Relatório do Projeto

Yan Luca Viana de Araújo Fontenele

1. Apresentação do Projeto

O projeto desenvolvido teve como base explorar os recursos disponíveis na plataforma BitDogLab, desde de dispositivos de entrada, como botões e joystick, até os dipositivos de saída, como matriz de leds, tela Oled e buzzer.

Navegando pelo joystick e selecionando uma opção por um botão, foi disponibilizado um menu de opções, onde o usuário pode escolher entre ver um emoji na matriz de leds, verificar a temperatura lida por um sensor de temperatura, ver a distância até algum objeto calculada pelo sensor de ultrassom ou ouvir uma música através do buzzer.

2. Título do Projeto

Menu de Funcionalidades da BitDogLab

3. Objetivos do Projeto

O objetivo principal deste projeto é desenvolver um sistema interativo utilizando a BitDogLab, explorando sensores, atuadores e uma interface visual para interação do usuário. O projeto demonstra a capacidade da plataforma em lidar com entradas e saídas digitais e analógicas, proporcionando um sistema dinâmico e funcional.

4. Principais Requisitos

1. Criar um menu interativo para seleção de funcionalidades.
2. Utilizar o joystick para navegar entre as funcionalidades.
3. Utilizar um botão para selecionar uma opção.
4. Utilizar um botão para voltar ao menu principal.
5. Exibir informações no display OLED.
6. Utilizar matriz de LED para exibir padrões visuais.
7. Implementar leitura de temperatura com o sensor DHT11.
8. Implementar leitura de distância com o sensor ultrassônico HC-SR04.
9. Reproduzir sons/melodias com o buzzer.

5. Descrição do Funcionamento

O sistema inicia com um menu interativo exibido no display OLED. O joystick é utilizado para navegar entre as opções, e um botão é usado para selecionar a funcionalidade desejada. Dependendo da escolha do usuário, o sistema pode:

- Exibir expressões faciais na matriz de LED.
- Medir temperatura e umidade com o DHT11.
- Medir distância usando o sensor ultrassônico HC-SR04.
- Reproduzir músicas através do buzzer.

6. Justificativa

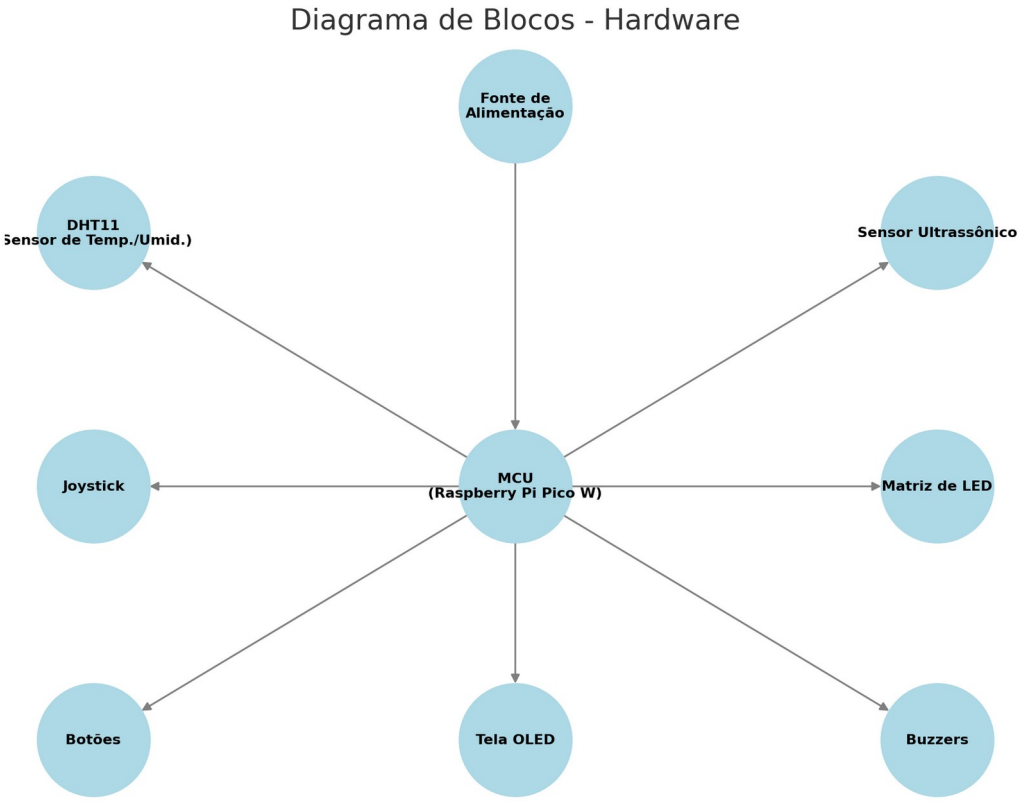
Este projeto permite explorar múltiplas funcionalidades da BitDogLab, proporcionando uma plataforma didática para aprendizado de sistemas embarcados, interação com sensores e controle de dispositivos.

