```c
/*-------------------------------------------------------------------------------------#
#                         ProJeto_Lampada_AJustavel - PROJ_FINA1.1.5                  #
#                                 ======================                              #
#             O objetivo deste projeto é mostrar o uso da aplicação de um sistema simples  #
#                     que pode ser aperfeicoado para uso em situações reais           #
#                Autor: Antonio josé #File: PROJ_01_INT #Date : março, 2025           #
# ------------------------------------------------------------------------------- ---*/
/*       ========= Aplicação utilizando Display OLED, PWM, LEDS e BOTÕES ============       #
#                                                                                     #
#           ============== ProJeto_Residencia: EmbarcaTech =====================        #
#                                                                                     #
# ------------------------------------------------------------------------------------*/

/* ------------------------------------------------------------------------------------#
#                              BIBLIOTECAS UTILIZADAS                                 #
#                           ================================                          #
# ------------------------------------------------------------------------------------#*/
#include <stdio.h> //Entrada e saida padrao                                           #
#include <string.h> //Manipulação de estring(ex: menset, snprintf)                    #
#include <stdlib.h> //Utilidade geral e alocação na memoria                           #
#include "pico/stdlib.h" //Guarda-Chuva                                               #
#include "pico/binary_info.h" //Para informacoes binarias e depuração                 #
#include "inc/ssd1306.h" //Controle do DISPLAY_OLED SSD1306                          #
#include "hardware/i2c.h" //Comunicação I2C                                          #
#include "hardware/pwm.h" //Controle do PWM                                          #
/* ------------------------------------------------------------------------------------*/

/* ------------------------------------------------------------------------------------#
#                             DEFINICOES E PINOS UTILIZADOS                           #
#                           ================================                          #
# ------------------------------------------------------------------------------------#*/
#define VERDE 11 //Unico utilizado no programa                                        #
#define AZUL 12 //definido                                                            #
#define VERMELHO 13 //definido                                                        #
// #define BTA 5 // Botão A antes utilizado sem o circuito para incrementar a luminosidade  #
#define BTA 3 // Botão A                                                              #
#define BTB 6 // Botão B                                                              #
const uint I2C_SDA = 14; //PINO SDA utilizado para comunicação com o display         #
const uint I2C_SCL = 15; //PINO SCL utilizado para comunicação com o display         #
/* ------------------------------------------------------------------------------------*/


/* ------------------------------------------------------------------------------------#
#                           VARIAVEIS DE CONFIGURACAO DO PWM                          #
#                           ================================                          #
# ------------------------------------------------------------------------------------#*/
const uint16_t periodo = 200; //Periodo do PWM (valor máximo de tensão)               #
const float divisor_pwm = 16.0; //Divisor fracional do clock do PWM                   #
const uint16_t stemp_pwm = 40; //Passo de incremento e decremento do duty cycle do LED  #
/* ------------------------------------------------------------------------------------*/


/* ------------------------------------------------------------------------------------#
#                        VARIAVEIS PARA AJUSTE DE LUMINOSIDADE                        #
#                           ================================                          #
# ------------------------------------------------------------------------------------#*/
uint8_t nivelLuminosidade = 0; // Nível inicial da luminosidade (1 a 5)              #
uint8_t contadorA = 0; // Contador para oS botoes                                     #
/* ------------------------------------------------------------------------------------*/


/* ------------------------------------------------------------------------------------#
