Relatório do projeto final do curso de formação básica em software embarcado - EmbarcaTech CEPEDI

Metrônomo audiovisual com Presets alternados por interrupção

Aluno:

Melk Silva Braga

Vitória da Conquista, 24 de Fevereiro de 2025

ÍNDICE

1. ESCOPO DO PROJETO

1.1 Apresentação do projeto

O projeto consiste em um metrônomo audiovisual que combina sinais sonoros e visuais para auxiliar músicos na marcação de tempo. Ele permite a configuração de diferentes Presets (predefinições) de BPM (batidas por minuto) e exibe, em loop, o número da batida do compasso de forma visual, além do sinal sonoro. O dispositivo também indica o contratempo através de um LED, e permite a mudança de predefinição através de um botão.

1.2 Título do projeto

Metrônomo audiovisual com Presets alternados por interrupção.

1.3 Objetivos do projeto

- Desenvolver um metrônomo que auxilie músicos na prática de ritmos variados.
- Permitir a configuração de múltiplos Presets de BPM e batidas.
- Oferecer feedback visual e sonoro para uma experiência de uso mais rica.
- Facilitar a mudança de Presets durante a execução musical.

1.4 Descrição do funcionamento

O metrônomo inicia solicitando ao usuário a configuração da quantidade de Presets desejados. Para ajustar esse número, o usuário movimenta um joystick para cima (incrementando) ou para baixo (decrementando), com um mínimo de 1 e um máximo de 20 Presets. Após definir a quantidade, o usuário confirma a seleção pressionando um botão.

Em seguida, para cada Preset, deve-se configurar:

- BPM (batidas por minuto): valor entre 1 e 300.
- Número de Beats (batidas por compasso): valor entre 2 e 9.

Durante a execução, o metrônomo gera bips audíveis por meio de buzzers e exibe os números correspondentes a cada Beat em uma matriz de LEDs. Além disso, um LED RGB desempenha duas funções:

- Contratempo: pisca em azul no meio tempo entre cada Beat.

- Indicação de troca de Preset: acende a luz vermelha quando um botão de troca de Presets é pressionado, permanecendo assim até que o loop atual termine e o novo Preset seja iniciado.

1.5 Justificativa

A execução deste projeto se justifica pela necessidade dos músicos de contar com um dispositivo versátil e personalizável para a prática rítmica. A integração de sinais visuais e sonoros, aliada à capacidade de alternar rapidamente entre diferentes Presets com um simples acionamento, proporciona uma experiência mais completa e adaptável às necessidades individuais.

Ao utilizar um Footswitch – um botão projetado para ser acionado com o pé –, por exemplo, o dispositivo se torna ainda mais prático, especialmente para bateristas. Com essa funcionalidade, um músico pode configurar um repertório inteiro dentro do metrônomo e alternar entre as músicas de forma fluida, sem a necessidade de interromper a performance. Basta pressionar o Footswitch com o pé para trocar de Preset, garantindo uma transição rápida e natural entre diferentes ritmos.

1.6 Originalidade

Existem inúmeros projetos de metrônomos conhecidos, sejam eles analógicos ou digitais. No entanto, o metrônomo desenvolvido neste projeto se destaca por sua abordagem única, combinando funcionalidades comuns a diferentes tipos de metrônomos e adicionando um sistema de Presets personalizáveis que podem ser alternados se maneira prática, algo não encontrado em modelos tradicionais.

Os metrônomos analógicos, como o clássico metrônomo de Maelzel, possuem um mecanismo simples baseado em um pêndulo oscilante. Embora sejam eficazes para marcar o tempo, eles carecem de flexibilidade, não permitindo ajustes dinâmicos ou a memorização de configurações personalizadas.

Já os metrônomos eletrônicos e digitais, como o Korg MA-2 (projetado como sistema embarcado) e o Metrônomo do Cifra Club (projetado como aplicativo para dispositivos móveis), oferecem maior precisão e customização de ritmos, mas não possuem a capacidade de alternar entre múltiplos Presets de forma rápida e intuitiva.

