UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA
DCET - DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MELK SILVA BRAGA

DESENVOLVIMENTO DE UM METRÔNOMO AUDIOVISUAL COM PRESETS PROGRAMÁVEIS E ALTERNÂNCIA POR INTERRUPÇÃO UTILIZANDO O RASPBERRY PI PICO

VITÓRIA DA CONQUISTA – BA
2025

MELK SILVA BRAGA

DESENVOLVIMENTO DE UM METRÔNOMO AUDIOVISUAL COM PRESETS PROGRAMÁVEIS E ALTERNÂNCIA POR INTERRUPÇÃO UTILIZANDO O RASPBERRY PI PICO

Projeto entregue à disciplina Trabalho Supervisionado II como requisito para obtenção do Grau de Bacharel em Ciência da Computação pela Universidade Estadual do Sudoeste da Bahia.

Orientador: Prof. Me. Marco Antonio Dantas Ramos

VITÓRIA DA CONQUISTA – BA
2025

DEDICATÓRIA

Ao meu avô, cuja memória me inspira e me trouxe até aqui. À minha mãe, que sempre me manteve no caminho com muito amor. Ao meu irmão, que está sempre do lado. Ao meu padrasto, que sempre nos ajudou. E a todos os familiares e amigos que tornam esta jornada mais leve e valiosa.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ADC	Analog-to-Digital Converter (Conversor Analógico-Digital)
ARM	Advanced RISC Machines (Máquinas RISC Avançadas)
BPM	Beats Per Minute (Batidas Por Minuto)
CPU	Central Processing Unit (Unidade Central de Processamento)
EEPROM	Electrically Erasable Programmable Read-Only Memory (Memória Somente Leitura Programável e Apagável Eletricamente)
GPIO	General-Purpose Input/Output (Entradas e Saídas de Uso Geral)
I²C	Inter-Integrated Circuit (Circuito Interintegrado)
IDE	Integrated Development Environment (Ambiente de Desenvolvimento Integrado)
LCD	Liquid Crystal Display (Display de Cristal Líquido)
LED	Light Emitting Diode (Diodo Emissor de Luz)
MIDI	Musical Instrument Digital Interface (Interface Digital para Instrumentos Musicais)
OLED	Organic Light Emitting Diode (Diodo Orgânico Emissor de Luz)
PIO	Programmable Input/Output (Entradas e Saídas Programáveis)
PWM	Pulse-Width Modulation (Modulação por Largura de Pulso)
RAM	Random Access Memory (Memória de Acesso Aleatório)
RGB	Red, Green, Blue (Vermelho, Verde, Azul)
RX	Receive (Receber)
SCL	Serial Clock Line (Linha de Clock Serial)
SDA	Serial Data Line (Linha de Dados Seriais)
SDK	Software Development Kit (Kit de Desenvolvimento de Software)
SPI	Serial Peripheral Interface (Interface Serial Periférica)
TX	Transmit (Transmitir)
UART	Universal Asynchronous Receiver/Transmitter (Receptor/Transmissor Assíncrono Universal)

RESUMO

A necessidade de ferramentas de apoio ao estudo musical que ofereçam precisão rítmica, flexibilidade e boa usabilidade tem impulsionado o desenvolvimento de dispositivos digitais mais acessíveis e personalizados. Neste contexto, este trabalho apresenta o desenvolvimento de um metrônomo digital audiovisual baseado na placa Raspberry Pi Pico W, construída com o microcontrolador RP2040 e integrada à BitDogLab, uma plataforma educacional para desenvolvimento de sistemas embarcados. O sistema utiliza sinais sonoros e visuais sincronizados, empregando LEDs WS2812, buzzers ativos, um LED RGB e um display OLED para *feedback* imediato ao usuário. Além disso, incorpora a alternância dinâmica entre *presets* por meio de interrupções, permitindo ao músico trocar configurações em tempo real sem interromper o fluxo de execução. A interface utiliza um joystick analógico e botões físicos, integrando múltiplos periféricos via PWM, PIO e I²C, todos programados em linguagem C. A metodologia envolveu levantamento de requisitos, modelagem modular do firmware, implementação progressiva dos módulos, testes funcionais e documentação no GitHub. Os resultados demonstraram a eficácia do dispositivo na execução estável de pulsações rítmicas, na alternância fluida entre *presets* e na clareza visual do padrão dos tempos e contratempos por meio dos LEDs, tornando-o adequado tanto para prática individual quanto para uso didático. Como contribuições, destacam-se a aplicação de técnicas de programação embarcada, o emprego de hardware de baixo custo e a criação de um protótipo funcional que evidencia o potencial do Raspberry Pi Pico W para dispositivos musicais interativos. Conclui-se que o sistema atendeu aos objetivos propostos e constitui uma base sólida para expansões futuras, como persistência de *presets* e versões com comunicação sem fio.

Palavras-chave: metrônomo digital, Raspberry Pi Pico W, metrônomo audiovisual, sistemas embarcados, BitDogLab.

ABSTRACT

The need for tools that support musical study by offering rhythmic precision, flexibility, and good usability has driven the development of more accessible and customizable digital devices. In this context, this work presents the development of an audiovisual digital metronome based on the Raspberry Pi Pico W board, built with the RP2040 microcontroller and integrated into the BitDogLab platform, an educational environment for embedded systems development. The system uses synchronized audio and visual signals, employing WS2812 LEDs, an active buzzer, an RGB LED, and an OLED display to provide immediate feedback to the user. In addition, it incorporates dynamic preset switching through hardware interrupts, allowing musicians to change configurations in real time without interrupting execution. The interface uses an analog joystick and physical buttons, integrating multiple peripherals through PWM, PIO, and I²C, all programmed in the C language. The methodology involved requirement analysis, modular firmware design, progressive implementation of modules, functional testing, and documentation on GitHub. The results demonstrated the effectiveness of the device in the stable execution of rhythmic pulses, the smooth alternation between presets, and the clear visual indication of beats and offbeats through the LEDs, making it suitable for both individual practice and didactic use. The main contributions include the application of embedded programming techniques, the use of low-cost hardware, and the creation of a functional prototype that highlights the potential of the Raspberry Pi Pico W for interactive musical devices. It is concluded that the system met the proposed objectives and provides a solid foundation for future improvements, such as preset persistence and wireless communication.

Keywords: digital metronome, Raspberry Pi Pico W, audiovisual metronome, embedded systems, BitDogLab.

SUMÁRIO

1. Introdução

O ritmo é frequentemente considerado o “esqueleto” da música — é ele que organiza o tempo, estrutura as composições, e permite que diferentes músicos toquem de forma sincronizada (SABRA, 2025). Um dispositivo fundamental para treinar e manter essa precisão rítmica é o metrônomo: seja na sua forma tradicional, mecânica, ou em versões digitais, o metrônomo oferece uma pulsação constante que auxilia músicos a internalizar o tempo e melhorar a consistência de execução.

Sendo originalmente um pêndulo de dupla haste movido por corda, projetado para manter batidas regulares ajustáveis em batidas por minuto, o metrônomo é, em suma, um instrumento para estabelecer o tempo musical (Encyclopedia Britannica, 2025a).

Apesar da existência de diversos metrônomos digitais e até aplicações em software, muitos deles utilizam apenas *feedback* auditivo (clique sonoro), o que pode ser uma limitação, principalmente em ambientes com alta interferência sonora. Sendo assim, uma alternativa para mitigar tal problema é a utilização de interfaces visuais, que permitam ao usuário visualizar a pulsação rítmica. Com o avanço da eletrônica embarcada e a acessibilidade a plataformas de desenvolvimento, tornou-se viável explorar novas formas de apresentar o pulso rítmico — por exemplo, por meio de sinais visuais sincronizados — expandindo os modos de uso do metrônomo.

Sistemas embarcados são sistemas computacionais dedicados, projetados para desempenhar funções específicas e integrados a dispositivos maiores. Visam monitorar ou controlar um determinado processo ou função. Em geral, são sistemas simples, de baixo custo, compostos por um número limitado de componentes (Souza, F. 2022). Tais sistemas têm sido amplamente empregados no desenvolvimento de dispositivos musicais digitais.

Neste contexto, surgiu a motivação para o desenvolvimento deste trabalho, que busca resolver o problema técnico da limitação de metrônomos baseados exclusivamente em clique sonoro, especialmente quanto à percepção da pulsação rítmica em ambientes com muito ruído. Assim, propõe-se o projeto e a implementação de um metrônomo digital audiovisual baseado na placa Raspberry Pi Pico W, a qual utiliza o microcontrolador RP2040. O sistema combina saída sonora com sinalização luminosa (LEDs), fornecendo múltiplos estímulos sensoriais ao usuário, além de funcionalidades como *presets* (conjuntos de configurações armazenadas)

alternáveis por interrupção, em tempo de execução, permitindo ao músico trocar configurações sem interromper o fluxo de execução.

A interface do sistema utiliza um conjunto de dispositivos integrados à BitDogLab, uma plataforma educacional de hardware aberto voltada ao ensino de eletrônica, programação e sistemas embarcados. Desenvolvida no âmbito do projeto School Project 4.0 da Universidade Estadual de Campinas (UNICAMP), a BitDogLab é baseada na Raspberry Pi Pico e integra componentes como botões, joystick analógico, LEDs e display OLED, permitindo a exploração e o desenvolvimento de aplicações embarcadas de forma organizada e segura (BitDogLab, 2025).

1.1. Objetivo Geral

Estudar técnicas e ferramentas de desenvolvimento em hardware e software (hardware–software co-design) para projetar e implementar um protótipo funcional de metrônomo digital audiovisual, capaz de fornecer *feedback* sonoro e visual sincronizado, como alternativa à limitação de metrônomos baseados exclusivamente em sinal sonoro, sendo o sistema versátil, acessível e adequado tanto para músicos iniciantes quanto para usuários experientes.

1.2. Objetivos Específicos

Especificamente, o sistema visa:

- Implementar a geração de pulsos rítmicos precisos controlados por software.
- Integrar *feedback* audiovisual por meio de sinais sonoros e luminosos sincronizados.
- Permitir a alternância de *presets* em tempo real por meio do uso de interrupções.
- Desenvolver uma interface simples, responsiva e intuitiva para interação com o usuário.

1.3. Justificativa e Relevância

O desenvolvimento deste sistema justifica-se pela relevância de ferramentas de uso transparente que apoiam o estudo musical e, especialmente, o treinamento rítmico, promovendo o progresso técnico de estudantes e instrumentistas.

A relevância acadêmica e técnica deste trabalho se manifesta em dois aspectos. Primeiro, como um exercício de design e implementação de sistema embarcado com múltiplos

periféricos e controle em tempo real. E, segundo, como uma contribuição prática para a comunidade musical, provando que é possível criar dispositivos úteis, com custo reduzido, e de fácil prototipação, que gerem valor além dos metrônomos tradicionais ou simples aplicativos.

O presente trabalho está organizado da seguinte forma: no capítulo 2, é apresentada uma fundamentação teórica. O capítulo 3 traz o estado da arte com uma revisão de produtos comerciais e implementações de desenvolvedores independentes. O capítulo 4 descreve a especificação e implementação do sistema desenvolvido. O capítulo 5 detalha os testes e os resultados obtidos. O capítulo 6 discute possíveis aprimoramentos e trabalhos futuros. O capítulo 7 apresenta a documentação fotográfica do dispositivo e, por fim, são apresentadas as conclusões gerais sobre o projeto.

