Grupo 4 - Amigo Perto

Nome: Eric Senne Roma, Vitor Gomes e Antônio Almeida

[**Etapa 1 Semana 1 2**](#_ysyh0la208u)

[Perfis de clientes 2](#_8q2mq0nnwemv)

[Diagrama de Contexto 4](#_etboanxuzfcc)

[Tabelas - Requisitos Funcionais e Não Funcionais 6](#_rxoz9nwt37kp)

[Requisitos Funcionais 6](#_898cufyus8z5)

[Requisitos de Sistema 6](#_xt7wgide7lei)

[Requisitos de Software 8](#_tv7n9h70uyam)

[Requisitos Não Funcionais 9](#_ci9pe0nw1brg)

[TRL 10](#_bicide2upstn)

[**Etapa 1 Semana 2 11**](#_qoagnrrr4cur)

[Requisitos Consolidados 11](#_kcz2cszht4tp)

[Requisitos Funcionais 11](#_pporiize07p0)

[Requisitos de Sistema 11](#_ngg4j04oh8cx)

[Requisitos de Software 13](#_4lqp07u8ksu8)

[Requisitos Não Funcionais 15](#_9fxbrnj4vsyt)

[Diagrama de Blocos 18](#_ch4e3kuvu8y0)

[Diagrama de Hardware 18](#_o1c8ftinyyx)

[Diagrama de Software 19](#_seosxjcxlt2e)

[Diagrama de camadas 20](#_sclj6ex7ghf7)

[Interfaces e Dependências principais 25](#_7r022ddvgtyz)

[**Etapa 1 Semana 3 30**](#_kti9xgg64mu2)

[Plano de Abstração (HAL) 30](#_6ht3acehcm06)

[Arquitetura de Software 30](#_e6bxq0ellj1o)

[Camada HAL (Hardware Abstraction Layer) 36](#_f6hfsqvnqcog)

[HAL do BLE 36](#_450bt0370rx)

[HAL do Buzzer 38](#_e9dnaq8he8uu)

[HAL da Bateria 38](#_2z16aig0rok3)

[HAL do Botão 39](#_lp8czxfgakfo)

[Comparação microcontrolador 39](#_cph9b5wfc08x)

[Justificativas técnicas e econômicas 40](#_q58ijqw9xv6c)

[Diagrama consolidado do sistema 41](#_dwz4xju3bmaz)

## 

## 

## 

# Etapa 1 Semana 1

## **Perfis de clientes**

**1 - Tutores de cães domésticos (usuários finais principais)**

* **Perfil:** Pessoas que têm cães como animais de estimação, especialmente em **áreas urbanas** ou **semiurbanas**.
* **Necessidades:**
  + Garantir a **segurança** do animal (evitar fugas e perdas).
  + Acompanhar **localização em tempo real**.
  + **Educar/adestrar** o cão com métodos não agressivos (alerta sonoro).
* **Benefícios:**
  + Tranquilidade e praticidade para o tutor.
  + Redução de risco de perda ou acidentes.
  + Melhor controle do comportamento e movimentação.

**2 - Tutores em áreas rurais ou com grandes espaços**

* **Perfil:** Pessoas que moram em chácaras, sítios ou fazendas, onde o cão tem liberdade para circular.
* **Necessidades:**
  + Delimitar **áreas seguras** de circulação.
  + Evitar que o cão entre em regiões perigosas (estradas, plantações, áreas com animais silvestres).
  + Encontrar o animal facilmente em terrenos amplos.
* **Benefícios:**
  + Controle à distância, mesmo em áreas extensas.
  + Segurança para o animal e para o ambiente.

**3 - Adestradores profissionais**

* **Perfil:** Profissionais de comportamento animal que utilizam ferramentas tecnológicas para treinamento.
* **Necessidades:**
  + Ferramentas de **reforço positivo e estímulos controlados**.
  + Registro de comportamento e movimentação.
  + Tecnologia que facilite o treinamento à distância.
* **Benefícios:**
  + Aprimora a **precisão e consistência** no adestramento.
  + Pode servir como **equipamento auxiliar moderno** em sessões de treinamento.

**4 - Clínicas veterinárias e pet shops**

* **Perfil:** Estabelecimentos que oferecem **hospedagem, passeios ou day care** para cães.
* **Necessidades:**
  + Monitorar a **localização de vários animais** em tempo real.
  + Evitar que algum cão fuja durante passeios ou hospedagens.
* **Benefícios:**
  + Aumento da **segurança** operacional.
  + Diferencial competitivo no serviço oferecido ao cliente.

**5 - ONGs e protetores de animais**

* **Perfil:** Organizações e pessoas que cuidam de múltiplos cães, frequentemente soltos em áreas abertas.
* **Necessidades:**
  + Controlar deslocamento e comportamento de vários animais.
  + Reduzir perdas e acidentes.
  + Acompanhar rotinas e movimentação sem grande infraestrutura.
* **Benefícios:**
  + Melhora a **gestão de segurança e bem-estar animal**.
  + Permite uma resposta rápida em caso de fuga.

**6 - Tutores idosos ou com mobilidade reduzida**

* **Perfil:** Pessoas que têm dificuldade em correr ou buscar o cão em caso de fuga.
* **Necessidades:**
  + Sistema de **rastreamento fácil de usar**.
  + Alerta automático se o cão se afasta.
* **Benefícios:**
  + **Independência e tranquilidade**.
  + Reduz o esforço físico em emergências.

## **Diagrama de Contexto**

**“Amigo Perto-BLE” se relaciona com atores externos e principais componentes**.  
(não detalha a arquitetura interna, apenas **define fronteiras (domínios), entradas e saídas**.

**Atores Externos**

1. **Tutor / Usuário (App móvel)**
   * Interage via aplicativo (BLE e interface gráfica).
   * Define raio, perfis e comandos (“Encontrar”, “Modo Perdido”).
2. **Coleira Inteligente**
   * Dispositivo embarcado com MCU e rádio BLE.
   * Mede RSSI, estima distância e aciona buzzer.
3. **Ambiente físico**
   * Afeta RSSI (interferências, obstáculos, distância).
4. **Servidor / Nuvem (opcional futuro)**
   * Sincronização, firmware update ou histórico de uso (fora do escopo inicial, mas possível evolução).

**Sistema em foco: “Coleira Inteligente BLE”**

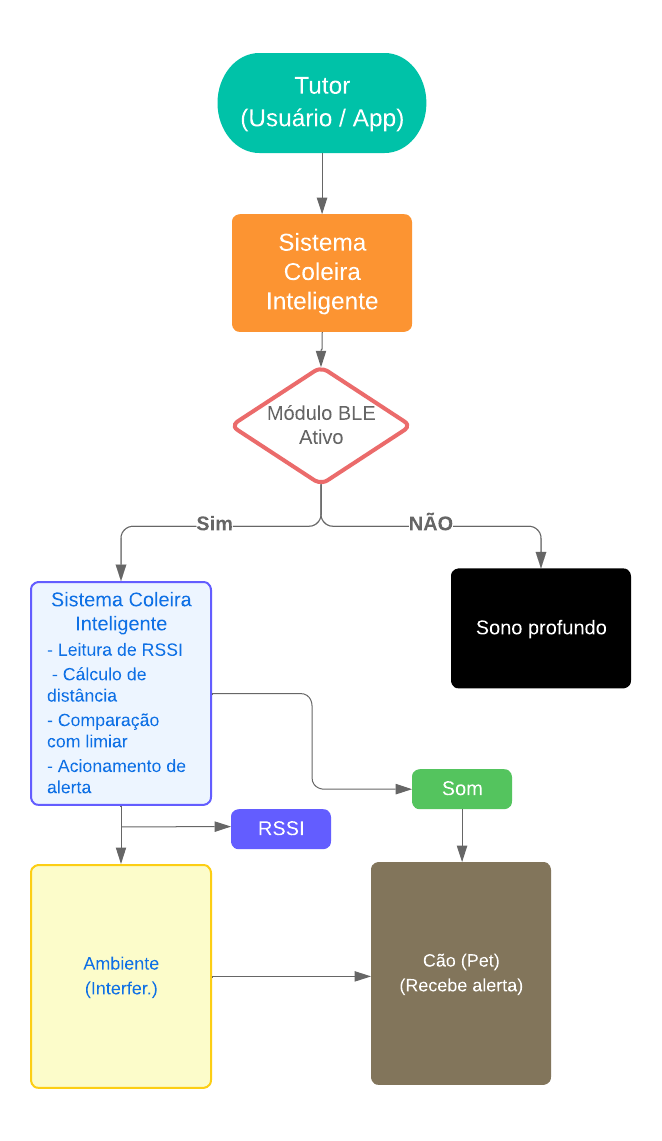
Composto de dois módulos lógicos principais:

* **Módulo Coleira (HW embarcado)**
  + MCU + BLE + buzzer + sensores.
* **Módulo App Móvel (SW no celular)**
  + Interface, configuração e controle via BLE.

|  |  |  |  |
| --- | --- | --- | --- |
| **Origem** |  | **Destino** |  |
| App → Coleira |  | Parâmetros de configuração (raio, perfil, comandos) |  |
| Coleira → App |  | RSSI, status de conexão, eventos de alerta |  |
| Coleira → Cão |  | Sinal sonoro de alerta |  |
| Ambiente → Coleira |  | Atenuação do sinal BLE (RSSI) |  |
| Tutor ↔ App |  | Interação de configuração, visualização e controle |  |
| (opcional) App ↔ Servidor |  | Backup, logs, atualizações |  |

