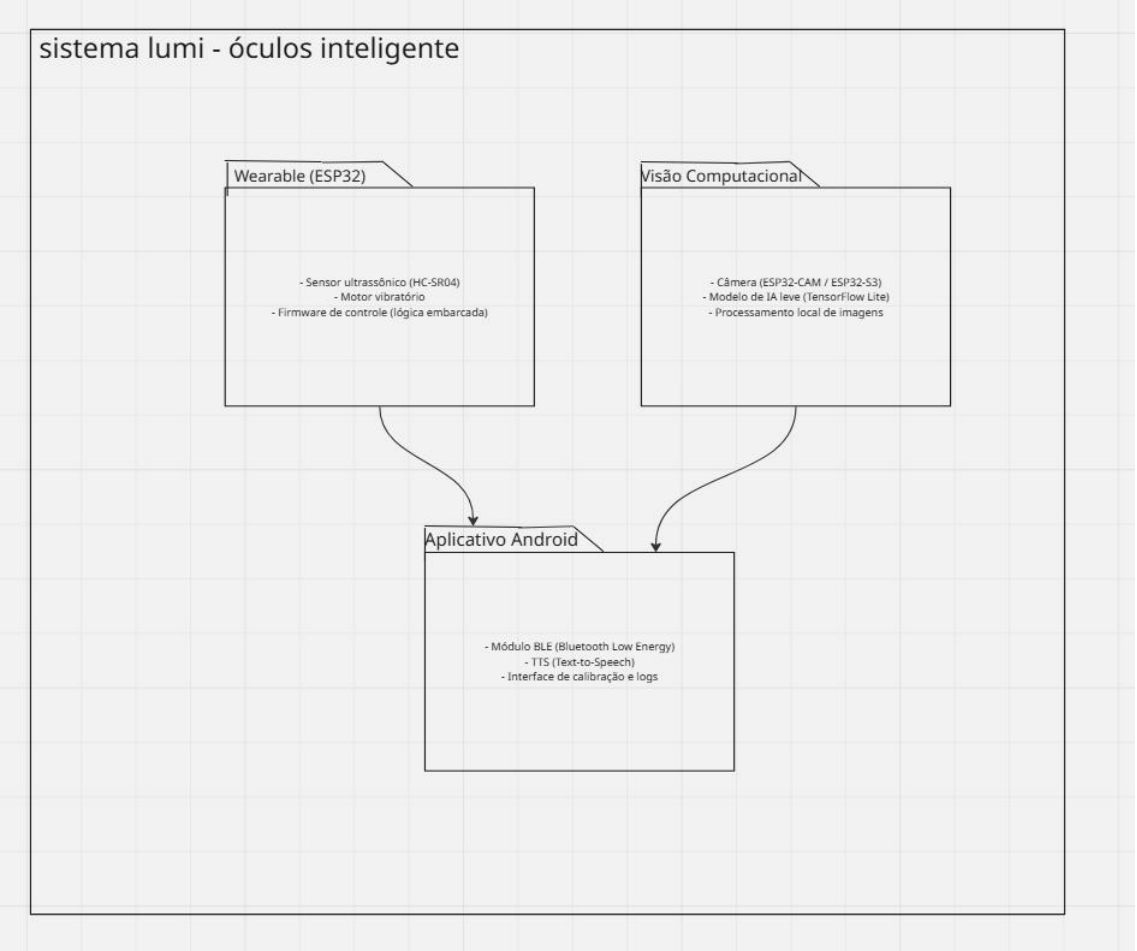
- Testes realizados em ambientes controlados (interno e externo leve).
- Smartphone Android com Bluetooth e TTS compatível.
- Alimentação via bateria recarregável de 3.7–5 V.

3. Modelagem de Subsistemas

3.1 Diagrama de Pacotes

Descrição:
O sistema é dividido em três subsistemas principais:

1. **Wearable** – Controle de sensores e atuadores (ESP32).
2. **Visão Computacional** – Captura e inferência de imagens.
3. **Aplicativo Android** – Gerenciamento de configuração, TTS e registro de logs.



(Imagem – Diagrama de Pacotes do Sistema)

3.2 Descrição dos Subsistemas

Subsistema	Descrição	Interfaces
Sensoriamento Tátil	Realiza medições de distância e aciona vibração conforme proximidade.	Sensor HC-SR04, motor vibratório, ESP32.
Visão Computacional	Reconhece objetos e classes prioritárias (fio, degrau, pessoa, porta, obstáculo).	Câmera ESP32-CAM / modelo TFLite.
Aplicativo Android	Exibe status, calibra sensores, reproduz TTS e armazena logs.	BLE, tela do usuário, TTS.

3.3 Diagrama Visual do Sistema

Esta seção apresenta a representação visual consolidada do fluxo de dados e comunicação entre os principais componentes do sistema "**Óculos Inteligente de Baixo Custo para Auxílio à Locomoção de Pessoas com Deficiência Visual**".

O Diagrama Visual do Sistema (Figura X) ilustra a interação entre os módulos embarcados (ESP32, ESP32-S3, sensores e atuadores), o aplicativo Android e o servidor intermediário. Ele demonstra o ciclo completo de detecção, processamento, transmissão e resposta tátil e sonora para o usuário final.