Diferentemente desses modelos, o metrônomo desenvolvido neste projeto:

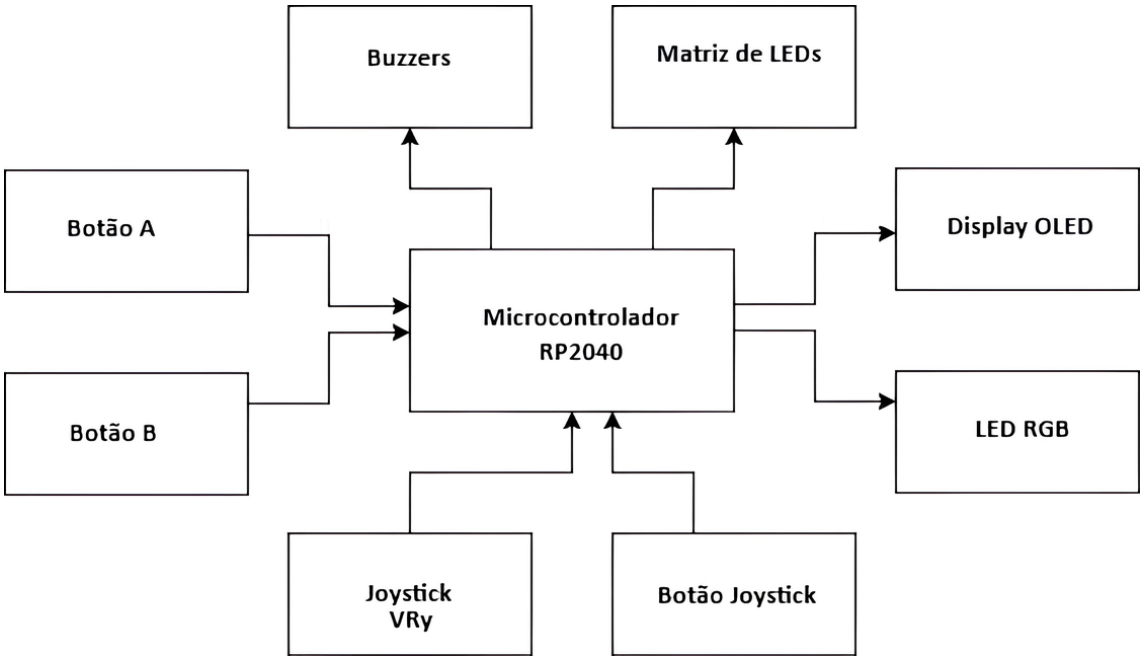
- Permite a configuração de múltiplos Presets, possibilitando ao usuário alternar rapidamente entre diferentes BPMs e assinaturas rítmicas sem precisar reconfigurar o dispositivo a cada alteração.
- Utiliza um sistema audiovisual, combinando sinais sonoros emitidos por buzzers, exibição numérica dos Beats em uma matriz de LEDs e um LED RGB que marca contratempo e indica transição entre Presets.
- Opera com controle simplificado via joystick e botões, tornando a interface intuitiva e facilitando a configuração mesmo durante apresentações ou ensaios.
- Permite a configuração de valores extremos de BPM, variando de 1 até 300.

Assim, este projeto representa uma inovação dentro do campo dos metrônomos digitais, combinando facilidade de uso, feedback visual e personalização rápida, algo pouco explorado em modelos comerciais.

2. ESPECIFICAÇÃO DO HARDWARE

2.1 Diagrama em bloco

Para o desenvolvimento do projeto, foi utilizada a BitDogLab, uma placa de desenvolvimento que conta com um Raspberry Pi Pico, um microcontrolador desenvolvido pela Raspberry Pi Foundation. A ferramenta dispõe de todos os periféricos necessários ao metrônomo.



2.2 Função de cada bloco

- **Microcontrolador:** Controla todas as operações do metrônomo.
- **Matriz de LEDs:** Exibe o número da batida atual do compasso, o Beat.
- **LED RGB:** Indica o contratempo e a mudança de Presets.
- **Buzzers:** Emitem sons para marcar o tempo.
- **Display OLED:** Exibe as informações de configuração.
- **Joystick VRy:** Permite alterar os valores de BPM e Beats ao configurar os Presets.
- **Botão Joystick:** Permite alternar entre os Presets configurados.
- **Botão A:** Pausa a execução do metrônomo.
- **Botão B:** Confirma os valores ao configurar os Presets.