**Título:***Monitoramento de Proximidade BLE para Cães*

**Elementos e conexões:**



## **Tabelas - Requisitos Funcionais e Não Funcionais**

## Requisitos Funcionais

### **Requisitos de Sistema**

|  |  |  |
| --- | --- | --- |
| **ID** | **Título** | **Descrição resumida** |
| **SYS-01** | Tecnologia de comunicação | Usar BLE 5.x ou superior para comunicação entre coleira e dispositivo do tutor. |
| **SYS-02** | Monitoramento de RSSI | Monitorar RSSI continuamente para detectar variações de proximidade. |
| **SYS-03** | Estimativa de distância | Estimar distância a partir do RSSI usando modelo de propagação calibrado (ex.: Log-Distance) com filtragem para reduzir ruído. |
| **SYS-04** | Comparação com limiar | Comparar distância estimada com limiar configurável e gerenciar estado do sistema (seguro/afastado). |
| **SYS-05** | Alerta Físico Local | Acionar alerta físico (beep) quando limiar for excedido. |
| **SYS-06** | Estabilidade (Histerese) | Aplicar histerese para evitar alertas repetitivos próximos ao limiar. |
| **SYS-07** | Conexão Segura e Exclusiva | Conectar apenas ao dispositivo do tutor previamente emparelhado, ignorando outros sinais BLE. |
| **SYS-08** | Configuração via BLE | Permitir ajuste remoto de parâmetros essenciais via comunicação BLE. |
| **SYS-09** | Interface do Usuário (App) | Disponibilizar aplicativo móvel funcional para configuração mínima e monitoramento do estado da coleira via BLE. |
| **SYS-10** | Dome virtual de proximidade | Criar zona de proximidade configurável, alterar estado do sistema e acionar alerta quando limite for ultrapassado. |

### **Requisitos de Software**

|  |  |  |
| --- | --- | --- |
| **ID** | **Título** | **Descrição resumida** |
| **SW-01** | Leitura periódica de RSSI | Realizar leituras periódicas do RSSI para proximidade em tempo real. |
| **SW-02** | Algoritmo de Estimativa de Distância | Implementar algoritmo de conversão de RSSI em distância estimada com filtros para reduzir variações e ruídos. |
| **SW-03** | Comparação com limiar | Comparar continuamente distância estimada com limiar configurado, gerando eventos internos para alertas. |
| **SW-04** | Controle de alerta físico | Gerenciar acionamento do buzzer conforme eventos de ultrapassagem ou retorno ao limite. |
| **SW-05** | Implementação de histerese | Aplicar histerese para evitar alertas repetitivos. |
| **SW-06** | Autenticação e filtragem de dispositivos | Autenticar dispositivo do tutor durante emparelhamento e ignorar sinais BLE não autorizados. |
| **SW-07** | Configuração via BLE | Expor características BLE para ajuste remoto de parâmetros essenciais da coleira. |
| **SW-08** | Implementação Interface do Usuário (App) | Disponibilizar app funcional para configuração mínima, monitoramento de proximidade e alertas, podendo ser simplificado. |
| **SW-09** | Implementação do Dome Virtual de Proximidade | Implementar lógica do dome virtual: manter raio configurável, comparar distância, alterar estado e acionar alerta físico. |

## Requisitos Não Funcionais

|  |  |  |
| --- | --- | --- |
| **ID** | **Título** | **Descrição resumida** |
| **SYS-NF01** | Fonte de energia  compacta | O sistema deve utilizar uma fonte de alimentação de dimensões reduzidas. |
| **SYS-NF02** | Eficiência energética | Minimizar consumo para operação contínua com fonte restrita. |
| **SYS-NF03** | Compacidade e leveza | Design compacto e leve para uso permanente na coleira. |
| **SYS-NF04** | Sinalização segura | Alerta perceptível sem causar incômodo ou risco ao PET. |
| **SYS-NF05** | Robustez BLE | Comunicação BLE estável e resiliente a interferências. |
| **SYS-NF06** | Evolução de TRL | Avançar de protótipo para hardware dedicado com testes documentados. |
| **SYS-NF07** | PCB dedicada | Empregar PCB própria, otimizada para espaço, peso e RF. |
| **SYS-NF08** | Ergonomia e segurança | Materiais e acabamentos seguros e confortáveis para o PET. |
| **SYS-NF09** | Confiabilidade ambiental | Operar em condições externas típicas (temperatura, umidade, respingos). |
| **SYS-NF10** | Usabilidade do app | App simples, intuitivo e fácil de usar,mesmo com interface mínima via BLE. |
| **SYS-NF11** | Plataforma dedicada | Não usar BitDogLab; adotar microcontrolador + rádio BLE em PCB própria. |

## **TRL**

**Nível atual**

**TRL 2 -** Estamos na etapa de pesquisa aplicada: já definimos a aplicação, estamos estudando o hardware e o perfil do cliente, e elaborando os requisitos iniciais.

**Nível a ser atingido**

**TRL 5 -** A expectativa é de termos ao final do projeto um protótipo que apresente os requisitos iniciais testados em um ambiente controlado.

[Etapa 1 Semana 2](#_qoagnrrr4cur)

# Etapa 1 Semana 2

## Requisitos Consolidados

### Requisitos Funcionais

### **Requisitos de Sistema**

**[SYS-01] Tecnologia de comunicação**

O sistema deve utilizar Bluetooth Low Energy (BLE) versão 5.x ou superior como protocolo de comunicação entre a coleira e o dispositivo do tutor.

*Rationale:* O BLE é essencial para atingir os objetivos de autonomia, alcance, baixo consumo de energia e interoperabilidade com os dispositivos móveis atuais.

**[SYS-02] Monitoramento de RSSI**

O sistema deve monitorar continuamente a intensidade do sinal BLE (RSSI) entre a coleira e o dispositivo do tutor, com taxa de amostragem suficiente para detectar variações de proximidade.

*Rationale:* O monitoramento contínuo do RSSI é essencial para identificar mudanças de distância e sustentar a lógica de estimativa e alerta.

**[SYS-03] Estimativa de distância**

O sistema deve estimar a distância entre os dispositivos a partir do valor do RSSI , utilizando um modelo de propagação de sinal calibrado e eficiente. Devem ser aplicadas técnicas de filtragem para reduzir ruídos e variações nas medições, assegurando maior estabilidade e precisão nos resultados.

*Rationale:* A estimativa de distância a partir do RSSI é essencial para converter a intensidade do sinal em uma métrica física (metros), viabilizando a definição de zonas de alcance e limites de segurança. O uso de um modelo de propagação calibrado e filtros adequados melhora significativamente a confiabilidade das estimativas, tornando o sistema mais robusto frente às variações do ambiente.

**[SYS-04] Comparação com limiar configurado**

O sistema deve comparar a distância estimada com um limiar de proximidade configurável e gerenciar o estado do sistema (seguro/afastado) conforme essa comparação.

*Rationale:* Essa lógica permite identificar quando o animal ultrapassa o raio seguro definido, funcionando como base para o acionamento de alertas.

**[SYS-05] Alerta Físico Local**

Quando o limiar for excedido, o sistema deve acionar um alerta físico sonoro (beep) na coleira, cessando automaticamente quando o animal retornar à zona segura.

*Rationale:* O alerta físico local é o principal mecanismo de aviso, permitindo resposta imediata e intuitiva, sem depender de conectividade adicional.

**[SYS-06] Estabilidade (Histerese)**

O sistema deve aplicar histerese no processo de decisão para evitar o acionamento repetitivo de alertas quando o animal oscila na região próxima ao limiar configurado.

*Rationale:* A histerese melhora a estabilidade e reduz falsos alarmes, evitando desconforto para o animal e irritação para o usuário.

**[SYS-07] Conexão Segura e Exclusiva**

O sistema deve estabelecer conexão BLE apenas com o dispositivo do tutor previamente emparelhado, ignorando sinais de outros dispositivos Bluetooth.

*Rationale:* Garante que a lógica de proximidade e alertas seja baseada somente no dispositivo autorizado, evitando interferências e problemas de segurança.

**[SYS-08] Configuração via BLE**

O sistema deve permitir ajuste remoto de parâmetros essenciais (como o raio seguro e intensidade do alerta) por meio da comunicação BLE com o dispositivo do tutor.

*Rationale:* A configuração sem fio aumenta a praticidade e evita ajustes físicos na coleira, simplificando o uso e a calibração.

**[SYS-09] Interface do Usuário (App)**O sistema deve disponibilizar um aplicativo móvel que permita ao usuário configurar os parâmetros essenciais da coleira e acompanhar o estado de conexão e proximidade. O aplicativo deve se comunicar com o dispositivo por meio de BLE, podendo apresentar interface simples ou funcionalidade mínima necessária para configuração e monitoramento.

*Rationale:* Um aplicativo móvel é fundamental para a usabilidade do sistema, pois permite ajustar parâmetros e visualizar informações sem necessidade de interação física direta com a coleira. Manter o aplicativo leve e modular garante flexibilidade para evolução futura sem comprometer o funcionamento básico.

**[SYS-10] Dome virtual de proximidade**O sistema deve estabelecer um raio de proximidade configurável, definindo um “dome virtual” ao redor do dispositivo do tutor. Quando esse limite for ultrapassado, o sistema deve alterar seu estado e acionar o alerta local.