#                    VARIAVEIS PARA O TRATAMENTO DO DEBOUCE NOS BOTOES                #
#                           ================================                          #
# ------------------------------------------------------------------------------------#*/
//Definições para tratamento do debaunce pra garantir que n aJam leituras incorretas  #
uint32_t last_button_a_time = 0; //Armazena o ultimo tempo em que o botao A foi precionado  #
uint32_t last_button_b_time = 0; //Armazena o ultimo tempo que o botao B foi precionado  #
const uint32_t DEBOUNCE_DELAY = 200; //Tempo de debounce em mile segundos             #
/* ------------------------------------------------------------------------------------*/


/* ------------------------------------------------------------------------------------#
#                         FUNÇÃO PARA HABILITAR O PWM DO PINO                         #
#                           ================================                          #
# ------------------------------------------------------------------------------------#*/
void config_pwm(int led) { //Recebe um parametro do pino do led                       #
    uint slice; //Variável para obter o slice do PWM, divisor da frequência         #
    gpio_set_function(led, GPIO_FUNC_PWM); //Configura para saida PWM                #
    slice = pwm_gpio_to_slice_num(led); //Retorna um slice permitindo o controle do sinal  #

    //Configuração do periodo wrap duty cycle / aJuste do sinal de saida            #
    pwm_set_clkdiv(slice, divisor_pwm); //Define o divisor do clock do PWM          #
    pwm_set_wrap(slice, periodo); //Configura o valor máximo do contador(periodo pwm)  #
    pwm_set_gpio_level(led, stemp_pwm); //Define o nivel de pwm no pino correspondente  #
    pwm_set_enabled(slice, true); //Habilita o pwm no slice correspondente
}

/*------------------------------- FUNÇÕES PARA O DISPLAY ----------------------------------#
#                           FUNÇÃO DE REDENRIZAÇÃO DO DISPLAY                         #
#                         ===================================                         #
#                                                                                     #
# A função Render_on_display e responssavel por redenrizar o conteudo do buffer SSD na area #
# especificado pelo parametro do Display                                              #
# ------------------------------------------------------------------------------------ #
#                       ============== PARAMETROS ==============                      #
#           uint8_t *ssd: ponteiro para o buffer que contem os dados a serem exibidos  #
#          struct render_area *area: ponteiro para uma estrutura que define a area do  #
#           display onde o conteudo sera redenrizado                                  #
#                    ======= render_on_display atualiza a tela ==========             #
# ------------------------------------------------------------------------------------*/
```

```c
void render_on_display(uint8_t *ssd, struct render_area *area);


/*--------------------------------------------------------------------------------------------#
#                        FUNÇÃO PARA DESENHAR A ESTRING NO DISPLAY                            #
#                          ====================================                               #
#                                                                                             #
# A função extern_void ssd1306_drow_string é responsavel por desenhar umma estring no         #
# buffer SSD a partir de coordenadas, manipula o buffer para incluir a reprentação grafica    #
# nas cordenadas corretas no limite do display                                                #
#                                                                                             #
# ------------------------------------------------------------------------------------------- #
#                         =============== PARAMETROS ===============                          #
#                      uint8_t *ssd: buffer onde a string sera desenhada                      #
#               int16_t x: coordenada x onde a string comeca a ser desenhada                  #
#               int16_t y: coordenada y onde a string comeca a ser desenhada                  #
#               char *string: A string a ser desenhada                                        #
#             ====== ssd1306_draw_string lida com a renderização de textos =======            #
#---------------------------------------------------------------------------------------------#*/
extern void ssd1306_draw_string(uint8_t *ssd, int16_t x, int16_t y, char *string);


/*--------------------------------------------------------------------------------------------#