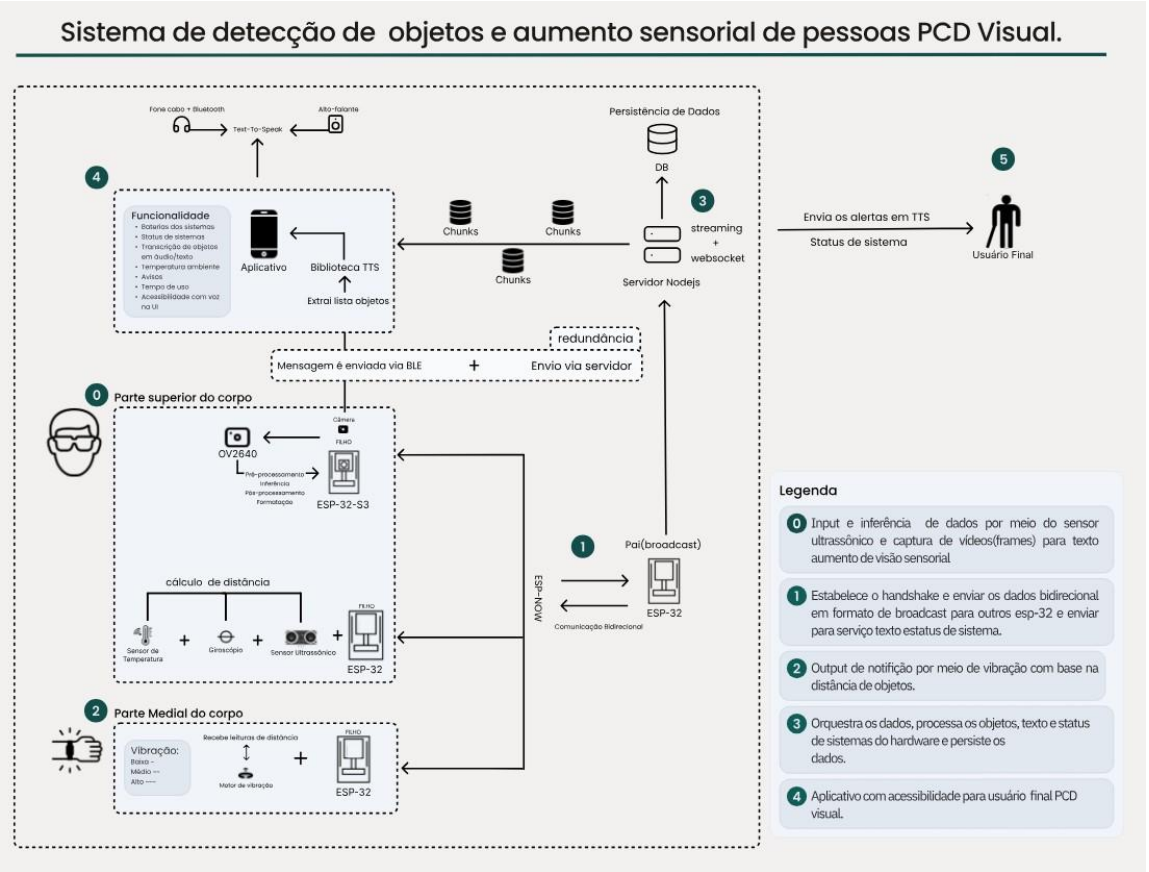
A arquitetura é organizada de forma hierárquica e redundante, permitindo que múltiplos **módulos ESP-32** atuem de forma cooperativa (modelo *pai–filho*) com comunicação bidirecional via BLE e Wi-Fi. O diagrama também evidencia o uso de streaming de vídeo, inferência local com pós-processamento, e envio de alertas TTS ao usuário.

Principais fluxos representados:

- **Entrada de dados: sensores ultrassônicos, câmera OV2640, giroscópio e sensores de temperatura;**
- **Processamento: pré-processamento, inferência e formatação dos dados;**
- **Comunicação: transmissão de informações entre ESPs e o servidor via WebSocket;**

- **Saída: feedback tátil (vibração) e sonoro (TTS);**
- **Persistência: armazenamento de logs e status de sistema no banco de dados;**

Interface: aplicativo Android com acessibilidade por voz e controle de calibração.



