# Roteiro de Experimento: PWM com Interrupção na BitDogLab

## 1. Título do Experimento

Controle de um Buzzer via PWM e Interrupção no RP2040 (Projeto: isr\_pwm\_bitdoglab)

## 2. Objetivo

- Compreender o funcionamento do PWM no RP2040.  
- Aprender a configurar interrupções associadas ao PWM (evento de wrap).  
- Gerar sinais sonoros controlados automaticamente usando ISR.

## 3. Materiais Necessários

- Placa BitDogLab.  
- Buzzer integrado conectado ao GPIO21.  
- Computador com VSCode e Pico SDK configurado.

## 4. Diagrama de Montagem

• Buzzer conectado ao GPIO21.  
• (A BitDogLab já integra o Buzzer, não sendo necessário montagem adicional.)

## 5. Procedimento

### 5.1. Inicialização do projeto

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

### 5.2. Código Fonte Básico

#include <stdio.h>  
#include "pico/stdlib.h"  
#include "hardware/pwm.h"  
#include "hardware/irq.h"  
  
#define PWM\_GPIO 21  
#define PWM\_FREQ\_HZ 1000  
#define CLOCK\_DIV 2.0f  
#define PWM\_WRAP (uint16\_t)(125000000 / (PWM\_FREQ\_HZ \* CLOCK\_DIV))  
#define CICLOS\_PARA\_1S PWM\_FREQ\_HZ  
  
volatile int contador = 0;  
volatile bool tocando = false;  
  
void pwm\_wrap\_handler() {  
 uint slice = pwm\_gpio\_to\_slice\_num(PWM\_GPIO);  
 pwm\_clear\_irq(slice);  
  
 contador++;  
 if (contador == 0) return;  
  
 if (contador >= CICLOS\_PARA\_1S) {  
 contador = 0;  
 tocando = true;  
 pwm\_set\_gpio\_level(PWM\_GPIO, PWM\_WRAP / 2);  
 printf(" BEEP!\n");  
 }  
 if (tocando && contador == 100) {  
 pwm\_set\_gpio\_level(PWM\_GPIO, 0);  
 tocando = false;  
 }  
}  
  
int main() {  
 stdio\_init\_all();  
 sleep\_ms(2000);  
 printf(" PWM + IRQ no GPIO21 (Buzzer BitDogLab)\n");  
  
 gpio\_set\_function(PWM\_GPIO, GPIO\_FUNC\_PWM);  
 uint slice = pwm\_gpio\_to\_slice\_num(PWM\_GPIO);  
  
 pwm\_config cfg = pwm\_get\_default\_config();  
 pwm\_config\_set\_clkdiv(&cfg, CLOCK\_DIV);  
 pwm\_config\_set\_wrap(&cfg, PWM\_WRAP);  
 pwm\_init(slice, &cfg, true);  
  
 pwm\_set\_gpio\_level(PWM\_GPIO, 0);  
  
 pwm\_clear\_irq(slice);  
 pwm\_set\_irq\_enabled(slice, true);  
 irq\_set\_exclusive\_handler(PWM\_IRQ\_WRAP, pwm\_wrap\_handler);  
 irq\_set\_enabled(PWM\_IRQ\_WRAP, true);  
  
 while (true) {  
 tight\_loop\_contents();  
 }  
}

## 6. Análises e Observações

- O PWM gera uma interrupção a cada ciclo completo (wrap).  
- A cada segundo aproximadamente, o buzzer emite um bipe rápido (~100ms).  
- O controle é feito pela contagem de ciclos no handler de interrupção.  
- Discuta:  
 • Vantagens do uso de interrupções.  
 • Como modificar o tempo entre os bipes ou a duração do bipe.

## 7. Extensões e Desafios

- Alterar a frequência do PWM para mudar o tom do bipe.  
- Criar padrões de alertas sonoros (curto-curto-longo, etc).  
- Acionar o bipe manualmente por meio de um botão.

## 8. Importante

- Sempre limpar a interrupção do PWM dentro do handler.  
- Ajustar corretamente o CLOCK\_DIV para evitar overflow.  
- A duração do bipe é baseada na contagem de ciclos do PWM.