7. Originalidade

O diferencial deste projeto é a poder explorar vários recursos dentro da mesma aplicação, mostrando o que pode ser feito com o hardware disponível e integrando sensores a placa de desenvolvimento, criando um sistema interativo que pode ser facilmente expandido para outras aplicações.

8. Hardware

8.1 Diagrama em Blocos



8.2 Função de Cada Bloco

- **Raspberry Pi Pico W**: Unidade central de processamento.
- **Display OLED**: Exibição das informações.
- **Matriz de LED**: Exibição de padrões visuais.
- **Joystick**: Controle de navegação no menu.
- **Botão**: Confirma seleção de opção.
- **Buzzer**: Emissão de sons.
- **DHT11**: Medida de temperatura e umidade.
- **HC-SR04**: Medida de distância.

8.3 Lista de Materiais

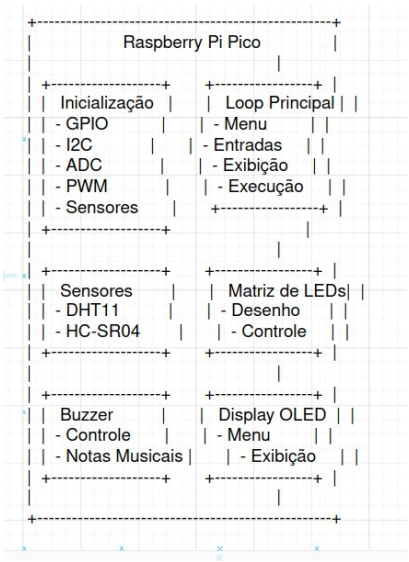
- BitDogLab (Raspberry Pi Pico W)
- Display OLED
- Matriz de LED
- Joystick
- Botão
- Buzzer
- Sensor DHT11
- Sensor HC-SR04
- Jumpers

8.4 Descrição da Pinagem

- Display OLED
    - SDA – GPIO14
    - SCL – GPIO15
- Matriz de LED
    - GPIO7
- Joystick
    - GPIO26
- Botão
    - BUTTON_A – GPIO5
    - BUTTON_B_GPIO6
- Buzzer
    - GPIO10
- Sensor DHT11
    - GPIO8
- Sensor HC-SR04
    - Trigger - 17
    - Echo - 18

9. Software

9.1 Blocos Funcionais



9.2 Definição das Variáveis

F unções de Inicialização e Configuração:

pwm_set_freq(uint slice_num, uint16_t freq): Configura a frequência do PWM.

buzzer_init(): Inicializa o buzzer.

hc_sr04_init(): Inicializa o sensor HC-SR04.

npInit(uint pin): Inicializa a matriz de LEDs.

ssd1306_init(): Inicializa o display OLED.

read_from_dht(dht_reading *result): Lê dados do sensor DHT11.

Funções de Controle de LEDs:

npSetLED(const uint index, const uint8_t r, const uint8_t g, const uint8_t b): Define a cor de um LED.

npClear(): Limpa a matriz de LEDs.

npWrite(): Envia os dados para a matriz de LEDs.

drawHeart(): Desenha um coração na matriz de LEDs.

draw1(), draw2(), draw3(), draw4(): Desenham números na matriz de LEDs.

drawRostoFeliz(), drawRostoBravo(), drawRostoTriste(): Desenham emojis na matriz de LEDs.

drawMusic(): Desenha um padrão musical na matriz de LEDs.

exibir_matriz_led(int opcao): Exibe um padrão na matriz de LEDs com base na opção.

exibir_matriz_led_Emojis(int opcao): Exibe emojis na matriz de LEDs.

exibir_matriz_led_Musica(int opcao): Exibe padrões musicais na matriz de LEDs.

limparMatriz(): Limpa a matriz de LEDs.