2.3 Configuração de cada bloco

- **Microcontrolador:** Configurado para gerenciar entradas e saídas digitais e analógicas.
- **Matriz de LEDs:** Controlada via PIO para exibir padrões numéricos.
- **LED RGB:** Controlado por GPIO para piscar e mudar de cor.
- **Buzzers:** Controlados por PWM para emitir sons em diferentes frequências.
- **Display OLED:** Utiliza comunicação I2C para exibir informações.
- **Joystick VRy:** Configurado para enviar sinais analógicos ao microcontrolador, utilizando o conversor Analógico-Digital.
- **Botão Joystick:** Controlado por GPIO com resistor de pull-up interno para que quando pressionado, retornar nível lógico baixo.
- **Botão A:** Controlado por GPIO com resistor de pull-up interno para que quando pressionado, retornar nível lógico baixo
- **Botão B:** Controlado por GPIO com resistor de pull-up interno para que quando pressionado, retornar nível lógico baixo

2.4 Comandos e registros utilizados:

- **Microcontrolador:**
    - stdio_init_all(); // Inicializa entrada e saída padrão
    - adc_init(); // Inicializa o conversor Analógico-Digital

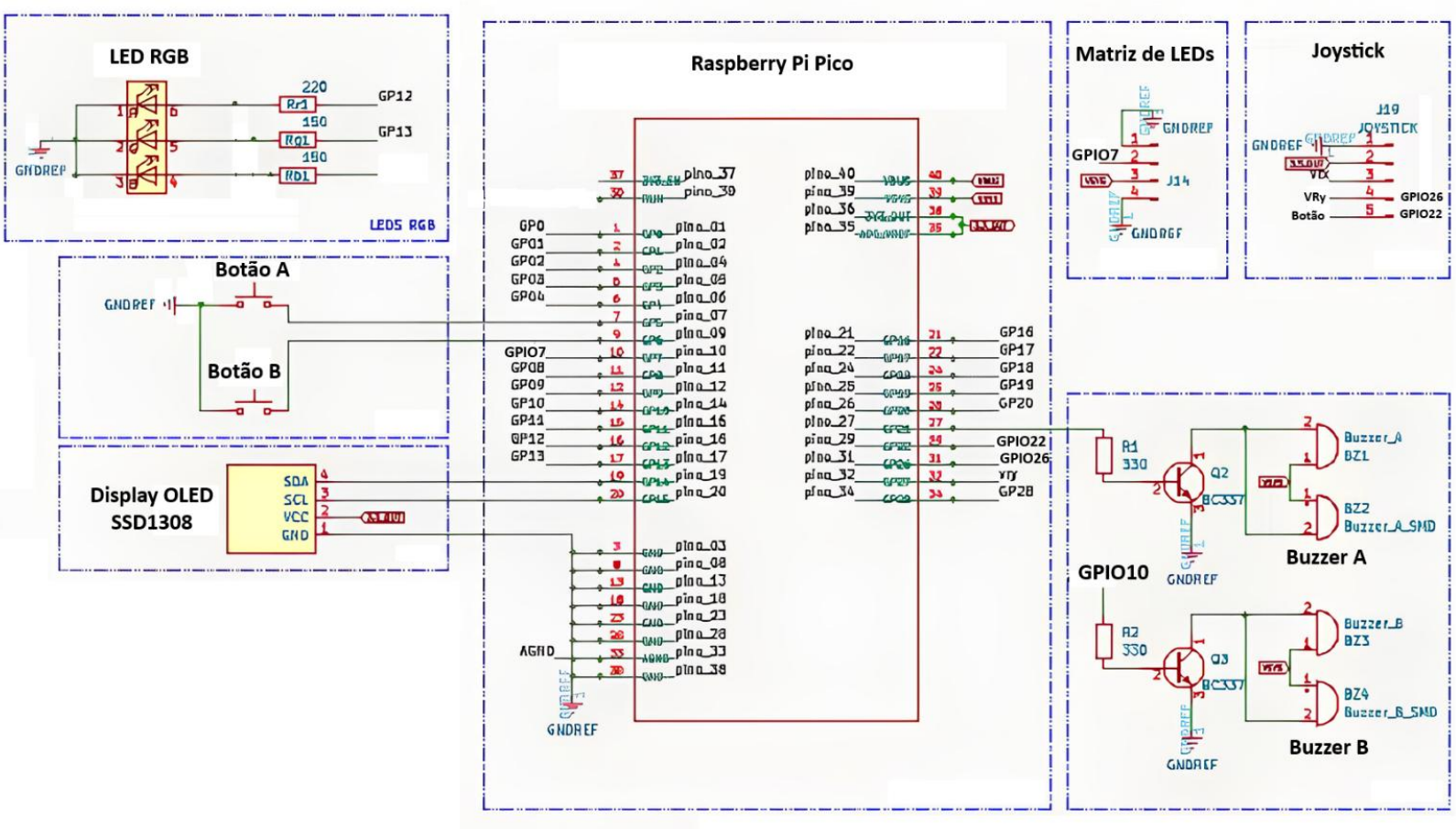
- **Matriz de LEDs:**
    - uint offset = pio_add_program(pio0, &ws2818b_program); // Carrega programa PIO
    - PIO np_pio = pio0; // Seleciona um dos blocos
    - uint sm = pio_claim_unused_sm(np_pio, true); // Reserva uma state machine livre
    - ws2818b_program_init(np_pio, sm, offset, pin, 800000.f); // Inicializa LEDs
- **LED RGB:**
    - gpio_init(gpio); // Inicializa o pino GPIO para a respectiva cor do LED
    - gpio_set_dir(gpio, GPIO_OUT); // Define o pino como saída
    - gpio_put(gpio, 1); gpio_put(gpio, 0); // Muda nível lógico no pino
- **Buzzers:**
    - pwm_config config = pwm_get_default_config(); // Configuração padrão do PWM
    - pwm_set_wrap(slice_num, 12500); // Define período do PWM (ajusta frequência do som)
    - pwm_set_chan_level(slice_num, PWM_CHAN_A, duty_cycle); // Ajusta volume (duty cycle)
    - pwm_set_enabled(slice_num, true); // Habilita PWM no buzzer
- **Display OLED:**
    - i2c_init(i2c0, 400 * 1000); // Inicializa I2C na velocidade de 400kHz
    - gpio_set_function(14, GPIO_FUNC_I2C); // Define SDA
    - gpio_set_function(15, GPIO_FUNC_I2C); // Define SCL
    - ssd1306_init(); // Inicializa o display
- **Joystick VRy:**
    - adc_gpio_init(26); // Configura GPIO 26 para leitura analógica
    - adc_select_input(0); // Seleciona canal ADC correspondente
    - uint16_t leitura = adc_read(); // Lê valor analógico (0-4095)
- **Botões A, B e do Joystick:**
    - gpio_init(Botão); // Inicializa o botão
    - gpio_set_dir(Botão, GPIO_IN); // Define o pino como entrada
    - gpio_pull_up(Botão); // Ativa pull-up interno