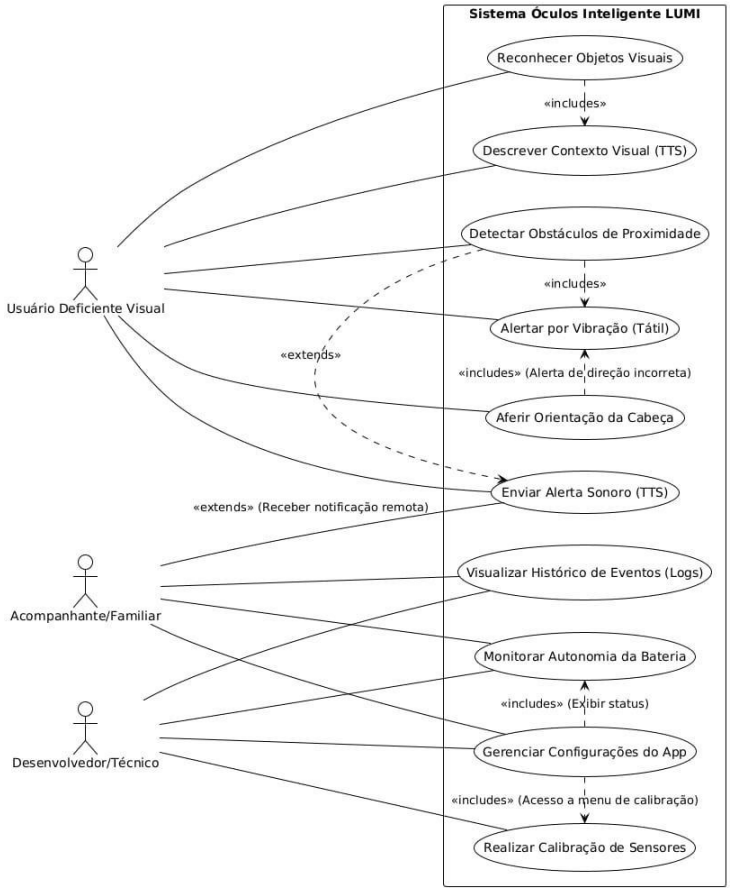
(Imagem – Diagrama de Visual)

4. Modelagem de Casos de Uso

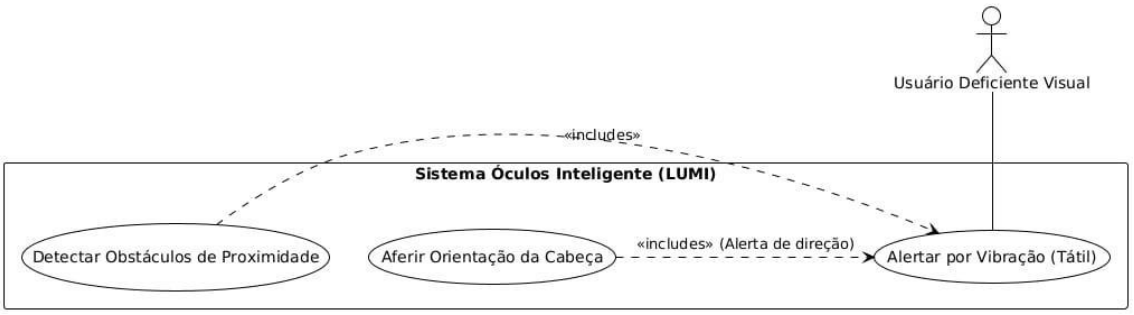
4.1 Atores

Ator	Descrição
Usuário	Utiliza o dispositivo durante a locomoção.
Aplicativo Android	Sistema auxiliar que fornece interface, TTS e calibração.
Operador/Testador	Auxilia no teste, calibração e coleta de dados.

4.2 Diagrama de Casos de Uso



(Imagem – Diagrama de Casos de Uso do Sistema)



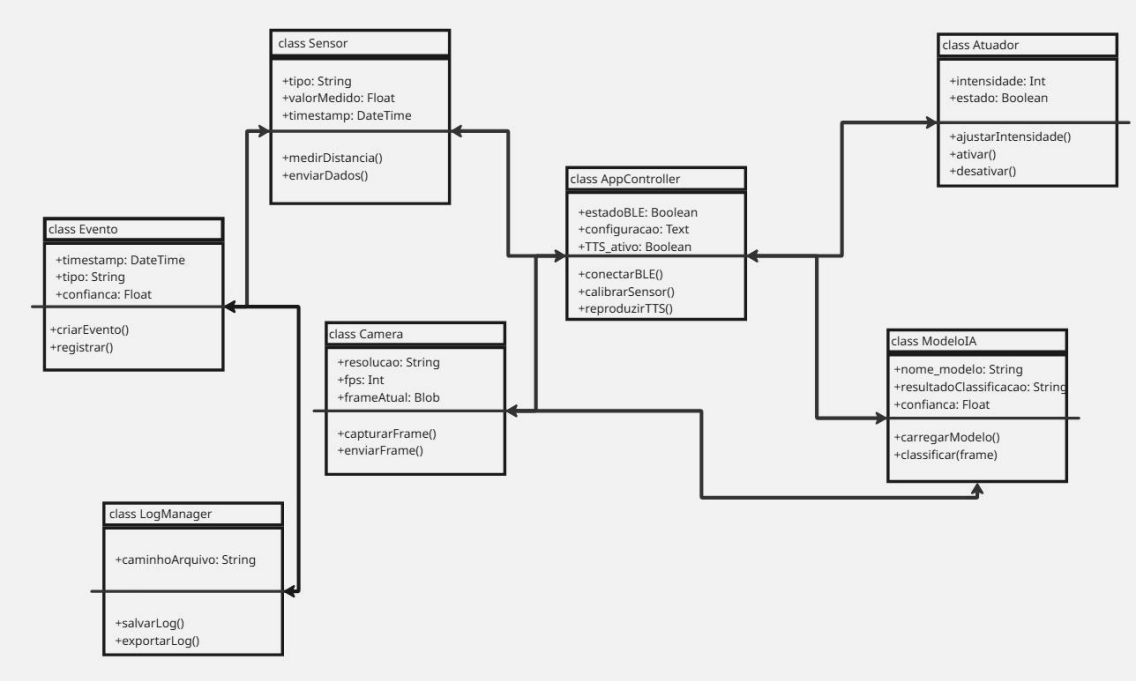
(Imagem – Diagrama de Casos de Uso do Sistema(específico))

