Aluna: PRISCILA DO ESPIRITO SANTO SOUSA

link do vídeo: https://drive.google.com/drive/my-drive

Documentação do Sistema de Alerta de Segurança com Microfone

Introdução

Este projeto implementa um sistema de alerta de segurança utilizando a placa Raspberry Pi Pico, um microfone, um display OLED, LEDs de estado, um buzzer e botões. O sistema detecta sons no ambiente e aciona um alarme se o som detectado for muito alto, como o som de uma porta abrindo. A frequência do som foi ajustada para uma sensibilidade adequada.

Hardware Utilizado

- Placa Raspberry Pi Pico
- Microfone
- Display OLED (SSD1306)
- Buzzer
- LEDs (Vermelho, Verde, Azul)
- Botões (Botão A e Botão B)

Configurações de Pinos:

const uint I2C_SDA = 14; // Pino SDA para I2C

const uint I2C_SCL = 15; // Pino SCL para I2C

const uint MIC_PIN = 28; // Pino do Microfone

const uint BUZZER_PIN = 21; // Pino do Buzzer

const uint BUTTON_A_PIN = 5; // Pino do Botão A

const uint BUTTON_B_PIN = 6; // Pino do Botão B

const uint LED_RED_PIN = 10; // Pino do LED Vermelho

const uint LED_GREEN_PIN = 11; // Pino do LED Verde

const uint LED_BLUE_PIN = 12; // Pino do LED Azul

Configurações do ADC:

const float SOUND_OFFSET = 1.65; // Offset de som

const float HIGH_SOUND_THRESHOLD = 0.15; // Limite de som alto para detecção

const float ADC_REF = 3.3; // Referência do ADC

```
const int ADC_RES = 4095; // Resolução do ADC
```

Estado do Sistema:

```
bool system_active = false; // Estado do sistema (ativado/desativado)
bool alarm_active = false; // Estado do alarme (ativado/desativado)
```

Inicialização do PWM para o Buzzer:

```
void pwm_init_buzzer(uint pin) {
    gpio_set_function(pin, GPIO_FUNC_PWM);
    uint slice_num = pwm_gpio_to_slice_num(pin);
    pwm_config config = pwm_get_default_config();
    pwm_config_set_clkdiv(&config, 4.0f);
    pwm_init(slice_num, &config, true);
    pwm_set_gpio_level(pin, 0);
}
```

Configuração dos LEDs de Estado:

```
void configure_leds() {
    gpio_init(LED_RED_PIN);
    gpio_set_dir(LED_RED_PIN, GPIO_OUT);
    gpio_put(LED_RED_PIN, 0);

    gpio_init(LED_GREEN_PIN);
    gpio_set_dir(LED_GREEN_PIN, GPIO_OUT);
    gpio_put(LED_GREEN_PIN, 0);

    gpio_init(LED_BLUE_PIN);
    gpio_set_dir(LED_BLUE_PIN, GPIO_OUT);
    gpio_put(LED_BLUE_PIN, 1);
}
```

Atualização dos LEDs de Estado:

```
void update_led_status(bool active, bool sound_detected) {
```

```
    if (!active) {
        gpio_put(LED_RED_PIN, 0);
        gpio_put(LED_GREEN_PIN, 0);
        gpio_put(LED_BLUE_PIN, 1);
    } else if (sound_detected) {
        gpio_put(LED_RED_PIN, 1);
        gpio_put(LED_GREEN_PIN, 0);
        gpio_put(LED_BLUE_PIN, 0);
    } else {
        gpio_put(LED_RED_PIN, 0);
        gpio_put(LED_GREEN_PIN, 1);
        gpio_put(LED_BLUE_PIN, 0);
    }
}
```

Função para Tocar o Alarme:

```
void play_alarm(uint pin, uint frequency, uint duration_ms) {
    uint slice_num = pwm_gpio_to_slice_num(pin);
    uint32_t clock_freq = clock_get_hz(clk_sys);
    uint32_t top = clock_freq / frequency - 1;

    pwm_set_wrap(slice_num, top);
    pwm_set_gpio_level(pin, top / 2);

    sleep_ms(duration_ms);
    pwm_set_gpio_level(pin, 0);
    sleep_ms(50); // Pausa entre alarmes
}
```

Função para Reproduzir o Alarme de Segurança:

```
void play_security_alarm(uint pin) {
    alarm_active = true;
    while (alarm_active) {
```

```
            play_alarm(pin, alarm_frequency, alarm_duration);

            // Verifique se o botão B foi pressionado para interromper o alarme
            if (gpio_get(BUTTON_B_PIN) == 0) {
                alarm_active = false;
                break;
            }
        }
    }
```