#                      FUNÇÂO PARA CALCULAR O CUMPRIMENTO DO BUFFER                           #
#                          ====================================                               #
#                                                                                             #
# A função calculate_render_area_buffer_length calcula o cumprimento do buffer necessario     #
# para redenrizar uma area especifica do Dysplay                                              #
# A implementação calcula o tamanho do buffer com base nas dimensoes da area(largura altura)#
#
#                                                                                             #
# ------------------------------------------------------------------------------------------- #
#                         =============== PARAMETROS ===============                          #
#   struct render_area *area: ponteiro para estrutura que define a area a ser redenrizada.    #
#                                                                                             #
# == calculate_render_area_buffer_length ajuda a garantir que haja espaço suficiente no ==    #
# == para a área que se deseja renderizar.buffer  ==                                          #
#
#---------------------------------------------------------------------------------------------#*/
void calculate_render_area_buffer_length(struct render_area *area);


/*--------------------------------------------------------------------------------------------#
#                      FUNCAO PARA EXIBIR A MENSAGEM NO DISPLAY                              #
#                          ====================================                               #
#                                                                                             #
# ------------------------------------------------------------------------------------------- #
#                         =============== PARAMETROS ===============                          #
# uint8_t *ssd: ponteiro para o buffer onde os dados da mensagem serao armazenados            #
# struct render_area *frame_area: ponteiro para a area do display para exibição da mensagem #
# const char *mensagem: string que sera exibida                                               #
#---------------------------------------------------------------------------------------------#*/
void exibir_mensagem(uint8_t *ssd, struct render_area *frame_area, const char *mensagem) {
    memset(ssd, 0, ssd1306_buffer_length); //Usa o tamanho correto do buffer para: Limpar o buffer ------------------------------------------------- #
    ssd1306_draw_string(ssd, 5, 0, mensagem); //chama a função ssd1306_draw_string para desenhar a mensagem a partir das cordenadas do dispaly(5,0)--#
    render_on_display(ssd, frame_area);// chama a função render_on_display para redenrizar o conteudo do buffer do Dysplay na area especifica -------#
} // -------------------------------------------------------------------------------------------------------------------------------------------#


/*--------------------------------------------------------------------------------------------#
#                      FUNCOES PARA EXIBIR AS MENSAGENS NO DISPLAY                            #
#                          ====================================                               #
#                                  LAMPADA DESLIGADA                                          #
#    A FUNÇÃO da função exibir lampada é exibir a mensagem LAMDADA DESLIGADA no display       #                                                       #
# ------------------------------------------------------------------------------------------- #
#                         =============== PARAMETROS ===============                          #
# uint8_t *ssd: ponteiro para o buffer onde os dados da mensagem serao armazenados            #
# struct render_area *frame_area: ponteiro para area do dispay                                #
# declara um array de char de 30 caracteris para armazenar a mensagem                         #
# utiliza o snprintf() para formatar e armazenar em mensagem                                  #
# chamada da função EXIBIR MENSAGEM passando buffer, area de redenrização  e a mensagem...    #
# formatada.                                                                                  #
#                                                                                             #
#---------------------------------------------------------------------------------------------#*/
void exibir_lampada_desligada(uint8_t *ssd, struct render_area *frame_area) {
    char mensagem[30]; // array de 30 caracteris -------------------------------------------------------------------------#
