Anexo 5 – Temporizador One Shot 2

# Descrição

Este é um outro exemplo que demonstra o funcionamento de um temporizador não-repetitivo (One Shot) sendo executado fora do Loop Infinito, liberando desta forma a CPU para tratar de outras tarefas.

# Objetivos:

- Adicionar um timer não-repetitivo que, após a ocorrência de um evento externo (acionamento de um botão), chama uma função de Callback que realiza várias tarefas: ativar debounce, chamar função de Callback de debouncer, acender os três LEDs, bloquear novos acionamentos do botão e a de agendar um alarme para, após 3 segundos, chamar uma outra função de Callback;

- Essa próxima função de Callback realizará duas tarefas; apagar um dos leds e agendar um alarme para apagar o próximo LED;

- Mais duas funções de Callback são chamadas, sempre dentro de agendamentos feitos na função de Callback anterior que também apagarão um dos LEDs e agenderá temporizador para apagar o próximo LED.

- Observar que todo o processo ocorre uma única vez;

- Liberar a CPU para tratar das tarefas do Loop Infinito.

# Código-fonte (main.c)

#include <stdio.h>

#include "pico/stdlib.h"

#include "hardware/gpio.h"

#include "hardware/timer.h"

// Definições de pinos

#define LED\_VERDE 11

#define LED\_AZUL 12

#define LED\_VERMELHO 13

#define BOTAO 5

// Variáveis de controle

volatile bool pode\_pressionar = true;

volatile bool debounce\_ativo = false;

// Callback para desligar o led verde

int64\_t turn\_off\_callback\_verde(alarm\_id\_t id, void \*user\_data) {

gpio\_put(LED\_VERDE, 0);

pode\_pressionar = true;

return 0;

}

// Callback para desligar o led vermelho

int64\_t turn\_off\_callback\_vermelho(alarm\_id\_t id, void \*user\_data) {

gpio\_put(LED\_VERMELHO, 0);

add\_alarm\_in\_ms(3000, turn\_off\_callback\_verde, NULL, false);

return 0;

}

// Callback para desligar o led azul

int64\_t turn\_off\_callback\_azul(alarm\_id\_t id, void \*user\_data) {

gpio\_put(LED\_AZUL, 0);

add\_alarm\_in\_ms(3000, turn\_off\_callback\_vermelho, NULL, false);

return 0;

}

// Callback para debounce do botão

int64\_t debounce\_callback(alarm\_id\_t id, void \*user\_data) {

debounce\_ativo = false;

return 0;

}

// Callback chamado ao pressionar o botão (com debounce)

void tratar\_botao(uint gpio, uint32\_t eventos) {

if (!pode\_pressionar || debounce\_ativo) return; // Evita acionamentos indesejados

debounce\_ativo = true; // Ativa debounce

add\_alarm\_in\_ms(200, debounce\_callback, NULL, false); // 200 ms de debounce

// Liga os leds

gpio\_put(LED\_AZUL, 1);

gpio\_put(LED\_VERMELHO, 1);

gpio\_put(LED\_VERDE, 1);

pode\_pressionar = false; // Bloqueia novos acionamentos até o ciclo terminar

// Agenda desligamento dos leds com atraso de 3 segundos cada

add\_alarm\_in\_ms(3000, turn\_off\_callback\_azul, NULL, false);

}

int main() {

stdio\_init\_all();

// Inicializa os leds

gpio\_init(LED\_AZUL);

gpio\_set\_dir(LED\_AZUL, GPIO\_OUT);

gpio\_put(LED\_AZUL, 0);

gpio\_init(LED\_VERMELHO);

gpio\_set\_dir(LED\_VERMELHO, GPIO\_OUT);

gpio\_put(LED\_VERMELHO, 0);

gpio\_init(LED\_VERDE);

gpio\_set\_dir(LED\_VERDE, GPIO\_OUT);

gpio\_put(LED\_VERDE, 0);

// Inicializa botão com pull\_up

gpio\_init(BOTAO);

gpio\_set\_dir(BOTAO, GPIO\_IN);

gpio\_pull\_up(BOTAO);

// Configura interrupção do botão para detectar borda de descida

gpio\_set\_irq\_enabled\_with\_callback(BOTAO, GPIO\_IRQ\_EDGE\_FALL, true, tratar\_botao);

while (true) {

sleep\_ms(100); // Aguarda para evitar consumo excessivo da CPU

}

}

# Observações Práticas

- Incluir as Bibliotecas “hardware/timer.h” e “hardware/gpio.h” no CMakeList.txt, como indicado abaixo e, em seguida, salvar o arquivo CMakeList.txt:

