Politécnico de Coimbra

Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática

Tecnologias e Aplicações Móveis

Ano Letivo 2025/2026

Sistemas Embebidos – Análise e planeamento

Elaborado em: 18/10/2025

Gabriel Lopes, a2019153399

Índice

Lista de Figuras

1 Introdução

O presente documento visa estruturar e clarificar a primeira etapa de um projeto de Sistemas Embebidos. Assim, funcionará como um guia de referência para todo o ciclo de desenvolvimento, tornando-se essencial para definir a visão global do sistema, estabelecer a arquitetura tecnológica adequada e organizar o trabalho de forma lógica e eficiente.

Para além de identificar os componentes de hardware necessários e justificar as escolhas tecnológicas, este relatório procura também delinear um planeamento claro, desde a conceção inicial até à implementação final.

Assim, este planeamento não se limita apenas à descrição inicial do projeto, tendo o objetivo de estabelecer uma orientação estratégica, garantindo que cada decisão técnica esteja fundamentada e alinhada com os objetivos definidos.

2 Definição do projeto

SafeSenior, é um sistema embebido baseado em Arduino com conectividade Wi-Fi, concebido para aumentar a segurança e a qualidade de vida de idosos em contexto doméstico. O sistema visa combinar sensores de movimento e sinais vitais com um botão físico de SOS e um módulo de localização, permitindo deteção automática de quedas, acompanhamento de parâmetros de saúde e comunicação imediata de emergências para familiares ou cuidadores.

Um objetivo diferenciador deste dispositivo seria fornecer, ao idoso, funcionalidades básicas (*e.g.* Alarme, relógio, medida de tensão), para incentivar o seu uso constante. Desta forma, mantendo o interesse do utilizador, o dispositivo mantém-se na sua posse o máximo de tempo possível. Estas funcionalidades, para o atual contexto, poderão ser de carácter opcional, sendo o principal objetivo desenvolver o sistema de deteção de eventos perigosos para idosos.

2.1 Objetivos

O SafeSenior tem como finalidade proporcionar uma monitorização contínua e inteligente de idosos em ambiente doméstico, com foco na prevenção de acidentes e na deteção precoce de problemas de saúde.

O problema que se propõe a resolver prende-se com a alta incidência de acidentes domésticos entre idosos, especialmente quedas, que são frequentemente graves e podem ter consequências fatais ou debilitantes. Além disso, muitos idosos vivem sozinhos ou têm vigilância limitada, o que dificulta a reação rápida a emergências.

A função do sistema é, portanto, detetar eventos críticos em tempo real, notificar familiares ou cuidadores imediatamente, e fornecer histórico de saúde e eventos, permitindo intervenções rápidas e tomadas de decisão informadas.

2.2 Componentes

Esta secção descreve todos os componentes de hardware necessários para o funcionamento do sistema SafeSenior, incluindo sensores, atuadores e módulos de comunicação.

2.2.1 Sensores (Opcional)

Os sensores detêm a essencial função de captar dados físicos e biométricos, permitindo que o sistema detete quedas, monitorize sinais vitais e identifique emergências. Para o presente projeto, este módulo detém um carácter opcional, sendo dado foco a outros componentes, nomeadamente, os atuadores e comunicação de valores.

1. **Acelerómetro + Giroscópio:**
    - **Função:** Detetar movimentos bruscos ou anómalos, como quedas ou alterações de postura.
    - **Justificação:** Este sensor combina acelerómetro e giroscópio num único módulo, permitindo medir aceleração em 3 eixos e orientação angular. É essencial para identificar quedas com precisão e enviar alertas imediatos.
2. **Sensor de Batimentos Cardíacos / oxigénio:**
    - **Função:** Monitorizar frequência cardíaca e saturação de oxigénio no sangue.

- **Justificação:** Permite avaliar o estado de saúde do idoso em tempo real. Alterações súbitas nos sinais vitais podem indicar emergências médicas, complementando a deteção de quedas.