2. Fundamentação Teórica

Este capítulo apresenta os fundamentos teóricos que embasam o desenvolvimento do metrônomo audiovisual proposto neste trabalho. São abordados conceitos relacionados à história do metrônomo, aos diferentes tipos existentes e aos aspectos relevantes da percepção rítmica.

2.1. Ritmo e Percepção Temporal na Música

Do ponto de vista teórico, o ritmo é definido como a organização dos eventos sonoros no tempo, estruturando pulsação, métrica e padrões regulares que orientam a execução musical (Encyclopedia Britannica, 2025b). Pode ser entendido como a percepção temporal estruturada, baseada na expectativa e repetição de padrões, sendo uma habilidade cognitiva essencial para qualquer músico. A capacidade de manter o tempo estável, portanto, é uma competência desenvolvida por meio de prática deliberada e de ferramentas que auxiliam na precisão rítmica.

Pesquisas em cognição musical mostram que a percepção rítmica humana depende de um sistema de *entrainment* — um tipo de sincronização natural entre estímulos externos e o movimento corporal (Repp, 2005). Isso significa que dispositivos que fornecem pulsos regulares, como o metrônomo, desempenham papel essencial para treinar o senso interno de tempo, aperfeiçoar a execução instrumental e reduzir flutuações de andamento (variações involuntárias e não intencionais na velocidade do tempo musical durante a execução).

A prática com metrônomo é amplamente recomendada por professores de música, compositores e pedagogos, pois melhora a consistência temporal, fortalece a disciplina motora e auxilia no desenvolvimento técnico.

> A cognição depende de experiências baseadas em possuir um corpo com capacidades sensório-motoras; essas capacidades estão inseridas em um contexto biológico, psicológico e cultural mais amplo. (Iyer, 2002, tradução nossa)

Portanto, ritmo e tempo não são apenas medidas objetivas, mas experiências corporificadas.

2.2. Contexto Histórico do Metrônomo

O desenvolvimento do metrônomo está diretamente ligado à busca histórica por mecanismos capazes de produzir pulsos temporais regulares, fundamentais para a organização e a padronização do tempo musical.

> Em 1696, Étienne Loulié descreveu pela primeira vez um chronomètre utilizado para definir andamentos musicais. O chronomètre consiste em um pêndulo formado por um peso de chumbo móvel suspenso por um cordão. Ao elevar ou abaixar o peso, o período de oscilação diminuía ou aumentava e, como o período correspondente a um determinado comprimento efetivo do pêndulo é mais ou menos uniforme, tornou-se possível, pela primeira vez, padronizar taxas e andamentos musicais. Embora confiável, versátil (Loulié definiu 72 andamentos distintos) e razoavelmente preciso, o chronomètre mostrou-se pouco prático, em parte porque o período de oscilação precisava ser determinado visualmente, em parte porque o movimento não podia ser mantido sem interrupções e, sobretudo, porque um pêndulo com quase um metro de comprimento era necessário para produzir um andamento de uma batida por segundo (m.m. = 60). Durante o século XVIII, foram realizados esforços consideráveis para aprimorar a invenção de Loulié. (Manasse, 2022, tradução nossa)

No entanto, a consolidação do metrônomo ocorreu apenas em 1815, quando Johann Nepomuk Maelzel patenteou um instrumento baseado em um pêndulo com uma longa barra metálica oscilante. A velocidade do pulso era ajustada por meio do deslocamento de um peso ao longo da barra, o que alterava o período de oscilação do pêndulo e, consequentemente, a frequência das batidas, permitindo definir diferentes andamentos musicais (MetronomeBot apud Souza, W. 2018).

Embora Maelzel seja frequentemente considerado o criador do metrônomo comercial, devido à sua patente, a história revela nuances importantes:

> Apesar de todos esses esforços, apenas por volta de 1812 foi alcançado um avanço significativo e fundamental, cujo crédito deve ser atribuído ao mecânico holandês Dietrich Winkel. Foi ele quem percebeu que um pêndulo composto, em vez de um pêndulo simples com haste e peso único, permitiria obter andamentos lentos com comprimentos

> reduzidos. Um pêndulo composto consiste em uma haste rígida vertical pivotada, com dois pesos fixados em lados opostos do ponto de apoio. O peso superior pode ser movido para cima ou para baixo ao longo da haste, alterando o centro de gravidade e, assim, diminuindo ou aumentando o andamento. [...] Foi durante uma visita a Amsterdã, por volta dessa época, que Maelzel encontrou Winkel e apropriou-se de sua ideia do novo e superior metrônomo. Maelzel reconheceu claramente a praticidade, versatilidade e utilidade da invenção de Winkel e, após a recusa deste em lhe vender os direitos sobre o metrônomo, Maelzel patenteou o novo dispositivo sob seu próprio nome. (Manasse, 2022, tradução nossa)

Em 1938, Franz introduziu um metrônomo movido por motor síncrono capaz de manter grande estabilidade temporal graças à precisão da corrente alternada controlada. Esse modelo permaneceu em produção por décadas e representou um ponto de inflexão entre os dispositivos puramente mecânicos, como o de Maelzel, e as tecnologias eletrônicas modernas. Nas décadas seguintes, o metrônomo, de modo geral, incorporou melhorias adicionais, como mecanismos de nivelamento automático e ajustes finos de compensação (Geiger, 2022).

Sob o impulso do avanço dos microcontroladores e da eletrônica digital, surgiram metrônomos programáveis, aplicativos móveis e dispositivos multimodais que combinam estímulos sonoros, visuais e até táteis. Essa evolução tecnológica abriu espaço para inovações que reduzem custo, ampliam a acessibilidade e oferecem interfaces mais intuitivas. Nas próximas sessões, serão melhor detalhados alguns tipos de metrônomos, bem como suas funcionalidades.

2.3. Tipos de Metrônomos

Nesta seção são apresentados os principais tipos de metrônomos, classificados de acordo com suas características de funcionamento e recursos disponíveis.

2.3.1. Metrônomos Mecânicos

São descendentes diretos do modelo de Maelzel, baseados em pêndulo e mola. Produzem um som mecânico característico e permitem ajuste de tempo por deslocamento do peso na barra. Embora clássicos, apresentam limitações como menor precisão e ausência de recursos adicionais.

Figura 1: Wittner Taktell 836 Piccolo, exemplo de metrônomo mecânico (Prelude Musical, 2025).

2.3.2. Metrônomos Eletrônicos

Popularizados no século XX, utilizam circuitos osciladores estáveis. Oferecem volume maior, precisão mais consistente e variações sonoras, além de alguns modelos permitirem subdivisões e acentuações programáveis.

Figura 2: Seiko Quartz Metronome, exemplo de metrônomo eletrônico (Amazon, 2025).

2.3.3. Metrônomos Digitais

Baseados em microcontroladores, são os mais comuns no mercado. Incluem funcionalidades como: memória de *presets*, telas LCD/OLED, integração MIDI e Bluetooth, toque sensível e variações complexas de compasso (estrutura métrica que organiza as batidas em grupos recorrentes). Podem ser tanto aparelhos independentes (*stand-alone*), quanto aplicativos. No entanto, ambos são caracterizados pelo processamento digital.

Figura 3: DB-90 “Dr. Beat”, exemplo de metrônomo digital (BOSS, 2025).

2.3.4. Metrônomos Audiovisuais

Integram estímulos sonoros e visuais de forma sincronizada. Essa categoria, em crescente adoção, oferece suporte aprimorado à percepção temporal, já que combina múltiplas modalidades sensoriais. O protótipo desenvolvido neste trabalho se enquadra nesse tipo.

Figura 4: Korg MA-1, exemplo de metrônomo audiovisual (Korg, 2025).

2.3.5. Metrônomos Táteis

Os metrônomos táteis utilizam vibração como principal forma de sinalização rítmica, permitindo que o músico perceba o tempo por meio de estímulos táteis. Esse tipo de metrônomo é especialmente útil em ambientes ruidosos, tais como em palcos, por exemplo.

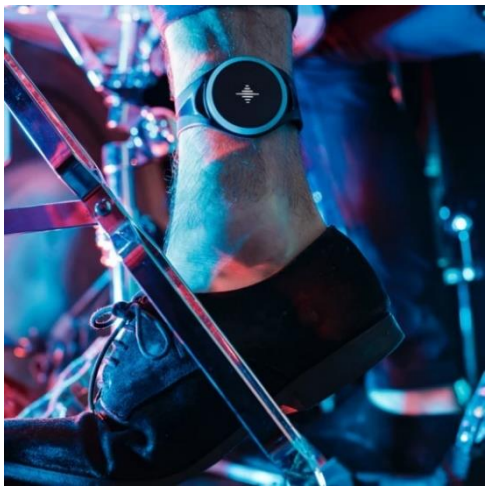

Figura 5: Soundbrenner Pulse, um dispositivo estilo relógio de pulso anexado à perna.

2.4. Percepção Rítmica e Dispositivos Multimodais

A sincronização entre percepção temporal e execução motora é um exemplo de processo multimodal, ou seja, se trata da sincronização entre modos diferentes de interação.

> A integração de múltiplas modalidades sensoriais, como entradas visuais e auditivas, é essencial para promover uma percepção adequada e respostas comportamentais em ambientes dinâmicos, já que esses sinais costumam derivar de eventos ou objetos externos comuns. (Macaluso e Driver apud Lapenta et al., 2023, tradução nossa)

Além dos aspectos perceptivos, a literatura também destaca os efeitos da integração audiovisual sobre a coordenação motora:

> Estímulos audiovisuais simultâneos e co-localizados não são benéficos apenas para a percepção, mas também para tarefas de coordenação motora, incluindo a coordenação com objetos em movimento. (Meyer et al. apud Lapenta et al., 2023, tradução nossa)

Por estas razões, dispositivos audiovisuais tendem a melhorar a percepção do pulso pelo músico em relação aos metrônomos exclusivamente sonoros, além de potencialmente facilitar a execução instrumental, já que o próprio instrumento se torna um objeto em movimento enquanto é tocado. Em ambientes com alto nível de ruído, o estímulo visual pode complementar ou até substituir o clique sonoro, ampliando as possibilidades de adaptação a diferentes contextos de prática musical.

3. Estado da Arte

Atualmente, o metrônomo ultrapassa sua função tradicional de simples marcador temporal e passa a integrar um ecossistema mais amplo de ferramentas digitais voltadas ao apoio à prática musical. No mercado, observa-se a presença de soluções que variam desde equipamentos dedicados de uso profissional até dispositivos experimentais e aplicações baseadas em sistemas embarcados, explorando diferentes formas de interação.

3.1. Produtos Comerciais Representativos

Este item apresenta uma análise de metrônomos amplamente difundidos no mercado, considerando suas principais características funcionais.

3.1.1. Metrônomos Mecânicos Tradicionais

Os metrônomos mecânicos, popularizados desde o século XIX, continuam presentes no mercado contemporâneo e são amplamente utilizados por entusiastas do modelo clássico.