2.5 Descrição da pinagem usada

Pinagem utilizada do microcontrolador Raspberry Pi Pico.

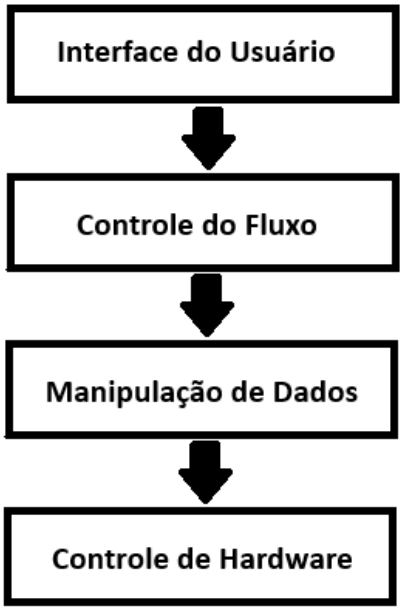
- **LED RGB (Para cor vermelha):** GPIO 13 - Pino 17
- **LED RGB (Para cor azul):** GPIO 12 - Pino 16
- **Matriz de LEDs:** GPIO 7 - Pino 10
- **Botão A:** GPIO 5 - Pino 7
- **Botão B:** GPIO 6 - Pino 9
- **Botão do Joystick:** GPIO 22 - Pino 29
- **Joystick VRy:** GPIO 26 - Pino 31
- **Display OLED (I2C_SDA):** GPIO 14 - Pino 19
- **Display OLED (I2C_SCL):** GPIO 15 - Pino 20
- **BUZZER A (para primeiro Beat do compasso):** GPIO 21 - Pino 27
- **BUZZER B (para demais Beats do compasso):** GPIO 10 - Pino 14