Função Principal:

```
int main() {
    stdio_init_all();

    adc_init();
    adc_gpio_init(MIC_PIN);
    adc_select_input(2);

    // Inicialização do PWM para o buzzer
    pwm_init_buzzer(BUZZER_PIN);

    configure_leds();

    // Inicialização do display OLED
    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);

    ssd1306_init();
```

```
gpio_init(BUTTON_A_PIN);
gpio_set_dir(BUTTON_A_PIN, GPIO_IN);
gpio_pull_up(BUTTON_A_PIN);

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

struct render_area frame_area = {
    .start_column = 0,
    .end_column = ssd1306_width - 1,
    .start_page = 0,
    .end_page = ssd1306_n_pages - 1
};

calculate_render_area_buffer_length(&frame_area);

// Limpar display
uint8_t ssd[ssd1306_buffer_length];
memset(ssd, 0, ssd1306_buffer_length);
render_on_display(ssd, &frame_area);

// Mensagem inicial
ssd1306_draw_string(ssd, 0, 0, "Alerta de Segurança");
ssd1306_draw_string(ssd, 0, 16, "Aguardando ativação...");
render_on_display(ssd, &frame_area);

while (true) {
    // Verificar se o botão A foi pressionado
    if (gpio_get(BUTTON_A_PIN) == 0) {
```

```
    system_active = true;
    update_led_status(true, false);
    ssd1306_draw_string(ssd, 0, 16, "Sistema ativado    ");
    render_on_display(ssd, &frame_area);
    sleep_ms(200);
}

// Verificar se o botão B foi pressionado
if (gpio_get(BUTTON_B_PIN) == 0) {
    system_active = false;
    alarm_active = false;
    update_led_status(false, false);
    ssd1306_draw_string(ssd, 0, 16, "Sistema desativado ");
    render_on_display(ssd, &frame_area);
    sleep_ms(200);
}

if (system_active) {
    // Leitura do microfone
    uint16_t raw_adc = adc_read();
    float voltage = (raw_adc * ADC_REF) / ADC_RES;
    float sound_level = fabs(voltage - SOUND_OFFSET);

    if (sound_level > HIGH_SOUND_THRESHOLD && !alarm_active) {
        alarm_active = true;
        update_led_status(true, true);
        ssd1306_draw_string(ssd, 0, 32, "Alerta de Som Detetado!");
        render_on_display(ssd, &frame_area);

        // Tocar alarme
        play_security_alarm(BUZZER_PIN);
```

```
            } else {
                update_led_status(true, false);
            }
        } else {
            update_led_status(false, false);
        }

        sleep_ms(100);
    }

    return 0;
}

int main() {
    stdio_init_all();

    adc_init();
    adc_gpio_init(MIC_PIN);
    adc_select_input(2);

    // Inicialização do PWM para o buzzer
    pwm_init_buzzer(BUZZER_PIN);

    configure_leds();

    // Inicialização do display OLED
    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);
```

```
ssd1306_init();

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

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

struct render_area frame_area = {
    .start_column = 0,
    .end_column = ssd1306_width - 1,
    .start_page = 0,
    .end_page = ssd1306_n_pages - 1
};

calculate_render_area_buffer_length(&frame_area);

// Limpar display
uint8_t ssd[ssd1306_buffer_length];
memset(ssd, 0, ssd1306_buffer_length);
render_on_display(ssd, &frame_area);

// Mensagem inicial
ssd1306_draw_string(ssd, 0, 0, "Alerta de Segurança");
ssd1306_draw_string(ssd, 0, 16, "Aguardando ativação...");
render_on_display(ssd, &frame_area);

while (true) {
    // Verificar se o botão A foi pressionado
```

```
if (gpio_get(BUTTON_A_PIN) == 0) {
    system_active = true;
    update_led_status(true, false);
    ssd1306_draw_string(ssd, 0, 16, "Sistema ativado    ");
    render_on_display(ssd, &frame_area);
    sleep_ms(200);
}

// Verificar se o botão B foi pressionado
if (gpio_get(BUTTON_B_PIN) == 0) {
    system_active = false;
    alarm_active = false;
    update_led_status(false, false);
    ssd1306_draw_string(ssd, 0, 16, "Sistema desativado ");
    render_on_display(ssd, &frame_area);
    sleep_ms(200);
}

if (system_active) {
    // Leitura do microfone
    uint16_t raw_adc = adc_read();
    float voltage = (raw_adc * ADC_REF) / ADC_RES;
    float sound_level = fabs(voltage - SOUND_OFFSET);

    if (sound_level > HIGH_SOUND_THRESHOLD && !alarm_active) {
        alarm_active = true;
        update_led_status(true, true);
        ssd1306_draw_string(ssd, 0, 32, "Alerta de Som Detetado!");
        render_on_display(ssd, &frame_area);

        // Tocar alarme
```

```
            play_security_alarm(BUZZER_PIN);
        } else {
            update_led_status(true, false);
        }
    } else {
        update_led_status(false, false);
    }

    sleep_ms(100);
  }

  return 0;
}
```