Embora existam diversas marcas e fabricantes, desde modelos tradicionais de alta qualidade, como os da Wittner, representado pela Figura 1, até versões mais acessíveis de marcas como Aroma ou Cherub, a forma geral do produto permanece praticamente a mesma, o que dificulta destacar uma única marca como representativa. Assim, considera-se o próprio formato mecânico pendular como o produto comercial consolidado dentro dessa categoria.

3.1.2. BOSS DB-90 “Dr. Beat”

O BOSS DB-90, apresentado na Figura 3, é um metrônomo digital profissional que combina metronomia avançada com funções de treinamento de ritmo, memória de *presets*, geração de timbres variados, integração MIDI e capacidade de sincronização com dispositivos externos. É frequentemente citado nas listas “top” de metrônomos por sua riqueza de recursos e por atender a músicos profissionais.

3.1.3. Peterson StroboPLUS HDC

Embora conhecido como afinador de alta precisão, o Peterson StroboPLUS HDC disponibiliza também uma função de metrônomo integrada com controles detalhados de parâmetros e saída por alto-falante ou fone. Representa modelos multiferramenta que combinam afinação e metrônomo em um mesmo equipamento.

Figura 6: StroboPLUS HDC (Peterson Electro-Musical Products, 2025).

3.1.4. Korg MA-series

A linha Korg MA (MA-1 / MA-2), apresentado na Figura 4, ilustra o segmento de metrônomos portáteis e acessíveis, com displays de batida, várias subdivisões rítmicas, função *tap tempo* (recurso que ajusta o BPM conforme o usuário toca repetidamente — em um botão ou na tela, por exemplo — seguindo o ritmo desejado) e, em alguns modelos, afinador integrado. São exemplos de dispositivos simples, robustos e amplamente utilizados por estudantes de música.

3.1.5. Soundbrenner Pulse

O Soundbrenner Pulse, apresentado na Figura 5, é um dispositivo háptico que fornece um metrônomo baseado em vibração, complementado por LEDs para sinalização visual e por um aplicativo que permite a criação de padrões e *presets*. Ele exemplifica a linha de produtos que privilegiam *feedback* tátil para prática silenciosa ou em palco, além de integração via aplicativo para ajustes detalhados.

3.1.6. Metrônomo do Cifra Club

O Metrônomo do Cifra Club é um aplicativo digital amplamente utilizado por estar disponível em dispositivos móveis. Ele permite ajuste fino de BPM, subdivisões rítmicas, acentuação de tempos, detecção de tempo por meio da função *tap tempo*, seleção de diferentes timbres de clique e a possibilidade de favoritar *presets* personalizados.

Figura 7: Captura de tela do aplicativo Metrônomo, do Cifra Club (Cifra Club, 2025).

3.1.7. Walrus Audio Canvas Rehearsal Pedal

O pedal Canvas Rehearsal é um *all-in-one* (dispositivo ou sistema que reúne várias funções em um único equipamento) projetado para ensaio e prática musical: ele combina amplificador para fone de ouvido, entrada auxiliar ou Bluetooth para *backing tracks* (faixas de acompanhamento pré-gravadas usadas para estudo ou performance*)*, e um metrônomo embutido com controle de BPM, assinatura de tempo, divisão de batidas e acentuação. A ferramenta conta com display OLED para exibir BPM e configuração do metrônomo, botão *tap tempo*, controle de volume independente para instrumento, batida e auxiliar, além de saída para fones ou amplificação, tornando-o uma solução muito prática para guitarristas ou músicos em geral que desejam praticar com metrônomo sem usar computador, mesa de som ou demais aparelhos.

Figura 8: Canvas Rehearsal (MusicRadar, 2025).

Produtos como o Canvas Rehearsal Pedal e o DB-90 representam o extremo "profissional" (muitas funções, integração MIDI, diferentes interfaces de conexão e configurações complexas), enquanto Korg MA-series são soluções econômicas e portáteis. Aparelhos táteis como o Soundbrenner exploram outra dimensão, em que a tecnologia apresentada é valiosa para ambientes em que o som não pode ser utilizado ou não é preferível.

3.2. Implementações de Desenvolvedores Independentes Utilizando Microcontroladores

Diversos desenvolvedores independentes têm publicado implementações de metrônomos baseados em microcontroladores em plataformas de código aberto, como o GitHub. Esses projetos exploram microcontroladores amplamente utilizados no desenvolvimento de sistemas embarcados, apresentando diferentes abordagens para a geração e a sinalização do pulso rítmico. Há, por exemplo, repositórios públicos que demonstram a utilização de placas Arduino associadas a LEDs para a indicação visual do tempo (braveness23, 2024), bem como implementações mais simples baseadas no microcontrolador ESP32, que utilizam buzzers e um único LED como forma de *feedback* auditivo e visual (Umadi, 2023).

Essas implementações abertas servem como referências de engenharia, demonstrando estratégias de tratamento temporal, organização modular de firmware e uso de bibliotecas otimizadas para manipulação de periféricos. No contexto específico do Raspberry Pi Pico, é possível encontrar exemplos que vão desde um metrônomo implementado em CircuitPython e baseado em material técnico da plataforma de ensino Adafruit (Abidin, 2022), até projetos que utilizam MicroPython para controlar anéis de LEDs WS2812B com temporização sincronizada (Pinto Neto, 2021).

Tais implementações, embora não constituam produtos comerciais, fornecem bases técnicas úteis e ilustram abordagens adotadas pela comunidade para resolver desafios semelhantes aos enfrentados pelo protótipo desenvolvido neste trabalho.

3.3. Síntese: Lacunas e Oportunidades

Ao comparar produtos comerciais e os projetos experimentais com microcontroladores, constata-se que:

- Há uma forte presença de soluções audiovisuais e hápticas comerciais, e de soluções profissionais que oferecem grande funcionalidade, mas poucos

dispositivos combinam *presets* alternáveis por interrupção com uma implementação otimizada em plataformas como o Raspberry Pi Pico W integrado à BitDogLab;

- A literatura sugere que abordagens multimodais podem melhorar a robustez da percepção rítmica em diferentes contextos, mas ainda existem oportunidades para estudos aplicados que avaliem protótipos multimodais embarcados de baixo custo no ecossistema educacional;
- Projetos acadêmicos que articulem avaliação técnica e usabilidade (experimentos com músicos) de um dispositivo metrônomo em um único trabalho são menos frequentes, o que abre espaço para uma monografia que una implementação, medição técnica e avaliação com usuários.

Portanto, o protótipo proposto neste trabalho (um metrônomo audiovisual com *presets* programáveis e alternância por interrupção utilizando o Raspberry Pi Pico W) encontra seu lugar no estado da arte ao oferecer uma solução econômica, programável, e testável que utiliza técnicas embarcadas modernas. As etapas de implementação, bem como os resultados alcançados, serão apresentadas na sessão seguinte.

4. Especificação e Implementação do Sistema

A escolha do hardware é uma etapa essencial no desenvolvimento de sistemas embarcados. Neste trabalho a placa BitDogLab com Raspberry Pi Pico W foi selecionada por ter sido amplamente explorada durante um programa gratuito de capacitação profissional técnica voltado a estudantes de nível superior nas áreas de Tecnologias da Informação e Comunicação e Engenharias, com foco em Sistemas Embarcados. O programa, denominado Embarcatech, foi uma iniciativa do Ministério da Ciência, Tecnologia e Inovação (MCTI), coordenada pela Softex e executada por instituições credenciadas pelo Comitê da Área de Tecnologia da Informação (CATI) (Embarcatech, 2025). Além do contexto formativo, a plataforma foi escolhida por sua viabilidade econômica, bom desempenho e recursos adequados às exigências de temporização e controle de um metrônomo audiovisual.

4.1. Raspberry Pi Pico W e BitDogLab

O Raspberry Pi Pico W é uma placa de desenvolvimento que utiliza o microcontrolador RP2040, projetado pela Raspberry Pi Foundation. O chip possui um dual-core ARM Cortex-M0+ a 133 MHz, com 264 KB de RAM. Sua versatilidade, baixo custo e suporte a múltiplas interfaces de comunicação — como I²C, SPI, UART, ADC, PWM e o poderoso subsistema PIO (Programmable I/O) — a tornam adequada para sistemas embarcados que demandam precisão temporal, controle de LEDs, leitura de sensores e execução de rotinas síncronas e assíncronas (Raspberry Pi Fundation, 2025).

A BitDogLab, por sua vez, é uma plataforma acadêmica amplamente usada em disciplinas de sistemas embarcados, composta por:

- Raspberry Pi Pico W integrado;
- Matriz de LEDs WS2812**;**
- Display OLED SSD1306;
- LED RGB;
- Joystick analógico (eixo X/Y e botão central);
- Botões adicionais;
- Buzzers ativos;
- Microfone;
- Bateria recarregável.

Além de circuitos de proteção, resistores, reguladores e conectores didáticos, bem como energização via cabo, não se limitando ao uso de bateria. Essa integração de periféricos facilita a prototipagem de dispositivos interativos, que é exatamente o caso do projeto deste metrônomo. Ainda, vale ressaltar o caráter educacional da plataforma, que é voltada para o ensino de sistemas embarcados, programação e eletrônica. Seus elementos foram selecionados para incentivar o aprendizado prático e progressivo. Totalmente aberta, a ferramenta pode ser copiada e aprimorada por qualquer usuário (BitDogLab, 2025).

4.2. Requisitos e Funcionamento

A plataforma de prototipação, possui diversas funcionalidades disponíveis para implementação de projetos. Neste trabalho foram utilizadas as seguintes, entre os requisitos funcionais e não-funcionais:

4.2.1. Requisitos Funcionais

- RF01 — Permitir a seleção de diferentes tempos.
- RF02 — Emitir sinal sonoro sincronizado com o tempo.
- RF03 — Emitir sinal luminoso sincronizado, distinguindo tempo e contratempo.
- RF04 — Exibir informações de tempo e *presets* no display OLED.
- RF05 — Permitir alternância imediata entre *presets* através de interrupção.
- RF06 — Registrar e atualizar automaticamente o *preset* atual.
- RF07 — Navegar pelas opções utilizando joystick e botões.

4.2.2. Requisitos Não Funcionais

- RNF01 — Baixo consumo de energia.
- RNF02 — Alta responsividade, com latência mínima entre processamento interno e saída sonora/visual.
- RNF03 — Código modular e organizado, seguindo boas práticas de desenvolvimento.
- RNF04 — Clareza visual e sonora, com sinais facilmente perceptíveis.
- RNF05 — Utilização exclusiva de hardware embarcado, sem dependência de software externo.

4.2.3. Funcionamento

O metrônomo inicia solicitando ao usuário, através do display OLED, a configuração da quantidade de *presets* desejados. Para ajustar esse número, o usuário movimenta o 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 loop: valor entre 2 e 9.

Durante a execução, o metrônomo gera bips audíveis por meio dos buzzers e exibe, na matriz de LEDs, os números correspondentes a cada batida do ciclo rítmico. Esse ciclo — o loop de batidas — consiste em uma sequência repetitiva cujo primeiro *beat* é acentuado, seguido pelos demais, que se repetem continuamente enquanto o metrônomo estiver ativo.

Além disso, o 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.

