Rorras Neves da silva / 20251RSE.MTC0099

Complementação do Projeto TempCycleDMA Não foi utilizado outra estratégia para gerenciar o tempo de execução das tarefas em função da Tarefa 1, considerara a principal.

Qual a melhoria que deve ser realizada no novo projeto:
Sincronizar as tarefas em função da primeira utilizado add_repeating_timer_ms nas demais tarefas e repeating_timer_callback para a tarefa 1.

```c
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/watchdog.h"
#include "hardware/timer.h"

#include "setup.h"
#include "tarefa1_temp.h"
#include "tarefa2_display.h"
#include "tarefa3_tendencia.h"
#include "tarefa4_controla_neopixel.h"
#include "neopixel_driver.h"
#include "testes_cores.h"
#include "pico/stdio_usb.h"

// declar as funcoes das tarefas c
void tarefa_3();
void tarefa_5();
void tarefa_4();
void tarefa_2();
void tarefa_1();

// variavelis
float media;
tendencia_t tend;
absolute_time_t ini_tarefa1, fim_tarefa1, tempo1_us, ini_tarefa2, fim_tarefa2, tempo2_us,
    ini_tarefa3, fim_tarefa3, tempo3_us, ini_tarefa4, fim_tarefa4, tempo4_us;

typedef struct
{
        const char *name;     // Nome da tarefa
        uint32_t interval_ms; // Intervalo em ms
        uint32_t last_run;    // Última execução em ms desde boot
        void (*task_fn)();    // Ponteiro para função da tarefa
} Task;

// Definição da estrutura da tarefa/task
Task tasks[] = {
    {"Tarefa 1", 10, 0, tarefa_1},
    {"Tarefa 2", 10, 0, tarefa_2},
    {"Tarefa 3", 10, 0, tarefa_3},
    {"Tarefa 4", 10, 0, tarefa_4},
    {"Tarefa 5", 10, 0, tarefa_5},
};

const int num_tasks = sizeof(tasks) / sizeof(Task);
// === Callback da Tarefa (NAO FUNCIONOU DESSA FORMA)===
bool tarefas_callback(struct repeating_timer *t)
{
        uint32_t now = to_ms_since_boot(get_absolute_time());

        for (int i = 0; i < num_tasks; i++)
```

```
        {
                if (now - tasks[i].last_run >= tasks[i].interval_ms)
                {
                        tasks[i].last_run = now;
                        tasks[i].task_fn();
                }
        }
        watchdog_update();
        return true; // Mantém o timer repetindo
}

int main()
{
        setup(); // Inicializações: ADC, DMA, interrupções, OLED, etc.
        sleep_ms(2000);

        watchdog_enable(3000, false);

        // Inicializa tempos de última execução das tarefas
        uint32_t now = to_ms_since_boot(get_absolute_time());
        for (int i = 0; i < num_tasks; i++)
        {
                tasks[i].last_run = now;
        }

        // Inicia o executor de tarefas
        //struct repeating_timer timer;
        //bool ok = add_repeating_timer_ms(550, tarefas_callback, NULL, &timer);

        while (true)
        {
                uint32_t now = to_ms_since_boot(get_absolute_time());

        for (int i = 0; i < num_tasks; i++)
        {
                if (now - tasks[i].last_run >= tasks[i].interval_ms)
                {
                        tasks[i].last_run = now;
                        tasks[i].task_fn();
                }
        }
        watchdog_update();
        //tight_loop_contents(); // Economiza energia, faz ocioso
        }
}

/*******************************/

void tarefa_1()
{
        // --- Tarefa 1: Leitura de temperatura via DMA ---
        ini_tarefa1 = get_absolute_time();
        media = tarefa1_obter_media_temp(&cfg_temp, DMA_TEMP_CHANNEL);
        fim_tarefa1 = get_absolute_time();
        int64_t tempo1_us = absolute_time_diff_us(ini_tarefa1, fim_tarefa1);
}

/*******************************/
void tarefa_2()
{
        // --- Tarefa 3: Análise da tendência térmica ---
        ini_tarefa3 = get_absolute_time();
        tend = tarefa3_analisa_tendencia(media);
        fim_tarefa3 = get_absolute_time();
        int64_t tempo2_us = absolute_time_diff_us(ini_tarefa2, fim_tarefa2);
}
/*******************************/
```

```c
void tarefa_3()
{
        // --- Tarefa 2: Exibição no OLED ---
        ini_tarefa2 = get_absolute_time();
        tarefa2_exibir_oled(media, tend);
        fim_tarefa2 = get_absolute_time();
        int64_t tempo3_us = absolute_time_diff_us(ini_tarefa3, fim_tarefa3);
}
/*****************************/
void tarefa_4()
{
        // --- Tarefa 4: Cor da matriz NeoPixel por tendência ---
        ini_tarefa4 = get_absolute_time();
        tarefa4_matriz_cor_por_tendencia(tend);
        fim_tarefa4 = get_absolute_time();
        int64_t tempo4_us = absolute_time_diff_us(ini_tarefa4, fim_tarefa4);
}
void tarefa_5()
{
        printf("Temperatura: %.2f °C | T1: %.3fs | T2: %.3fs | T3: %.3fs | T4: %.3fs | Tendência:
%s\n",
               media,
               tempo1_us / 1e6,
               tempo2_us / 1e6,
               tempo3_us / 1e6,
               tempo4_us / 1e6,
               tendencia_para_texto(tend));

        // --- Tarefa 5: Extra ---
        if (media < 1)
        {
                npSetAll(COR_BRANCA);
                npWrite();
                sleep_ms(1000);
                npClear();
                npWrite();
        }
}
```