*Rationale:*Formaliza a zona de segurança como comportamento funcional do sistema, assegurando resposta determinística e previsível ao afastamento.

### **Requisitos de Software**

**[SW-01] Leitura periódica de RSSI**O software deve realizar leituras periódicas do RSSI durante a conexão BLE, com frequência mínima suficiente para permitir estimativa de proximidade em tempo real.

*Rationale:* Sem medições periódicas, não é possível calcular distância nem acionar alertas de forma responsiva.

**[SW-02] Algoritmo de Estimativa de Distância**O software deve implementar um algoritmo para converter RSSI em distância estimada, aplicando filtragem adequada para suavizar variações de sinal.

*Rationale:* O RSSI é naturalmente instável; um modelo calibrado e filtragem são necessários para produzir estimativas consistentes.

**[SW-03] Lógica de comparação com limiar**O software deve comparar continuamente a distância estimada com o limiar configurado, determinando o momento em que o limite é ultrapassado e gerando eventos internos para acionar alertas.

*Rationale:* Essa lógica é o núcleo funcional do sistema de proximidade, permitindo identificar o estado “dentro” ou “fora” da zona segura.

**[SW-04] Controle de alerta físico**O software deve gerenciar o acionamento do buzzer conforme os eventos de ultrapassagem e retorno ao limite configurado.

*Rationale:* O alerta físico é a resposta imediata para indicar afastamento e ajudar na localização do PET. O controle preciso do alerta garante feedback imediato e reduz acionamentos desnecessários.

**[SW-05] Implementação de histerese**O software deve aplicar histerese na lógica de comparação, prevenindo alertas intermitentes quando o RSSI varia próximo ao limiar.

*Rationale:* Reduz “chattering”. A histerese evita instabilidade e melhora a experiência do usuário e do PET.

**[SW-06] Autenticação e Filtragem de Dispositivos**O software deve autenticar o dispositivo do tutor durante o emparelhamento BLE e ignorar sinais RSSI de dispositivos não autorizados.

*Rationale:* Evita interferências externas e mantém a confiabilidade e segurança do sistema.

**[SW-07] Configuração via BLE**O software deve expor características BLE configuráveis que permitam ajustar remotamente os parâmetros principais, como o limiar de distância e intensidade do alerta.

*Rationale:* Permite calibração simples via interface BLE, sem necessidade de aplicativo complexo.

**[SW-08] Interface do Usuário (App)**O software deve disponibilizar um aplicativo móvel funcional que permita ao usuário:

* Configurar parâmetros essenciais da coleira (como raio de proximidade e intensidade do alerta);
* Visualizar informações básicas de estado, incluindo conexão BLE, proximidade e alertas;
* Receber notificações e sinais de alerta do dispositivo.

Podendo ser implementado com interface gráfica mínima ou uma versão simplificada que execute as funções essenciais.

*Rationale:* O aplicativo móvel garante usabilidade e controle remoto do dispositivo, permitindo operação e monitoramento sem interação física direta. Manter o requisito flexível permite que a interface seja simples ou evolua no futuro, sem comprometer o funcionamento do principal do sistema.

**[SW-09] Implementação do Dome Virtual de Proximidade**O software deve implementar a lógica de dome virtual, responsável por:

* Manter o raio de proximidade configurável ao redor do dispositivo do tutor;
* Comparar continuamente a distância estimada com o limite configurado;
* Alterar o estado do sistema (seguro / afastado) quando o limite é ultrapassado;
* Acionar o alerta físico local conforme o estado.

A implementação deve ser modular, permitindo ajustes futuros no cálculo da distância, limites ou comportamento do alerta, sem depender de interfaces adicionais ou modos avançados.

*Rationale:* O dome virtual formaliza a zona de segurança da coleira, garantindo que a transição de estado e o acionamento do alerta sejam determinísticos e previsíveis, assegurando a função principal de manter o animal dentro do perímetro seguro.

### **Requisitos Não Funcionais**

**[SYS-NF01] Fonte de alimentação compacta**

O sistema deve utilizar uma fonte de alimentação de dimensões reduzidas (ex.: bateria tipo moeda, Li-Po de pequeno porte ou equivalente), garantindo operação contínua dentro das especificações de autonomia e funcionalidade definidas.

*Rationale:* Como o sistema será acoplado à coleira do animal, a fonte de energia deve ser pequena e leve, de modo a não causar desconforto ou restringir seus movimentos. A adoção de uma bateria compacta também reduz peso e facilita a manutenção, porém impõe limitações de capacidade e corrente, exigindo otimização do hardware e do firmware para baixo consumo e eficiência energética.

**[SYS-NF02] Eficiência energética**

A arquitetura do sistema deve minimizar consumo para viabilizar operação contínua com embalagem compacta e fonte de energia restrita.

*Rationale:* Baixo consumo é crítico para autonomia e viabilidade de uso diário.

**[SYS-NF03] Compacidade e leveza**

O sistema deve ser compacto e leve, adequado ao uso permanente na coleira.

*Rationale:* Dimensão e massa impactam diretamente no conforto, aceitação e segurança do animal.

**[SYS-NF04] Sinalização embarcada perceptível e segura**O sistema de alerta (beep) deve ser claramente perceptível para o tutor, sem causar incômodo excessivo ou risco ao animal.

*Rationale:* A sinalização deve cumprir o objetivo de alerta sem comprometer o bem-estar do PET.

**[SYS-NF05] Robustez da comunicação BLE**A comunicação BLE deve ser estável e resiliente a interferências, minimizando falsos alarmes por oscilações de link e perdas de conexão.

*Rationale:* Robustez do enlace é essencial para a confiabilidade do sistema de proximidade.

**[SYS-NF06] Evolução de TRL**O desenvolvimento deve progredir de protótipo em placa para hardware dedicado, com documentação e evidências de teste em cada fase, até atingir o nível de prontidão tecnológica definido pelo projeto. (TRL a ser atingida: 5)

*Rationale:* A evolução do TRL garante maturidade técnica, reduz riscos e prepara a solução para uso real.

**[SYS-NF07] PCB dedicada ao formato de coleira**A solução deve empregar PCB própria, considerando restrições de espaço, peso e robustez, além de boas práticas de RF e manufatura.

*Rationale:* Uma PCB dedicada garante não apenas o desempenho ideal da antena e a integridade mecânica, mas também permite utilizar apenas os componentes necessários, resultando em um design mais compacto, leve e eficiente.

**[SYS-NF08] Ergonomia e segurança animal**Materiais, acabamentos do sistema devem priorizar conforto e segurança, evitando arestas, aquecimento local e pontos de pressão.

*Rationale:* O bem‑estar do PET é requisito básico para adoção do produto.

**[SYS-NF09] Confiabilidade ambiental**O sistema deve operar de forma confiável em condições típicas externas (variações de temperatura, umidade, respingos).

*Rationale:* Uso em ambientes reais exige resistência a condições ambientais comuns.

**[SYS-NF10] Usabilidade**O aplicativo móvel deve ser simples, intuitivo e de fácil operação, permitindo que o usuário configure os parâmetros essenciais da coleira e monitore seu estado sem dificuldades. A interface deve priorizar clareza e eficiência, independentemente de ter front-end gráfico completo ou implementação mínima via BLE.

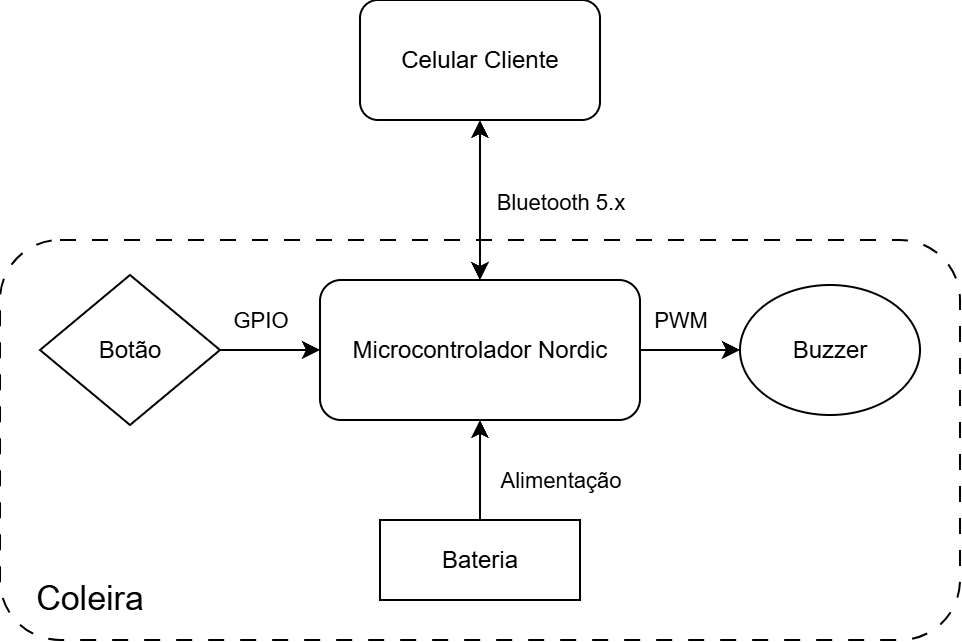
*Rationale:* Uma interface de fácil utilização reduz erros operacionais e agiliza a configuração, mesmo que a aplicação seja funcionalmente mínima.

