Anexo 2 – Blink COM Timer

# Descrição

Este exemplo demonstra a utilização de Timers para tratar eventos fora do laço infinito, sem utilizar recursos valiosos da CPU, liberando a mesma para tarefas que exijam maior poder de processamento.

# Objetivo

- Piscar um LED AZUL (GPIO 12) no loop infinito (tarefa principal, representando aqui a tarefa que necessitaria de maior poder de processamento da CPU);  
- Utilizar um temporizador repetitivo (periódico) e uma função de Callback para realizar outra tarefa, fora do loop infinito e, portanto, sem necessidade da CPU;  
- Após decorrido um determinado intervalo de tempo, acionará o LED VERDE (GPIO 11), fora do loop infinito (representando aqui a tarefa secundária), e após 3 segundos, desliga o mesmo.

# Requisitos

- Placa BitDogLab (RP2040W)  
- SDK oficial do RP2040  
- LED AZUL no GPIO 12  
- LED VERDE no GPIO 12  
- Resistores de 220 Ω (somente necessários no Wokwi)

# Código-fonte (main.c)

#include <stdio.h> // Biblioteca padrão de entrada e saída para uso de funções como printf.

#include "pico/stdlib.h" // Biblioteca padrão do Raspberry Pi Pico para controle de GPIO, temporização e comunicação serial.

#include "hardware/timer.h" // Biblioteca para controle e configuração de temporizadores de hardware.

// Definições de pinos

#define LED\_VERDE 11

#define LED\_AZUL 12

// Função Callback para desligar o LED VERDE que será chamada a cada intervalo definido pelo alarme.

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

gpio\_put(LED\_VERDE, 0); // Desliga o LED.

return 0;

}

// Função Callback para ligar o LED VERDE que será chamada a cada intervalo definido pelo temporizador.

bool repeating\_timer\_callback(struct repeating\_timer \*t) {

gpio\_put(LED\_VERDE, 1); // Liga o LED.

// Adiciona um alarme de 3 segundos para chamar a função de Callback que desligará o LED VERDE.

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

return 1; // Retorna true para manter o temporizador repetindo esta função.

// Se retornasse false, o temporizador pararia de chamar a função de callback.

}

int main()

{

stdio\_init\_all();

gpio\_init(LED\_VERDE); // Inicializa o pino GPIO 11, preparando-o para ser usado como saída digital.

gpio\_set\_dir(LED\_VERDE, GPIO\_OUT); // Configura a direção do pino 11 como saída.

gpio\_put(LED\_VERDE, 0); // Garante que o LED VERDE inicia apagado.

gpio\_init(LED\_AZUL); // Inicializa o pino GPIO 12, preparando-o para ser usado como saída digital.

gpio\_set\_dir(LED\_AZUL, GPIO\_OUT); // Configura a direção do pino 12 como saída.

gpio\_put(LED\_AZUL, 0); // Garante que o LED AZUL inicia apagado.

// Declara uma estrutura para armazenar informações sobre o temporizador repetitivo.

struct repeating\_timer timer;

// Configura um temporizador repetitivo que chama a função 'repeating\_timer\_callback' a cada 5 segundos (5000 ms).

// Parâmetros:

// 5000: Intervalo de tempo em milissegundos (5 segundos).

// repeating\_timer\_callback: Função de callback que será chamada a cada intervalo.

// NULL: Dados adicionais que podem ser passados para a função de callback (não utilizado aqui).

// &timer: Ponteiro para a estrutura que armazenará informações sobre o temporizador.

add\_repeating\_timer\_ms(5000, repeating\_timer\_callback, NULL, &timer);

while (true) {

gpio\_put(LED\_AZUL, 1); // Acende o LED AZUL.

sleep\_ms(500); // Mantem o LED aceso por 500 ms

gpio\_put(LED\_AZUL, 0); // Apaga o LED AZUL.

sleep\_ms(500); // Mantem o LED apagado por 500 ms.

// O loop infinito é usado para realizar outras tarefas, como por exemplo, piscar o LED AZUL.

// Como o temporizador e a função de callback estão gerenciando o LED VERDE, nenhuma instrução

// acerca do LED VERDE é necessária aqui no Loop infinito.

}

}

# Observações Práticas

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

