


Relatório do Projeto: Cronômetro Binário Interativo com Feedback Sonoro

Nome do arquivo: **U7T_RNSRibeiro.pdf**

Autor: **Rodrigo Nunes Sampaio Ribeiro**

Link do video do projeto: <https://www.youtube.com/watch?v=kfQC4YPR4o0>

 BinaryClock - Embarcotech - BitDogLab

Código do projeto: <https://github.com/rnsribeiro/BinaryClock-BitDogLab>

1. Título

Cronômetro Binário com Representação em Matriz de LEDs

2. Objetivos

Desenvolver um sistema embarcado baseado na Raspberry Pi Pico W (BitDogLab) que funcione como um cronômetro digital controlado por botões, exibindo o tempo decorrido em um display OLED (formato decimal) HH:MM:SS e em uma matriz de LEDs 5x5 (formato binário), com feedback sonoro via buzzer para interação do usuário. O sistema deve iniciar e parar a contagem com um botão "Start/Stop", zerar com um botão "Reset", um buzzer emitirá um som sempre que um botão for pressionado e oferecer uma representação visual inovadora do tempo por meio de LEDs binários.

3. Justificativa

Este projeto integra conceitos fundamentais de sistemas embarcados, como temporização, controle de GPIOs, comunicação I2C, PWM e manipulação de LEDs, aplicados de forma criativa, combinando manipulação de hardware e exibição de dados em diferentes formatos. A combinação de exibição decimal e binária, somada ao feedback sonoro, torna o sistema educativo e funcional, podendo ser utilizado como uma ferramenta de gerenciamento de tempo para estudos ou atividades produtivas. O cronômetro, em particular, permite ao usuário contar o tempo dedicado a atividades como trabalho ou estudos, oferecendo uma interface visual única e prática para medir períodos de tempo com precisão.

4. Requisitos e Funcionamento

● Requisitos:

- Iniciar/parar a contagem de tempo ao pressionar o botão "Start/Stop".
- Zerar o cronômetro ao pressionar o botão "Reset".
- Exibir o tempo em horas, minutos e segundos no display OLED (ex:"01:25:56").
- Representar os segundos em formato binário na matriz de LEDs 5x5.
- Emitir um bipe no buzzer a cada pressionamento de botão.

- **Funcionamento**

- O sistema inicia com o cronômetro zerado. Ao pressionar "Start/Stop", a contagem começa, atualizando o display OLED e a matriz de LEDs em tempo real, com um bip confirmando a ação. Um novo pressionamento de "Start/Stop" pausa a contagem, novamente com bip. O botão "Reset" zera o tempo e emite um bipe, preparando o sistema para uma nova medição.

5. Originalidade

A originalidade está na combinação de duas formas de visualização do tempo (decimal no OLED e binária nos LEDs) em um único sistema embarcado, com interação sonora para feedback. Embora cronômetros e relógios binários existam, a integração desses elementos em uma Raspberry Pi Pico W com controle simplificado por botões adiciona um grau de inovação pedagógica e prática. Projetos existentes focam em relógios ou displays genéricos, não em cronômetros com exibição dupla.

5.1 Projetos correlatos:

- [Stopwatch Using Raspberry Pi Pico](#): Implementa um cronômetro com display TM1637 de 4 dígitos 7 segmentos, controlado por botões. Não utiliza matriz de LEDs para exibição binária, limitando-se a display decimal.
- [LED-Matrix-Clock](#): Utiliza Raspberry Pi Pico com uma matriz de 4 segmentos 8x8 controlada por MAX7219 para mostrar hora, temperatura e umidade. Possui botões para mudar modos, mas é um relógio, não um cronômetro, e não inclui display decimal separado nem buzzer.
- [Striking Binary Clock](#): Relógio binário com LEDs RGB individuais, não uma matriz, e focado em exibir hora atual, não tempo decorrido. Não menciona buzzer ou display decimal.
- [Pico Binary Clock](#): Relógio binário em painel LED, usando Raspberry Pi Pico e módulo RTC. É um relógio, não cronômetro, e não inclui display decimal ou buzzer.
- [Electronic Clock for Raspberry Pi Pico](#): Relógio digital com LED digit display, buzzer e botões, mas é um relógio, não um cronômetro, e não usa matriz de LEDs.