Funções de Controle de Áudio:

play_note(int frequency, int duration): Toca uma nota no buzzer.

buzzer_init(): Inicializa o buzzer.

Funções de Leitura de Sensores:

hc_sr04_read_distance(): Lê a distância do sensor HC-SR04.

read_from_dht(dht_reading *result): Lê temperatura e umidade do DHT11.

Funções de Exibição no Display OLED:

displayMenu(uint8_t *ssd, struct render_area *frame_area, int menuIndex): Exibe o menu principal no OLED.

displayMenuEmojis(uint8_t *ssd, struct render_area *frame_area, int menuIndex): Exibe o menu de emojis no OLED.

displayMenuTemperatura(uint8_t *ssd, struct render_area *frame_area, float temperatura): Exibe a temperatura no OLED.

displayMenuDistancia(uint8_t *ssd, struct render_area *frame_area, float distancia): Exibe a distância no OLED.

displayMenuMusica(uint8_t *ssd, struct render_area *frame_area, int menuIndex): Exibe o menu de música no OLED.

Funções Auxiliares:

executarOpcao(int opcao): Executa uma opção do menu.

calculate_render_area_buffer_length(struct render_area *frame_area): Calcula o tamanho do buffer para renderização no OLED.

render_on_display(uint8_t *ssd, struct render_area *frame_area): Renderiza o conteúdo no display OLED.

Principais Variáveis

Variáveis de Configuração:

DHT_PIN: Pino GPIO conectado ao DHT11.

TRIG_PIN, ECHO_PIN: Pinos GPIO conectados ao HC-SR04.

JOY_Y, JOY_X: Pinos ADC conectados ao joystick.

BUTTON_A, BUTTON_B: Pinos GPIO conectados aos botões.

LED_COUNT: Número de LEDs na matriz.

LED_PIN: Pino GPIO conectado à matriz de LEDs.

BUZZER_PIN: Pino GPIO conectado ao buzzer.

I2C_SDA, I2C_SCL: Pinos I2C conectados ao display OLED.

Variáveis de Estado:

menuIndex: Índice do menu atual.

opcao_atual: Opção selecionada no menu principal.

opcao_atual_emojis: Opção selecionada no menu de emojis.

opcao_atual_musica: Opção selecionada no menu de música.

currentMenu: Menu atualmente exibido.

Variáveis de Dados:

leds[LED_COUNT]: Buffer de pixels da matriz de LEDs.

melody[], melody2[]: Arrays com as notas das melodias.

durations[], durations2[]: Arrays com as durações das notas.

ssd[ssd1306_buffer_length]: Buffer de dados para o display OLED.

frame_area: Área de renderização no display OLED.

Variáveis de Sensores:

dht_reading reading: Estrutura para armazenar leituras do DHT11.

float distance: Distância medida pelo HC-SR04.