4.3 Descrição dos Casos de Uso Principais

ID	Nome do Caso de Uso	Descrição	Atores	Requisitos Relacionados
CU01	Detectar Obstáculo	Mede a distância à frente e aciona vibração conforme proximidade.	Usuário	RF01, RF02
CU02	Reconhecer Objeto	Captura imagem e identifica classes prioritárias.	Usuário, App	RF03, RF04
CU03	Emitir Alerta TTS	O app anuncia o tipo e distância do obstáculo.	App	RF06
CU04	Calibrar Sistema	Ajusta thresholds de distância, vibração e volume.	Operador	RF10
CU05	Exportar Logs	Exporta registros de eventos para análise.	Operador	RF15

5. Modelo Estrutural

5.1 Diagrama de Classes (ou Entidade-Relacionamento)



(Imagem – Diagrama de Classes do Sistema)

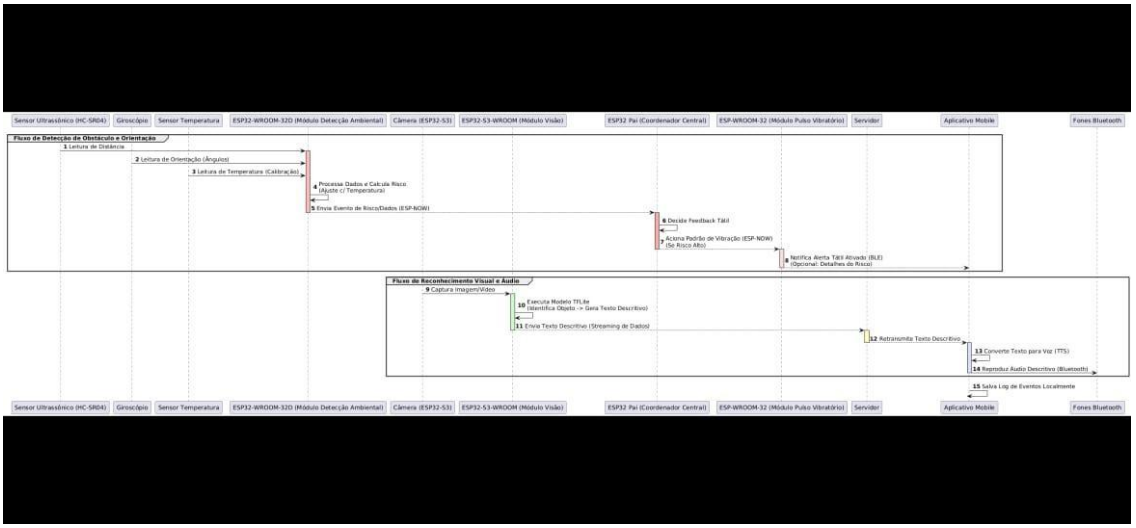
Descrição textual das principais classes:

- **Sensor**: armazena dados de medição de distância.
- **Atuador**: controla intensidade de vibração.
- **Câmera**: captura e envia frames para inferência.
- **ModeloIA**: executa classificação TFLite.
- **AppController**: gerencia BLE, calibração e TTS.
- **Eventos** registro de alertas (timestamp, tipo, confiança).

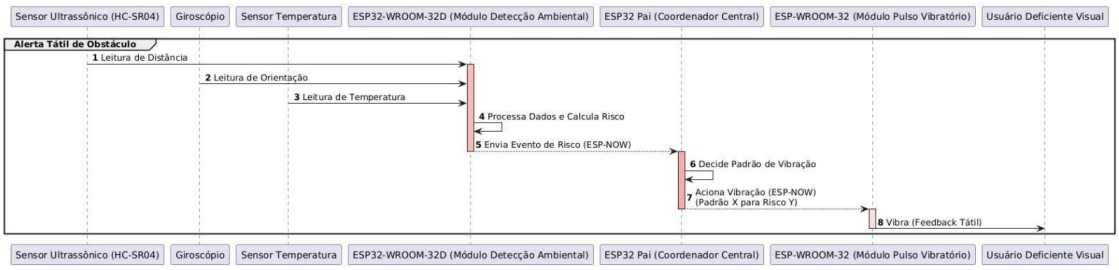
- **LogManager**: salva e exporta logs.

6. Modelo de Interação

6.1 Diagrama de Sequência – Fluxo de Detecção e Alerta



(Imagem – Diagrama de Sequência)



(Imagem – Diagrama de Sequência(específico))

Fluxo Descrito:

1. Sensor detecta obstáculo.
2. ESP32 processa a distância.
3. Se limiar ultrapassado, aciona vibração.
4. Envia evento ao App via BLE.
5. App converte evento em mensagem TTS.
6. Log é salvo localmente.