    snprintf(mensagem, sizeof(mensagem), "Lampada desligada"); // mensagem e a função sizeof para percorrer os caracteris --#
    exibir_mensagem(ssd, frame_area, mensagem); //buffer, area-redenrização, mensagem formatada/realiza a tarefa de exibição#
} //-----------------------------------------------------------------------------------------------------------------------#

/*--------------------------------------------------------------------------------------------#
#                      FUNÇÂO PARA ATUALIZAR A MENSAGEM DE LUMINOSIDADE                       #
#                          ====================================                               #
#                         Obetivo: exibir o nivel de luminosidade                             #
# ------------------------------------------------------------------------------------------- #
#                         =============== PARAMETROS ===============                          #
# uint8_t *ssd: ponteiro para o buffer onde os dados da mensagem serao armazenados            #
# struct render_area *frame_area: ponteiro para area do dispay para exibição de imagem        #
# declara um array de char de 20 caracteris para armazenar a mensagem                         #
# utiliza o nsprintf(para formatae e gravar a string, o %d é substituido pelo valor da...     #
# variavel nivel de luminosidade.                                                             #
# chama a função Exibir_mensagem: passando o buffer e area de redenrização da mensagem...     #
# linpando o buffer e desenhando a mensagem                                                   #
#                                                                                             #
#---------------------------------------------------------------------------------------------#*/
void exibir_nivel_luminosidade(uint8_t *ssd, struct render_area *frame_area) {
    char mensagem[20];//array 20caracteris
-----------------------------------------------------------------------------------------------------------------------#
    snprintf(mensagem, sizeof(mensagem), "Luminosidade: %d", nivelLuminosidade); // mensagem e a função sizeof para percorrer os caracteris, exibir o nivel de
luninosidade#
```

```
    exibir_mensagem(ssd, frame_area, mensagem);//limpa e desenha a mensagem
-------------------------------------------------------------------------------------------#
}//-------------------------------------------------------------------------------------------------------------------------------------------------------
----#



/*-------------------------------------------------------------------------------------------#
#              FUNÇÃO PARA INCREMENTAR E DECREMENTAR A MENSAGEM DE LUMINOSIDADE              #
#                          ====================================                             #
#        Obetivo: ajustar os clicks no botao e luminosidade e exibir as mensagens           #
# ------------------------------------------------------------------------------------------ #
#                         =============== PARAMETROS ==============                        #
# pino do led                                                                               #
# pino do Botao                                                                             #
# uint8_t *ssd: ponteiro para o buffer onde os dados da mensagem serao armazenados          #
# struct render_area *frame_area: ponteiro para area do dispay para exibição de imagem      #
#                                                                                           #
#-------------------------------------------------------------------------------------------#*/
void Incrementar(int led, int botao, uint8_t *ssd, struct render_area *frame_area) {
    uint32_t current_time = to_ms_since_boot(get_absolute_time()); //a função que trata de Temporização, ela obtem o tempo atual em mile segundos sendo armazenada por
esta variavel o tempo atual -----------------#
    uint32_t *last_time = (botao == BTA) ? &last_button_a_time : &last_button_b_time; //a variavel last_time determina qual botao foi precionado e armazena o tempo do
ultimo precionamento em um ponteiro(last time)#
    //depois verifica se o botao foi precionado  e se o tempo desde o ultimo precionamento e maior que um a trazo de debouce, para que n tenha leitura incorreta
-----------------------------------------------------#

    // Verifica se o botão foi pressionado e se passou tempo suficiente desde o último pressionamento
    if (gpio_get(botao) == 0 && (current_time - *last_time >= DEBOUNCE_DELAY)) {
        *last_time = current_time; // Atualiza o tempo do último pressionamento

        if (botao == BTA && nivelLuminosidade < 5) {// if o botao for clicado e o nivel de luminosidade estiver menor que 5