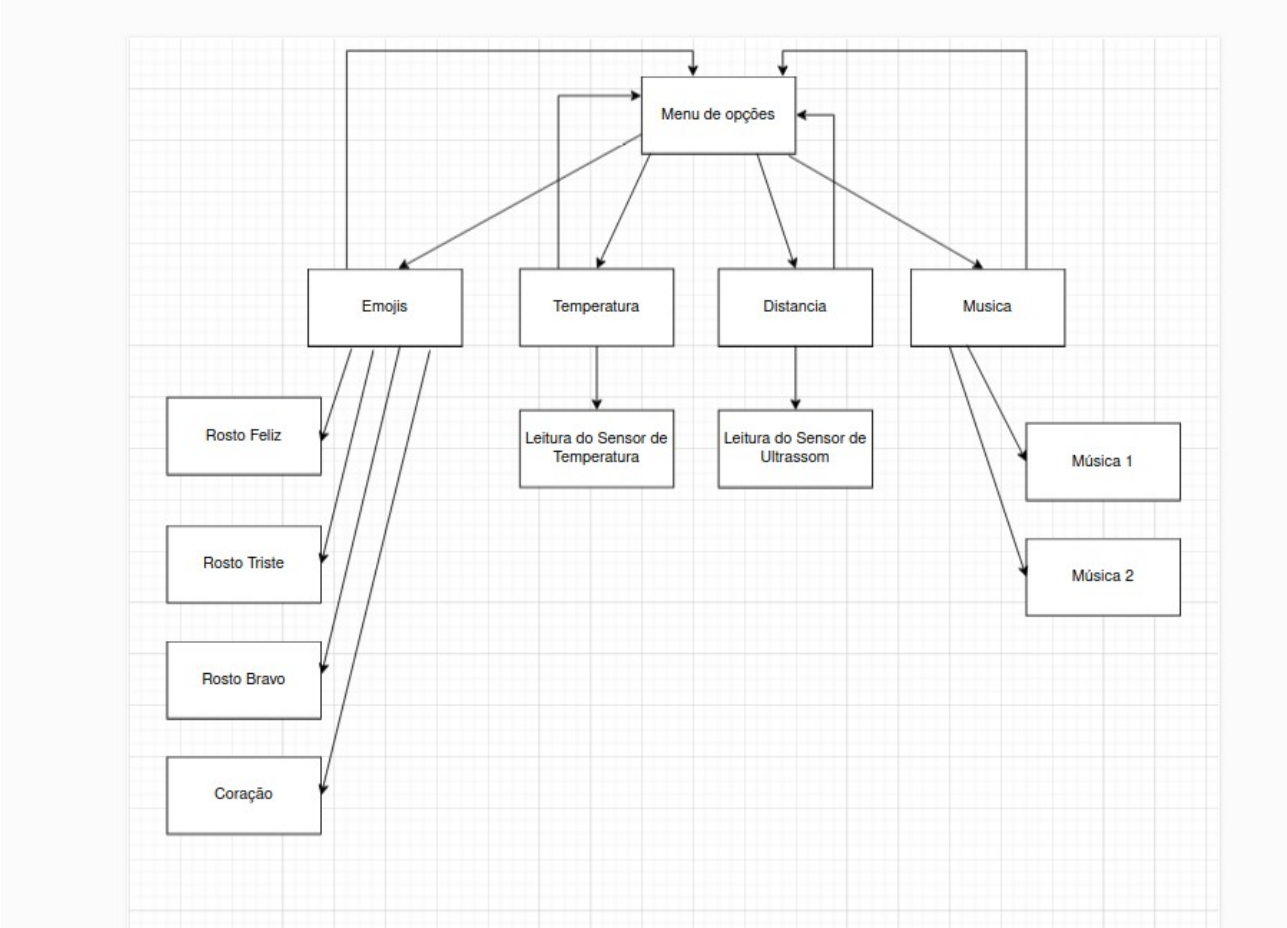
Variáveis de Hardware:

np_pio: Instância do PIO para controle da matriz de LEDs.

sm: Máquina de estado do PIO.

adc_y_raw, adc_x_raw: Leituras brutas do joystick.

9.3 Fluxograma



9.4 Inicialização

Deve-se umar o Pico Sdk para compilar o projeto.
Depois colocar a Raspberry Pi Pico W em modo de boot, mantendo pressionado o botão “boot” ao conectá-la ao computador;

Depois, com o comando Run, do Pico Sdk, enviar o programa para a placa.
Ela iniciará automaticamente o programa, o qual já vai aparecer na tela o menu de opções.

10. Execução do Projeto

10.1 Metodologia

O projeto foi desenvolvido através do Pico SDK no vscode, por partes, testando e entendendo o funcionamento de cada recurso para integrar no produto final.

10.2 Testes de Validação

Testes manuais, visualizando o resultado diretamente na placa.

10.3 Discussão dos Resultados

O desempenho foi satisfatório, porém não foi estabelecida corretamente a leitura dos sensores externos.

10.4 Vídeo de 3 Min

Link no github: https://github.com/yan-luca/projetoFinalEmabrcaTech