3. **Localizador GPS:**
    - **Função:** Determinar a localização do idoso em emergências.
    - **Justificação:** Caso o idoso esteja fora de casa, o GPS fornece informação de posição para que a ajuda chegue rapidamente ao local correto.
4. **Sensor de Temperatura / Humidade:**
    - **Função:** Monitorizar o ambiente próximo do idoso, garantindo condições adequadas de conforto e segurança.
    - **Justificação:** Permite detetar condições ambientais extremas (ex.: calor excessivo, humidade elevada) que possam agravar riscos de saúde ou quedas.
5. **Sensor de Luz:**
    - **Função:** Detetar luminosidade do ambiente, podendo servir como trigger para alertas ou operação noturna.
    - **Justificação:** Permite adaptar notificações e funcionamento do sistema conforme a hora do dia (ex.: não enviar alertas sonoros à noite).

2.2.2 Atuadores

Os atuadores permitem que o sistema interaja fisicamente com o utilizador ou com o ambiente, garantindo que alertas e notificações sejam efetivos.

1. **Botão SOS:**
    - **Função:** Permitir que o utilizador acione manualmente um alerta em caso de emergência.
    - **Justificação:** Garante que, mesmo se o sistema automático não detetar a situação crítica, o idoso pode solicitar ajuda imediata. É simples, intuitivo e confiável.
2. **LEDs Indicadores:**
    - **Função:** Sinalizar visualmente estado do sistema (ativo, alerta, SOS enviado).
    - **Justificação:** Permitem ao idoso ou a terceiros saber rapidamente se o sistema está a funcionar corretamente ou se houve um evento crítico.
3. **Buzzer:**
    - **Função:** Emitir alertas sonoros locais em caso de emergência ou falha do sistema.
    - **Justificação:** Aumenta a eficácia de notificações imediatas, especialmente se o idoso estiver próximo do dispositivo.

2.2.3 Módulos

Os módulos permitem a ligação do Arduino com a *Cloud*, *APIs* e dispositivos móveis, viabilizando notificações e monitorização remota.

1. **Wi-Fi:**
    - **Função:** Comunicação com a API e envio de dados para backend e notificações.
    - **Justificação:** Permite integração do sistema com a *Cloud*, garantindo que alertas e dados de monitorização cheguem em tempo real a familiares ou cuidadores.
2. **Energia (Baterias + fonte de carregamento / pilhas):**

- **Função:** Fornecer energia estável ao sistema.
- **Justificação:** Garante funcionamento contínuo do sistema.

3 Arquitetura e tecnologias

3.1 Plataforma embebida

Para o projeto *SafeSenior*, a plataforma escolhida, com base nos requisitos do sistema foi o Arduino WIFI. Esta seleção, debruça-se sobre justificações como o custo, consumo de energia, capacidade de processamento, entre outras.

3.1.1 Custo

O projeto SafeSenior pretende ser uma solução económica e replicável para idosos e cuidadores, evitando custos de equipamentos médicos profissionais. As placas Arduino com Wi-Fi integrado, como o ESP8266 ou ESP32, são de baixo custo, o que torna possível desenvolver e produzir o sistema em escala sem encarecer o produto final. Alternativas com maior capacidade de processamento, como *Raspberry Pi*, seriam desnecessariamente caras e energeticamente mais exigentes, contrariando o objetivo.

3.1.2 Consumo de energia

SafeSenior é um sistema que deve funcionar continuamente, alimentado por bateria/pilhas. Portanto, o consumo energético é um fator crítico. Placas como o ESP8266 e o ESP32 apresentam modos de poupança de energia (*Deep sleep*) que permitem suspender o processamento quando não há eventos críticos, reduzindo drasticamente o consumo.