7. Requisitos Funcionais e Não Funcionais (Resumo)

7.1 Requisitos Funcionais

Principais requisitos já definidos no DR:

- RF01 – Medição de Distância
- RF02 – Alerta Tátil
- RF03 – Captura de Imagem
- RF04 – Reconhecimento de Objetos
- RF05 – Comunicação BLE
- RF06 – Alerta TTS
- RF07 – Registro de Logs
- RF10 – Ajuste de Parâmetros
- RF15 – Exportação de Logs

7.2 Requisitos Não Funcionais

- **Latência:** ≤ 400 ms
- **Acurácia:** ≥ 70%
- **Autonomia:** ≥ 2 horas
- **Peso:** ≤ 200 g
- **Usabilidade:** padrões de vibração distinguíveis
- **Segurança:** logs acessíveis apenas no app autenticado

8. Glossário

Termo	Definição
TTS (Text-to-Speech)	Recurso de voz do app que lê mensagens.
BLE (Bluetooth Low Energy)	Protocolo de comunicação entre o óculos e o app.
ESP32 / ESP32-CAM	Microcontroladores usados no protótipo.
TFLite (TensorFlow Lite)	Framework de IA leve para inferência embarcada.
PoC (Proof of Concept)	Prova de conceito – protótipo experimental de curta duração.
Vibracall	Atuador tátil de vibração usado no wearable.

10. Referências

- Documentação Técnica ESP32 (Espressif, 2024).
- TensorFlow Lite Micro Guide.
- Android BLE API Reference.
- Diretrizes de Acessibilidade W3C / WCAG 2.1.

ID	Tipo	Descrição do Requisito (Resumo)
RF01	Funcional	Medição de Distância (uso do HC-SR04 em 200 ms).
RF02	Funcional	Alerta Tátil Proporcional (vibração com intensidade proporcional à distância).
RF03	Funcional	Captura de Imagem (a cada 100 ms se a distância for crítica).
RF04	Funcional	Reconhecimento de Objetos (disponibilizar inferência local ou via App).
RF05	Funcional	Comunicação com App (transferência de dados, timestamp, confidence via BLE).
RF06	Funcional	TTS no App (mensagens de áudio correspondentes ao evento).
RF07	Funcional	Armazenamento Local (eventos/logs por 30 dias para análise offline).
RF08	Funcional	Controles por Hardware (botão liga/desliga, silêncio temporário, etc.).
RF09	Funcional	Falha de Processamento (notificar se a inferência local falhar).
RF10	Funcional	Atualização de Parâmetros via App (distância, sensibilidade, volume TTS).
RF11	Funcional	Verificar dispositivos pelo App (status de conexão/bateria).
RF12	Funcional	Alerta de Bateria (ativação de alerta sonoro/TTS quando atingir nível crítico).
RF13	Funcional	Confirmação de Inicialização (vibração + TTS "Sistema Ativo").
RF14	Funcional	Modo de Silêncio Temporário (silenciar temporariamente todos os alertas).
RF15	Funcional	Exportação de Logs de Teste (CSV/JSON para análise de desempenho offline).
RNF01	Não Funcional	Latência ponta-a-ponta (vibração < 400 ms).
RNF02	Não Funcional	Acurácia mínima de 70% em ambiente controlado.
RNF03	Não Funcional	Autonomia mínima de ~ 4 horas de uso típico.

RNF04	Não Funcional	Interface de calibração acessível no App.
RNF Outros	Não Funcional	Disponibilidade: 99.9% durante a sessão de uso.
RNF Outros	Não Funcional	Dados de usuários e logs armazenados localmente.

Caso de Uso Relacionado	Módulo / Componente
UC04 - Medir Distância	Módulo de Sensoriamento
UC02 - Emitir Alerta Tátil	Módulo de Feedback Tátil
UC07 - Capturar Imagem	Módulo de Captura
UC08 - Reconhecer Objeto	Módulo de Processamento
UC09 - Transmitir Dados	Módulo de Comunicação (BLE)
UC03 - Emitir Alerta Sonoro	Módulo de Áudio (App)
UC10 - Armazenar Log	Módulo de Armazenamento
UC11 - Controlar Dispositivo	Controlador Principal
UC12 - Notificar Falha	Módulo de Processamento
UC05 - Calibrar Sensores	Interface de Configuração (App)
UC13 - Monitorar Status	Módulo de Comunicação (App)
UC03 - Emitir Alerta Sonoro	Hardware / Energia
UC01 - Detectar Obstáculo	Módulo de Feedback Tátil
UC11 - Controlar Dispositivo	Controlador Principal
UC10 - Armazenar Log	Módulo de Comunicação (App)
—	Módulo de Sensoriamento
—	Módulo de Sensoriamento
—	Hardware / Energia

—	Interface de Configuração (App)
—	Confiabilidade
—	Segurança

Caso de Teste Relacionado	Prioridade	Status
CT01	Alta	Em Desenvolvimento
CT01, CT02, CT05	Alta	Em Desenvolvimento
CT03	Alta	Pendente
—	Alta	Pendente
CT03	Alta	Aprovado
—	Média	Pendente
—	Alta	Aprovado
—	Média	Em Desenvolvimento
—	Média	Pendente
—	Média	Pendente
—	Média	Aprovado
—	Alta	Em Análise
—	Alta	Em Desenvolvimento
—	Média	Aprovado
—	Média	Pendente
CT01 (Implícito)	Alta	Em Desenvolvimento
CT04 (Implícito)	Alta	Em Análise
CT07 (Implícito)	Alta	Em Análise

CT05 (Implícito)	Média	Pendente
—	Alta	Em Análise
—	Alta	Aprovado

RELATÓRIO DE CONTROLE E REGISTRO DE CONFIGURAÇÃO (STATUS ACCOUNTING REPORT)

Projeto: ÓCULOS INTELIGENTE DE BAIXO CUSTO (LUMI) **Versão do Documento:** 1.1 **Data:** 09/11/2025 **Aprovador:** Prof. Jean Carlos Figueiredo (CCB) **Revisão:** Equipe DEV – Engenharia de Software FAMETRO

1. Propósito

Este documento apresenta o Status Accounting (Controle e Registro de Configuração) do projeto LUMI, com o objetivo de garantir a rastreabilidade e visibilidade do progresso de artefatos de software e hardware durante o ciclo de vida do produto.