**[SYS-NF11] Plataforma de hardware dedicada**O sistema não deve utilizar a placa de desenvolvimento BitDogLab, mas sim um microcontrolador e um rádio BLE adequados à aplicação, integrados em uma PCB própria.

*Rationale:* O uso de hardware dedicado garante otimização de consumo, redução de tamanho e peso, uma melhor qualidade do *Bluetooth*, além de maior confiabilidade e viabilidade atendendo às restrições do formato de coleira.

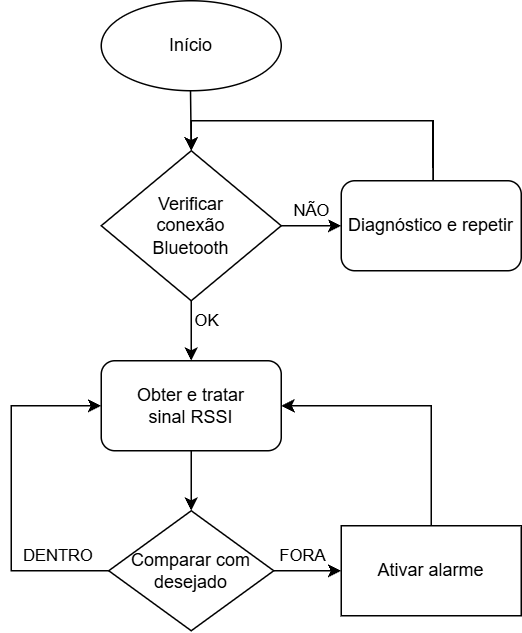
## Diagrama de Blocos

### Diagrama de Hardware

****

* O celular do cliente tem papel fundamental no sistema, ele que se comunicará com o dispositivo que estará junto ao cachorro.
* O microcontrolador é a peça central do sistema. O número de modelo ainda será definido, mas sabemos que será de um fabricante da Nordic, referência em microcontroladores com Bluetooth. Ele será responsável por receber o sinal RSSI, gerenciar a conexão Bluetooth e acionar o alarme quando o limiar for ultrapassado.
* O botão servirá para ações do usuário, como parear o celular com a coleira e talvez ligar/desligar o dispositivo, ações a serem definidas.
* O buzzer servirá como alarme, tocará quando o cachorro estiver mais longe do que o dono determinou que ele deveria.
* A bateria servirá para alimentar toda a parte do sistema presente na coleira, o objetivo é usar uma bateria pequena e que consiga entregar boa longevidade para o sistema.

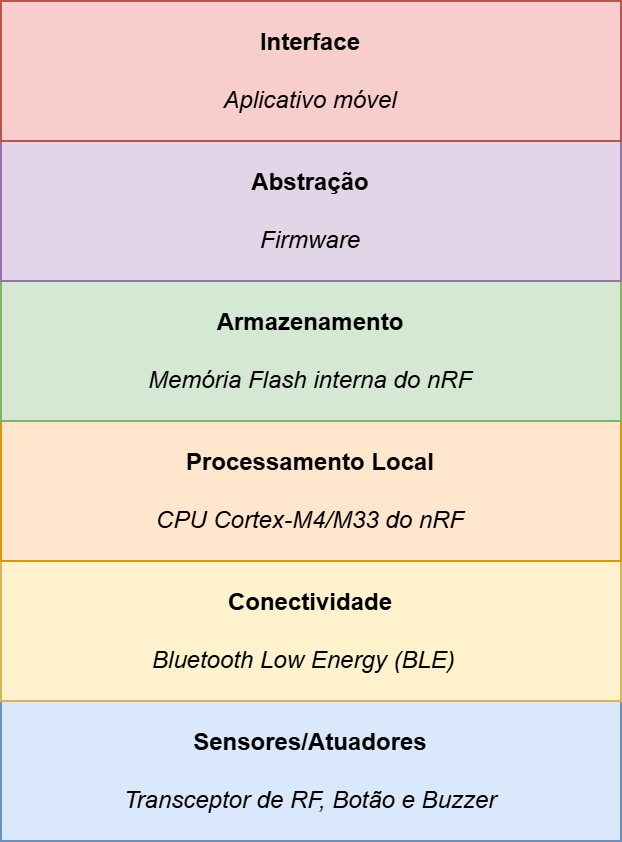
### Diagrama de Software

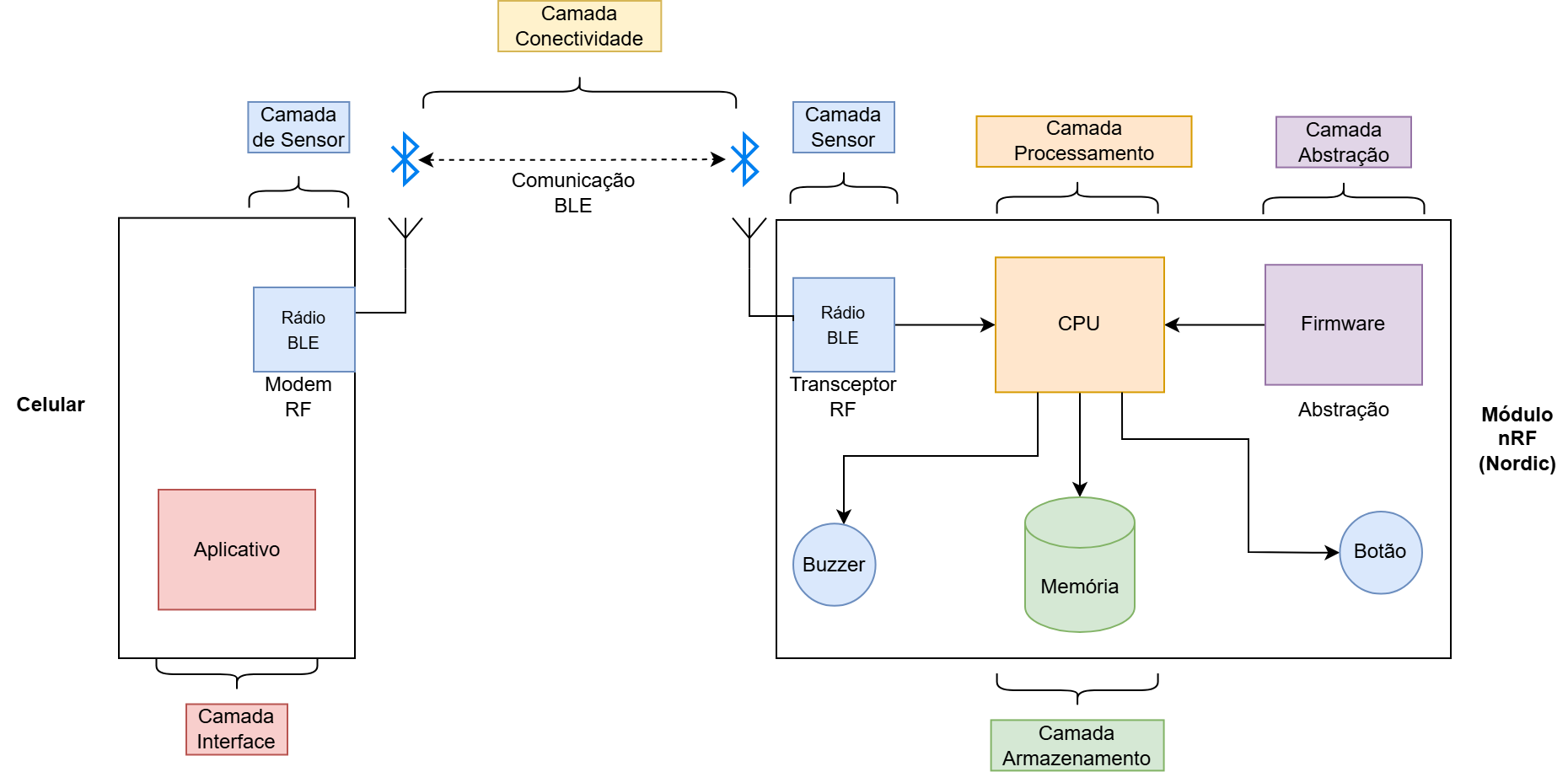
****

Ao iniciar o sistema, ele verificará se tem condições de conectar ao bluetooth da coleira, caso já exista algum pareamento anterior, caso seja a primeira vez, será necessário configurar. Caso consiga comunicar, segue o fluxo, do contrário, gera um diagnóstico de possíveis erros e tenta corrigi-los, após isso, tenta comunicar novamente e fica nesse loop até obter sucesso.

Com o pareamento concluído, o sinal RSSI obtido deve passar por uma série de tratamentos, como filtros e histerese para determinar qual distância está e quando alarmar. O alarme deve ser acionado quando tiver certeza de que a coleira está mais distante que o determinado pelo cliente, caso esteja dentro do dome setado, nada deve ser realizado. E assim se repete enquanto o sistema estiver ativo, verifica a conexão bluetooth, checa a distância e age.

## Diagrama de camadas





**1️) Camada de Sensores/Atuadores**

**Função:** Interação física com o ambiente (entrada e saída).

**Componentes:**

* **Buzzer (atuador)** → emite o beep de alerta quando o pet ultrapassa o limite de distância.
* **Botão (sensor)** → Servirá para ações do usuário
* **Transceptor de RF (sensor)** → O transceptor de RF (módulo responsável por TX/RX de pacotes BLE) possui internamente um circuito de detecção de potência de sinal. Esse circuito analógico mede a intensidade do sinal recebido (RSSI).

