# Roteiro de Experimento: Leitura de ADC com Interrupção na BitDogLab

## 1. Título do Experimento

Leitura periódica do valor analógico de um joystick usando ADC e Interrupção no RP2040 (Projeto: isr\_adc\_irq\_bitdoglab)

## 2. Objetivo

- Compreender o uso da interrupção de fim de conversão do ADC no RP2040.  
- Ler o valor do joystick através do ADC usando interrupções.  
- Acender ou apagar um LED de acordo com o valor lido.

## 3. Materiais Necessários

- Placa BitDogLab.  
- Joystick analógico integrado.  
- LED integrado (GPIO12).  
- Computador com VSCode e Pico SDK configurado.

## 4. Diagrama de Montagem

• Joystick eixo Y conectado ao GPIO27 (ADC1).  
• LED Azul ao GPIO12 (ânodo comum).  
• (Toda a montagem já é interna à BitDogLab.)

## 5. Procedimento

### 5.1. Inicialização do projeto

• Criar um novo projeto no VSCode com suporte ao Pico SDK.  
• Nome sugerido: isr\_adc\_irq\_bitdoglab.

### 5.2. Código Fonte Básico

#include <stdio.h>  
#include "pico/stdlib.h"  
#include "hardware/adc.h"  
#include "hardware/irq.h"  
  
#define JOY\_Y\_ADC\_CHANNEL 1  
#define LED\_VERMELHO 12  
#define LIMITE\_ACIONAMENTO 3500  
  
volatile bool acionado = false;  
  
void adc\_irq\_handler() {  
 uint16\_t valor = adc\_fifo\_get();  
  
 if (valor > LIMITE\_ACIONAMENTO) {  
 if (!acionado) {  
 acionado = true;  
 gpio\_put(LED\_VERMELHO, 0);  
 printf("Acionado! Valor do ADC: %d\n", valor);  
 }  
 } else {  
 if (acionado) {  
 acionado = false;  
 gpio\_put(LED\_VERMELHO, 1);  
 printf("Desativado. Valor do ADC: %d\n", valor);  
 }  
 }  
}  
  
int main() {  
 stdio\_init\_all();  
 sleep\_ms(2000);  
 printf(" Sistema iniciado. Aguarde...\n");  
  
 gpio\_init(LED\_VERMELHO);  
 gpio\_set\_dir(LED\_VERMELHO, GPIO\_OUT);  
 gpio\_put(LED\_VERMELHO, 1);  
  
 adc\_init();  
 adc\_gpio\_init(27);  
 adc\_select\_input(JOY\_Y\_ADC\_CHANNEL);  
 adc\_fifo\_setup(true, true, 1, false, false);  
 irq\_set\_exclusive\_handler(ADC\_IRQ\_FIFO, adc\_irq\_handler);  
 irq\_set\_enabled(ADC\_IRQ\_FIFO, true);  
 adc\_irq\_set\_enabled(true);  
  
 adc\_run(true);  
 adc\_fifo\_drain();  
  
 while (true) {  
 adc\_run(true);  
 adc\_fifo\_drain();  
 adc\_hw->cs |= ADC\_CS\_START\_ONCE\_BITS;  
 sleep\_ms(50);  
 }  
}

## 6. Análises e Observações

- A leitura do ADC ocorre por interrupção, sem necessidade de polling ou timers.  
- O LED acende ou apaga com base no valor do joystick lido.  
- O uso da interrupção torna o sistema mais eficiente.  
- Discuta:  
 • Diferenças entre Timer + ADC e Interrupção ADC.  
 • Vantagens da interrupção em termos de eficiência de CPU.

## 7. Extensões e Desafios

- Ajustar o valor de limite para calibrar o joystick.  
- Implementar ação para outro eixo do joystick.  
- Fazer o LED piscar proporcionalmente à posição do joystick.  
- Implementar detecção de movimentos rápidos (variações abruptas de valor).

## 8. Importante

- Sempre iniciar a conversão ADC manualmente no loop principal.  
- A ISR deve ser curta e eficiente.  
- Cuidar para limpar o FIFO do ADC para evitar acúmulo de dados antigos.