5.2 Justificativa de Originalidade

O projeto proposto é original por combinar os seguintes elementos de forma única:

- **Cronômetro funcional**: Diferente de relógios, mede tempo decorrido, iniciado e parado por botões, algo presente em poucos projetos, como o "Stopwatch Using Raspberry Pi Pico", mas sem matriz de LEDs binária.
- **Exibição dupla**: Utiliza display decimal (ex.: OLED) e matriz de LEDs 5x5 para formato binário, algo não observado em nenhum projeto analisado. Projetos como [LED-Matrix-Clock](#) mostram tempo em matriz, mas sem display decimal separado.
- **Feedback sonoro**: Inclui buzzer para feedback ao pressionar botões, ausente em quase todos os projetos, exceto em produtos comerciais como o "Electronic Clock for Raspberry Pi Pico", que não é um cronômetro.

- **Plataforma específica:** Usa Raspberry Pi Pico W RP2040 BitDogLab, alinhando-se a projetos como [Striking Binary Clock](#), mas com foco em cronômetro, não relógio.

5.3 Conclusão

Após análise, conclui-se que o projeto é original devido à combinação única de funcionalidades não encontradas em projetos correlatos. Ele atende aos critérios de autoria e inovação, sendo uma contribuição significativa no campo de sistemas embarcados com Raspberry Pi Pico.

6. Hardware

6.1 Diagrama de blocos

Este diagrama reflete o fluxo de dados: os botões enviam entrada para o Microcontrolador Raspberry Pi Pico W RP2040, que processa e envia saída para o display OLED (vía I2C), matriz de LEDs 5x5 (via GPIOs) e buzzers (via GPIOs com PWM).

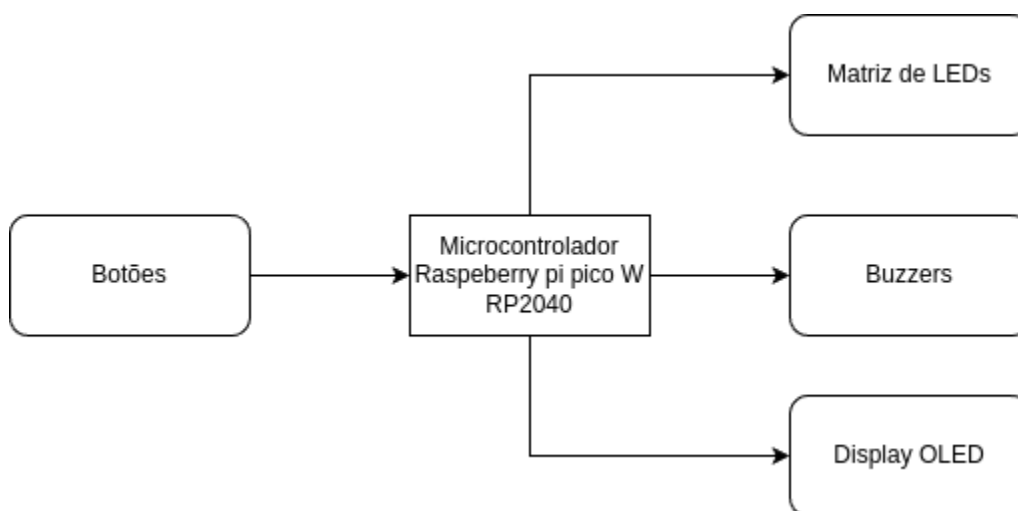


Figura 1: Diagrama de blocos mostrando a esquerda um bloco referente aos botões como entrada, microcontrolador no centro e a direita os blocos representando a matriz de leds, buzzers e o display OLED.

Fonte: Autor, criado em 20/02/2025.

6.2 Configuração de Cada Bloco

- **Raspberry Pi Pico W:** Configurado com clock padrão, inicializando GPIOs como entrada (botões) ou saída (matriz, buzzers), I2C para OLED e PWM para buzzers.
- **Display OLED:** Configurado para comunicação I2C a 100kHz, inicializado com endereço padrão (ex.: 0x3C para SSD1306), exibindo texto em formato decimal.
- **Matriz de LEDs 5x5:** Linhas e colunas configuradas como saída via GPIOs (assumindo matriz de cátodo comum, com varredura para acender LEDs correspondentes aos bits binários do tempo).

- **Botões:** Configurados como entrada com pull-up interno, detectando nível baixo (0V) em pressionamento, com debounce básico via software (delay de 200ms).
- **Buzzers:** Configurados como saída PWM (GPIOs diferentes), gerando tons de 1kHz e 500Hz, respectivamente, com duty cycle de 50% para duração de 100ms por bipe.

