Ficha de proposta de projeto

Nome do Aluno: Eder Renato da Silva Cardoso Casar
Polo: Bom Jesus da Lapa
Data: 21/04/2025

Projeto de Revisão da Capacitação em Sistemas Embarcados

Objetivo Geral

1. Matriz de LEDs 5x5 (WS2812B)

- Exibe um **contador numérico**.
- **Botão A**: Incrementa o valor.
- **Botão B**: Decrementa o valor.

2. Controle de LED RGB via Joystick

- O **movimento do joystick** ajusta a cor do LED RGB usando **PWM**.

3. Display SSD1306 (128x64 pixels)

- Exibe um **ponto de 8x8 pixels**.
- Posição inicial: **centralizada**.
- O **joystick** controla o movimento do ponto.

4. Buzzer

- Dois comandos para reproduzir melodias:
  - **buzzer_a**: Toca uma melodia no primeiro buzzer.
  - **buzzer_b**: Toca outra melodia no segundo buzzer.

.

Descrição Funcional

O sistema possui dois modos principais de operação:

1. **Modo Interativo Principal (Core 0)**:
   - Monitora continuamente o joystick e botões
   - Atualiza o display OLED com a posição do cursor
   - Controla os LEDs RGB conforme o movimento do joystick
   - Gerencia a exibição de números na matriz de LEDs
2. **Modo de Comunicação Serial (Core 1)**:
   - Fica em espera por comandos via serial
   - Executa sequências musicais no buzzer quando recebe comandos específicos

Componentes e Funcionalidades

1. Matriz de LEDs 5x5 (WS2812B)

Lógica de Operação:

- Utiliza PIO (Programmable I/O) para comunicação precisa com os LEDs
- Implementa um mapeamento especial para organizar os LEDs em formato de matriz
- Cada número (0-9) tem um padrão de cores pré-definido
- A função display_numerico() renderiza o número atual na matriz

Funções Principais:

- npInit(): Inicializa o controlador PIO para os LEDs
- npSetLED(): Define a cor de um LED individual
- npWrite(): Envia os dados para a matriz de LEDs
- getIndex(): Mapeia coordenadas (x,y) para o índice linear do LED

2. Display OLED (SSD1306)

Lógica de Operação:

- Comunicação via I2C
- Mostra um cursor que segue a posição do joystick
- O fundo alterna entre preto e branco quando o botão do joystick é pressionado

Funções Principais:

- display_init(): Configura o display
- ssd1306_draw_char(): Desenha o cursor na posição atual

3. Controle RGB com PWM

Lógica de Operação:

- Canal vermelho controlado pelo eixo X do joystick
- Canal azul controlado pelo eixo Y do joystick
- Canal verde alternado pelo botão do joystick
- Usa PWM para controle preciso da intensidade

Funções Principais:

- init_pwm(): Configura os canais PWM
- set_pulse(): Define o duty cycle do PWM

4. Joystick Analógico

Lógica de Operação:

- Usa o ADC para ler os valores dos eixos X e Y
- Normaliza os valores para um range de 0-100%
- Mapeia as posições para coordenadas do display

Funções Principais:

- read_adc(): Lê valores do conversor analógico-digital

- normalize_value(): Converte valores brutos do ADC para porcentagem
- map_value(): Mapeia valores para coordenadas de tela

5. Botões

Lógica de Operação:

- Implementa debounce para evitar leituras múltiplas
- Botão A: incrementa o contador (0-9)
- Botão B: decrementa o contador (9-0)
- Botão do Joystick: alterna o LED verde

Funções Principais:

- button_callback(): Trata as interrupções dos botões

6. Buzzer Musical

Lógica de Operação:

- Usa PWM para gerar tons musicais
- Dois temas musicais pré-programados
- Acionado por comandos via serial no core 1

Funções Principais:

- init_buzzer_pwm(): Configura o PWM para o buzzer
- set_buzzer_tone(): Define a frequência do tom
- sweet_child() e gran_vals(): Sequências musicais

7. Comunicação Serial (Core 1)

Lógica de Operação:

- Fica em loop esperando comandos
- "buzzer_a": Toca "Sweet Child O'Mine"
- "buzzer_b": Toca "Gran Vals"

Fluxo Principal

1. Inicialização de todos os periféricos
2. Lançamento do core 1 para lidar com comunicação serial
3. Loop principal no core 0:
    - Leitura do joystick
    - Atualização do display
    - Controle dos LEDs RGB
    - Exibição do número atual na matriz de LEDs
    - Tratamento de interrupções dos botões

Diagrama de Estados

1. **Estado Inicial**:
    - Display limpo

- Matriz mostra '0'
- LEDs RGB respondem ao joystick

2. **Botão A/B Pressionado**:
   - Incrementa/decrementa contador
   - Atualiza matriz de LEDs
3. **Botão do Joystick Pressionado**:
   - Alterna LED verde
   - Inverte cores do display
4. **Comando Serial Recebido**:
   - Toca música correspondente

Este sistema demonstra uma integração eficiente de múltiplos periféricos e técnicas de programação embarcada, incluindo tratamento de interrupções, PWM, ADC, I2C, PIO e multithreading com os dois cores do RP2040.

Uso dos Periféricos da BitDogLab

1. **Joystick (Potenciômetros X/Y)**
   - Controla o **LED RGB** (X = vermelho, Y = azul) e move um cursor no **display OLED**.
2. **Botões (A, B e Joystick)**
   - **A**: Incrementa um contador (0-9) e atualiza a **matriz de LEDs**.
   - **B**: Decrementa o contador.
   - **Botão do Joystick**: Liga/desliga o **LED verde**.
3. **Display OLED**
   - Mostra um cursor que segue o joystick e inverte cores quando o botão é pressionado.
4. **Matriz de LEDs 5x5**
   - Exibe números coloridos (0-9) conforme o contador dos botões.
5. **LED RGB**
   - Vermelho/Azul: Controlados pelo joystick via **PWM**.
   - Verde: Ligado/desligado pelo botão do joystick.
6. **Buzzer**
   - Toca músicas ("Sweet Child O'Mine" ou "Gran Vals") quando recebe comandos pela **serial**.
7. **Interrupções e Debounce**
   - Evitam leituras acidentais dos botões com um **delay de 300ms**.

Lógica Geral

- **Core 0**: Gerencia joystick, display, LEDs e botões.
- **Core 1**: Recebe comandos pela serial para tocar músicas no buzzer.

Links para acesso ao código e ao vídeo.

Repositório: https://github.com/EderRenato/Tarefa_Revisao

Video demonstrativo: https://youtube.com/shorts/C3Dsy58RiMg?feature=share