RORRAS NEVES DA SILVA / 20251RSE.MTC0099

Tarefa 1: Self-Test
• Testar LEDs RGB: acender e apagar sequencialmente.
• Testar Buzzer: gerar som simples.
• Testar botões A, B, C e joystick SW.
• Testar Joystick analógico (ADC0 e ADC1).
• Testar Microfone (ADC2).
• Imprimir resultados na porta USB com pequenas pausas para visualização.
• Deletar a si própria após concluir (vTaskDelete(NULL)).
Tarefa 2: Alive Task
• Piscar o LED vermelho (GPIO 13) indicando o funcionamento.
• Ciclo de 1000ms (500ms ligado e 500ms desligado).
Tarefa 3: Monitor de Joystick e Alarme
• Ler e imprimir as tensões dos eixos X (ADC1) e Y (ADC0) do joystick.
• Imprimir leituras a cada 50ms na porta USB.
• Se qualquer eixo exceder 3.00V, ativar o buzzer (GPIO 21).
• Desligar o buzzer quando os valores voltarem ao normal.

```c
#include "pico/stdlib.h"
#include "pico/stdio_usb.h"
#include "hardware/adc.h"
#include "FreeRTOS.h"
#include "task.h"
#include <stdio.h>
#include "hardware/pwm.h"
#include "pwm.h"


// GPIOs
#define LED_RED 13
#define LED_GREEN 11
#define LED_BLUE 12
//#define BUZZER 21
#define BUTTON_A 5
#define BUTTON_B 6
#define JOY_BTN 22
#define JOY_VRX 27   // ADC1
#define JOY_VRY 26   // ADC0
#define MIC_INPUT 28 // ADC2

#define STEP_DELAY_MS 1000

// Handles
TaskHandle_t LEDsHandle, BuzzerHandle, ButtonsHandle, JoystickHandle, MicrophoneHandle, AliveHandle,
MonitorJoyHandle;

void test_leds_task(void *params)
{
    ulTaskNotifyTake(pdTRUE, portMAX_DELAY); // espera ser ativada

    gpio_init(LED_RED);
    gpio_set_dir(LED_RED, GPIO_OUT);
    gpio_init(LED_GREEN);
    gpio_set_dir(LED_GREEN, GPIO_OUT);
    gpio_init(LED_BLUE);
    gpio_set_dir(LED_BLUE, GPIO_OUT);
```

```
    gpio_put(LED_RED, 1);
    vTaskDelay(pdMS_TO_TICKS(500));
    gpio_put(LED_RED, 0);
    printf("LED vermelho testado.\n");
    gpio_put(LED_GREEN, 1);
    vTaskDelay(pdMS_TO_TICKS(500));
    gpio_put(LED_GREEN, 0);
    printf("LED verde testado.\n");
    gpio_put(LED_BLUE, 1);
    vTaskDelay(pdMS_TO_TICKS(500));
    gpio_put(LED_BLUE, 0);
    printf("LED azul testado.\n");

    xTaskNotifyGive(BuzzerHandle);
    vTaskDelete(NULL);
}

void test_buzzer_task(void *params)
{
    ulTaskNotifyTake(pdTRUE, portMAX_DELAY); // espera ser ativada

    buzzer_pwm_init();         // Liga o buzzer com tom
    vTaskDelay(pdMS_TO_TICKS(1000));            // Toca por 1 segundo
    buzzer_pwm_stop();         // Para o som
    printf("Buzzer testado.\n");

    xTaskNotifyGive(ButtonsHandle);
    vTaskDelete(NULL);
}

void test_buttons_task(void *params)
{
    ulTaskNotifyTake(pdTRUE, portMAX_DELAY); // espera ser ativada

    gpio_init(BUTTON_A);
    gpio_set_dir(BUTTON_A, GPIO_IN);
    gpio_pull_down(BUTTON_A);
    gpio_init(BUTTON_B);
    gpio_set_dir(BUTTON_B, GPIO_IN);
    gpio_pull_down(BUTTON_B);
    gpio_init(JOY_BTN);
    gpio_set_dir(JOY_BTN, GPIO_IN);