Inclui informações sobre:

- Itens de configuração (CIs) identificados;
- Status atual de cada item;
- Mudanças aprovadas pela Change Control Board (CCB);
- Versões liberadas e *baseline* vigente.

2. Itens de Configuração (Configuration Items – CIs)

ID	Tipo	Descrição	Versão	Repositório / Local	Status	Responsável
CI-SW-001	Documento	Documento de Requisitos (DR)	1.3	GitHub / docs/requirements	Aprovado	Mateus M. Miranda
CI-SW-002	Código-fonte	Firmware ESP32 (Sensores + Vibração)	1.1	GitHub / src/esp32_main	Em Teste (QA)	Victor Abreu
CI-SW-003	Código-fonte	Módulo IA (TFLite ESP32-CAM)	1.0	GitHub / src/vision_ai	Em Teste (QA)	Victor Fernandes
CI-SW-004	Aplicativo Android	App “LUMI Assist” (TTS + BLE + Logs)	1.0	GitHub / app/android_lumi	Em Teste (QA)	Joao Pereira
CI-HW-001	Hardware	Montagem do protótipo (sensor HC-SR04 + ESP32)	Rev. B	Laboratório FAMETRO	Integrado	Luis Oliveira
CI-TEST-001	Documento	Plano de Teste Funcional	1.1	docs/testing/PT-LUMI.pdf	Aprovado	Mateus M. Miranda
CI-MGMT-001	Documento	Plano de Gerenciamento de Configuração	1.1	docs/config/CM_PLAN_LUMI.pdf	Aprovado	José Filho
CI-DOC-001	Documento	Documento de Análise de Requisitos	1.4	docs/analysis	Aprovado	Heloisa Cativo

3. Baseline de Configuração Vigente

Tipo de Baseline	Data	Itens Incluídos	Observações
Functional Baseline (FB)	01/10/2025	CI-SW-001, CI-DOC-001	Requisitos e arquitetura aprovados pelo CCB.
Allocated Baseline (AB)	15/10/2025	CI-SW-002, CI-HW-001, CI-TEST-001	Integração parcial dos módulos sensoriais e app.
Product Baseline (PB)	09/11/2025	CI-SW-003, CI-SW-004, CI-SW-002, CI-TEST-001	Todos os artefatos de software e teste em fase de QA final para submissão à INOVATECH.

4. Controle de Mudanças (Change Requests – CRs)

ID	Origem	Descrição	Data da Solicitação	Decisão do CCB	Status	Hash
CR-001	PO	Commit Inicial	06/09/2025	Aprovada	Implementada	37abf1d
CR-002	PO	Adiciona a estrutura inicial (scaffold) do projeto backend e seu esquema de banco de dados.	20/09/2025	Aprovada	Implementada	ed8cab6
CR-003	PO	Adicionar código do sensor ultrassônico HCSR04 em C++.	12/10/2025	Aprovada	Implementada	1f8f21
CR-004	PO	Atualizar título do projeto no README.md.	12/10/2025	Aprovada	Implementada	fd18058
CR-005	PO	Remover código do sensor ultrassônico HCSR04 e adicionar configuração do projeto	12/10/2025	Aprovada	Implementada	2e4b8b9
CR-006	PO	Merge na branch main	12/10/2025	Aprovada	Implementada	f34276c
CR-007	PO	Adiciona script reset-project (script para reiniciar o projeto para inicializar a estrutura do projeto.	21/10/2025	Aprovada	Implementada	2885dee
CR-008	PO	Atualiza o package.json para simplificar o script ‘start’ e adicionar o comando de build do EAS.	21/10/2025	Aprovada	Implementada	ad8db63
CR-009	PO	Atualização do README.md	23/10/2025	Aprovada	Implementada	2e68b5e
CR-010	PO	Adiciona a estrutura inicial do projeto e implementa a comunicação o ESP-NOW para detecção de distância e controle de vibração.	26/10/2025	Aprovada	Implementada	cbc669
CR-011	PO	Remove arquivos obsoletos (descontinuados) e atualiza a estrutura do projeto.	26/10/2025	Aprovada	Implementada	00f5113
CR-012	PO	Implementa o servidor de desenvolvimento com WebSocket para PCD Visual	28/10/2025	Aprovada	Implementada	60e720d
CR-013	PO	Adiciona o componente Bateria (Battery) e atualiza os mapeamentos de ícones no IconSymbol	28/10/2025	Aprovada	Implementada	428e4c7
CR-014	PO	Substituir IconSymbol por Imagem (Image) dos óculos da HomeScreen	28/10/2025	Aprovada	Implementada	4aa4aff
CR-015	PO	Atualiza a configuração do dependabot para o ecossistema de pacotes	28/10/2025	Aprovada	Implementada	6015feb
CR-016	DEV	Adicionar contexto do App	28/10/2025	Aprovada	Implementada	2a17d03
CR-017	DEV	Adicionar estrutura base	28/10/2025	Aprovada	Implementada	6e6b8d2
CR-018	PO	Adicionar servidor de visão para PCD com suporte a Websocket e rotas HTTP	29/10/2025	Aprovada	Implementada	f4ff1cc
CR-019	PO	Inicializa a API de Visão (Vision API) com endpoints de extratos de extremidade (edge) de verificação de saúde (health check) e detecção	29/10/2025	Aprovada	Implementada	58048df
CR-020	PO	Adiciona o cliente Websocket para ESP32-CAM com manipulação de detecção e comandos	29/10/2025	Aprovada	Implementada	936b497
CR-021	PO	Adiciona arquivos via upload (inventario de itens de configuraçao)	04/11/2025	Aprovada	Implementada	2bf3c75