Comparado com microcontroladores de maior potência, o Arduino Wi-Fi consome muito menos energia, o que prolonga a autonomia da bateria e aumenta a fiabilidade do dispositivo em uso diário. Desta forma, tendo o consumo de energia como foco, a plataforma selecionada, mostra-se significativamente benéfica.

3.1.3 Capacidade de processamento

O SafeSenior precisa de realizar tarefas de monitorização e deteção em tempo real, mas sem a necessidade de processamento intensivo (*e.g.* os dados apenas são comunicados se fugirem à norma estipulada).

Desta forma, o Arduino com Wi-Fi tem capacidade suficiente para ler múltiplos sensores, aplicar algoritmos simples de deteção de quedas e enviar dados para a Cloud. A arquitetura baseada em microcontrolador (e não microprocessador) é ideal para este tipo de aplicação, rápida resposta, operação previsível e baixo overhead de software.

3.1.4 Conectividade

A comunicação em tempo real é um dos pilares do SafeSenior. O sistema deve enviar alertas imediatos a cuidadores através da *Cloud*.

O Arduino WiFi inclui o módulo Wi-Fi integrado (*e.g.,* ESP8266/ESP32), eliminando a necessidade de componentes externos, o que simplifica o hardware e reduz falhas de comunicação. Conexões WiFi

oferecem alcance e velocidade suficientes para o ambiente doméstico, permitindo enviar dados para servidores de forma rápida e segura.

A fraqueza desta abordagem passa pelas situações em que uma conexão WIFI não pode ser estabelecida (*e.g.* fora do alcance doméstico), que possivelmente terá que ser combatida através de outras ferramentas. No entanto, outras tecnologias como Bluetooth seriam demasiado limitadas em alcance, e GSM/LTE aumentariam significativamente o custo e o consumo.

3.1.5 Facilidade de desenvolvimento e integração

O Arduino é amplamente reconhecido pela sua facilidade de prototipagem e integração com sensores e módulos. Isto porque, dispõe de uma grande comunidade, bibliotecas prontas e documentação abundante, o que acelera o desenvolvimento e a manutenção do sistema. Além disso, é compatível com a maioria dos sensores utilizados, permitindo uma integração direta sem necessidade de hardware adicional complexo.

Outra vantagem é que o Arduino Wi-Fi representa um equilíbrio ideal entre simplicidade, consumo e conectividade, adequando-se perfeitamente ao perfil do projeto, monitorização leve, mas contínua, envio ocasional de dados para a Cloud, baixo custo de implementação e manutenção e confiabilidade e autonomia energética.

3.1.6 Conclusões

A opção por um Arduino com Wi-Fi integrado é a mais adequada para o SafeSenior pelo facto dos seus requisitos serem adequados para as vantagens oferecidas mas também pelas fraquezas não constituírem impasses para o seu cumprimento. Esta é a plataforma ideal para garantir eficiência, fiabilidade e sustentabilidade no funcionamento de um sistema que requere um baixo custo, facilidade de desenvolvimento e manutenção e conectividade, enquanto mantém um ótimo processamento, eficiente energeticamente.

3.2 Stack

A Stack tecnológica do *SafeSenior* define as ferramentas, *frameworks* e plataformas utilizadas em cada componente do sistema, garantindo integração eficiente entre *hardware*, *backend* e *frontend*.

O diagrama de arquitetura apresentado, ilustra as comunicações entre os diferentes módulos do sistema, evidenciando o fluxo de dados desde a captura de sinais biométricos e eventos críticos pelo hardware até à visualização e notificação em tempo real via interface web.