**Responsável por:**

* Executar a resposta física ao evento de afastamento e ações do usuário.
* Receber intensidade do sinal recebido via transceptor de RF.

**Requisitos relacionados:**

* [SYS-02]: Monitoramento de RSSI
* [SYS-05]: Alerta Físico Local
* [SW-01]: Leitura periódica de RSSI
* [SW-04]: Controle de alerta físico

**2️) Camada de Conectividade**

**Função:** Comunicação sem fio entre a coleira e o smartphone do tutor.

**Componentes:**

* **Rádio BLE 5.x** embutido no SoC **nRF da Nordic** (ex.: nRF52832 / nRF52840 / nRF54L15).
* **Antena integrada** (PCB antenna ou chip antenna).

**Responsável por:**

* Transmitir/receber pacotes BLE.
* Medir e disponibilizar o **RSSI** (via hardware do transceptor).
* Garantir conexão segura (autenticação e emparelhamento).

**Requisitos relacionados:**

* [SYS-01]: Tecnologia de comunicação
* [SYS-07]: Conexão Segura e Exclusiva

**3) Camada de Processamento Local**

**Função:** Processamento dos dados e lógica de decisão.

**Componente principal:** CPU Cortex-M4/M33 do nRF.

**Tarefas do firmware:**

* Leitura periódica do RSSI.
* Cálculo da distância estimada.
* Aplicação de filtros.
* Comparação da distância com o limiar configurado.
* Implementação de histerese.
* Acionamento e desligamento do buzzer.

**Requisitos relacionados:**

* [SYS-02]: Monitoramento de RSSI
* [SYS-03]: Estimativa de distância
* [SYS-04]: Comparação com limiar
* [SYS-05]: Alerta Físico Local
* [SYS-06]: Estabilidade (Histerese)
* [SW-01]: Leitura periódica de RSSI
* [SW-02]: Algoritmo de Estimativa de Distância
* [SW-03]: Comparação com limiar
* [SW-04]: Controle de alerta físico
* [SW-05]: Implementação de histerese

**4) Camada de Armazenamento**

**Função:** Guardar parâmetros e dados essenciais.

**Componentes:** Memória Flash interna do nRF (ex.: 512 KB – 1 MB)

**Dados armazenados:**

* Limiar configurado (raio de proximidade).
* Intensidade do alerta.
* Dados de calibração do modelo de distância.
* Identificador do tutor emparelhado.

**Requisitos relacionados:**

* [SW-02]: Algoritmo de Estimativa de Distância
* [SW-03]: Comparação com limiar
* [SW-04]: Controle de alerta físico
* [SW-05]: Implementação de histerese
* [SW-06]: Autenticação e filtragem de dispositivos

**5️) Camada de Abstração**

**Função:** interface entre hardware e software.

**Componentes / Software:**

* HAL (Hardware Abstraction Layer) da Nordic SDK.
* Drivers BLE, GPIO, PWM, Timer.
* Middleware BLE Stack (Zephyr BLE stack).

**Responsável por:**

* Esconder detalhes do hardware.
* Oferecer APIs seguras e padronizadas para as camadas superiores.
* Implementar callbacks e eventos BLE (conexão, leitura RSSI, etc.)

**Requisitos relacionados:**

* [SW-02]: Algoritmo de Estimativa de Distância
* [SW-03]: Comparação com limiar
* [SW-04]: Controle de alerta físico
* [SW-05]: Implementação de histerese
* [SW-06]: Autenticação e filtragem de dispositivos
* [SW-07]: Configuração via BLE
* [SW-09]: Implementação do Dome Virtual de Proximidade

**6️) Camada de Interface**

**Função:** Interação com o usuário (tutor do animal).

**Componentes:** Aplicativo móvel BLE

**Responsável por:**

* Comunicação BLE com a coleira
* Envio de comandos de configuração.
* Exibir status de conexão e RSSI.
* Mostrar estado da coleira (seguro / afastado).

**Requisitos relacionados:**

* [SYS-09]: Interface do Usuário (App)
* [SW-08]: Implementação Interface do Usuário

## Interfaces e Dependências principais

**Bluetooth Low Energy (BLE)**:  
Um *periférico* (coleira) anuncia sinais BLE, enquanto uma *central* (celular) mede a intensidade do sinal (RSSI) para estimar a distância e acionar alertas quando o cão se afastar além do limite.

**As interfaces e dependências principais** devem, neste estágio do projeto, ser mantidas de forma modular, preservando uma arquitetura genérica que permita evoluções futuras conforme a escolha do hardware (por exemplo: nRF54L15, XIAO BLE, ESP32, etc.).

**Arquitetura de Alto Nível**

**Componentes principais:**

|  |  |  |
| --- | --- | --- |
| **Camada** | **Papel** | **Implementação esperada** |
| **Dispositivo BLE (coleira)** | Anuncia *advertisements BLE* com ID único e, opcionalmente, GATT para status/configuração | nRF54L15 / Zephyr / C SDK, Bare Metal, etc. |
| **Aplicativo Móvel (tutor)** | Detecta o beacon BLE, mede RSSI, calcula distância e dispara alertas caso necessário | Android (Java/Kotlin) |

**Interfaces principais entre os componentes**

**Interface 1: BLE Advertising (Coleira → Celular)**

**Função:** transmitir dados mínimos e frequentes.

|  |  |  |
| --- | --- | --- |
| **Campo** | **Tipo** | **Descrição** |
| UUID (ex: 0x180F) | 16-bit | Serviço base |
| ID do pet | 4 bytes | Identificador único |
| Nível de bateria | 1 byte | 0–100% |
| Estado (normal/alerta) | 1 byte | Status da coleira |
| TX Power | 1 byte | Para calibrar distância |
| RSSI (medido pelo celular) | — | usado apenas pelo central |

\*\*Essa interface é **somente broadcasting**. O celular apenas escuta e mede RSSI.

**Interface 2: BLE GATT (opcional)**

**Função:** configuração ou diagnóstico via BLE (quando o celular conecta manualmente).

|  |  |  |  |
| --- | --- | --- | --- |
| **Serviço** | **UUID** | **Características** | **Ação** |
| **AmigoPerto Service** | 128-bit custom | BuzzerControl, BatteryLevel, DeviceID, FirmwareVersion | Leitura/escrita BLE |
| **DFU / OTA Service** | padrão Nordic | Atualização de firmware via app é uma funcionalidade já inclusa na placa pela fabricante (opcional) | OTA update |

\*\*Esta interface só é usada quando o app “conecta” ao dispositivo — para configurar distância, atualizar firmware, etc.

**Dependências principais por camada** (A placa sugerida pode ser alterada; contudo, enquanto o fabricante permanecer o mesmo, os requisitos se mantêm. Caso se opte por outro fabricante, será necessária uma nova avaliação do sistema operacional, da IDE e das bibliotecas e drivers necessários.)

**Coleira (nRF54L15 / Zephyr / Bare Metal)**

* zephyr/bluetooth/bluetooth.h
* zephyr/bluetooth/hci.h
* zephyr/bluetooth/gatt.h
* drivers/gpio.h (para buzzer)
* drivers/sensor.h (opcional — bateria)
* device.h / pm.h (baixo consumo)

Dependências (build):

* nRF Connect SDK com Zephyr RTOS  
   (as linhas **nRF52**, **nRF53** e **nRF54** são suportadas — porém, **apenas a nRF54** oferece também um ambiente *bare metal*, permitindo desenvolver sem RTOS, para aplicações que exigem latência mínima ou uso muito restrito de recursos)<https://devzone.nordicsemi.com/nordic/nordic-blog/b/blog/posts/a-technical-dive-into-the-nrf-connect-sdk-bare-metal-option?utm_campaign=Product%20Update%20Notifications%20-%20PUN&utm_medium=email&_hsenc=p2ANqtz--TffQVOcyTuIroNpXDhY9aHm1UqySXID6ayAnrSnyAuNY9lisnU5Qpi1USCG_Ze6z5lOq-ITJ0NQ2jjGnpdPH1LJdV7f6O9vpqClnRXTc8tbj-zCY&_hsmi=383875339&utm_content=383876137&utm_source=hs_email>
* west, cmake, ninja
* bt\_enable(), bt\_le\_adv\_start()

**Aplicativo Android** (lib´s nativas)

* BluetoothLeScanner, ScanCallback, ScanResult
* BluetoothGatt (para conexões)
* RSSI e TxPowerLevel para cálculo da distância
* Dependências:
  + Android SDK 30+ ou superior
  + androidx.bluetooth (Jetpack)
  + LiveData / Room (armazenamento local)
  + OkHttp / Retrofit (para API)

Estrutura sugerida do projeto



**Eventos e fluxos principais**

|  |  |  |  |
| --- | --- | --- | --- |
| **Evento** | **Origem** | **Destino** | **Ação** |
| Advertising detectado | Coleira | App | Calcular RSSI |
| RSSI < threshold | App | Buzzer (via BLE write) | Ativar som |
| RSSI > threshold | App | Buzzer | Desligar |
| Alerta persistente | App | Durante distanciamento identificado | Registrar alerta |
| Nova config distância | App | Coleira | Atualizar GATT value |