Ainda na execução, há um botão dedicado para avançar para o próximo *preset*, outro para retornar ao *preset* anterior e um terceiro para pausar a execução.

4.3. Especificação do Hardware

Nesta seção é apresentada a especificação do hardware utilizado no desenvolvimento do sistema, descrevendo os principais componentes e sua organização funcional.

4.3.1. Diagrama em Blocos

Com o objetivo de fornecer uma visão geral da arquitetura do sistema, a Figura 9 apresenta o diagrama em blocos do hardware, destacando os principais módulos e sua interligação com o microcontrolador central.

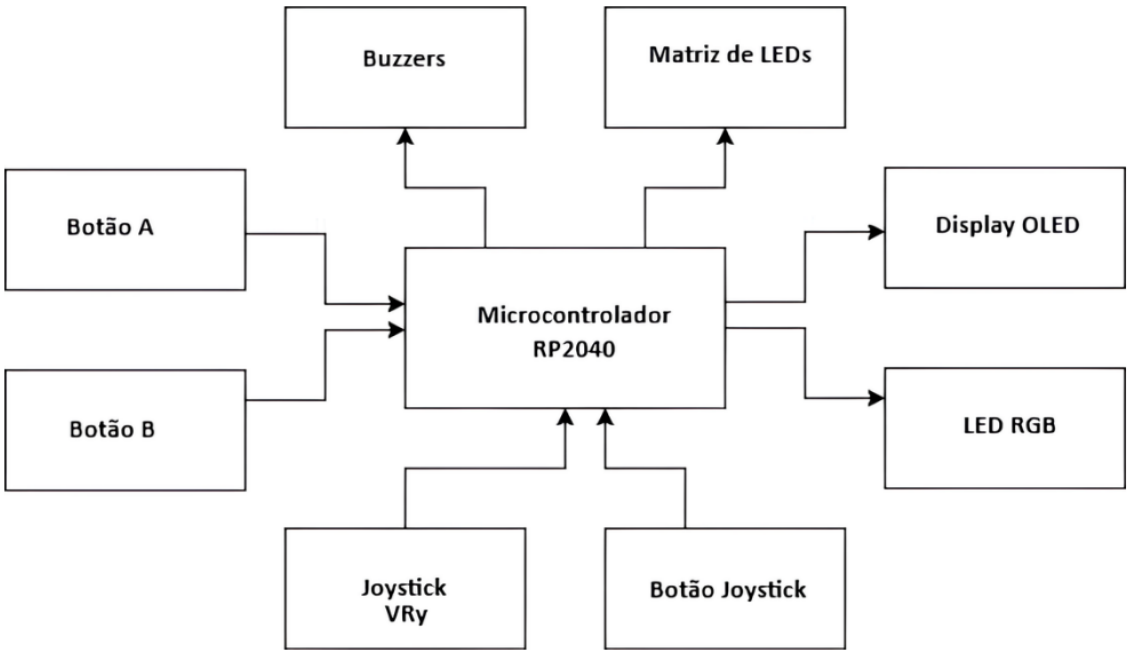

Figura 9: Diagrama em blocos (elaboração própria, 2025).

4.3.2. Configuração e Função de Cada Bloco

- Microcontrolador: Configurado para gerenciar entradas e saídas digitais e analógicas. Controla todas as operações do metrônomo.
- Matriz de LEDs: Controlada via PIO para exibir padrões numéricos. Exibe o número da batida atual do loop, o *beat*.
- LED RGB: Controlado por GPIO para piscar e mudar de cor. Indica o contratempo e a mudança de *presets*.
- Buzzers: Controlados por PWM para emitir sons em diferentes frequências. Emitem os bips para marcar o tempo.
- Display OLED: Utiliza comunicação I²C para exibir dados. Exibe as informações de configuração.
- Joystick VRy: Configurado para enviar sinais analógicos ao microcontrolador, utilizando o conversor analógico-digital. Permite alterar os valores de BPM e *beats* ao configurar os *presets*.
- Botão Joystick: Controlado por GPIO com resistor de *pull-up* interno para que quando pressionado, retornar nível lógico baixo. Permite pausar a execução do metrônomo.
- Botão A: Controlado por GPIO com resistor de *pull-up* interno para que quando pressionado, retornar nível lógico baixo. Confirma os valores durante as

configurações e permite avançar para o próximo *preset* quando o metrônomo está em execução.

- Botão B: Controlado por GPIO com resistor de *pull-up* interno para que quando pressionado, retornar nível lógico baixo. Permite retornar ao *preset* anterior.

4.3.3. Circuito Completo do Hardware e Pinagem Utilizada

No Raspberry Pi Pico W, os pinos GPIO (General Purpose Input/Output) constituem a interface fundamental para comunicação com sensores, atuadores e dispositivos externos. Cada GPIO pode operar como entrada ou saída digital, além de assumir funções alternativas como PWM, ADC, I²C, SPI, UART. A escolha e organização da pinagem determinam como cada periférico é conectado ao microcontrolador, garantindo que sinais elétricos, comunicação serial e controle temporal funcionem de forma correta e sincronizada no projeto. Configurou-se da seguinte forma:

- 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 loop): GPIO 21 - Pino 27
- Buzzer B (para demais *beats* do loop): GPIO 10 - Pino 14

A Figura 10 ilustra o circuito completo do sistema, detalhando a ligação dos periféricos ao microcontrolador central.

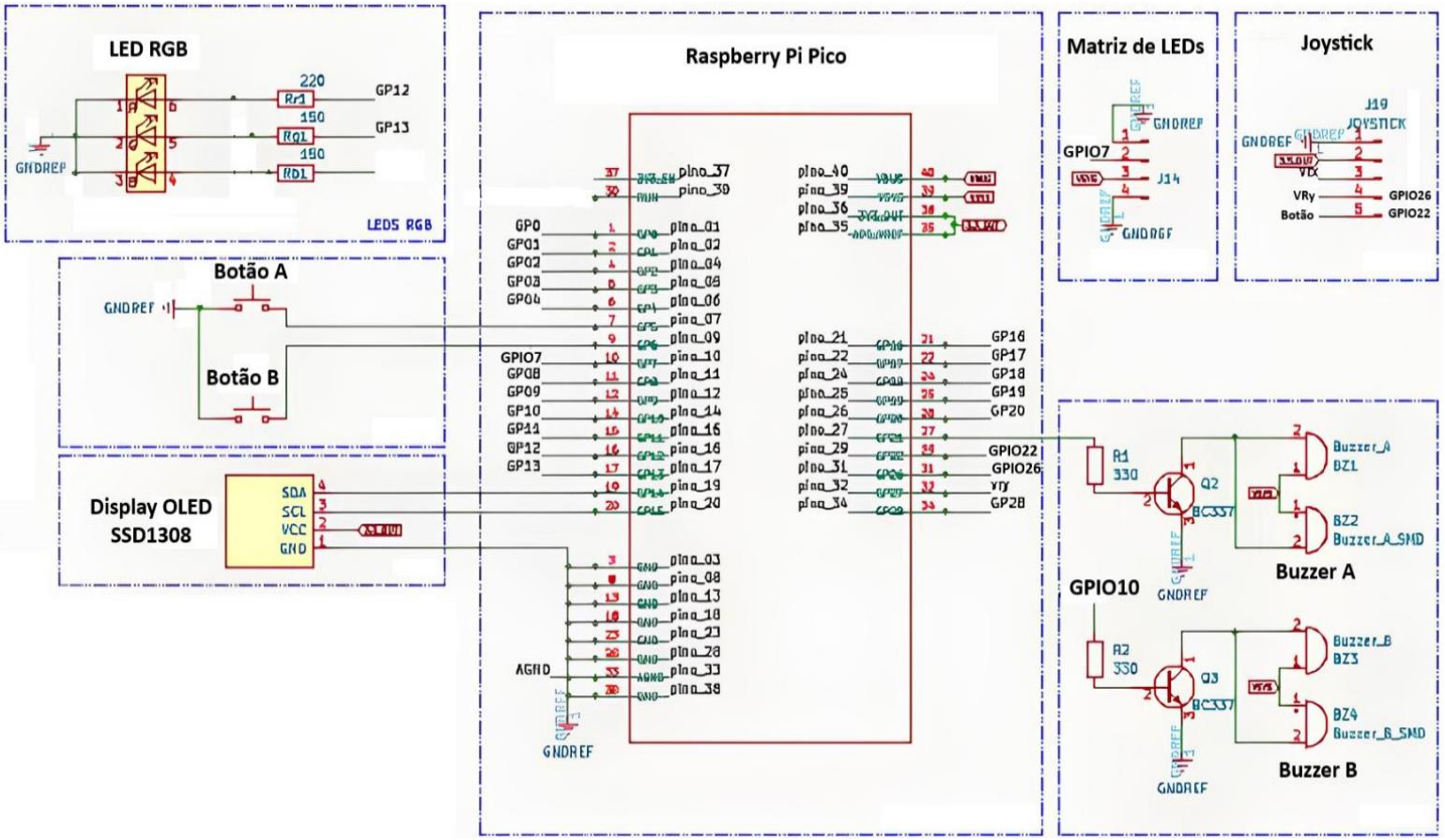

Figura 10: Circuito do hardware (elaboração própria, 2025).

4.4. Especificação do Firmware

O firmware do metrônomo audiovisual foi desenvolvido integralmente em 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, bem como realizado o setup do SDK, incluindo a integração das bibliotecas necessárias para o controle de GPIO, PWM, ADC, PIO e I²C. Também foram adicionados módulos específicos para o funcionamento do display OLED SSD1306 e da matriz de LEDs WS2812. Para cada função implementada, foi utilizado o ambiente de depuração do Visual Studio Code para identificar e remover defeitos encontrados ou comportamentos indesejados, adicionando *breakpoints* (pontos definidos no código onde a execução do programa para temporariamente, permitindo inspeção de variáveis) e monitorando expressões.

A linguagem C foi escolhida por oferecer maior eficiência e controle direto sobre o hardware, características essenciais em sistemas embarcados com restrições de memória, processamento e tempo real. Em comparação com linguagens interpretadas, como Python, o código em C é compilado diretamente para código de máquina, resultando em menor latência e maior previsibilidade temporal, fundamentais para a precisão rítmica do metrônomo.

O código segue uma abordagem modular, separando as rotinas de exibição, controle de LEDs, entradas do usuário para configuração dos *presets*, interrupções e geração sonora. Essa estrutura está alinhada às boas práticas de Engenharia de Software: Medeiros (2025) enfatiza que a combinação entre modularidade e desenvolvimento incremental simplifica a depuração, reduz riscos e aumenta a confiabilidade do sistema como um todo.

Para garantir transparência neste trabalho, bem como reprodutibilidade e facilidade de auditoria por outros pesquisadores e desenvolvedores, todos os módulos — incluindo o código-fonte completo, arquivos auxiliares e documentação — estão disponíveis publicamente no repositório oficial do projeto: https://github.com/MelkBraga/Metronomo_de_Presets.

4.4.1. Blocos Funcionais

A Figura 11 ilustra os blocos funcionais do sistema embarcado, evidenciando a divisão das responsabilidades entre seus principais módulos.

Figura 11: Blocos funcionais (elaboração própria, 2025).

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* do loop 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 para exibição do contratempo ou indicação da troca de *presets*.