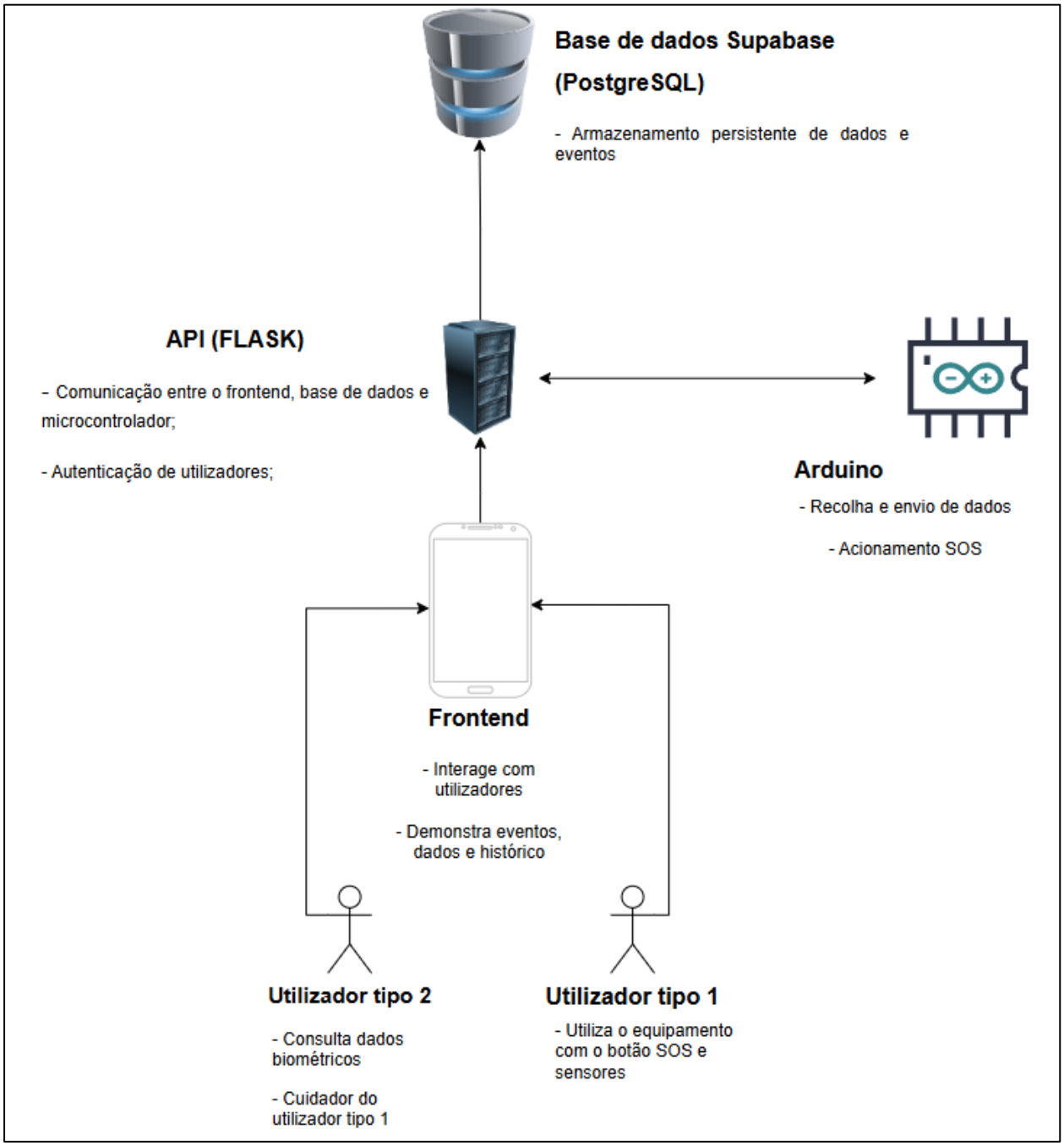

Figura 1 - Diagrama de Arquitetura

3.2.1 Backend

O backend funciona como o cérebro do *SafeSenior*. Recebe dados em tempo real, enviados pelo Arduino (por HTTP), e integra-os com a base de dados para registar medições, eventos e alertas. Para manter leveza e eficiência, a comunicação será realizada através de APIs RESTful, em *Python/Flask*.