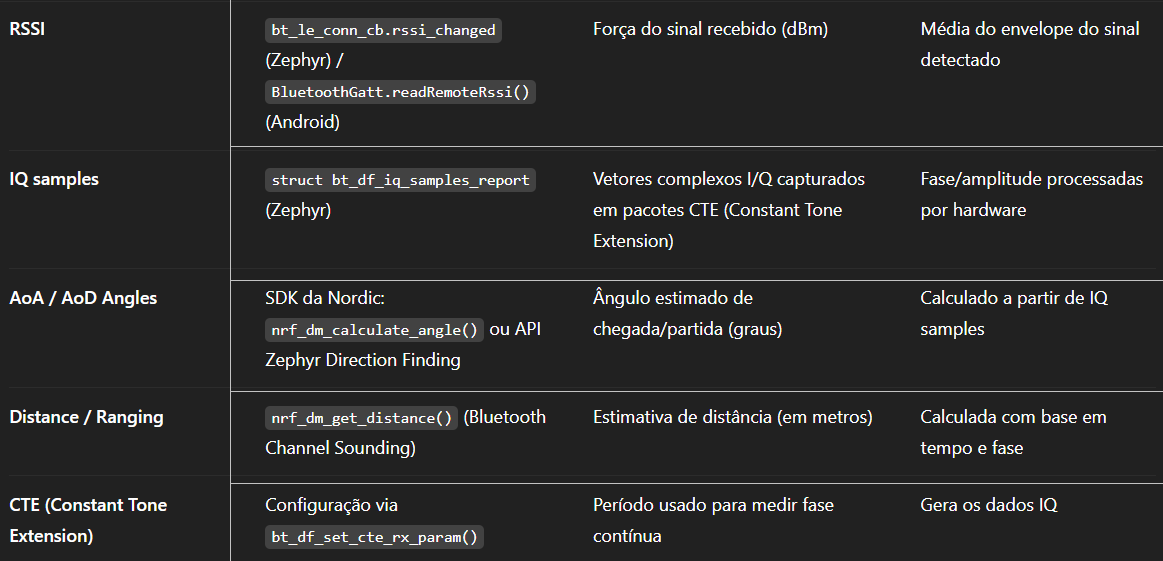
**Uso de LE Codec**

**LE 1M, LE 2M e LE Coded (**explicado intuitivamente**)**

|  |  |  |  |
| --- | --- | --- | --- |
| **PHY** | **Velocidade** | **Alcance** | **Uso típico** |
| **LE 1M** | Médio (1 Mb/s) | Padrão (~10 m) | Conexões gerais BLE |
| **LE 2M** | Alto (2 Mb/s) | Curto | Transferência rápida de dados, p.ex. firmware update |
| **LE Coded** | Baixo (125–500 kb/s) | Longo (~100 m ou mais) | Sensores distantes, comunicação IoT de longo alcance |

O **LE Coded** usa **Forward Error Correction (FEC)** e **redundância** para permitir comunicação mesmo com sinais fracos — ideal para sensores em campo.

A stack BLE (exemplo: **nRF Connect SDK**, **Zephyr**, ou **Android Bluetooth API**) **métricas derivadas e interpretadas**:



**Sugestão de MVP**

1. Firmware BLE simples anunciando ID e TX Power

2. App Android detectando advertising e medindo RSSI

3. Buzzer aciona na coleira

4. (Opcional) Escrever para GATT → já ativa buzzer

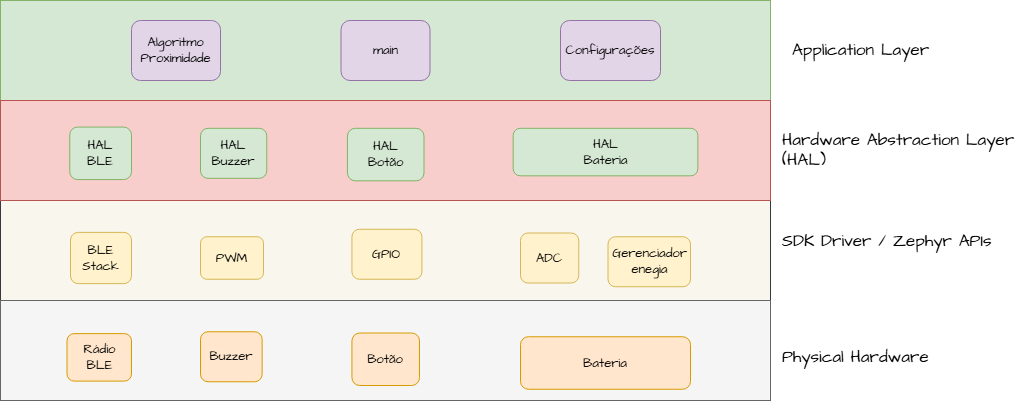
5. (Opcional) Gráfico básico de distância em tempo real.

# Etapa 1 Semana 3

## Plano de Abstração (HAL)

O objetivo da HAL (Hardware Abstraction Layer) é isolar o firmware de aplicação das particularidades do hardware e do SDK, facilitando testes e simulações (mocks), além de permitir a substituição do hardware ou da pilha de software sem a necessidade de reescrever a lógica de alto nível. Essa camada garante o atendimento aos requisitos funcionais e não funcionais do projeto. Para isso, é essencial primeiramente ter uma visão geral da arquitetura de software:

## Arquitetura de Software



🟩 **1. Application Layer**

**Componentes:**

* **main**
  + Inicializa o sistema, configura os periféricos, o BLE e inicia as tarefas (Zephyr) ou o *super loop* no bare-metal.
  + Faz o link entre as camadas HAL e o algoritmo de proximidade.
  + Cumpre requisitos [SW-01] a [SW-04], inicializando o loop principal e os módulos de leitura RSSI, estimativa de distância e controle do buzzer.
* **Algoritmo de Proximidade**
  + Implementa:  
    - Leitura periódica do RSSI ([SW-01])
    - Estimativa de distância via modelo calibrado + filtragem ([SW-02])
    - Comparação com o limiar e histerese ([SW-03], [SW-05])
    - Acionamento do alerta sonoro ([SW-04])
  + Define a lógica do “**dome virtual de proximidade**” ([SW-09], [SYS-10]).
  + Essa lógica será executada ciclicamente, atualizando o estado (“seguro” / “afastado”).
* **Configurações**
  + Gerenciar os parâmetros de operação da coleira, incluindo configurações locais e remotas via BLE (Advertising ou GATT).
  + Inicialização e carregamento de parâmetros
  + Atualização dinâmica via BLE
  + Salvamento em memória não volátil
  + Permite alterar parâmetros como:  
    - Limiar de distância
    - Intensidade do alerta
    - Intervalo de leitura RSSI
  + **BLE Advertising:** transmite parâmetros básicos periodicamente;
  + **BLE GATT (opcional):** permite leitura/escrita remota conforme [SW-07] e [SYS-08].

**🟥 2. Hardware Abstraction Layer (HAL)**

Camada intermediária que abstrai o acesso direto ao hardware, expondo APIs simples e independentes do driver específico.

**Módulos:**

* **HAL BLE**
  + Encapsula as funções de comunicação BLE (advertising, conexão, leitura de RSSI, GATT).
  + Permite que a camada de aplicação apenas solicite “obter RSSI” ou “atualizar parâmetro” sem conhecer o stack BLE.
  + Implementa segurança de emparelhamento[SYS-07], [SW-06]).
* **HAL Buzzer**
  + Controla o PWM ou GPIO responsável pelo som.
  + Oferece funções simples como buzzer\_on(intensidade) e buzzer\_off().
  + Relaciona-se com [SYS-05], [SYS-NF04].
* **HAL Botão**
  + Permite entrada física (opcional) para reset, pareamento ou modo de configuração.
  + Usa GPIOs via Zephyr drivers.
* **HAL Bateria**
  + Lê a tensão via ADC, gerencia estados de carga e fornece informação de nível de bateria para o app.
  + Cumpre [SYS-NF01], [SYS-NF02].

🔹 **Função principal da HAL:** Permitir que a camada de aplicação interaja com os periféricos **sem depender das APIs específicas do Zephyr ou Nordic SDK**, facilitando manutenção e portabilidade.

**🟨 3. SDK Driver / Zephyr APIs**

Camada fornecida pelo Zephyr e pelo nRF Connect SDK.

**Principais interfaces:**

* **BLE Stack (Zephyr BT)**
  + Implementa o protocolo BLE 5.x (advertising, GATT, RSSI).
  + Cumpre [SYS-01], [SYS-07], [SYS-08], [SYS-09].
* **PWM Driver**
  + Controla o buzzer.
  + Fornece interface para HAL\_Buzzer.
* **GPIO Driver**
  + Lida com entradas e saídas digitais (botões).
* **ADC Driver**
  + Faz leitura da bateria, utilizada em HAL\_Bateria.
* **Gerenciador de Energia (Zephyr Power Management)**
  + Controla modos *sleep*, *idle*, *active* para reduzir consumo[SYS-NF02]).

Esses drivers são utilizados pela HAL e, no caso do uso do Zephyr RTOS, são gerenciados pelo sistema operacional, que cuida do task scheduling, event handling e do baixo consumo de energia. No caso de uma implementação bare-metal, não há escalonador de tarefas, o controle é feito pelo super loop principal e por interrupções. A temporização baseia-se em timers de hardware, e o gerenciamento de energia e eventos é responsabilidade direta do firmware. Nessa abordagem, os drivers utilizam funções fornecidas pelo SDK Connect da Nordic.

