Nome completo: Marcony Henrique Bento Souza
Matrícula: 20251RSE.MTC0089
Email: marconyhenrique321@gmail.com
Link do wokwi: https://wokwi.com/projects/431111907582992385

Enunciado: Semáforo de Trânsito Interativo Criar um semáforo de trânsito, com acionamento de travessia para pedestres e indicação de tempo restante

Código c usado na BitDogLab:

```c
/*
  Aluno: Marcony Henrique Bento Souza
  Email: marconyhenrique321@gmail.com
  Matrícula: 20251RSE.MTC0089
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include "pico/stdlib.h"
#include "hardware/timer.h"
#include "hardware/clocks.h"
#include "hardware/pwm.h"
#include "pico/binary_info.h"
#include "hardware/i2c.h"
#include "inc/ssd1306.h"

#define BUTTON_A 5
#define BUTTON_B 6
#define RED_LED 13
#define GREEN_LED 11
#define BUZZER_PIN 10
#define BUZZER_FREQUENCY 20000

#define I2C_SDA 14
#define I2C_SCL 15

uint8_t ssd[ssd1306_buffer_length];

// Preparar área de renderização para o display (ssd1306_width pixels
por ssd1306_n_pages páginas)
struct render_area frame_area = {
    start_column : 0,
```

```
    end_column : ssd1306_width - 1,
    start_page : 0,
    end_page : ssd1306_n_pages - 1
};

int64_t long_time = 10000;
int64_t short_time = 3000;
bool debounce = false;
alarm_id_t Ids[3];
int color = 1;

void pwm_init_buzzer(uint pin);
void beep(uint pin);
void button_callback(uint gpio, uint32_t events);
int64_t red_led_callback(alarm_id_t id, void *user_data);
int64_t yellow_led_callback(alarm_id_t id, void *user_data);
int64_t green_led_callback(alarm_id_t id, void *user_data);
int64_t turn_on_buzzer_callback(alarm_id_t id, void *user_data);
int64_t turn_off_buzzer_callback(alarm_id_t id, void *user_data);
void setup();

int main() {
    stdio_init_all();

    setup();

    gpio_set_irq_enabled_with_callback(BUTTON_A, GPIO_IRQ_EDGE_FALL,
true, &button_callback);
    gpio_set_irq_enabled(BUTTON_B, GPIO_IRQ_EDGE_FALL, true);

    add_alarm_in_ms(0, red_led_callback, NULL, true);


    while (true){}
}

void pwm_init_buzzer(uint pin) {
    // Configurar o pino como saída de PWM
    gpio_set_function(pin, GPIO_FUNC_PWM);

    // Obter o slice do PWM associado ao pino
    uint slice_num = pwm_gpio_to_slice_num(pin);
```

```
    // Configurar o PWM com frequência desejada
    pwm_config config = pwm_get_default_config();
    pwm_config_set_clkdiv(&config, clock_get_hz(clk_sys) /
(BUZZER_FREQUENCY * 4096)); // Divisor de clock
    pwm_init(slice_num, &config, true);

    // Iniciar o PWM no nível baixo
    pwm_set_gpio_level(pin, 0);
}

void beep(uint pin) {
    // Obter o slice do PWM associado ao pino
    uint slice_num = pwm_gpio_to_slice_num(pin);

    // Configurar o duty cycle para 50% (ativo)
    pwm_set_gpio_level(pin, 2048);
}

void button_callback(uint gpio, uint32_t events)
{
    // sleep_ms(50); // Debounce de 50ms
    if(gpio == BUTTON_A && !debounce)
    {
        debounce = true;
        for(int i = 0; i < 3; i++)
        {
        cancel_alarm(Ids[i]);
        }
        printf("Botão de Pedestres A acionado\n");

        memset(ssd, 0, ssd1306_buffer_length);
        render_on_display(ssd, &frame_area);

        ssd1306_draw_string(ssd, 5, 0, "Botão A acionado");
        render_on_display(ssd, &frame_area);

        color = -1;
        add_alarm_in_ms(0, yellow_led_callback, NULL, true);
    }
    else if(gpio == BUTTON_B && !debounce)
    {
        debounce = true;
        for(int i = 0; i < 3; i++)
```

```
        {
        cancel_alarm(Ids[i]);
        }
        printf("Botão de Pedestres B acionado\n");

        memset(ssd, 0, ssd1306_buffer_length);
        render_on_display(ssd, &frame_area);

        ssd1306_draw_string(ssd, 5, 0, "Botão B acionado");
        render_on_display(ssd, &frame_area);
        color = -1;
        add_alarm_in_ms(0, yellow_led_callback, NULL, true);
        return;
    }
}

int64_t countdown_callback(alarm_id_t id, void *user_data)
{