Funções principais:

1. Receção de dados do Arduino (movimento, batimentos cardíacos, GPS, SOS, etc.) enviados através de pedidos HTTP (POST, GET, *etc.*).
2. Processamento e validação dos dados, analisando valores anómalos (*e.g.*, queda detetada, batimento irregular).
3. Gestão da lógica de negócio:
    - Registo de alertas;
    - Envio de notificações para familiares/cuidadores (via email, SMS, *push notification*, *etc*);
    - Armazenamento de dados históricos de saúde.
4. Comunicação com a base de dados.
5. Fornecimento de *endpoints* para o *frontend* aceder aos dados do utilizador, histórico de eventos, localização, *etc*.
6. Autenticação e segurança, garantindo que apenas utilizadores registados (idosos, familiares, cuidadores) têm acesso aos dados.

Protocolos de comunicação:

- *HTTP REST API* – Para comunicações diretas *Arduino >> Servidor >> App*.
- *JSON* – Formato padrão de troca de dados entre Arduino, backend e frontend.

Justificação da escolha:

- *Flask* é leve, rápido e facilmente escalável.
- Python oferece vastas bibliotecas para manipulação de dados (*e.g.,* pandas, numpy) e integração IoT.
- Permite rápida implementação e integração com bases de dados remotas.
- Baixa curva de aprendizagem e alta legibilidade de código.

3.2.2 Base de dados

A base de dados será alojada na plataforma *Supabase*, que utiliza PostgreSQL como motor central e fornece uma API RESTful automática, autenticação integrada e gestão de permissões, ideal para integração rápida com o *backend Flask*.

Funções principais:

- Armazenamento persistente de:
    - Dados biométricos – Batimentos cardíacos, pressão arterial;
    - Registos de eventos – Quedas, SOS, localização, alertas;
    - Dados do utilizador – Perfil do idoso, familiares, cuidadores
    - Contexto temporal – Para acompanhamento de saúde e log de eventos ou interações no sistema.

Justificação da escolha:

- PostgreSQL é robusto, seguro e altamente escalável, suportando transações complexas e integridade referencial.
- Solução gratuita/open-source nas fases iniciais do projeto, podendo escalar facilmente para produção.

3.2.3 Frontend

O frontend do projeto, será desenvolvido em *React*, utilizando *Tailwind CSS* para uma personalização rápida e responsiva. Esta abordagem, baseada em componentes permite criar uma interface modular e de fácil manutenção, acelerando o desenvolvimento e evitando a escrita manual de CSS.

A interface permitirá:

- Consultar, em tempo real, os dados biométricos e de movimento medidos pelos sensores.
- Monitorizar eventos críticos, como quedas e acionamento do botão SOS.
- Consultar histórico de eventos e medições com *timestamps*.
- Receber alertas visuais de emergências de forma imediata.

Justificação da escolha:

- Desenvolvimento ágil e modular com React.
- Layouts rápidos e responsivos com Tailwind CSS.
- Integração simples com APIs RESTful do backend Flask.
- Acessível via web, dispensando desenvolvimento nativo adicional.
- Esta abordagem garante que o *frontend* seja funcional, intuitivo e rápido de desenvolver, permitindo focar os recursos do projeto no núcleo crítico, o sistema embebido e a monitorização dos sensores.

4 Planeamento

O desenvolvimento do sistema *SafeSenior*, será conduzido através de uma abordagem híbrida, combinando princípios da metodologia ágil no desenvolvimento do software do sistema com uma etapa posterior de testagem em hardware.

Nesta primeira fase, o código será desenvolvido e validado através de um simulador Arduino, garantindo que toda a lógica está funcional antes da montagem física. Esta estratégia permite acelerar o desenvolvimento do software, reduzir custos iniciais e garantir que o código do sistema embebido e do backend se encontram estáveis antes da integração física.