**🟫 4. Physical Hardware**

Camada mais baixa, composta pelos componentes físicos conectados ao módulo nRF.

**Elementos:**

* **Rádio BLE** — transceptor integrado do SoC (2,4 GHz).
* **Buzzer** — alerta sonoro controlado via PWM.
* **Botão** — entrada física
* **Bateria** — fonte de energia do sistema.

Como explicado, o sistema pode operar em **duas abordagens de configuração BLE**: via **Advertising** ou via **GATT**.

* **Modo Advertising:**
  + Nesse modo, a coleira apenas transmite pacotes de *advertising* (broadcast), e o celular realiza a leitura do RSSI desses pacotes para estimar a distância.
  + É um modelo unidirecional (*coleira → celular*).
  + O algoritmo de proximidade é executado no aplicativo móvel, que decide quando acionar o alerta.
  + Entre as transmissões, a coleira entra em deep sleep, otimizando o consumo de energia.
* **Modo GATT:**
  + Nesse modo, a comunicação é bidirecional, permitindo que o celular configure parâmetros da coleira (como limites, potência ou intervalos).
  + O algoritmo de proximidade pode ser executado diretamente no hardware da coleira, tornando-a mais autônoma.

Dependendo da abordagem, a estrutura de diretórios da arquitetura de software pode variar:

***Modo Advertising:***

amigo\_perto/

├── src/

│ ├── app/

│ │ ├── main.c # Aplicação principal

│ │ └── config.c/.h # Configurações BLE

│ │

│ └── hal/

│ ├── ble\_hal.c/.h # Advertising wrapper

│ ├── buzzer\_hal.c/.h # PWM wrapper

│ └── battery\_hal.c/.h # ADC wrapper

│ └── botao\_hal.c/.h # Wrapper GPIO pro Botão

│

├── mobile\_app/ # App móvel (separado)

│ └── proximity.js/.dart # Lógica RSSI→distância (celular)

│

├── boards/

│ └── nrf52832\_collar.overlay

│

├── CMakeLists.txt

├── prj.conf

└── README.md

***Modo GATT:***

amigo\_perto/

├── src/

│ ├── main.c # Aplicação principal

│ ├── proximity.c/.h # Dome virtual + RSSI

│ ├── gatt\_config.c/.h # GATT Server

│ │

│ └── hal/

│ ├── ble\_hal.c/.h # Wrapper Zephyr BLE

│ ├── buzzer\_hal.c/.h # Wrapper PWM

│ └── battery\_hal.c/.h # Wrapper ADC

│ └── botao\_hal.c/.h # Wrapper GPIO pro Botão

│

├── boards/

│ └── nrf52832\_collar.overlay # Device Tree Overlay

│

├── CMakeLists.txt

├── prj.conf # Kconfig do Zephyr

└── README.md

## Camada HAL (Hardware Abstraction Layer)

### HAL do BLE

|  |  |  |
| --- | --- | --- |
| **Função / Estrutura** | **Descrição** | **Observações** |
| **ble\_adv\_data\_t** | Estrutura de dados transmitida no advertising packet | Contém UUID do serviço, ID do PET, nível de bateria, estado e potência TX |
| **ble\_adv\_event\_cb\_t** | Callback opcional para eventos de advertising | Pode ser usada para tratar eventos de início/parada do advertising |
| **ble\_hal\_init()** | Inicializa o BLE em modo broadcaster | Parâmetros: nome do dispositivo, ID do PET, potência TX; retorna 0 em sucesso |
| **ble\_hal\_start\_advertising()** | Inicia o advertising com intervalo definido | Parâmetro: intervalo em ms (100–10000) |
| **ble\_hal\_stop\_advertising()** | Para o advertising | Retorna 0 em sucesso |
| **ble\_hal\_update\_adv\_data()** | Atualiza todos os dados do advertising packet | Parâmetro: ponteiro para ble\_adv\_data\_t |
| **ble\_hal\_update\_battery()** | Atualiza apenas o nível de bateria | Envio otimizado, valor 0–100% |
| **ble\_hal\_update\_state()** | Atualiza o estado da coleira | 0 = normal, 1 = alerta |
| **ble\_hal\_get\_tx\_power()** | Retorna a potência de transmissão configurada | Em dBm |
| **ble\_hal\_is\_advertising()** | Verifica se o advertising está ativo | Retorna true se ativo, false se parado |
| **BLE\_ADV\_SERVICE\_UUID** | UUID base do serviço | Ex.: 0x180F (Battery Service) |

### 

***Modo GATT:***

|  |  |  |
| --- | --- | --- |
| **Função / Estrutura** | **Descrição** | **Observações** |
| **ble\_rssi\_cb\_t** | Callback para leitura periódica de RSSI | Recebe valor RSSI em dBm |
| **ble\_conn\_cb\_t** | Callback para eventos de conexão/desconexão | Recebe ponteiro para conexão e estado |
| **ble\_central\_hal\_init()** | Inicializa BLE em modo central | Parâmetros: nome do dispositivo, endereço do tutor autorizado; retorna 0 em sucesso |
| **ble\_central\_hal\_start\_scan()** | Inicia scan BLE | Conecta somente ao dispositivo autorizado; intervalo em ms |
| **ble\_central\_hal\_stop\_scan()** | Para o scan BLE | Sem retorno de erro |
| **ble\_central\_hal\_connect()** | Conecta ao dispositivo emparelhado | Retorna 0 em sucesso, negativo em erro |
| **ble\_central\_hal\_disconnect()** | Desconecta do dispositivo ativo | — |
| **ble\_central\_hal\_get\_connection()** | Obtém ponteiro para conexão ativa | Retorna NULL se não conectado |
| **ble\_central\_hal\_get\_rssi()** | Lê RSSI da conexão ativa | Parâmetro: ponteiro para int8\_t; retorna 0 em sucesso |
| **ble\_central\_hal\_set\_rssi\_callback()** | Configura callback periódico de RSSI | Parâmetros: função callback e intervalo em ms |
| **ble\_central\_hal\_set\_conn\_callback()** | Configura callback de conexão/desconexão | — |

### HAL do Buzzer

|  |  |  |
| --- | --- | --- |
| **Função / Estrutura** | **Descrição** | **Observações** |
| **buzzer\_hal\_init()** | Inicializa o hardware do buzzer | Deve ser chamada antes de ler tensão ou nível; retorna 0 em sucesso |
| **buzzer\_hal\_on()** | Liga o buzzer com frequência e volume definidos | Parâmetros: freq\_hz (Hz), volume (0–100%); retorna 0 em sucesso |
| **buzzer\_hal\_off()** | Desliga o buzzer | Retorna 0 em sucesso |
| **buzzer\_hal\_beep()** | Emite count bipes de duração definida | Parâmetros: count (número de bips), duration\_ms (duração de cada beep em ms); retorna 0 em sucesso |

### HAL da Bateria

|  |  |  |
| --- | --- | --- |
| **Função / Estrutura** | **Descrição** | **Observações** |
| **battery\_hal\_init()** | Inicializa a medição da bateria | Deve ser chamada antes de ler tensão ou nível; retorna 0 em sucesso |
| **battery\_hal\_get\_voltage\_mv()** | Retorna a tensão da bateria em milivolts | Tipo uint16\_t |
| **battery\_hal\_get\_level\_percent()** | Retorna o nível da bateria em porcentagem | Tipo uint8\_t, valor de 0 a 100% |

### 

### HAL do Botão

|  |  |  |
| --- | --- | --- |
| **Função / Estrutura** | **Descrição** | **Observações** |
| **button\_hal\_init()** | Inicializa o botão e configura interrupção | Deve ser chamada antes de usar; retorna 0 em sucesso |
| **button\_hal\_set\_callback()** | Configura callback para eventos do botão | Função callback é chamada quando o botão é pressionado ou liberado; |

## 

## **Comparação microcontrolador**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parâmetro** | **IN100** | **nRF52840** | **nRF52832** | **nRF54L15** |
| Bluetooth | 5.3 | 5.4 | 5.4 | Até 6.0 |
| Potência de Transmissão (TX) | Até +5dBm | Até +8dBm | Até +4dBm | Até +8dBm |
| Sensibilidade de Recepção (RX) | Valor não encontrado | Até -103dBm | Até -96dBm | Até -96dBm |
| Consumo de Corrente do Sistema para 3V | 650nA (Sleep Mode) | 0,4 μA – Sleep Mode  1,86 μA – Sistema OFF, com RAM totalmente retida  0,97 μA – Sistema ON, sem retenção de RAM  2,35 μA – Sistema ON, com RAM totalmente retida  3,16 μA – Sistema ON, com RAM totalmente retida + RTC ativo | 0,3 μA – Sleep Mode  1,2 μA – Todos os periféricos em modo ocioso (IDLE)  1,6 μA – Todos os periféricos em modo ocioso (IDLE) com 32 kHz + RTC ativos  20 nA por 4 KB – Retenção de RAM | 0,7 μA – System OFF  0,9 μA – with GRTC wakeup  1,7 μA – ON com 96kb de RAM |
| Tensão de Alimentação | 1.1V ~ 3.6V | 1.7V ~ 5.5V | 1.7V ~ 3.6V | 1.7V ~ 3.6V |
| Complexidade de desenvolvimento | Muito baixa: “sem código”, configuração via GUI | Média, muitos exemplos prontos, necessita integração com Zephyr, SDK VSCODE | Média, alguns exemplos prontos, necessita integração com Zephyr, SDK VSCODE | Média, alguns exemplos prontos, pode utilizar bare metal além do Zephyr, SDK VSCODE |
| Preço | Baixo: $6.10 | Baixo (fácil encontrar): R$40 | Baixo (difícil encontrar): R$30 | Médio (dificuldade média): R$90 |
| Tempo Estimado da Bateria (consumo médio) | Até 1 ano | Entre 3 e 6 meses | Entre 5 e 8 meses | Entre 2 a 6 meses |
| Recursos de Hardware | Baixo | Alto | Médio | Alto |

