Nome do Aluno:
Carlos Fernando Mattos do Amaral
Eric Senne Roma
Melvin Gustavo Maradiaga Elvir

HipSafe - Sistema de Monitoramento Ativo para Reabilitação de Quadril

Projeto Final da Fase 2: Etapa 2 – Arquitetura e Modelagem

Campinas (SP), 08 de Agosto de 2025

Sumário

Diagrama de Blocos Funcionais:

O dispositivo será composto por quatro elementos principais: Os sensores, a placa de desenvolvimento BitDogLab, o RTC e o periférico de memória Flash SDCard.

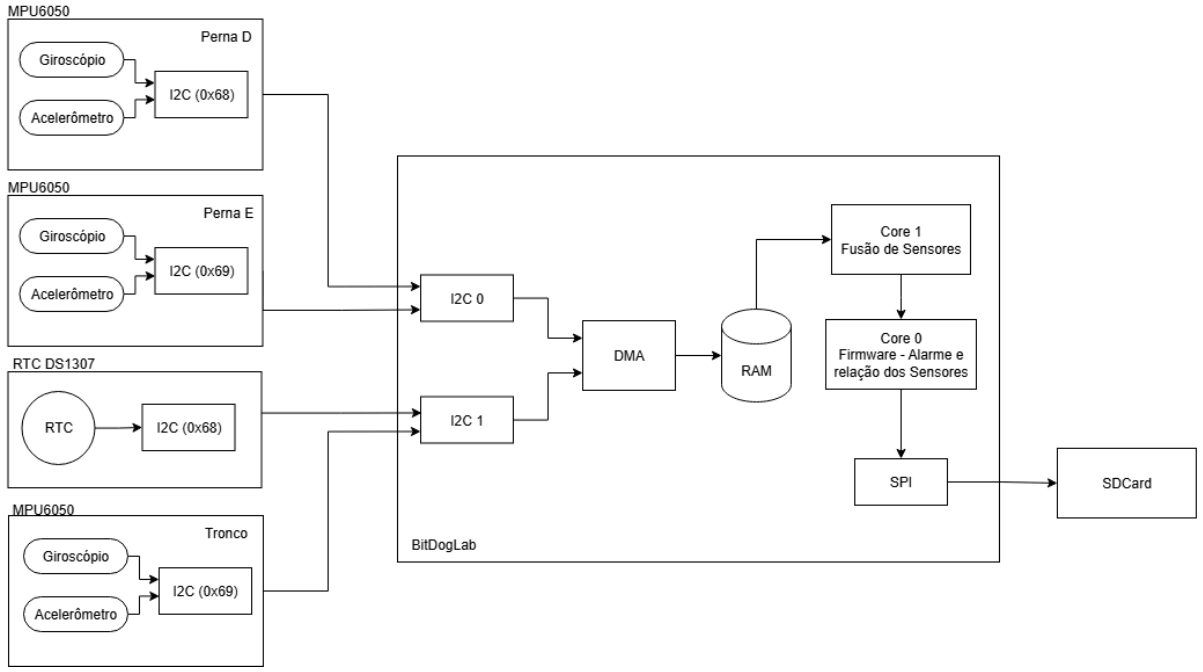

Figura 1 - Esquema de blocos funcionais. Fonte: Autoria Própria.

A Figura 1 apresenta o diagrama de blocos funcionais do sistema, detalhando os blocos do sistema e as interações entre os principais componentes envolvidos na aquisição, processamento e armazenamento de dados.

O sistema utiliza três módulos MPU6050, que integram acelerômetro e giroscópio, conectados via barramento I2C. Como os módulos MPU6050 possuem, por padrão, o mesmo endereço I2C (0x68), seria inviável conectá-los todos no mesmo barramento. No entanto, é possível alterar o endereço de um módulo para 0x69 aplicando 5V no pino ADO. Com isso, dois dos módulos tiveram seus endereços alterados para 0x69, permitindo a seguinte configuração: no barramento I2C0, há dois dispositivos MPU6050, um com endereço 0x68 e outro com 0x69; no barramento I2C1, há um terceiro MPU6050 com endereço 0x69 e um módulo de relógio de tempo real (RTC) DS1307, que possui endereço fixo 0x68.