            contadorA++; //incrementa 1
            nivelLuminosidade++; // incrementa a luminosidade
            printf("Botão A pressionado: %d - Nível de luminosidade da lampada: %d\n", contadorA, nivelLuminosidade); //exibe no display serial
            exibir_nivel_luminosidade(ssd, frame_area); // chama a função para exibir no display o nivel de luminosidade atual
        } else if (botao == BTB && nivelLuminosidade > 1) {
            nivelLuminosidade--; //decrementa a luminosidade
            printf("Botão B pressionado - Nível de luminosidade da lampada: %d\n", nivelLuminosidade);// exibi no monitor serial
            exibir_nivel_luminosidade(ssd, frame_area); // chama a função para exibir no display o nivel de luminosidade atual
        } else if (botao == BTB && nivelLuminosidade == 1) {// se o nivel da luminosodade ao clicar no botao chegar a 1
            nivelLuminosidade--;// ao clicar no botao e chagar em 0 decrementa
            if (nivelLuminosidade == 0) {
                gpio_put(led, 0); //chegando em 0 apaga
                printf("Botão B pressionado - Lampada desligada - Nível de luminosidade: %d\n", nivelLuminosidade); // exibi no monitor serial
                exibir_lampada_desligada(ssd, frame_area); //chama a função que exibe a mensagem de apagada
            }
        }
    }
}
/*------------------------------ FUNÇÃO PRINCIPAL ----------------------------------------*/
int main() {
    stdio_init_all(); // Inicializa os tipos stdio padrão presentes ligados ao binário

    // Inicialização do i2c
    i2c_init(i2c1, ssd1306_i2c_clock * 1000);//interface 12c1, com frequencia do clock definida
    gpio_set_function(I2C_SDA, GPIO_FUNC_I2C); //inicializa os pinos para saida i2c
    gpio_set_function(I2C_SCL, GPIO_FUNC_I2C); //inicializa os pinos para saida i2c
    gpio_pull_up(I2C_SDA);// utilizaa o resistor que coloca em nivel logico alto
    gpio_pull_up(I2C_SCL);// utilizaa o resistor que coloca em nivel logico alto

    // Inicializa os LEDs
    gpio_init(VERDE);
    gpio_set_dir(VERDE, GPIO_OUT);

     // Inicializa os botões
    gpio_init(BTA);
    gpio_set_dir(BTA, GPIO_IN);
    gpio_pull_up(BTA); // Resistor pull-up para o botão A

    gpio_init(BTB);
    gpio_set_dir(BTB, GPIO_IN);
    gpio_pull_up(BTB); // Resistor pull-up para o botão B

    config_pwm(VERDE); // configura o PWM para o LED verde chamando a função de configuração


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

    // Preparar área de renderização para o display (ssd1306_width pixels por ssd1306_n_pages páginas)
    struct render_area frame_area = {//especifica a ara do display onde ira redenrizar
        .start_column = 0,//define a coluna de incio 0
        .end_column = ssd1306_width - 1,// coluna de fim 1
        .start_page = 0, //pagina de inicio da redenrização e fim
        .end_page = ssd1306_n_pages - 1 //termina na coluna maxima do display abraangendo todas as paginasdisponiveis
    };
    // FUNÇÃO de calculo do cumprimento do buffer necessario por armazenar os dados da area redenrizada
    calculate_render_area_buffer_length(&frame_area);

    // Zera o display inteiro
    uint8_t ssd[ssd1306_buffer_length]; //declara um array chamado ssd usado como buffer para armazenar os dados que serao enviados para o display
    memset(ssd, 0, ssd1306_buffer_length);// inicializa todos os bytes do buffer SSD zerando o display
    render_on_display(ssd, &frame_area);

    // Parte do código para exibir a mensagem no display
    render_on_display(ssd, &frame_area);//envia os dados para o Display

    if (nivelLuminosidade == 0) {
        exibir_lampada_desligada(ssd, &frame_area);
    } else {
        exibir_nivel_luminosidade(ssd, &frame_area);
```

```
    }

    while (true) {
        // Ajusta o nivel de PWM baseado no nivel de luminosidade
        if (nivelLuminosidade > 0) { //se o nivel da luminosidade maior que 0
            pwm_set_gpio_level(VERDE, periodo / 5 * nivelLuminosidade);// aJusta o pwm. calcula a fração do periodo (valor maximo de tensao) / 5 ultiplicado pelo nivel
de luminosidade
        } else {
            pwm_set_gpio_level(VERDE, 0); // Certifica-se que o LED está desligado quando nivelLuminosidade é 0
        }

        // Chama a função Incrementar com os parâmetros apropriados
        Incrementar(VERDE, BTA, ssd, &frame_area); // Exemplo de chamada com LED Verde e Botão A. ssd bufer que contem os dados a serem exibidos e &frame_area: area de
redenrização usada na função
        Incrementar(VERDE, BTB, ssd, &frame_area); // Exemplo de chamada com LED Verde e Botão B.  ssd bufer que contem os dados a serem exibidos e &frame_area: area
de redenrização usada na função

        sleep_ms(10); // Delay de 1 segundo
    }
    return 0;
}
```