4.1 Fases do projeto

4.1.1 Fase 1 – Desenvolvimento e Testagem em Simulação

Nesta primeira fase, será adotada uma metodologia ágil, centrada em entregas iterativas curtas e testes frequentes.

O objetivo é desenvolver e validar todo o software recorrendo a simuladores Arduino e ambientes de teste virtuais, sem depender ainda do hardware físico.

Objetivos principais:

- Implementar e testar as rotinas de leitura de sensores (simulados).
- Desenvolver a API responsável pela comunicação com o Arduino e o backend.
- Simular a troca de dados via Wi-Fi, usando ferramentas como Tinkercad Circuits.
- Criar uma base de dados em PostgreSQL e validar a persistência de dados.
- Integrar um frontend simples apenas para visualização dos resultados.

Metodologia:

- Desenvolvimento dividido em sprints semanais, com entregas incrementais:
    1. Comunicação e envio de dados simulados (Arduino -> API).
    2. Processamento e alertas via backend Flask.
    3. Integração com a base de dados.
    4. Visualização simples dos dados.
- Cada sprint incluirá testes e validações funcionais dos módulos.

Vantagens:

- Ciclos de feedback rápidos e contínuos.
- Identificação precoce de falhas lógicas ou de comunicação.
- Possibilidade de testar o sistema sem necessidade de hardware físico.
- Redução de custos e riscos iniciais (baixa importância para o contexto académico atual).

Entrega:

- Ao final desta primeira fase, deverá existir um sistema simulado totalmente funcional, com:
    - Simulador do Arduino estável;
    - API Flask funcional e integrada;
    - Base de dados configurada e operacional;

- Frontend simples a exibir dados reais simulados.

4.1.2 Fase 2 – Montagem e testagem em Hardware

Após a validação completa em simulação, inicia-se a implementação física do sistema. Esta fase foca-se na montagem dos componentes eletrónicos, calibração e integração real com o backend.

Objetivos principais:

- Montar o hardware: Arduino, sensores (opcional), atuadores e módulo Wi-Fi.
- Validar a comunicação real com o backend (via rede Wi-Fi).
- Realizar testes de campo (*e.g.* envio de alertas, fiabilidade da ligação).
- Realizar ajustes, se necessário.

Metodologia:

- Testagem modular.
- Integração progressiva dos componentes até o sistema completo estar operacional.
- Comparação entre dados simulados e dados reais para avaliar a precisão.

Entrega desta fase:

- O protótipo físico funcional do SafeSenior deve ser capaz de:
  - Monitorizar e enviar dados reais;
  - Comunicar autonomamente com o backend e o frontend;
  - Emitir alertas sonoros e visuais em situações críticas;
  - Armazenar dados históricos na base de dados.

4.2 Milestones

Milestone	Foco	Objetivo Principal	Resultado Esperado
13 – 17 outubro	Planeamento e preparação do ambiente	Configuração do ambiente de desenvolvimento e simulador Arduino.	IDE Arduino, Flask e Supabase configurados.
18 – 24 outubro	Comunicação unidirecional	Envio de sinal do botão (Arduino → API)	API recebe pedidos do botão SOS via HTTP
25 – 31 outubro	Comunicação bidirecional	Resposta do backend (API → Arduino)	API envia instruções (acender LED) em resposta ao botão
01 – 07 novembro	Integração com base de dados	Registar eventos de SOS na base de dados	Eventos gravados no Supabase
8 – 14 novembro	Frontend simples	Exibir alertas e histórico de botões acionados	Interface básica funcional
15 – 21 novembro	Testes em simulação	Validar todo o fluxo em ambiente virtual	Comunicação end-to-end simulada
22–28 novembro	Montagem física	Montar Arduino com botão e LED reais	Comunicação física com o backend funcional
Entrega final	Testes de campo e otimização final	Testes finais em hardware	Protótipo funcional validado

Figura 2 - Tabela de Milestones