A comunicação entre os sensores e o processador da placa BitDogLab é gerenciada por dois barramentos I2C (I2C0 e I2C1), cujos dados são transferidos para a memória RAM utilizando um canal de DMA (Direct Memory Access). A utilização do DMA visa reduzir a carga de processamento da CPU, permitindo uma transferência eficiente de dados com menor intervenção do processador.

Para otimizar o desempenho do sistema, as tarefas foram distribuídas entre os dois núcleos de processamento disponíveis no RP2040. O Core 1 é responsável pela leitura dos dados da RAM e pela fusão dos sensores, isto é, a conversão dos dados brutos de aceleração e

rotação dos módulos MPU6050 em ângulos de orientação (roll, pitch e yaw). O Core 0, por sua vez, utiliza essas informações para realizar a correlação entre os ângulos dos sensores, identificar possíveis situações de risco postural ao paciente (como ângulos que indicam flexão perigosa do quadril), acionar alarmes quando necessário e registrar os eventos em um cartão SD via interface SPI para futura análise.

Diagrama de Hardware:

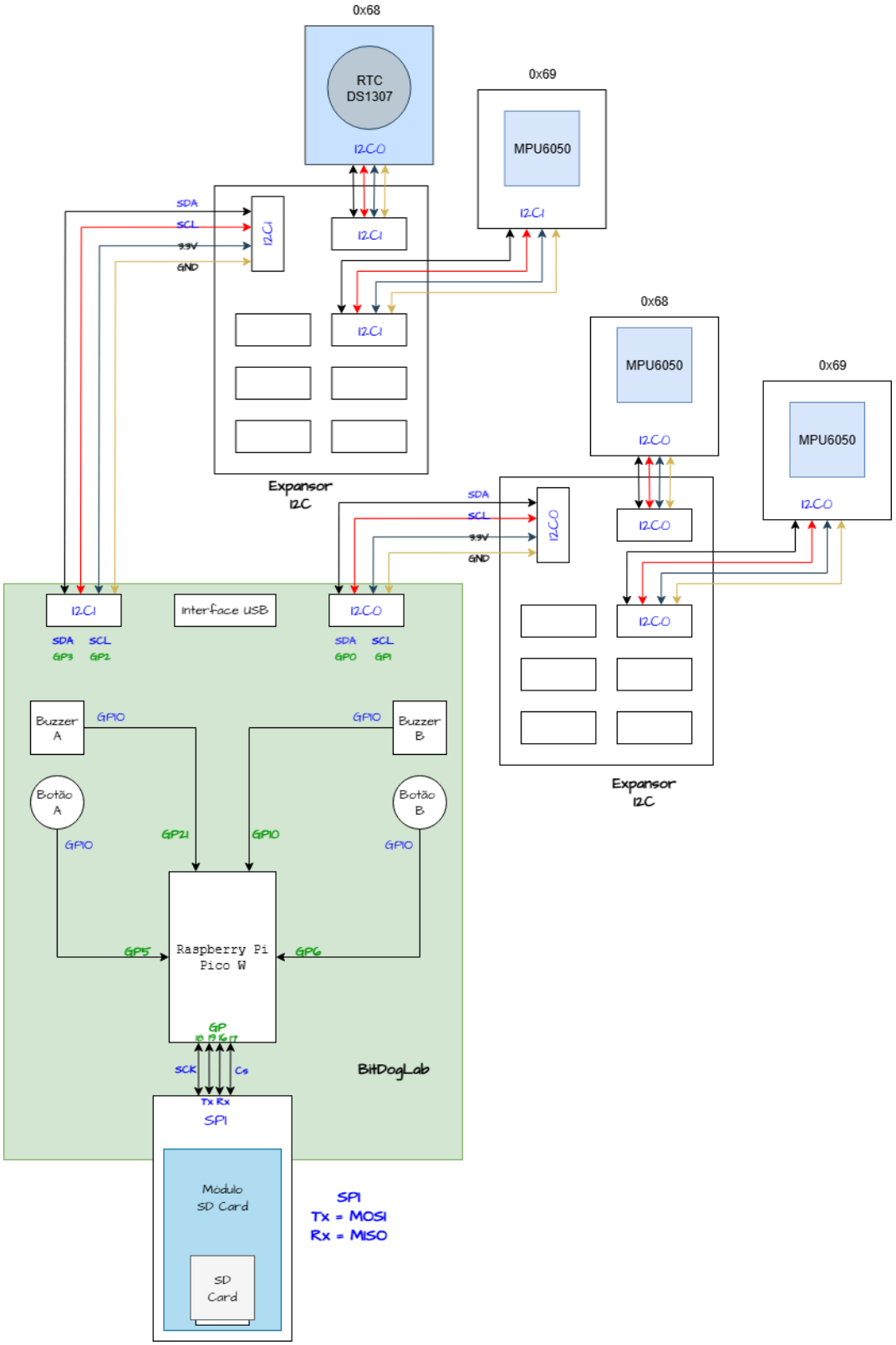

Figura 2 - Diagrama de Hardware do sistema. Fonte: Autoria Própria.