6.3 Especificações

As especificações técnicas atendem aos requisitos do usuário:

- **Precisão de tempo:** O Pico W possui temporizador preciso, garantindo contagem em milissegundos, essencial para o cronômetro.
- **Legibilidade do display:** OLED com resolução suficiente (ex.: 128x64) para mostrar "HH:MM:SS" claramente.
- **Resposta dos botões:** GPIOs com pull-up interno e debounce garantem acionamentos confiáveis, evitando múltiplos triggers.
- **Matriz de LEDs:** 5x5 permite representar até 25 bits, mais que suficiente para segundos (12 bits para 3600s, ou 1 hora).
- **Buzzers:** PWM permite tons distintos, facilitando feedback auditivo para ações diferentes.

6.4 Lista de Materiais

Componente	Modelo/Descrição	Quantidade	Especificações
Microcontrolador	Raspberry Pi Pico W (RP2040)	1	Processador dual-core ARM Cortex-M0+, 264KB SRAM, 2MB flash, 28 GPIOs, interfaces I2C, SPI, UART
Botões	Chave Tátil Push Button Botão 12x12x7.5 Mm	2	Botões táteis
Display	OLED SSD1306	1	128x64 pixels, 0.96", interface I2C
Matriz de LEDs	5x5 WS2812B RGB LEDs	1	LEDs individuais, comunicação via entrada de dados única, controle por PWM
Buzzers	buzzers passivos	2	Frequência típica 2-4 kHz, interface digital de saída

Tabela 1: Lista de Materiais especificando os componentes, modelo, descrição, quantidade e especificações.

6.5 Descrição da Pinagem Usada

Componente	Função	Pinos do Pico W
Display OLED	I2C (SDA)	GP14
Display OLED	I2C (SCL)	GP15
Matriz de LEDs 5x5	PWM	GP7
Botão A	Entrada (Start/Stop)	GP5
Botão B	Entrada (Reset)	GP6
Buzzer A	Saída (PWM)	GP21
Buzzer B	Saída (PWM)	GP10

Tabela 2: Descrição da pinagem usada apresentando informações sobre os componentes, função e os pinos utilizados.

6.6 Circuito Completo do Hardware

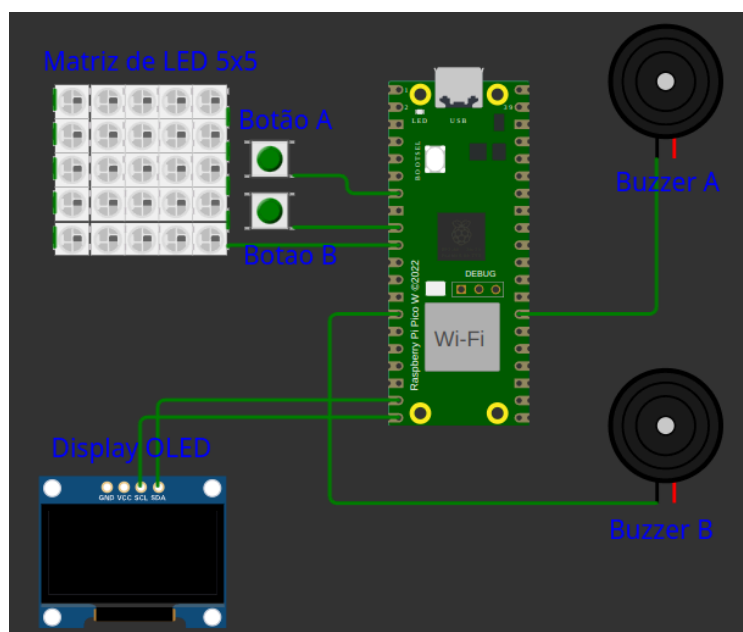


Figura 2: Circuito esquemático mostrando os componentes e as conexões na placa raspberry pi pico w.
Fonte: Autor criado em 20/02/2025

O circuito completo é pré-definido na placa BitDogLab, com as conexões descritas acima. Esquemáticamente, o Raspberry Pi Pico W está conectado ao OLED via I2C (GP14 e GP15), à matriz de LEDs 5x5 via GPIOs (GP7), aos botões via GP5 e GP6 (entradas com

pull-up interno), e aos buzzers via GP21 e GP10 (saídas PWM). A placa trata resistores e pull-ups internamente, eliminando a necessidade de componentes adicionais.