4.4.2. Fluxograma

Com o objetivo de detalhar a sequência de execução do sistema, a Figura 12 apresenta o fluxograma de funcionamento, evidenciando o fluxo lógico das operações realizadas.

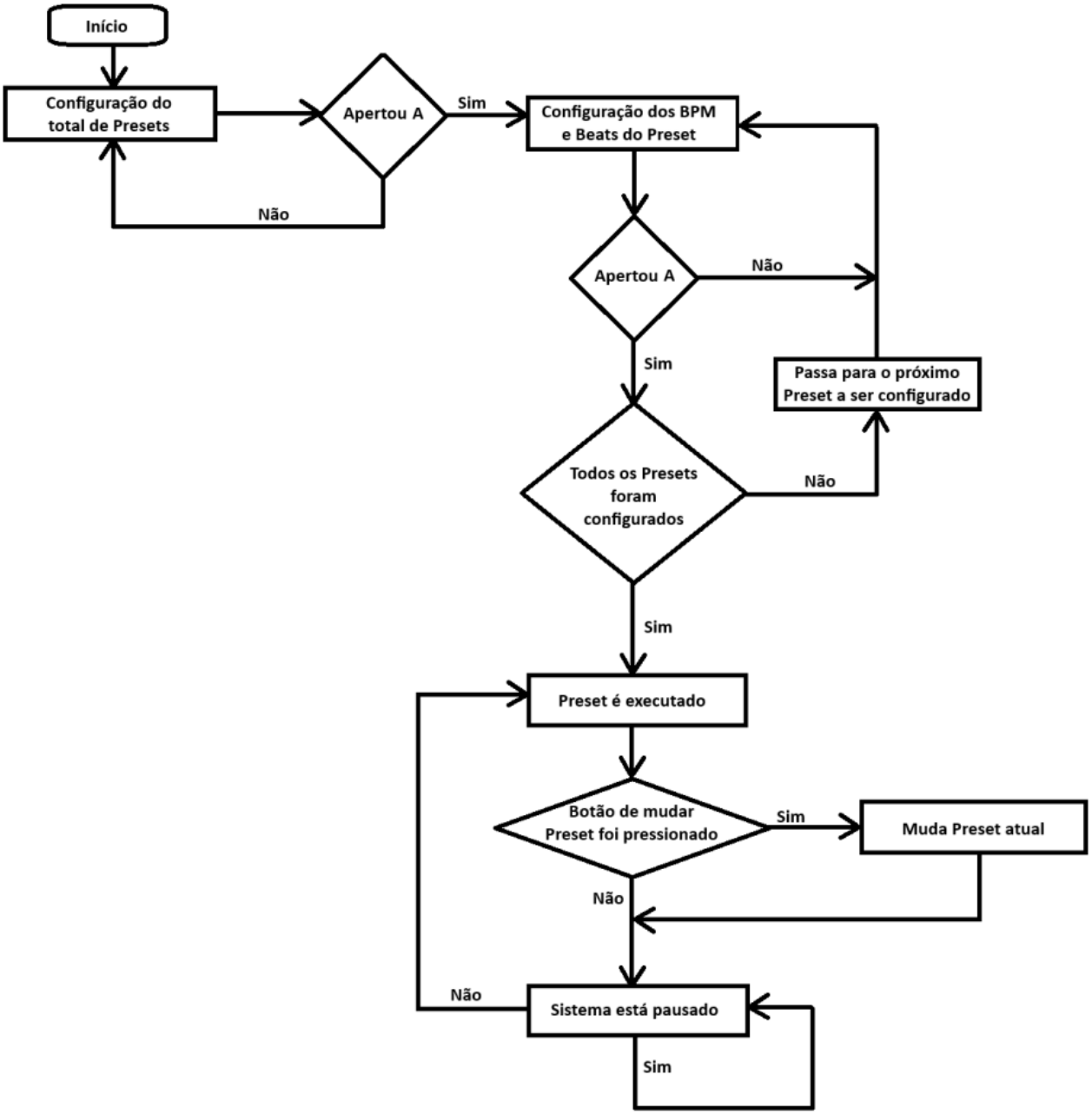

Figura 12: Fluxograma (elaboração própria, 2025).

4.4.3. Visão Geral das Funções-Chave do Firmware

- Inicialização da matriz de LED WS2812 via PIO

Aqui o firmware carrega um programa PIO pré-compilado para enviar sinais temporizados aos LEDs WS2812, garantindo precisão de microssegundos. O procedimento é chamado na inicialização do sistema, peça função setup().

```
void npInit(uint pin) {
    uint offset = pio_add_program(pio0, &ws2818b_program);
    np_pio = pio0;
    sm = pio_claim_unused_sm(np_pio, true);
    ws2818b_program_init(np_pio, sm, offset, pin, 800000.f);
}
```

Figura 13: Função npInit() (elaboração própria, 2025).

- Função de exibição visual na matriz LED

Acende os LEDs em um arranjo 5×5, e é chamada pela função main(). Devido à disposição física dos periféricos na BitDogLab, a operação do dispositivo torna-se mais ergonômica quando utilizado de cabeça para baixo. Por esse motivo, tanto a numeração exibida na matriz de LEDs quanto os textos apresentados no display OLED foram rotacionados.

```
void exibir_numero(int num) {
    npClear();

    // Tabela 5x5 com os padrões dos dígitos (0-9), invertidos.
    // Omitida aqui para brevidade. Código completo disponível em:
    // https://github.com/MelkBraga/Metronomo_de_Presets
    const uint8_t numeros[10][5][5] = {
        /* ... matriz omitida ... */
    };

    for (int row = 4; row >= 0; row--) {
        for (int col = 4; col >= 0; col--) {
            int led_index = zigzag_map(4 - row, 4 - col);
            if (numeros[num][row][col]) {
                npSetLED(led_index, 255, 0, 0);
            }
        }
    }

    npWrite();
}
```

Figura 14: Função exibir_numero() (elaboração própria, 2025).

A matriz contendo os padrões gráficos dos dígitos (0–9) foi omitida aqui neste trabalho por ser extensa e repetitiva. O código integral encontra-se disponível no repositório do projeto.

- Atualização do display OLED

O display mostra o *preset* atual, BPM e quantidade de *beats* do loop. O procedimento é chamado pelas funções main() e config_preset().

```
void atualizar_display(const char *mensagem) {
    ssd1306_fill(&ssd, false);
    ssd1306_draw_string(&ssd, mensagem, 0, 0);
    ssd1306_send_data(&ssd);
}
```

Figura 15: Função atualizar_display() (elaboração própria, 2025).

- Configuração dos *presets*

Permite definir quantos *presets* serão usados e ajustar, para cada um, os valores de BPM e *beats* por loop. O procedimento é chamado pela função main().

```
void config_preset() {
    int contador = 1;
    // Loop para definir o total de Presets
    while (true) {
        adc_select_input(0);
        uint16_t eixo_y = adc_read();
        if (eixo_y < 1000 && contador < 20) contador++;
        if (eixo_y > 3000 && contador > 1) contador--;
        if (gpio_get(BUTTON_A) == 0) break;
        char buffer[20];
        sprintf(buffer, "Total de\nPresets:\n  >%d", contador);
        atualizar_display(buffer);
        sleep_ms(80);
    }
    bpm_count = contador;
    // Loop para definir os BPM e Beats de cada Preset
    for (int i = 1; i <= bpm_count; i++) {
        int bpm = 60;
        int beats = 4;
        while (true) {
            char buffer[30];
            if (i <= 9){
                sprintf(buffer, " BPM do\n\nPreset %d\n  >%d", i, bpm);
            } else{
                sprintf(buffer, " BPM do\n\nPreset%d\n  >%d", i, bpm);
            }
            atualizar_display(buffer);
            sleep_ms(80);
            adc_select_input(0);
            uint16_t eixo_y = adc_read();
            if (eixo_y < 1000 && bpm < 300) bpm++;
            else if (eixo_y > 3000 && bpm > 1) bpm--;
            if (gpio_get(BUTTON_A) == 0) {
                while (true) {
                    char buffer[30];
                    if (i <= 9){
                        sprintf(buffer, "Beats do\nPreset %d\n   >%d", i, beats);
                    } else{
                        sprintf(buffer, "Beats do\nPreset%d\n   >%d", i, beats);
                    }
                    atualizar_display(buffer);
                    sleep_ms(80);
                    adc_select_input(0);
                    uint16_t eixo_y = adc_read();
                    if (eixo_y < 1000 && beats < 9) beats++;
                    else if (eixo_y > 3000 && beats > 2) beats--;
                    if (gpio_get(BUTTON_A) == 0) break;
                }
                break;
            }
        }
        bpm_values[i-1] = bpm;
        beats_values[i-1] = beats;
    }
}
```

Figura 16: Função config_preset() (elaboração própria, 2025).

- Função de *debounce*

Responsável por garantir leituras confiáveis dos botões eliminando acionamentos múltiplos indesejados. Para cada botão, é armazenado o instante do último acionamento válido e comparado com o tempo atual do sistema. Caso o intervalo mínimo de 200 ms seja respeitado, o acionamento é considerado válido e a função retorna verdadeiro. Caso contrário, o evento é descartado. A função é chamada pelo procedimento isr_botoes().

```
bool debounce(uint gpio) {
    static uint32_t ultimo_tempo_A = 0;
    static uint32_t ultimo_tempo_B = 0;
    static uint32_t ultimo_tempo_JOYSTICK_BUTTON = 0;

    uint32_t tempo_atual = to_ms_since_boot(get_absolute_time());

    if (gpio == BUTTON_A) {
        if (tempo_atual - ultimo_tempo_A > 200) {
            ultimo_tempo_A = tempo_atual;
            return true;
        }
    } else if (gpio == JOYSTICK_BUTTON) {
        if (tempo_atual - ultimo_tempo_JOYSTICK_BUTTON > 200) {
            ultimo_tempo_JOYSTICK_BUTTON = tempo_atual;
            return true;
        }
    } else if (gpio == BUTTON_B) {
        if (tempo_atual - ultimo_tempo_B > 200) {
            ultimo_tempo_B = tempo_atual;
            return true;
        }
    }
    return false;
}
```

Figura 17: Função debounce() (elaboração própria, 2025).

- Interrupção para troca de *presets* e pausa

O procedimento é chamado toda vez que um dos botões é pressionado, e a alternância ocorre instantaneamente, mesmo com o metrônomo em execução.

```
void isr_botoes(uint gpio, uint32_t events) {
    if (debounce(gpio)) {
        if (gpio == JOYSTICK_BUTTON && metronome_running == true) {
            paused = !paused;
        } else if (gpio == BUTTON_A && metronome_running == true) {
            current_bpm_index = (current_bpm_index + 1) % bpm_count;
        } else if (gpio == BUTTON_B && metronome_running == true) {
            if (current_bpm_index == 0)
                current_bpm_index = bpm_count - 1;
            else
                current_bpm_index--;
        }
    }
}
```

Figura 18: Função isr_botoes (elaboração própria, 2025).