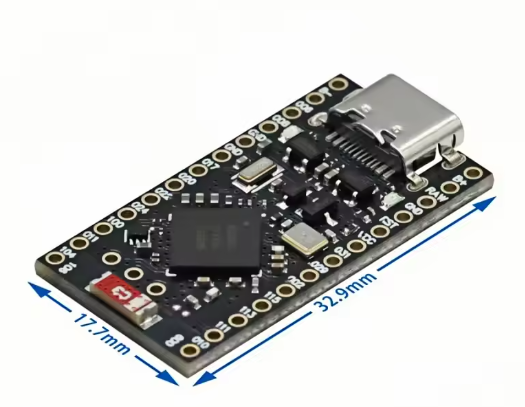
## **Justificativas técnicas e econômicas**

Para nosso sistema o consumo de energia é bem importante, dado que desejamos alimentar tudo com uma bateria tipo pilha, neste caso a escolha ideal seria o IN100, porém outro fator contribuiu para não o comprarmos, o fato de seu preço aumentar bastante por conta de frete e impostos, uma vez que seus módulos somente são encontrados em sites superfaturados, como a sparkfun ou mouser. Com isso definimos que usaremos os microcontrolados da Nordic Semiconductor, uma vez que nos possibilita grande quantidade de exemplos, placas prontas e SDK de fácil utilização. Entre os candidatos, destaque o nRF52840, que contempla a maior variação de módulos e exemplos prontos, justificando nossa primeira escolha. A segunda ficou entre a nova geração e a antiga, escolhemos a nova com o nRF54L15, com o ponto positivo principal sendo a possibilidade de programar em bare metal usando versões de SDK recentes e atualizadas, já que os outros das famílias antigas são atualizados com o sistema RTOS Zephyr. Os outros componentes são de fácil acesso e não precisamos definir algum PART NUMBER por agora, trata-se de um buzzer e uma bateria.

Com isso compramos os módulos:

**nRF54L15**

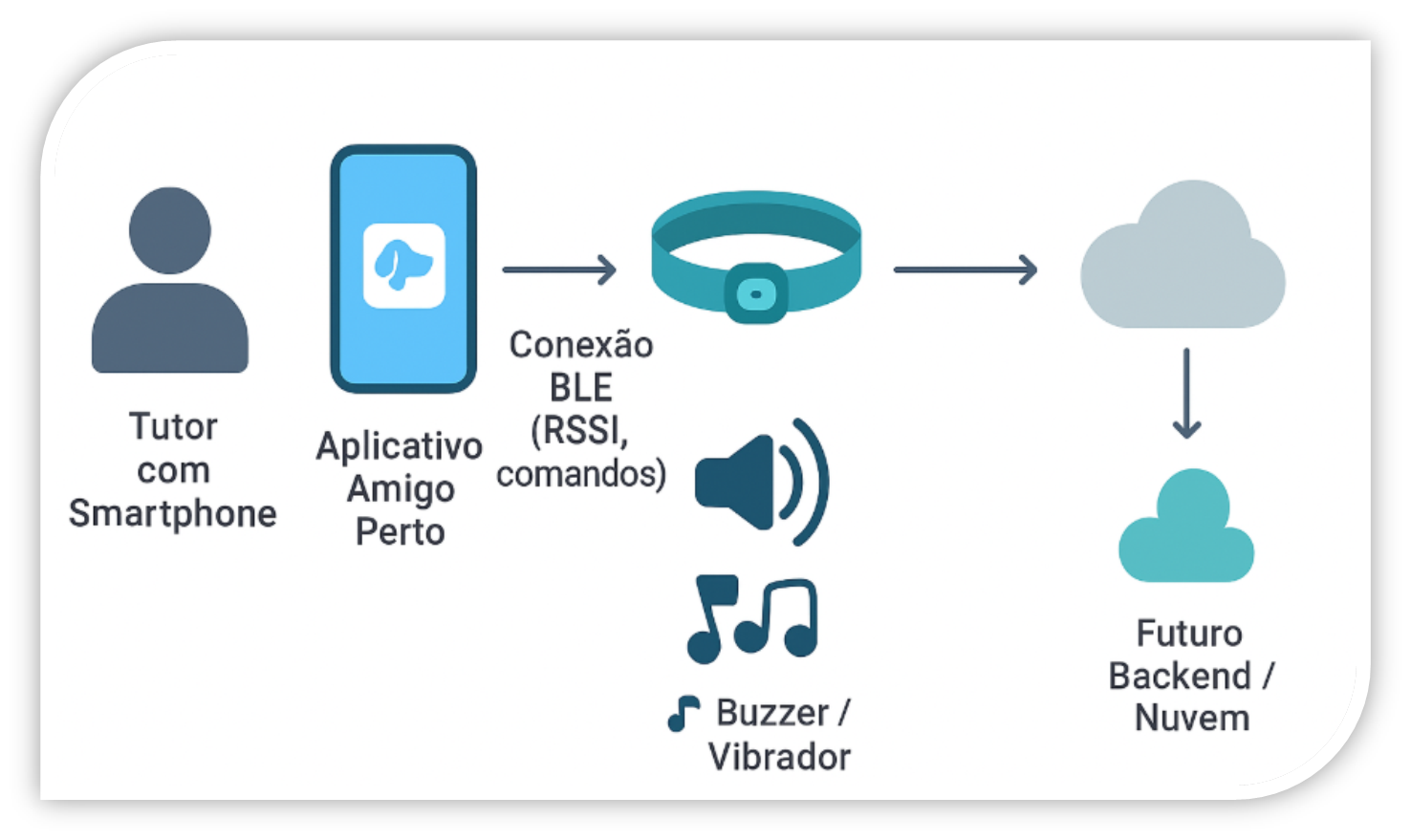
[](https://pt.aliexpress.com/item/1005009461952447.html?spm=a2g0o.detail.0.0.163bjs7Xjs7X0d&mp=1&pdp_npi=5%40dis%21BRL%21BRL%2098.89%21BRL%2098.89%21%21BRL%2098.89%21%21%21%402101d93a17616709753324917e32a1%2112000049188790645%21ct%21BR%214272877047%21%211%210&_gl=1*1s1nhhc*_gcl_aw*R0NMLjE3NjE2NzA5NjIuQ2p3S0NBancwNEhJQmhCOEVpd0E4akdOYmNfb3p6ZlZfME1tSElRYkNpVV80UG9OTE1qOXMzZXRkQXRaT1p1YTBqanRacTNraENfWEhob0N3LWdRQXZEX0J3RQ..*_gcl_dc*R0NMLjE3NjE0MDc5NjYuQ2p3S0NBanc2dkhIQmhCd0Vpd0FxNHp2QTZyWlpIMUk1THVabDZ5LWpzNGpIUHJkMTMxYzk1UXNOVzZNX1NWdnEtMUZBckl1Zk1iUjF4b0N3ckFRQXZEX0J3RQ..*_gcl_au*MTg5MTQwODYwMS4xNzU5MDYyMjg3*_ga*MTk3MTE1NzI1NS4xNzU5MDYyMjg3*_ga_VED1YSGNC7*czE3NjE2NzA3NjUkbzMzJGcxJHQxNzYxNjcwOTc3JGozMyRsMCRoMA..&gatewayAdapt=glo2bra)

**nRF52840**

## **Diagrama consolidado do sistema**

**o diagrama de arquitetura funcional simplificado**, mostrando domínio pretendido:

* os **atores principais** (tutor e coleira);
* o **meio de comunicação** (Bluetooth Low Energy, BLE);
* os **componentes físicos e lógicos** (aplicativo, firmware, buzzer/vibrador);
* e **camada futura opcional** (backend/nuvem).



**Componentes principais:**

1. **Coleira Inteligente (Dispositivo BLE)**
   * Microcontrolador: *Nordic nRF52840 / nRF54L15 / nRF52840 / IN100 (NanoBeacon)*
   * Atuadores: *Buzzer + Vibrador*
   * Energia: *Bateria (CR…)*
   * Comunicação: *Bluetooth Low Energy*
   * Firmware embarcado:  
     + Detecta intensidade de sinal RSSI
     + Aciona alerta (som/vibração) quando ultrapassa o limite definido
     + Recebe comandos do app (por exemplo: tocar ou parar buzzer)
2. **Aplicativo Mobile (Tutor)**
   * Detecta o dispositivo BLE
   * Mede RSSI (força do sinal) e calcula distância aproximada
   * Define **limite configurável de distância**
   * Aciona alerta no celular (som/vibração)
   * Envia **comando BLE** de retorno para a coleira (alerta do pet)
   * Registra **eventos locais** (data, hora, afastamento, retorno)
3. **Futuras Expansões (fase posterior)**
   * **Sincronização com nuvem (**via REST API ou WebSocket)
   * **Histórico de eventos e localização**
   * **Painel web** para monitoramento
   * **Controle de múltiplos pets/dispositivos**