    char temp_str[16];

    printf("Contagem regressiva: %d segundos\n", user_data);

    memset(ssd, 0, ssd1306_buffer_length);
    render_on_display(ssd, &frame_area);

    ssd1306_draw_string(ssd, 5, 0, "Contagem regressiva:");
    sprintf(temp_str, "%d segundos", user_data);
    ssd1306_draw_string(ssd, 5, 10, temp_str);
    render_on_display(ssd, &frame_area);

    if (user_data > 1)
    {
        add_alarm_in_ms(1000, countdown_callback, (void
*)((int)user_data - 1), true);
    }
    else
    {
        color = 0;
    }
    return 0;
}

int64_t turn_on_buzzer_callback(alarm_id_t id, void *user_data)
```

```
{
    beep(BUZZER_PIN);
    add_alarm_in_ms(100, turn_off_buzzer_callback, (void
*)((int)user_data), true);
    return 0;
}

int64_t turn_off_buzzer_callback(alarm_id_t id, void *user_data)
{
    if(user_data == 0)
    {
        pwm_set_gpio_level(BUZZER_PIN, 0);
        return 0;
    }
    else
    {
        pwm_set_gpio_level(BUZZER_PIN, 0);
        add_alarm_in_ms(20, turn_on_buzzer_callback, (void
*)((int)user_data - 1), true);
    }
    return 0;
}

int64_t red_led_callback(alarm_id_t id, void *user_data)
{
    Ids[0] = id;
    if(color == 1)
    {
        gpio_put(RED_LED, 1);
        gpio_put(GREEN_LED, 0);
        if(debounce)
        {
        add_alarm_in_ms(5000, countdown_callback, (void *)((int)5),
true);
        color = 1;
        }
        else
        {
            color = 0;
        }
        add_alarm_in_ms(long_time, green_led_callback, NULL, true);
        printf("Sinal: Vermelho\n");
```

```
        memset(ssd, 0, ssd1306_buffer_length);
        render_on_display(ssd, &frame_area);

        ssd1306_draw_string(ssd, 5, 0, "Sinal: Vermelho");
        render_on_display(ssd, &frame_area);
        add_alarm_in_ms(0, turn_on_buzzer_callback, (void *)(2), true);
    }
    return 0;
}

int64_t yellow_led_callback(alarm_id_t id, void *user_data)
{
    Ids[1] = id;
    if(color == -1)
    {
        gpio_put(RED_LED, 1);
        gpio_put(GREEN_LED, 1);
        add_alarm_in_ms(short_time, red_led_callback, NULL, true);
        printf("Sinal: Amarelo\n");

        memset(ssd, 0, ssd1306_buffer_length);
        render_on_display(ssd, &frame_area);

        ssd1306_draw_string(ssd, 5, 0, "Sinal: Amarelo");
        render_on_display(ssd, &frame_area);
        color = 1;
        add_alarm_in_ms(0, turn_on_buzzer_callback, (void *)(1), true);
    }
    return 0;
}

int64_t green_led_callback(alarm_id_t id, void *user_data)
{
    Ids[2] = id;
    if(color == 0)
    {
        debounce = false; //O botão só pode ser chamado novamente se
estiver no sinal verde
        gpio_put(RED_LED, 0);
        gpio_put(GREEN_LED, 1);
        add_alarm_in_ms(long_time, yellow_led_callback, NULL, true);
        printf("Sinal: Verde\n");
```

```
        memset(ssd, 0, ssd1306_buffer_length);
        render_on_display(ssd, &frame_area);

        ssd1306_draw_string(ssd, 5, 0, "Sinal: Verde   ");
        render_on_display(ssd, &frame_area);

        add_alarm_in_ms(0, turn_on_buzzer_callback, (void *)(0), true);
        color = -1;
    }

    return 0;
}

void setup()
{
    gpio_init(BUTTON_A);
    gpio_set_dir(BUTTON_A, GPIO_IN);
    gpio_pull_up(BUTTON_A);

    gpio_init(BUTTON_B);
    gpio_set_dir(BUTTON_B, GPIO_IN);
    gpio_pull_up(BUTTON_B);

    gpio_init(RED_LED);
    gpio_set_dir(RED_LED, GPIO_OUT);

    gpio_init(GREEN_LED);
    gpio_set_dir(GREEN_LED, GPIO_OUT);

    pwm_init_buzzer(BUZZER_PIN);

    i2c_init(i2c1, ssd1306_i2c_clock * 1000);
    gpio_set_function(I2C_SDA, GPIO_FUNC_I2C);
    gpio_set_function(I2C_SCL, GPIO_FUNC_I2C);
    gpio_pull_up(I2C_SDA);
    gpio_pull_up(I2C_SCL);

    // Processo de inicialização completo do OLED SSD1306
    ssd1306_init();

    calculate_render_area_buffer_length(&frame_area);

    // zera o display inteiro
```

```
    memset(ssd, 0, ssd1306_buffer_length);
    render_on_display(ssd, &frame_area);
}
```