- Geração de áudio com PWM

O bip é assíncrono, não travando o fluxo do metrônomo. O procedimento chamado pela função main(), dentro do loop central do programa.

```
void beep_async(uint gpio, int frequency, int duration_ms) {
    // Configurar PWM no GPIO do buzzer
    gpio_set_function(gpio, GPIO_FUNC_PWM);
    uint slice_num = pwm_gpio_to_slice_num(gpio);

    // Configurar frequência do PWM
    uint32_t clock = 125000000; // Clock padrão do RP2040
    uint32_t divider16 = clock / frequency / 4096 + (clock % (frequency * 4096) != 0);
    if (divider16 / 16 == 0) divider16 = 16;
    uint32_t wrap = clock * 16 / divider16 / frequency - 1;

    pwm_config config = pwm_get_default_config();
    pwm_config_set_clkdiv_mode(&config, PWM_DIV_FREE_RUNNING);
    pwm_config_set_clkdiv(&config, divider16 / 16.0f);
    pwm_config_set_wrap(&config, wrap);

    pwm_init(slice_num, &config, true);

    // 50% duty cycle para gerar o som
    pwm_set_gpio_level(gpio, wrap / 2);

    // Configurar o timer para desligar o beep após a duração
    add_alarm_in_ms(duration_ms, stop_beep_callback, (void*)(uintptr_t)gpio, false);
}
```

Figura 19: Função beep_async (elaboração própria, 2025).

- Interrupção do sinal sonoro

Função que faz o buzzer parar de emitir som. É chamada pelo procedimento beep_async().

```
int64_t stop_beep_callback(alarm_id_t id, void *user_data) {
    uint gpio = (uintptr_t)user_data;
    pwm_set_gpio_level(gpio, 0); // Desliga o som
    gpio_set_function(gpio, GPIO_FUNC_SIO); // Retorna o GPIO ao estado normal
    gpio_put(gpio, 0);
    return 0; // Nao reagendar o alarme
}
```

Figura 20: Função stop_beep_callback() (elaboração própria, 2025).

- Inicialização e configuração dos principais periféricos do sistema

São configuradas as interfaces de comunicação I²C para o display OLED, o conversor analógico-digital para leitura do joystick, bem como os pinos de entrada associados aos botões, incluindo a habilitação de resistores de *pull-up* e interrupções de cada botão. Adicionalmente, a função inicializa os pinos de saída para controle dos LEDs e configura da matriz de LEDs. O procedimento é chamado pela função main().

```c
void setup() {
    stdio_init_all();
    i2c_init(I2C_PORT, 400 * 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(&ssd, 128, 64, false, OLED_ADDR, I2C_PORT);
    ssd1306_config(&ssd);

    adc_init();
    adc_gpio_init(JOYSTICK_Y);

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

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

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

    gpio_set_irq_enabled_with_callback(BUTTON_A, GPIO_IRQ_EDGE_FALL, true, &isr_botoes);
    gpio_set_irq_enabled_with_callback(BUTTON_B, GPIO_IRQ_EDGE_FALL, true, &isr_botoes);
    gpio_set_irq_enabled_with_callback(JOYSTICK_BUTTON, GPIO_IRQ_EDGE_FALL, true, &isr_botoes);

    gpio_init(LED_RED);
    gpio_set_dir(LED_RED, GPIO_OUT);
    gpio_init(LED_BLUE);
    gpio_set_dir(LED_BLUE, GPIO_OUT);

    npInit(LED_MATRIX);
    npClear();
}
```

Figura 21: Função setup() (elaboração própria, 2025).

- Função principal com loop central do metrônomo

A função é responsável por inicializar o sistema, configurar os *presets* definidos pelo usuário e iniciar o laço principal de execução do metrônomo. O loop calcula o intervalo entre batidas com base no BPM do *preset* ativo e organiza a sequência exibição visual → bip → contratempo, verificando a cada iteração se o sistema está pausado e realizando o controle do LED azul para indicar o contratempo e do LED vermelho para indicar uma iminente troca de *preset*.

```c
int main() {
    setup();
    config_preset();
    metronome_running = true;

    // Loop principal de execução do metrônomo depois que os Presets foram configurados
    while (metronome_running) {
        int bpm = bpm_values[current_bpm_index];
        int led_flag = current_bpm_index;
        int intervalo = 60000 / bpm;
        int duracao = intervalo / 2;
        int contagem = beats_values[current_bpm_index];
        gpio_put(LED_RED, 0);
        absolute_time_t proximo_batimento = get_absolute_time();

        for (int i = 1; i <= contagem; i++) {
            if (paused) continue;
            char buffer[20];
            if ((current_bpm_index + 1) <= 9){
                if (bpm_values[current_bpm_index] <= 99){
                    sprintf(buffer, "Preset %d\nBPM: %d\n\nBeats:%d", current_bpm_index + 1,
                        bpm_values[current_bpm_index], beats_values[current_bpm_index]);
                } else {
                    sprintf(buffer, "Preset %d\nBPM:%d\n\nBeats:%d", current_bpm_index + 1,
                        bpm_values[current_bpm_index], beats_values[current_bpm_index]);
                }
            } else{
                if (bpm_values[current_bpm_index] <= 99){
                    sprintf(buffer, "Preset%d\nBPM: %d\n\nBeats:%d", current_bpm_index + 1,
                        bpm_values[current_bpm_index], beats_values[current_bpm_index]);
                } else{
                    sprintf(buffer, "Preset%d\nBPM:%d\n\nBeats:%d", current_bpm_index + 1,
                        bpm_values[current_bpm_index], beats_values[current_bpm_index]);
                }
            }

            atualizar_display(buffer);
            exibir_numero(i);

            if (i == 1) {
                beep_async(BUZZER_A, 800, duracao);
            } else {
                beep_async(BUZZER_B, 600, duracao);
            }

            // Pisca o LED azul no contratempo (metade do intervalo)
            absolute_time_t meio_tempo = delayed_by_ms(get_absolute_time(), intervalo / 2);

            // Aguarda o contratempo para piscar o LED azul
            if (led_flag == current_bpm_index){
                sleep_until(meio_tempo);
                gpio_put(LED_BLUE, 1);
                sleep_ms(duracao / 2);
                gpio_put(LED_BLUE, 0);
            } else{ // Liga LED vermelho para indicar que o preset vai mudar
                gpio_put(LED_RED, 1);
            }

            proximo_batimento = delayed_by_ms(proximo_batimento, intervalo);
            sleep_until(proximo_batimento);
        }
    }
    return 0;
}
```

Figura 22: Função main (elaboração própria, 2025).

Além das rotinas apresentadas, o firmware conta ainda com diversas outras funções de suporte, incluindo utilitários de escrita no display, manipulação de memória da matriz WS2812, ajustes finos de temporização. Embora não detalhadas individualmente, essas funções são essenciais para o correto funcionamento do sistema como um todo, assegurando a integração harmoniosa entre os periféricos, a estabilidade temporal do metrônomo e a consistência da interface audiovisual implementada.

4.4.4. Controle de Temporização do Sistema

O controle temporal do metrônomo é realizado integralmente por software, utilizando os recursos de temporização do microcontrolador RP2040. O andamento musical (BPM) definido pelo usuário é calculado pela relação $\text{Intervalo} = \frac{60000}{BPM}$, em que 60000 representa a quantidade de milissegundos em um minuto, determinando o tempo entre batidas consecutivas.

Esse controle é implementado principalmente no laço principal da função main(), onde o intervalo calculado é utilizado para agendar cada batimento do metrônomo. Para garantir precisão e estabilidade ao longo da execução, o firmware adota uma abordagem baseada em tempo absoluto, por meio das funções get_absolute_time(), delayed_by_ms() e sleep_until(), todas nativas do SDK do Raspberrry Pi Pico. Essa estratégia evita o acúmulo de erros temporais, assegurando regularidade no pulso rítmico mesmo durante execuções prolongadas.

Para geração do *feedback* sonoro, a função beep_async() configura o sinal PWM do buzzer e utiliza um temporizador por alarme para limitar a duração do bip, enquanto a função de retorno stop_beep_callback() é responsável por interromper o sinal após o tempo programado. Dessa forma, a duração de cada batida sonora permanece proporcional ao andamento configurado, sem bloquear o fluxo principal de execução.

O *feedback* visual é sincronizado com o controle temporal do sistema. A exibição dos números na matriz de LEDs e o acionamento dos LEDs indicadores ocorrem no mesmo instante em que cada batida é gerada, enquanto a marcação do contratempo é realizada por meio de um evento temporal intermediário, acionado na metade do intervalo entre batidas.

4.4.5. Tecnologias Utilizadas no Desenvolvimento do Sistema

A Tabela 1 mostra as principais tecnologias utilizadas no desenvolvimento do sistema.

Protocolos de comunicação	
I²C (Inter-Integrated Circuit)	Barramento de comunicação serial, bidirecional onde mais de um dispositivo pode assumir o papel de mestre, que usa uma linha de dados para enviar e receber informação (SDA) e uma linha de *clock* serial usada para sincronizar a comunicação entre dispositivos (SCL).
SPI (Serial Peripheral Interface)	Protocolo serial rápido e *full-duplex*, isto é, uma comunicação em que dois dispositivos podem transmitir e receber dados simultaneamente por linhas

	separadas, sem precisar alternar entre envio e recebimento.
UART (Universal Asynchronous Receiver/Transmitter)	Interface de comunicação assíncrona ponto a ponto, que transmite dados byte a byte usando um pino responsável por transmitir dados (TX) e um pino responsável por receber dados (RX), sem *clock* compartilhado.
Recursos internos do microcontrolador	
ADC (Analog-to-Digital Converter)	Conversor que transforma um sinal analógico (tensão contínua) em um valor digital que pode ser lido pelo microcontrolador.
PWM (Pulse-Width Modulation)	Técnica que controla potência variando a largura do pulso de um sinal digital, mantendo a frequência constante.
PIO (Programmable Input/Output)	Tratando-se especificamente do Raspberry Pi Pico, são blocos programáveis que permitem criar protocolos de comunicação personalizados e manipulação rápida de pinos, sem ocupar a CPU principal.
GPIO (General-Purpose Input/Output)	Pinos de uso geral do microcontrolador, configuráveis como entrada ou saída, usados para ler sensores ou acionar dispositivos.
Recurso de componente eletrônico	
Resistor de *pull-up*	Resistor conectado ao ponto de alimentação positiva de um circuito para manter um pino em nível alto quando ele não está sendo ativamente acionado, evitando instabilidade do sinal.

Tabela 1: Tecnologias utilizadas (elaboração própria, 2025).

5. Avaliação, Testes e Resultados

Este capítulo apresenta a avaliação do metrônomo audiovisual desenvolvido, contemplando a metodologia de testes adotada, os procedimentos experimentais realizados e a análise dos resultados obtidos, incluindo dados qualitativos e quantitativos.

5.1. Metodologia de Testes

Para verificar a precisão, estabilidade e usabilidade do dispositivo, foram conduzidos testes técnicos e empíricos, incluindo medições de tempo, análises de sincronização entre LEDs, buzzer e display, validação da interface física e avaliação prática com músicos. Os testes seguiram procedimentos controlados, com repetição sistemática das medições, garantindo confiabilidade nos resultados observados.

5.2. Testes Técnicos

A seguir, são apresentados os testes técnicos realizados e seus respectivos resultados.

5.2.1. Precisão de BPM

O objetivo destes testes foi avaliar a precisão temporal do *feedback* auditivo do metrônomo audiovisual.

