Anexo 6 - Temporizador de Sistema 1

# Descrição

Este experimento demonstra o uso de um Timer de Sistema. São utilizadas funções relacionadas ao tempo absoluto e verificações periódicas do tempo (time\_reached()). Ele define um tempo futuro (next\_wake\_time) e continuamente verifica se esse tempo foi alcançado. Quando o tempo é alcançado, uma mensagem é impressa, e o tempo futuro é ajustado para o próximo intervalo.

# Objetivos:

- Demonstrar o uso de Timer de Sistema;  
- Utilizar verificações periódicas de tempo;  
- Definir tempos futuros;  
- Executar uma tarefa quando o tempo é alcançado.

# Código-fonte (main.c)

#include <stdio.h>

#include "pico/stdlib.h" // Biblioteca padrão para funcionalidades básicas como GPIO, temporização e E/S.

#include "pico/time.h" // Biblioteca para gerenciamento de tempo, como manipulação de temporizadores e atrasos.

int main() {

// Inicializa a comunicação serial, permitindo o uso de funções como printf.

// Isso é essencial para enviar mensagens de depuração via USB ou UART.

stdio\_init\_all();

// Define um intervalo de tempo em milissegundos para o temporizador.

// Neste caso, o intervalo é de 1000 milissegundos, ou seja, 1 segundo.

uint32\_t interval = 1000; // Intervalo em milissegundos

// Calcula o próximo tempo absoluto em que a ação deve ocorrer.

// get\_absolute\_time() retorna o tempo atual do sistema.

// delayed\_by\_us() calcula um tempo futuro adicionando o intervalo em microsegundos ao tempo atual.

absolute\_time\_t next\_wake\_time = delayed\_by\_us(get\_absolute\_time(), interval \* 1000);

// Loop infinito que mantém o programa em execução contínua.

while (true) {

// Verifica se o tempo atual atingiu ou ultrapassou o tempo definido em next\_wake\_time.

// Se sim, a função time\_reached() retorna true.

if (time\_reached(next\_wake\_time)) {

// Imprime uma mensagem na saída serial indicando que 1 segundo se passou.

printf("1 segundo passou\n");

// Calcula o próximo tempo de despertar, adicionando o intervalo de 1 segundo ao tempo atual.

// Isso garante que a próxima execução aconteça exatamente 1 segundo após a última.

next\_wake\_time = delayed\_by\_us(next\_wake\_time, interval \* 1000);

}

// Introduz uma pequena pausa de 1 milissegundo para reduzir o uso da CPU.

// Isso evita que o loop rode desnecessariamente rápido, economizando recursos.

sleep\_ms(1);

}

}

# Observações Práticas

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