A Figura 2 apresenta o diagrama de hardware do projeto desenvolvido com a placa **Raspberry Pi Pico W**, componente central da plataforma **BitDogLab**. O sistema é responsável por adquirir dados de sensores inerciais e temporais, realizar o processamento interno, emitir alertas ao usuário e armazenar as informações localmente em um cartão SD.

A Raspberry Pi Pico W se comunica com os periféricos externos por meio das interfaces padrão I2C, SPI e GPIO. Para viabilizar a conexão simultânea de múltiplos dispositivos que compartilham endereços I2C semelhantes, o projeto faz uso de dois barramentos distintos, identificados como **I2C0** e **I2C1**, em conjunto com **expansores de barramento I2C**, os quais organizam fisicamente as conexões e evitam conflitos lógicos entre os dispositivos.

O sistema inclui três módulos **MPU6050**, que integram acelerômetro e giroscópio. Esses sensores são utilizados para medir os movimentos e inclinações do paciente. Por padrão, cada MPU6050 possui o mesmo endereço I2C (0x68), o que inviabilizaria sua utilização conjunta em um único barramento. Para resolver esse problema, dois dos módulos tiveram seus endereços alterados para 0x69, conectando o pino ADO de cada um deles à alimentação de 5V. Dessa forma, no barramento **I2C0**, foram conectados dois módulos MPU6050 com endereços distintos (0x68 e 0x69). Já no barramento **I2C1**, foi conectado um terceiro módulo MPU6050 com endereço 0x69, juntamente com o módulo **RTC DS1307**, que opera com endereço fixo 0x68.

O **RTC DS1307** fornece a referência de tempo real para o sistema, sendo essencial para a marcação temporal dos eventos registrados. Ele compartilha o barramento I2C1 com o terceiro sensor MPU6050, sem causar conflito, devido à diferença de endereços. Todos os dispositivos I2C são fisicamente organizados por meio de expansores, que distribuem os sinais **SDA**, **SCL**, **3V3** e **GND** de maneira clara e modular.

Para armazenamento dos dados processados, o sistema utiliza um **módulo de cartão SD**, conectado à placa por meio da interface **SPI**.

Além da coleta e processamento de dados, o sistema incorpora uma interface simples de interação com o usuário. Dois **botões físicos** (Botão A e Botão B) estão conectados a pinos GPIO da Raspberry Pi Pico W, permitindo a interrupção do buzzer quando ele é acionado. Também foram integrados dois **buzzers** (A e B), acionados via GPIO, que emitem alertas sonoros em caso de detecção de situações críticas, como movimentos perigosos durante a recuperação de um paciente submetido à artroplastia de quadril. Todo o sistema é alimentado e organizado a partir da Raspberry Pi Pico W, que atua como unidade central de controle.

Fluxograma do Software:

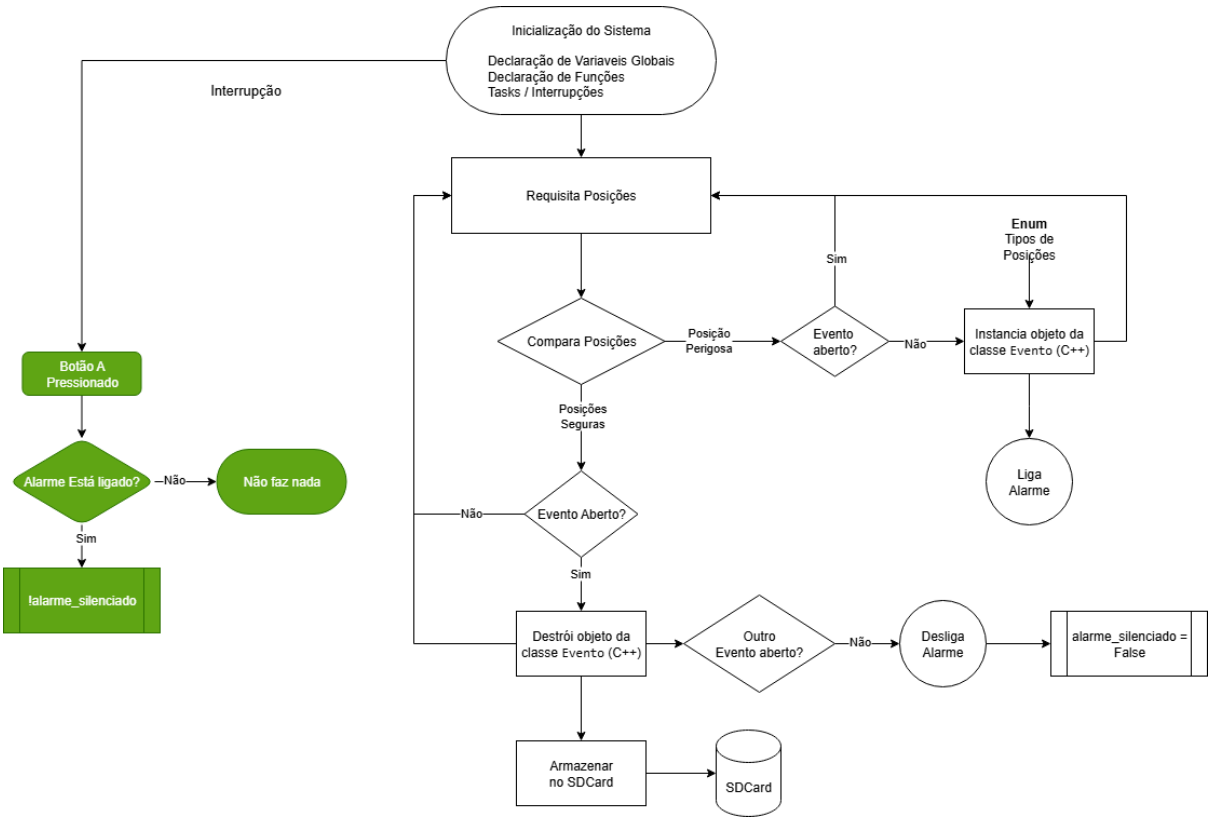

Figura 3 - Fluxograma do Software do sistema. Fonte: Autoria Própria.

A Figura 3 apresenta o fluxograma do sistema de monitoramento postural, detalhando a sequência lógica das ações realizadas desde a inicialização até o controle de alarmes e o armazenamento de eventos. Inicialmente, o sistema é configurado com a declaração de variáveis globais, funções e a ativação das tarefas e interrupções necessárias para o funcionamento.

O sistema monitora continuamente as posições captadas pelos sensores MPU6050, que estão conectados via dois barramentos I2C (I2C0 e I2C1), conforme descrito no diagrama de blocos funcionais. A transferência dos dados ocorre através de DMA para a memória RAM. A leitura e fusão dos dados brutos dos acelerômetros e giroscópios são feitas pelos núcleos de processamento, permitindo a conversão em ângulos de orientação (roll, pitch e yaw).

No fluxo principal, o sistema requisita as posições atuais dos sensores e as compara para determinar se estão em uma posição perigosa ou segura. Caso seja detectada uma posição perigosa, o sistema verifica se há um evento aberto. Se não houver evento aberto, instancia-se

um objeto da classe `Evento` (implementado em C++), que representa a ocorrência da situação de risco, e, em seguida, aciona o alarme correspondente.

Se a posição estiver segura e um evento estiver aberto, o sistema procede à destruição do objeto da classe `Evento`, indicando o encerramento da situação de risco. Após isso, verifica-se se há outro evento aberto; caso contrário, o alarme é desligado e a variável de controle `alarme_silenciado` é resetada para `False`.

Paralelamente, o sistema responde a interrupções externas, como o pressionamento do Botão A. Quando este botão é pressionado, o sistema verifica se o alarme está ativo. Se estiver, executa a rotina de silenciamento do alarme (`alarme_silenciado`), caso contrário, nenhuma ação é tomada. Por fim, todos os eventos identificados são armazenados em um cartão SD para posterior análise, garantindo o registro histórico das situações de risco detectadas.