Inicialmente, realizou-se a medição do intervalo entre batimentos por meio de um cronômetro externo, associada à contagem manual dos batimentos por minuto (BPM). O início do metrônomo e do cronômetro foi sincronizado no mesmo instante, e foram configurados três *presets* distintos: 60 BPM, 120 BPM e 180 BPM. Para cada configuração, procedeu-se à contagem manual dos cliques audíveis durante o intervalo de um minuto. Como evidenciado na Tabela 2, em todos os casos o número de batimentos contabilizados correspondeu exatamente ao valor de BPM previamente configurado no sistema.

***Preset* configurado (BPM)**	**BPM medido**	**Tempo de observação (segundos)**	**Diferença (BPM)**	**Erro relativo (%)**
60	60	60	0	0
120	120	60	0	0
180	180	60	0	0
240	240	60	0	0

Tabela 2: Validação por contagem manual dos batimentos (elaboração própria, 2025).

Adicionalmente, efetuou-se uma comparação direta com o metrônomo digital do aplicativo Metrônomo do Cifra Club, amplamente utilizado por músicos e reconhecido por sua alta precisão, adotado neste trabalho como referência confiável para validação. Nessa etapa, foram configurados quatro *presets* — 60 BPM, 120 BPM, 180 BPM e 240 BPM — tanto no metrônomo desenvolvido quanto no aplicativo. Ambos os dispositivos foram acionados simultaneamente, tomando-se como referência o instante de emissão do primeiro clique sonoro. Cada execução foi mantida por um período de um minuto e, ao longo de todas as medições, não foram observados descompassos ou variações perceptíveis entre os cliques emitidos pelos dois sistemas, como mostra a Tabela 3.

***Preset* configurado (BPM)**	**BPM de referência (aplicativo)**	**Tempo de observação (segundos)**	**Descompasso perceptível**
60	60	60	Não
120	120	60	Não
180	180	60	Não
240	240	60	Não

Tabela 3: Comparação com metrônomo de referência (elaboração própria, 2025).

5.2.2. Teste do Display OLED e do LED RGB Como Indicador de Mudança de *Preset*

O objetivo deste teste foi avaliar a correta exibição das informações no display OLED, bem como a legibilidade e estabilidade dos dados apresentados ao usuário. Adicionalmente, buscou-se verificar o funcionamento do LED RGB como indicador visual de mudança de *preset*, que deve mudar para a cor vermelha no momento da alternância entre *presets*.

Os ensaios consistiram na verificação da renderização adequada das informações exibidas no display OLED durante as etapas de configuração e execução do sistema. Também observados, em particular, a apresentação dos valores de BPM, do número de *beats* por loop e a indicação do *preset* ativo. Para esse teste, foram configurados 20 *presets* distintos, o que corresponde ao número máximo suportado pelo protótipo. Após a configuração, o metrônomo foi colocado em operação contínua e foi realizada a transição entre todos os *presets*.

Os resultados obtidos demonstraram que o display OLED apresentou leitura clara, estável e livre de artefatos visuais, exibindo corretamente todas as informações previstas tanto durante a configuração do número total de *presets* quanto, conforme apresentado na Tabela 4, durante a configuração e a execução de cada *preset* individual. Além disso, o LED RGB indicou de forma consistente todas as alternâncias.

Preset	**Informações exibidas corretamente durante a configuração**	**Número do *preset*, BPM e *beats* exibidos corretamente durante execução**	**LED RGB ficou vermelho na mudança para o próximo *preset***
1	Sim	Sim	Sim
2	Sim	Sim	Sim
3	Sim	Sim	Sim
4	Sim	Sim	Sim
5	Sim	Sim	Sim
6	Sim	Sim	Sim
7	Sim	Sim	Sim
8	Sim	Sim	Sim
9	Sim	Sim	Sim
10	Sim	Sim	Sim
11	Sim	Sim	Sim
12	Sim	Sim	Sim
13	Sim	Sim	Sim
14	Sim	Sim	Sim
15	Sim	Sim	Sim
16	Sim	Sim	Sim
17	Sim	Sim	Sim
18	Sim	Sim	Sim
19	Sim	Sim	Sim
20	Sim	Sim	Sim

Tabela 4: Verificação do display OLED e do LED RGB por *preset* (elaboração própria, 2025).

5.2.3. Teste da Matriz de LEDs e do LED RGB Como Indicador de Contratempo

O objetivo deste teste foi validar o funcionamento da matriz de LEDs e do LED RGB como elementos de *feedback* visual rítmico.

Para a realização do teste, foram configurados três *presets* distintos, com valores de 60 BPM, 120 BPM e 180 BPM, respectivamente. Todos os *presets* foram ajustados para conter 9 *beats* por loop, permitindo a avaliação da exibição de todos os números suportados pelo protótipo. Durante o intervalo entre *beats* consecutivos (por exemplo, entre os *beats* 1 e 2), o LED RGB deveria piscar em azul. Para cada *preset*, foi observado um loop completo de batidas.

Os resultados obtidos, apresentados na Tabela 5, indicaram a exibição correta dos *beats* na matriz de LEDs, bem como a temporização adequada do LED RGB.

Preset	**Exibição correta dos 9 *beats* na matriz de LEDs**	**LED RGB piscou em azul entre os *beats***
1	Sim	Sim
2	Sim	Sim
3	Sim	Sim

Tabela 5: Avaliação do *feedback* visual rítmico (elaboração própria, 2025).

5.2.4. Testes de Sincronização da Matriz de LEDs com Clique Sonoro

O objetivo deste teste foi validar a sincronização entre a exibição dos *beats* na matriz de LEDs e o sinal sonoro emitido pelos buzzers do sistema.

Para a realização do teste, foram configurados três *presets* distintos, com valores de 75 BPM, 131 BPM e 222 BPM, respectivamente. Todos os *presets* foram ajustados para conter 9 *beats* por loop, permitindo a avaliação da exibição de todos os números suportados pelo protótipo. Para cada *preset*, a execução foi acompanhada por um período de um minuto, durante o qual foram realizadas a audição dos sinais sonoros e a observação da exibição visual.

Em nenhum dos casos avaliados foram notados descompassos entre os cliques sonoros e a exibição dos *beats* na matriz de LEDs. Os resultados apresentados na Tabela 6 evidenciam a sincronização precisa entre os elementos visuais e sonoros.

Preset	**Tempo de observação (segundos)**	***Beats* visuais observados**	***Beats* sonoros escutados**	**Houve descompasso visual/sonoro**
1	60	9	9	Não
2	60	9	9	Não
3	60	9	9	Não

Tabela 6: Avaliação da sincronização entre matriz de LEDs e sinal sonoro (elaboração própria, 2025).

5.2.5. Testes dos Buzzers

O objetivo deste teste foi verificar a eficiência do *feedback* sonoro fornecido pelo sistema, considerando a intensidade do sinal emitido, sua duração e a diferenciação entre o *beat* principal e os demais *beats* do loop.

Os ensaios consistiram na validação auditiva da intensidade sonora e da duração do bip emitido, bem como da correta diferenciação entre o *beat* principal e os demais *beats*, realizada por meio da utilização de frequências distintas. Para a execução do teste, foram configurados dois *presets*: o primeiro com 80 BPM e cinco *beats*, e o segundo com 150 BPM e quatro *beats*. Para cada *preset*, a audição e observação foram realizadas durante um período de um minuto.

Os resultados obtidos indicaram que o *feedback* sonoro apresentou intensidade adequada, duração consistente e clara distinção entre os diferentes *beats*, conforme sintetizado na Tabela 7.

Preset	**Intensidade sonora perceptível**	**Duração do bip adequada**	**Diferenciação entre *beat* principal e demais *beats***
1	Sim	Sim	Sim
2	Sim	Sim	Sim

Tabela 7: Avaliação do *feedback* sonoro do metrônomo (elaboração própria, 2025).

5.2.6. Teste do Eixo Y do Joystick

O objetivo deste teste foi avaliar a confiabilidade do eixo Y do joystick, utilizado como interface de entrada para a alteração de valores durante o processo de configuração.

Para a realização do teste, durante as etapas de configuração do número total de *presets*, do valor de BPM de cada *preset* e do número de *beats* por *preset*, o eixo Y do joystick foi deslocado gradualmente até sua posição máxima, permitindo alcançar o valor máximo de cada parâmetro configurável. Em seguida, o eixo foi deslocado até a posição mínima, possibilitando a navegação até o menor valor permitido em cada configuração.

Os resultados obtidos demonstraram que o joystick permitiu a varredura completa de todos os valores possíveis, sem a ocorrência de saltos, além de possibilitar a seleção precisa de qualquer valor intermediário. Esses resultados são apresentados na Tabela 8.

Parâmetro configurado	**Valor mínimo alcançado**	**Valor máximo alcançado**	**Percorreu todos os valores sem saltos**	**Permitiu parada em valores intermediários**
Número de *presets*	Sim	Sim	Sim	Sim
BPM do *preset*	Sim	Sim	Sim	Sim
Beats do *preset*	Sim	Sim	Sim	Sim

Tabela 8: Avaliação do eixo Y do joystick (elaboração própria, 2025).

5.2.7. Testes dos Botões Pressionáveis

O objetivo deste teste foi avaliar a confiabilidade dos botões pressionáveis do sistema, bem como a eficácia da rotina de *debounce* implementada para mitigar múltiplos acionamentos indesejados durante interações rápidas.

Foram avaliadas as funcionalidades de confirmação de configurações (botão A), navegação entre *presets* (botões A e B) e pausa da execução do metrônomo (botão central do joystick). Para a realização do teste, foram configurados três *presets* com valores de BPM e número de *beats* arbitrários, uma vez que esses parâmetros não influenciam diretamente o comportamento analisado neste ensaio.

Durante a configuração do número total de *presets*, o botão A foi pressionado uma única vez para confirmação, não sendo observadas falhas nessa etapa. Na configuração individual dos *presets*, o botão A deveria ter sido acionado seis vezes ao todo (duas confirmações para cada *preset*). Entretanto, ao configurar o segundo *preset*, o sistema identificou um duplo acionamento durante a confirmação do valor de BPM, ocasionando, de forma não intencional, a confirmação imediata do número de *beats* do *preset*.

Durante a execução do metrônomo, o botão central do joystick foi acionado em cada *preset* para pausar o sistema, não sendo observados erros ou múltiplos acionamentos nessa funcionalidade. Adicionalmente, os botões A e B foram utilizados para alternar entre os *presets*, sendo pressionados quatro vezes cada. Contudo, em ambos os casos, foram identificados acionamentos múltiplos decorrentes de falhas na rotina de *debounce*, resultando em saltos de *presets* (do *preset* 1 diretamente para o *preset* 3, por exemplo), considerando o funcionamento cíclico do sistema, no qual o botão A avança e o botão B retrocede entre os *presets*.

Ao todo, o botão A foi pressionado 10 vezes, o botão B foi pressionado 4 vezes e o botão central do joystick foi pressionado 3 vezes. Os resultados indicaram funcionamento aceitável dos controles para uso geral, porém evidenciaram a necessidade de ajustes adicionais na rotina de *debounce*, a fim de reduzir a ocorrência de detecções múltiplas indesejadas. Os resultados são apresentados na Tabela 9.