2.6 Circuito completo do hardware



3. ESPECIFICAÇÃO DO FIRMWARE

3.1 Blocos funcionais



3.2 Descrição das funcionalidades

Interface do Usuário

- Exibe informações de configuração no display OLED.
- Permite ao usuário navegar pelas opções usando o joystick e botões.

Controle do Fluxo

- Inicia o metrônomo.
- Pausa e retoma a execução.
- Alterna entre Presets de forma dinâmica.

Manipulação de Dados

- Armazena BPM e beats para cada Preset.
- Atualiza as informações conforme as interações do usuário.

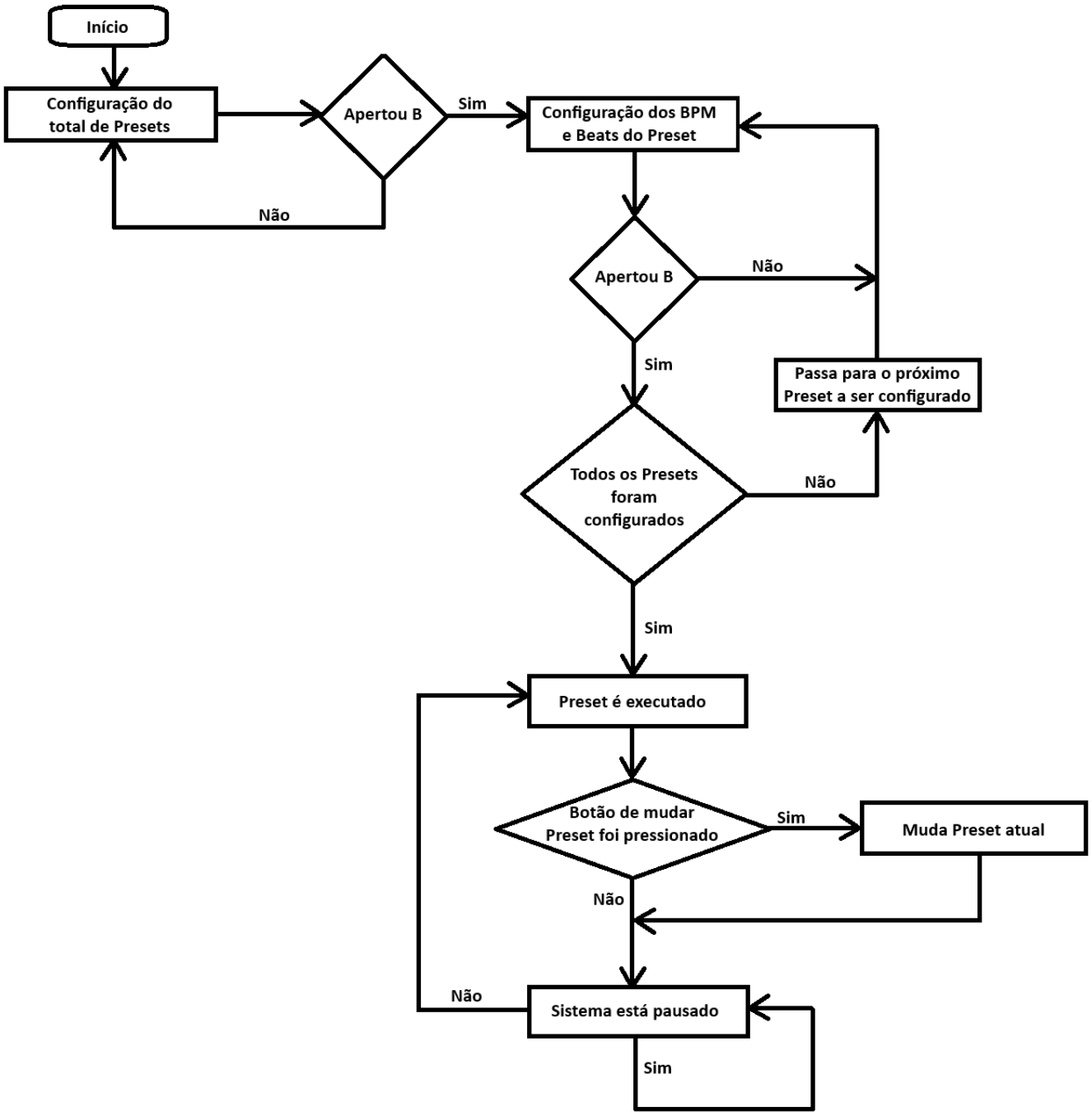
Controle de Hardware

- Gera sinais sonoros com os buzzers.
- Controla a matriz de LEDs para exibição visual do Beat.
- Gerencia o estado dos LEDs RGB.

3.3 Definição das variáveis

- **bpm_values[20]:** Armazena os valores de BPM de cada Preset.
- **beats_values[20]:** Armazena o número de Beats de cada Preset.
- **bpm_count:** Número total de Presets configurados.
- **current_bpm_index:** Índice do Preset atualmente em execução.
- **paused:** Indica se o metrônomo está pausado.
- **metronome_running:** Indica se o metrônomo está em execução.

3.4 Fluxograma



3.5 Inicialização

A função **setup()** realiza toda a inicilização do software. Nela, são inicializados o sistema de entrada e saída padrão (comando **stdio_init_all**), o display OLED via I2C, o ADC para leitura do joystick, os GPIOs dos botões com callback para a função de interrupção, o LED RGB e a matriz de LEDs. Os buzzers são inicializados na chamada da função **beep_async()**.

3.6 Configurações dos registros

- **Display OLED:** i2c_init, gpio_set_function.
- **Matriz de LEDs:** pio_add_program, pio_claim_unused_sm, ws2818b_program_init.
- **Buzzers:** pwm_init, pwm_set_gpio_level.
- **Botões e Joystick:** gpio_init, gpio_set_dir, gpio_pull_up.

3.7 Estrutura e formato dos dados

Os dados principais são armazenados em arrays (bpm_values, beats_values) e variáveis de controle (bpm_count, current_bpm_index, paused).