ID	Origem	Descrição	Data da Solicitação	Decisão do CCB	Status	Hash
CR-022	PO	Adiciona arquivos via upload	04/11/2025	Aprovada	Implementada	4de820e
CR-023	PO	Adiciona arquivos via upload	04/11/2025	Aprovada	Implementada	4fe048b
CR-024	PO	Rename status accounting report.pdf to delet	11/11/2025	Aprovada	Implementada	b84513e
CR-025	PO	Rename doc padrao.pdf to delet_2	11/11/2025	Aprovada	Implementada	b4b37fa
CR-026	PO	Rename inventario de itens de configuraçao.pdf to delet_3	11/11/2025	Aprovada	Implementada	c7cb6b2
CR-027	PO	atualizações locais	12/11/2025	Aprovada	Implementada	b215906
CR-028	PO	Adiciona novas pastas ao projeto	12/11/2025	Aprovada	Implementada	c1875d3
CR-029	PO	Adicionando novos documentos	12/11/2025	Aprovada	Implementada	dbeed15
CR-030	PO	Add kaz-image-captioning subproject	13/11/2025	Aprovada	Implementada	4dcb58e
CR-031	PO	Fix: Adiciona kaz-image-captioning como diretório normal (não submodule)	13/11/2025	Aprovada	Implementada	5a15e29
CR-032	PO	Refactor: Replace argparse alias with direct import and add run script for testing.	14/11/2025	Aprovada	Implementada	a2e0a3b
CR-033	PO	Add MPU6050 testing documentation and isolated test code.	15/11/2025	Aprovada	Implementada	9d7d2aa
CR-034	PO	Merge branch ‘main’ of github.com:fabiobrasileiroo/sistema_de_pagamento_escolar_por_nfc	15/11/2025	Aprovada	Implementada	a355993
CR-035	PO	Merge branch ‘main’ of github.com:fabiobrasileiroo/consciencia-espacial-PCD-visual	16/11/2025	Aprovada	Implementada	7198f12
CR-036	PO	feat: Add SSE and WebSocket test scripts for monitoring and debugging	16/11/2025	Aprovada	Implementada	22c0219
CR-037	PO	feat: Adiciona status do servidor e dispositivos conectados na tela de configurações	16/11/2025	Aprovada	Implementada	606b208
CR-038	PO	feat: Adiciona exibição da distância do objeto detectado e configuração da URL da API	16/11/2025	Aprovada	Implementada	2e801b8
CR-039	PO	style: Ajusta formatação e espaçamento no componente SettingsScreen	16/11/2025	Aprovada	Implementada	16f4088
CR-040	PO	Refactor Settings Screen: Replace emoji icons with Lucide icons, enhance loading state with SkeletonLoader, and update old styles file and consolidate styles into a new shared styles…	16/11/2025	Aprovada	Implementada	a4f1ce2
CR-041	PO	style: Ajusta a formatação do código nos componentes SettingsScreen e SkeletonLoader	16/11/2025	Aprovada	Implementada	2fdafaa
CR-042	PO	Create README.md for object detection project	16/11/2025	Aprovada	Implementada	dd5ffc4

ID	Origem	Descrição	Data da Solicitação	Decisão do CCB	Status	Hash
CR-043	PO	feat: Adiciona instalação do googletrans e atualiza script de execução com comandos de instalação de pacotes para…	16/11/2025	Aprovada	Implementada	d6c30cd
CR-044	PO	feat: Adiciona novos scripts para captura e processamento de imagens com ESP32-CAM, incluindo suporte a streaming e…	16/11/2025	Aprovada	Implementada	3ef875b
CR-045	PO	Merge remote-tracking branch ‘refs/remotes/origin/main’	17/11/2025	Aprovada	Implementada	6324c83
CR-046	PO	Add ESP32-CAM firmware with captive portal and camera configuration,	17/11/2025	Aprovada	Implementada	d057722
CR-047	PO	feat: Atualiza envio de detecções para servidor via HTTP POST e ajusta script de execução.	17/11/2025	Aprovada	Implementada	071250d
CR-048	PO	Merge remote-tracking branch ‘refs/remotes/origin/main’	17/11/2025	Aprovada	Implementada	3ab29c0
CR-049	PO	feat: Atualiza o módulo de motor de vibração com melhorias na configuração WiFi e mensagens de status, além de…	17/11/2025	Aprovada	Implementada	d029742
CR-050	PO	Merge branch ‘main’ of github.com:fabiobrasileiroo/sistema_de_pagamento_escolar_por_nfc	17/11/2025	Aprovada	Implementada	f0637dd
CR-051	PO	feat: Implement webcam capture functionality with realtime/manual movement and…	17/11/2025	Aprovada	Implementada	1d182f9
CR-052	PO	Add final model image to README	18/11/2025	Aprovada	Implementada	