Botão	**Número de pressionamentos**	**Número de acionamentos detectados pelo dispositivo**	**Erro relativo (%)**
A	10	12	20
B	4	5	25
Joystick	3	3	0

Tabela 9: Avaliação dos botões pressionáveis e da rotina de *debounce* (elaboração própria, 2025)

5.2.8. Testes com Músicos

O objetivo deste teste foi avaliar a usabilidade prática do metrônomo audiovisual em um contexto real de estudo musical.

O dispositivo foi avaliado por três músicos — um baterista, um guitarrista e um baixista — todos com experiência prévia em prática instrumental e estudo rítmico. Inicialmente, os participantes receberam uma breve explicação sobre o funcionamento do dispositivo e, em seguida, utilizaram o metrônomo em situações reais de estudo individual, de acordo com suas respectivas práticas musicais.

Ao final da utilização, cada participante respondeu a um questionário composto por oito perguntas relacionadas à usabilidade, clareza das informações, precisão temporal e eficiência dos *feedbacks* visual e sonoro. As respostas foram registradas de acordo com a seguinte escala de avaliação: 1 – Muito ruim; 2 – Ruim; 3 – Normal; 4 – Bom; 5 – Muito bom.

De modo geral, os músicos destacaram a boa percepção visual e sonora das batidas, a facilidade de alternância entre *presets* durante a execução, a elevada precisão temporal do sistema, a clareza na exibição dos números na matriz de LEDs e a correta indicação do contratempo por meio do LED RGB. Conforme apresentado na Tabela 10, todos os participantes consideraram o dispositivo plenamente utilizável em situações reais de prática musical, especialmente em contextos de estudo individual.

Questão avaliada	**Guitarrista**	**Baterista**	**Baixista**	**Média**
Clareza do *feedback* sonoro	5	5	5	5
Clareza do *feedback* visual	5	5	5	5
Precisão temporal percebida	5	5	5	5
Facilidade de uso geral	5	4	5	4,7
Facilidade de alternância entre *presets*	5	4	5	4,7
Qualidade dos elementos de interface (botões, joystick e indicadores visuais).	5	5	4	4,7
Utilização em estudo musical real	5	5	5	5

Tabela 10: Avaliação de usabilidade do metrônomo (elaboração própria, 2025)

6. Trabalhos Futuros, Documentação Fotográfica e Considerações Finais

Este capítulo reúne as considerações finais do trabalho, bem como as propostas de melhorias futuras e a documentação fotográfica do protótipo desenvolvido.

6.1. Melhorias Futuras

A avaliação do protótipo evidenciou possibilidades de aprimoramento das funcionalidades e da usabilidade do sistema. Entre elas, destaca-se a implementação de persistência de *presets* em memória não volátil, como *flash* ou EEPROM, assegurando a manutenção das configurações após o desligamento. Propõe-se também a substituição dos botões por um *footswitch*, permitindo a alternância de *presets* e o controle de pausa por meio dos pés durante a execução musical. Adicionalmente, a inclusão de conectividade Bluetooth possibilitaria o uso de fones de ouvido sem fio ou a conexão com sistemas de áudio externos. No aspecto sonoro, o bip atualmente utilizado pode ser aprimorado por meio de técnicas de síntese por PWM com filtragem e ajustes de envelope, visando à obtenção de timbres mais agradáveis e musicalmente adequados. Por fim, sugere-se a ampliação dos modos de visualização, com a criação de padrões gráficos para representar subdivisões rítmicas.

6.2. Documentação Fotográfica do Dispositivo Protótipo

Figura 23: Total de *presets* (elaboração própria, 2025).

Figura 24: BPM do *preset* (elaboração própria, 2025).

Figura 25: *Beats* do *preset* (elaboração própria, 2025).

Figura 26: Metrônomo em execução. LED RGB indica contratempo (elaboração própria, 2025).

Figura 27: LED RGB indica mudança de *preset* (elaboração própria, 2025).

6.3. Considerações Finais

O metrônomo audiovisual desenvolvido demonstrou desempenho sólido e confiável, atendendo plenamente aos objetivos estabelecidos no projeto. Os testes técnicos evidenciaram sincronização precisa entre os elementos visuais (matriz de LEDs), sonoros (buzzers A e B) e informativos (display OLED), resultando em um dispositivo de boa usabilidade e percepção rítmica clara, com variação imperceptível e, portanto, desprezível no intervalo entre batidas.

A presença de *presets* programáveis, aliada à possibilidade de personalizar valores de BPM e *beats* por loop, mostrou-se especialmente útil, tornando o sistema adaptável a diferentes contextos de estudo musical. A exibição dos números de batida na matriz de LEDs e a indicação correta do contratempo pelo LED RGB contribuíram significativamente para o caráter audiovisual do projeto, facilitando o acompanhamento rítmico por parte dos usuários.

Os testes com músicos confirmaram que o dispositivo é funcional e aplicável em práticas reais de estudo individual.

Apesar dos resultados positivos, foram identificados pontos de melhoria que podem ampliar a utilidade e robustez do sistema. Entre as possibilidades de trabalhos futuros, destacam-se a inclusão de memória persistente para armazenamento dos *presets* entre desligamentos, a adoção de um *footswitch* para operação com os pés durante performances ou práticas instrumentais, e a adição de conectividade Bluetooth, permitindo o uso com fones de ouvido ou caixas de som externas.

Conclui-se que o projeto resultou em um metrônomo digital audiovisual plenamente funcional, constituindo uma base sólida para expansões futuras e representando uma ferramenta útil para músicos e estudantes de música.

REFERÊNCIAS

SABRA. **Tipos de ritmo que todo músico iniciante precisa dominar**. SABRA – Sociedade Artística Brasileira, 22 ago. 2025. Disponível em: https://www.sabra.org.br/site/x-tipos-de-ritmo-que-todo-musico-iniciante-precisa-dominar/. Acesso em: 15 dez. 2025.

BRITANNICA EDITORS. **Metronome**. Encyclopedia Britannica, 5 mar. 2025a. Disponível em: https://www.britannica.com/art/metronome. Acesso em: 6 out. 2025.

SOUZA, Fábio. **O que são sistemas embarcados?**. *Embarcados*, 2022. Disponível em: https://embarcados.com.br/o-que-sao-sistemas-embarcados/. Acesso em: 13 dez. 2025.

BITDOGLAB. **BitDogLab**. GitHub repository. Disponível em: https://github.com/BitDogLab/BitDogLab?tab=readme-ov-file. Acesso em: 5 out. 2025.

CROSSLEY-HOLLAND, Peter. **Rhythm**. *Encyclopedia Britannica*, 9 ago. 2025. Disponível em: https://www.britannica.com/art/rhythm-music. Acesso em: 13 dez. 2025.

REPP, Bruno H. **Sensorimotor synchronization: a review of the tapping literature**. Psychonomic Bulletin & Review, v. 12, n. 6, p. 969–992, 2005.

IYER, Vijay. **Embodied mind, situated cognition, and expressive microtiming in African-American music**. Music Perception, v. 19, n. 3, p. 387–414, 2002.

MANASSE, Robert. **Johann Nepomuk Maelzel (1772–1838)**. Londres, 2022. Museum of Music History. Disponível em: https://momh.org.uk/exhibitions/johann-nepomuk-maelzel-1772-1838/. Acesso em: 13 dez. 2025.

SOUZA, Wyron Roberth Gomes de. **A utilização do metrônomo em aulas de ensino coletivo de violão para iniciantes**. 2018. Trabalho de Conclusão de Curso (Licenciatura em Educação Musical) – Universidade Federal de Alagoas, Instituto de Ciências Humanas, Comunicação e Arte, Maceió, 2018.

GEIGER, André. **Metrônomo e aplicativos de metrônomo: novas perspectivas na prática musical**. 2022. Trabalho de Conclusão de Curso (Licenciatura em Música) – Universidade Federal da Bahia, Salvador, 2022.

PRELUDE MUSICAL. **Metrônomo Wittner Taktell 836 Piccolo – Black**. Disponível em: https://preludemusical.com.br/produto/metronomo-wittner-taktell-836-piccolo-metronome-black/. Acesso em: 3 nov. 2025.

AMAZON. **Seiko Metrônomo de quartzo SQ50-V**. Disponível em: https://www.amazon.com.br/Seiko-Metr%C3%B4nomo-de-quartzo-SQ50-V/dp/B000LFCXL8. Acesso em: 1 dez. 2025.

BOSS. **Dr. Beat DB-90**. Disponível em: https://www.boss.info/br/products/db-90/. Acesso em: 3 nov. 2025.

KORG. **Metrônomo Digital MA-1 (BLBK)**. Disponível em: https://www.korg.com.br/produto/10270174-metronomo-digital-korg-ma-1-blbk. Acesso em: 5 nov. 2025.

SOUNDBRENNER. **Pulse Vibrating Metronome**. Disponível em: https://www.soundbrenner.com/products/pulse-vibrating-metronome. Acesso em: 6 nov. 2025.

LAPENTA, Olivia Morgan; KELLER, Peter E.; NOZARADAN, Sylvie; VARLET, Manuel. **Spatial and temporal (non)binding of audiovisual rhythms in sensorimotor synchronisation**. *Experimental Brain Research*, v. 241, 2023.

PETERSON ELECTRO-MUSICAL. **StroboPLUS HD**. Disponível em: https://www.petersontuners.com/products/stroboPLUSHDC/. Acesso em: 3 nov. 2025.

CIFRA CLUB. **Metrônomo – Aplicativo móvel**. Disponível em: https://www.cifraclub.com.br. Acesso em: 6 nov. 2025.

MUSICRADAR. **Walrus Audio Canvas Rehearsal – Review**. Disponível em: https://www.musicradar.com/guitars/guitar-pedals/walrus-audio-canvas-rehearsal-review. Acesso em: 6 nov. 2025.

BRAVENESS23. **VisualMetronome**. 2024. GitHub repository. Disponível em: https://github.com/braveness23/VisualMetronome. Acesso em: 13 dez. 2025.

UMADI, Ravi. **Digital-Metronome**. 2023. GitHub repository. Disponível em: https://github.com/raviumadi/Digital-Metronome. Acesso em: 13 dez. 2025.

ABIDIN, Idriz Zainal. **Pico WS2812 Metronome Example**. GitHub repository. Disponível em: https://gist.github.com/idriszmy/60e6e0641001dae274e4a133f0c126d4. Acesso em: 30 out. 2025.

PINTO NETO, Carlos Alberto da Costa. **Raspberry Pico WS2812 Controller**. GitHub repository. Disponível em: https://github.com/caneto/raspberry-pico-ws2812. Acesso em: 2 nov. 2025.

EMBARCATECH. **Embarcatech**. 2025. Disponível em: https://embarcatech.cepedi.org.br/. Acesso em: 13 dez. 2025.

RASPBERRY PI FOUNDATION. **RP2040: A microcontroller by Raspberry Pi**. Disponível em: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf. Acesso em: 2 out. 2025.

MEDEIROS, Anna Carolinne Albuquerque de. **Desenvolvimento de um sistema IoT de irrigação automatizada com Raspberry Pi Pico W**. 2025. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, Natal, 2025.