3.8 Protocolo de comunicação

O protocolo I2C é utilizado para comunicação com o display OLED. Nesse protocolo, duas conexões são utilizadas: SDA (dados) e SCL (clock). O microcontrolador envia os dados a serem exibidos para o endereço do display (0x3C).

3.9 Formato do pacote de dados

O pacote de dados enviado ao display OLED via I2C é dividido em dois frames: o frame de endereço, que contém o endereço do dispositivo (0x3C), e o frame de dados, seguido por um ou mais bytes representando os comandos ou os pixels a serem exibidos.

4. EXECUÇÃO DO PROJETO

4.1 Metodologia

Foram realizadas pesquisas sobre metrônomos eletrônicos já existentes e comercializados, para fins de comparação e análise da implementação das funcionalidades comuns aos metrônomos. Além disso, estudos sobre o objeto metrônomo foram feitos para compreender todos os aspectos do dispositivo.

O hardware escolhido para a execução do projeto foi a placa de desenvolvimento BitDogLab, que integra o microcontrolador RP2040 e dispõe de todos os elementos necessários para a implementação.

As funcionalidades do software foram definidas com base nos objetivos do projeto e programadas através de funções:

- Capacidade de salvar e alternar entre 20 Presets personalizados.
- Exibição do BPM e quantidade de batidas por compasso no display OLED.
- Sinalização visual do ritmo na matriz de LEDs WS2812B.
- Indicação sonora do tempo forte (primeiro Beat do compasso) e fraco (demais Beats do compasso) com frequências diferentes.
- Sinalização visual do contratempo no LED RGB.
- Controle por botões e joystick.

O software foi desenvolvido na linguagem C e a IDE utilizada foi o Visual Studio Code. Para isso foram instaladas as extensões C/C++, CMake, CMake Tools e Raspberry Pi Pico, além da instalação e configuração do SDK e bibliotecas necessárias para controle dos periféricos.