Arquitetura de Software:

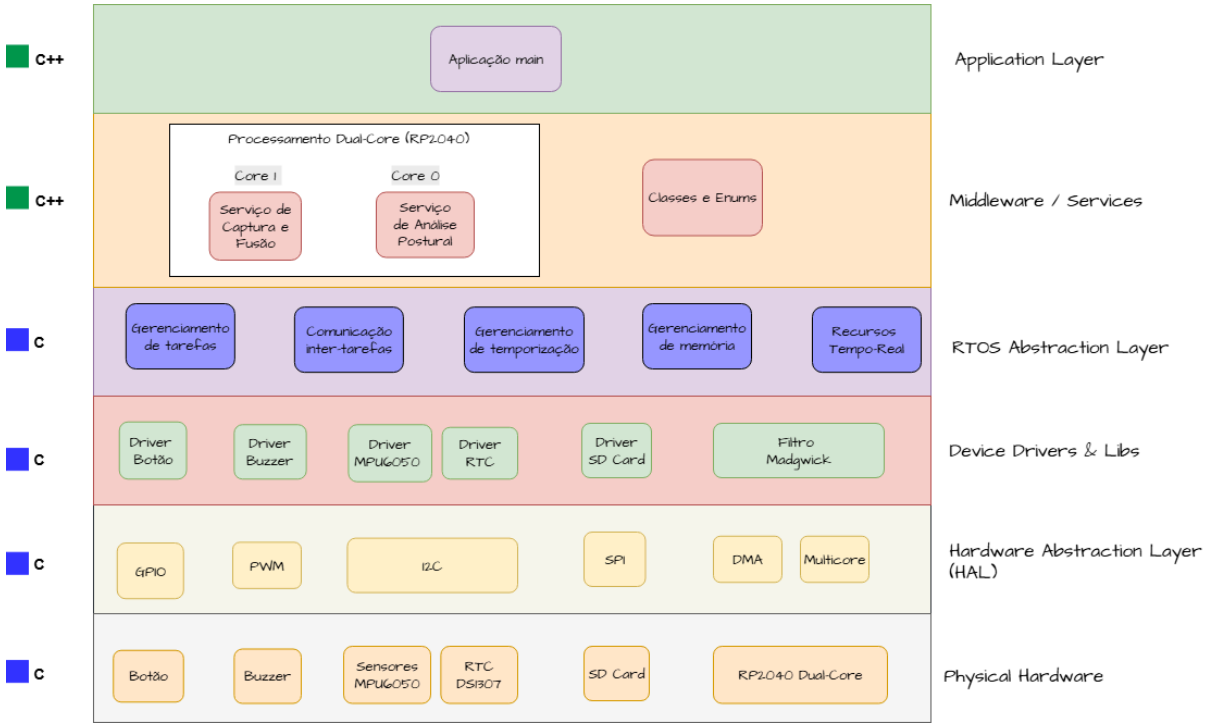

Figura 4 - Arquitetura de Software do sistema. Fonte: Autoria Própria.

1. Camada de Hardware

Inclui todos os componentes físicos essenciais para a operação do sistema embarcado:

- **RP2040 Dual-Core**: Microcontrolador ARM Cortex-M0+ com dois núcleos operando a até 133 MHz, memória SRAM interna e periféricos integrados. Este componente é responsável pelo controle geral do sistema, permitindo paralelismo entre as operações

de aquisição e análise de dados.

- **Sensores Inerciais MPU6050 (x3)**: Módulos que integram acelerômetro e giroscópio de 6 eixos, responsáveis pela medição da orientação espacial (roll, pitch e yaw) do corpo do paciente. São posicionados no tronco e nas pernas para permitir o cálculo dos ângulos relativos entre esses segmentos corporais e, assim, identificar posturas inadequadas.

- **Cartão SD (SDCard)**: Módulo de armazenamento conectado via SPI, utilizado para armazenamento não volátil dos eventos de postura crítica detectados durante o uso.

- **Botão**: Componente físico que permite ao usuário interagir com o sistema para silenciar o alarme.

- **Buzzer**: Dispositivo emissor de alertas sonoros, que notifica o paciente em tempo real sempre que uma postura incorreta for detectada.