5. Resumo do Status Atual

Área	Percentual Concluído	Observações
Firmware ESP32	95%	Comunicação BLE estável, calibração finalizada. Aguardando testes de QA.
Módulo IA (TFLite)	90%	Modelo leve implementado e integrado. Aguardando testes de QA.
Aplicativo Android	90%	Tela de calibração e logs concluídas. Exportação CSV implementada. Aguardando testes de QA.
Hardware Integrado	90%	Protótipo funcional testado em bancada.
Testes e QA	75%	Casos CT01–CT03 executados e aprovados. Início dos testes de integração (CT04-CT06).

6. Histórico de Versões e Auditorias

Data	Auditor	Escopo	Resultado	Ações Corretivas
05/10/2025	José Filho	Repositório Git (controle de branches)	Sem não-conformidades	—
18/10/2025	CCB	Requisitos vs Casos de Uso	100% rastreável	—
25/10/2025	QA	Plano de Testes	Corrigidos 2 desvios menores em CT03	—
02/11/2025	CCB	Status Accounting Review	Status OK	Nenhuma ação requerida
09/11/2025	QA	Product Baseline (PB) 1.0	Artefatos prontos para testes de aceitação.	Nenhuma ação requerida

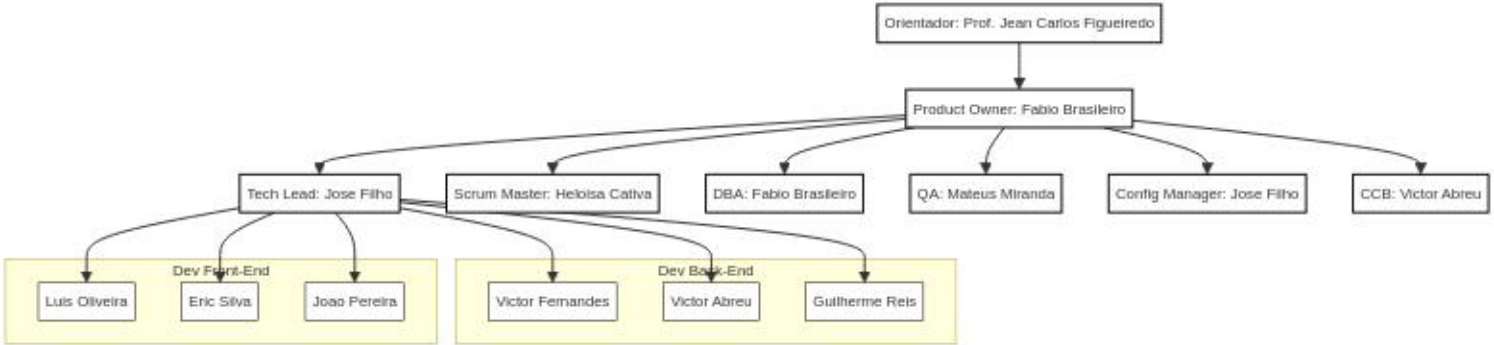
7. Conclusão e Próximos Passos

- O projeto LUMI encontra-se na fase de Qualidade e Testes (QA), com o **Product Baseline (PB) 1.0** estabelecido em 09/11/2025.
- Todos os Itens de Configuração de software (CI-SW-002, CI-SW-003, CI-SW-004) atingiram a Versão 1.0 e estão sob controle de configuração.
- A próxima atualização do Status Accounting (v1.2) incluirá:
  - Resultados de teste de campo e testes de aceitação;
  - Atualização final do Product Baseline (PB) após aprovação do CCB;
  - CRs implementados e revalidados.

Previsão de auditoria final interna: 18/11/2025. **Entrega PoC:** 22/11/2025.

8. Assinaturas

Nome	Cargo	Assinatura	Data
José Filho	Configuration Manager	_____	09/11/2025
Heloisa Cativo	Scrum Master	_____	09/11/2025
Jean Carlos Figueiredo	Orientador / CCB Chair	_____	09/11/2025


Orientador: Prof. Jean Carlos Figueiredo
Product Owner: Fabio Brasileiro
Tech Lead: Jose Filho
Scrum Master: Heloisa Cativa
DBA: Fabio Brasileiro
QA: Mateus Miranda
Config Manager: Jose Filho
CCB: Victor Abreu
Dev Front-End
Luis Oliveira
Eric Silva
Joao Pereira
Dev Back-End
Victor Fernandes
Victor Abreu
Guilherme Reis

File: merge_docs_to_pdf.py

Type: .py
Size: 4117 bytes
Path: /home/fabionote/inovatech-2025/consciencia-espacial-PCD-visual/docs/merge_docs_to_pdf.py

File: requirements-pdf.txt

```
pypdf>=3.0.0
Pillow>=10.0.0
pandas>=2.2.2
openpyxl>=3.1.2
```