Foram desenvolvidos os códigos para inicialização de hardware e controle dos periféricos, para implementação da lógica de configuração, seleção e execução dos Presets e para integração do controle dos periféricos utilizados. Para cada função implementada, foi utilizado o ambiente de depuração do VS Code para identificar e remover defeitos encontrados ou comportamentos indesejados, adicionando breakpoints e monitorando expressões e variáveis.

4.2 Testes de validação

4.2.1 Testes de BPM e sincronização

- Verificação da precisão do tempo entre batimentos utilizando um cronômetro externo.
- Verificação da quantidade de batimentos por minuto utilizando um cronômetro externo.
- Comparativo com aplicativos de metrônomo digitais em dispositivos móveis para equiparar a precisão.

4.2.2 Testes de display OLED

- Conferência da exibição correta dos valores de BPM e Beats por compasso em cada Preset, onde foram feitos ajustes na disposição das informações para melhor legibilidade.

4.2.3 Testes da matriz de LEDs

- Exibição dos números de batidas de forma clara e sincronizada.

4.2.4 Testes do buzzer

- Validação da duração e volume do bip para melhor percepção auditiva.

4.2.5 Testes de interação (Botões e Joystick)

- Resposta correta aos comandos para alterar valores, confirmar configuração, alternar Presets e pausar o metrônomo. Durante os testes, verificou-se especialmente a eficácia da função de debounce na prevenção de acionamentos indesejados.

4.2.6 Testes com músicos

- O dispositivo foi testado por três músicos, que validaram seu funcionamento e destacaram sua precisão e usabilidade.

4.3 Discussão dos resultados

O metrônomo audiovisual desenvolvido demonstrou um funcionamento confiável e preciso, atingindo os objetivos estabelecidos. A sincronização dos LEDs, do som e das informações no display garantiram uma boa usabilidade do dispositivo. A inclusão de Presets e a possibilidade de personalização do BPM e Beats por compasso, tornaram o sistema flexível para diferentes necessidades musicais.

A exibição dos Beats na matriz de LEDs e a alternância do LED RGB para indicar o contratempo foram bem visualizados e cumpriram a sua função de auxiliar no acompanhamento rítmico.

Como pontos de melhoria, a implementação de um sistema de memória persistente para armazenar Presets entre desligamentos e a utilização de um Footswitch no lugar do botão do joystick, poderiam aprimorar o sistema.

Conclui-se que o projeto atendeu aos requisitos e pode ser utilizado como um metrônomo funcional e personalizável, sendo uma ferramenta útil para músicos e estudantes de música.

5. LINKS DO PROJETO

Link para repositório com código-fonte do projeto:
https://github.com/MelkBraga/Metronomo_de_Presets.git

Link para vídeo demonstrativo do projeto:
https://youtu.be/xnk6HE-9JkA

REFERÊNCIAS

[1] "O Metrônomo de Malzel." Disponível em: https://www.metropolitana.pt/musicalia/o-metronomo-de-malzel/.

[2] "Metrônomo Digital Korg MA-2 BLBK." Disponível em: https://www.korg.com.br/produto/10270509-metronomo-digital-korg-ma-2-blbk.

[3] "Metrônomo – Cifra Club." Disponível em: https://www.cifraclub.com.br/metronomo/.

[4] B. F. D. Barros, *A utilização do metrônomo em aulas de ensino coletivo de violão para iniciantes*, Universidade Federal de Alagoas, 2022. Disponível em: https://www.repositorio.ufal.br/bitstream/123456789/9621/1/A%20utiliza%C3%A7%C3%A3o%20do%20metr%C3%B4nomo%20em%20aulas%20de%20ensino%20coletivo%20de%20viol%C3%A3o%20para%20iniciantes.pdf.

[5] "Metrônomo – Wikipédia." Disponível em: https://pt.wikipedia.org/wiki/Metr%C3%B4nomo.

[6] "Raspberry Pi Pico – Raspberry Pi Foundation." Disponível em: https://www.raspberrypi.com/products/raspberry-pi-pico/.

[7] "BitDogLab – GitHub." Disponível em: https://github.com/BitDogLab/BitDogLab.

[8] R. Souza, "BitDogLab: Uma jornada educativa com eletrônica embarcada e IA," *Embarcados*, 2023. Disponível em: https://embarcados.com.br/bitdoglab-uma-jornada-educativa-com-eletronica-embarcados-e-ia/.