---

2. Camada de Abstração de Hardware (HAL)

Inclui as bibliotecas do SDK. Fornece uma interface direta entre o software e os periféricos do microcontrolador, abstraindo os detalhes do hardware:

- **GPIO (General Purpose Input/Output):** Interface de controle de pinos digitais para entrada (botão) e saída (buzzer).

- **PWM (Pulse Width Modulation):** Utilizado para controlar a intensidade e frequência do sinal sonoro gerado pelo buzzer.

- **I2C (Inter-Integrated Circuit):** Interface de comunicação serial usada para adquirir dados dos sensores MPU6050 e do RTC.

- **SPI (Serial Peripheral Interface):** Interface utilizada para comunicação com o cartão SD.

- **DMA (Direct Memory Access):** Permite transferência direta de dados entre sensores e memória, sem sobrecarregar a CPU.

- **Multicore:** Recursos específicos do RP2040 para gerenciamento e comunicação entre os dois núcleos do microcontrolador.

---

3. Camada de Drivers e Bibliotecas

Esta camada implementa funcionalidades de mais alto nível, construídas sobre a HAL, oferecendo acesso modular e reutilizável aos periféricos:

- **Driver do Botão:** Driver responsável pela inicialização do botão, configuração da interrupção por borda e leitura do seu estado lógico por meio da interface GPIO.

- **Driver do Buzzer:** Driver responsável por controlar o buzzer através de sinais PWM, gerando alertas sonoros em diferentes padrões.

- **Driver dos Sensores MPU6050:** Gerencia a comunicação com os sensores inerciais via barramento I²C, permitindo a configuração de parâmetros como os intervalos de medição do acelerômetro e do giroscópio, além da leitura dos dados brutos de aceleração e rotação angular.

- **Driver RTC (DS1307):** Driver responsável pelo gerenciamento do módulo RTC DS1307, utilizado para obter carimbos de tempo precisos nos eventos registrados pelo sistema

- **Driver SD Card:** Gerencia a leitura e escrita de dados no cartão SD, salvando automaticamente as medições dos sensores em formato de tabela CSV, por meio do Raspberry Pi Pico, o que permite seu posterior registro e visualização.

- **Filtro Madgwick:** Biblioteca especializada para fusão sensorial, responsável por converter dados brutos dos sensores em orientações espaciais precisas (roll, pitch e yaw).

---

3. Camada do Sistema Operacional (FreeRTOS)

Responsável pelo gerenciamento dos recursos computacionais em tempo real:

- **Gerenciamento de Tarefas (Tasks)**: Divide a execução em múltiplas tarefas concorrentes, como aquisição, processamento, alarme e armazenamento.

- **Comunicação Inter-Tarefas**: Realizada através de filas (queues), semáforos (semaphores) e flags de evento para garantir sincronização e troca de dados segura.

- **Temporização e Controle de Atrasos**: Uso de temporizadores e delays precisos, necessários para amostragem periódica e resposta a eventos externos.

- **Gerenciamento de Memória Dinâmica**: Alocação eficiente de buffers e áreas temporárias em RAM.

- **Tratamento de Interrupções**: Permite que o sistema responda rapidamente quando o botão da BitDogLab é pressionado, garantindo que essa ação seja detectada e processada sem atrasos.

---

4. Camada de Middleware / Serviços

Representa os serviços intermediários, organizados por núcleo do microcontrolador:

Core 1 – Aquisição e Fusão de Dados

- Aquisição síncrona dos dados dos sensores via I²C.
- Fusão sensorial utilizando o filtro Madgwick.
- Envio dos dados angulares processados para o Core 0.

Core 0 – Análise e Alerta

- Análise postural com base na comparação dos ângulos medidos e limites clínicos definidos.
- Ativação do alerta sonoro sempre que posturas críticas forem detectadas.
- Armazenamento dos eventos com carimbo de tempo no cartão SD.

Classes e Enums

- Estruturas em C++ compartilhadas entre os serviços do sistema. As classes representam os eventos por meio de objetos instanciados e destruídos conforme a lógica de detecção, enquanto os enums classificam os tipos de posição corporal, permitindo distinguir eventos abertos e fechados.

---

5. Camada de Aplicação

Inclui a lógica de alto nível que define o comportamento do sistema:

- **Aplicação main:** Ponto de entrada do sistema, responsável por inicializar os periféricos, configurar os serviços e tarefas, e iniciar a execução do sistema operacional em tempo real.