Análise Completa da Atividade 2 – Controle de LEDs com Interrupções no Raspberry Pi Pico

# 1. Explicação Detalhada dos Códigos

Esta aplicação substitui o polling da Atividade 1 por uma abordagem baseada em interrupções. Cada botão (A, B e do joystick) alterna o estado de um LED (vermelho, azul e verde) utilizando tratamento de interrupção com debounce em software. O programa é dividido em três arquivos:

* - Atividade\_2.c: Função principal (main), configuração de GPIOs e interrupções, loop principal.
* - funcao\_atividade\_2.c: Implementação das funções auxiliares, incluindo o callback de interrupção.
* - funcao\_atividade\_2.h: Cabeçalho com protótipos, definições e variáveis globais.

## 1.1 Arquivo: Atividade\_2.c

A função main() realiza a inicialização dos pinos GPIO dos LEDs e botões. As interrupções são configuradas para detectar borda de descida (pressionamento). A função gpio\_callback é registrada como manipulador global de interrupções e o sistema permanece em modo de economia de energia usando \_\_wfi() até que uma interrupção ocorra. Quando uma flag de botão pressionado é detectada, o LED correspondente alterna seu estado.

## 1.2 Arquivo: funcao\_atividade\_2.c

Contém a função inicializar\_pino() para configurar GPIOs e a função gpio\_callback(), que age como ISR (rotina de interrupção). O callback compara o tempo atual com o último acionamento, implementando debounce de 40 ms. Se o tempo for suficiente, uma flag é ativada para que o loop principal trate o evento.

## 1.3 Arquivo: funcao\_atividade\_2.h

Define constantes como tempos de debounce e delay visual, pinos de botões e LEDs, e protótipos das funções. Também declara os vetores globais para controle de estados e comunicação entre a interrupção e o loop principal.

# 2. Respostas às Perguntas do Autor

## 2.1 Questão 1 – Arquitetura Foreground/Background

O código implementa a arquitetura foreground/background ao dividir a lógica em duas camadas:  
  
• Background: a função gpio\_callback() que atua como uma ISR rápida, apenas sinalizando o evento.  
• Foreground: o laço principal em main(), que consulta as flags de evento e executa as ações demoradas como alternar LEDs e aplicar delays.

Se a lógica de alternar LEDs estivesse na interrupção, haveria riscos como:  
• Travamentos causados por sleep\_ms() dentro da ISR.  
• Perda de eventos devido à demora.  
• Condições de concorrência ou corrupção de dados em sistemas maiores.

## 2.2 Questão 2 – Uso de volatile

A palavra-chave volatile é essencial porque as variáveis botoes\_pressionados[] são modificadas na interrupção e lidas no loop principal. Sem volatile, o compilador pode otimizar o código presumindo que o valor da variável não muda fora do escopo visível, levando a leituras incorretas. Volatile garante que a leitura sempre reflita o valor real da memória, evitando comportamento inesperado.

## 2.3 Questão 3 – Teste dos Botões

Durante os testes, cada botão alterna corretamente o estado do LED correspondente. O sistema responde bem a pressões rápidas e repetidas, com o debounce funcionando como esperado. Um possível efeito percebido é que pressões muito rápidas (menores que 40 ms) podem ser ignoradas — o que é desejável para evitar bouncing.