    gpio_pull_down(JOY_BTN);

    printf("Pressione os botões A, B e Joystick...\n");

    while (gpio_get(BUTTON_A) && gpio_get(BUTTON_B) && gpio_get(JOY_BTN))
    {
        vTaskDelay(pdMS_TO_TICKS(100));
    }
    printf("Botão A: %d | B: %d | Joy: %d\n", gpio_get(BUTTON_A), gpio_get(BUTTON_B),
gpio_get(JOY_BTN));

    xTaskNotifyGive(JoystickHandle);
    vTaskDelete(NULL);
}

void test_joystick_adc_task(void *params)
{
    ulTaskNotifyTake(pdTRUE, portMAX_DELAY); // espera ser ativada

    adc_gpio_init(JOY_VRY);
    adc_gpio_init(JOY_VRX);
    adc_select_input(0);
    uint16_t vry = adc_read();
    adc_select_input(1);
    uint16_t vrx = adc_read();
```

```
    printf("Joystick VRx: %u | VRy: %u\n", vrx, vry);

    xTaskNotifyGive(MicrophoneHandle);
    vTaskDelete(NULL);
}

void test_microphone_task(void *params)
{
    ulTaskNotifyTake(pdTRUE, portMAX_DELAY); // espera ser ativada

    adc_gpio_init(MIC_INPUT);
    adc_select_input(2);
    uint16_t mic = adc_read();
    printf("Microfone: %u\n", mic);

    printf("Todos os testes concluídos!\n");
    xTaskNotifyGive(AliveHandle);
    xTaskNotifyGive(MonitorJoyHandle);
    vTaskDelete(NULL);
}

void alive_task(void *params) {


    while (1) {
        gpio_put(LED_RED, 1);  // Liga LED
        vTaskDelay(pdMS_TO_TICKS(500));

        gpio_put(LED_RED, 0);  // Desliga LED
        vTaskDelay(pdMS_TO_TICKS(500));
    }
}

void monitor_joystick_task(void *params) {

    ulTaskNotifyTake(pdTRUE, portMAX_DELAY); // espera ser ativada

    while (1) {
        adc_select_input(1); // ADC1 - X
        uint16_t x = adc_read();

        adc_select_input(0); // ADC0 - Y
        uint16_t y = adc_read();

        printf("Joystick X: %u | Y: %u\n", x, y);

        // Verifica se algum eixo passou de 3000
        if (x > 3000 || y > 3000) {
            buzzer_pwm_init(); // Liga buzzer
        } else {
             buzzer_pwm_stop(); // Desliga buzzer
        }

        vTaskDelay(pdMS_TO_TICKS(50));
    }
}

int main()
{
    stdio_usb_init();
    sleep_ms(2000);
    printf("==== Iniciando Testes com FreeRTOS ====\n");
    adc_init();

    xTaskCreate(test_leds_task, "LEDs", 256, NULL, 5, &LEDsHandle);
    xTaskCreate(test_buzzer_task, "Buzzer", 256, NULL, 4, &BuzzerHandle);
    xTaskCreate(test_buttons_task, "Buttons", 512, NULL, 3, &ButtonsHandle);
```

```
    xTaskCreate(test_joystick_adc_task, "Joystick", 512, NULL, 2, &JoystickHandle);
    xTaskCreate(test_microphone_task, "Microfone", 512, NULL, 1, &MicrophoneHandle);
    xTaskCreate(alive_task, "Alive", 256, NULL, 0, NULL);  // prioridade mais baixa
    xTaskCreate(monitor_joystick_task, "JoyMonitor", 512, NULL, 1, &MonitorJoyHandle);

    // Inicia a primeira tarefa (LEDs)
    xTaskNotifyGive(LEDsHandle);

    vTaskStartScheduler();
    while (1)
    {
        tight_loop_contents();
    }
}
```