7. Software

7.1 Blocos Funcionais

Diagrama das Camadas do Software

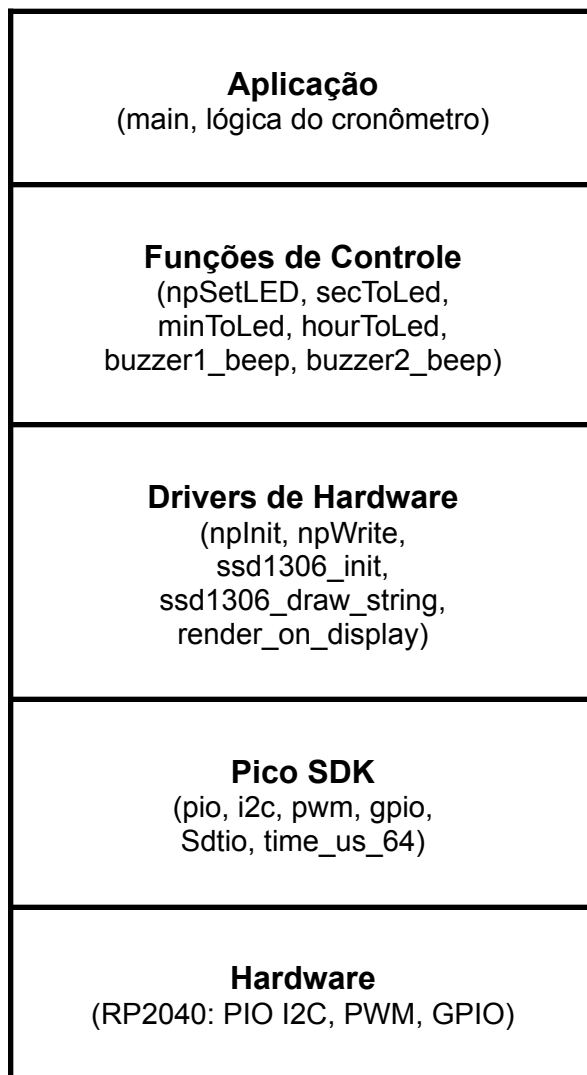


Tabela 3: Representação em blocos das camadas de software e suas funções.

Fonte: Autor criado em 20/05/2025

- **Camada de Aplicação:** Contém a lógica principal do cronômetro (**main**), gerenciando estados e interações do usuário.

- **Funções de Controle:** Funções específicas para configurar LEDs, buzzers e exibir informações (ex.: **secToLed**, **buzzer1_beep**).
- **Drivers de Hardware:** Intermediários entre o software e o hardware, inicializando e controlando periféricos (ex.: **nplnit**, **ssd1306_draw_string**).
- **Pico SDK:** Bibliotecas padrão do RP2040 para acesso a PIO, I2C, PWM, GPIO e tempo.
- **Hardware:** Os periféricos físicos do RP2040 na placa BitDogLab.

7.2 Descrição das Funcionalidades

- **Camada de Aplicação (main):**
 - Gerencia o fluxo do cronômetro, incluindo estados (rodando, pausado, reset), entrada dos botões, e chamadas às funções de controle.
- **Funções de Controle:**
 - **npSetLED:** Define a cor RGB de um LED específico no buffer.
 - **secToLed:** Converte segundos em binário e exibe em LEDs verdes.
 - **minToLed:** Converte minutos em binário e exibe em LEDs azuis.
 - **hourToLed:** Converte horas em binário e exibe em LEDs vermelhos.
 - **buzzer1_beep:** Gera um som de 1000 Hz no buzzer 1 por uma duração específica.
 - **buzzer2_beep:** Gera um som de 500 Hz no buzzer 2 por uma duração específica.
- **Drivers de Hardware:**
 - **nplnit:** Configura o PIO para controlar os LEDs WS2812.
 - **npWrite:** Envia o buffer de LEDs para a matriz física.
 - **ssd1306_init:** Inicializa o display OLED via I2C.
 - **ssd1306_draw_string:** Desenha uma string no buffer do OLED.
 - **render_on_display:** Atualiza o OLED com o conteúdo do buffer.
- **Pico SDK:**
 - Fornece funções de baixo nível para PIO (**pio_add_program**), I2C (**i2c_init**), PWM (**pwm_set_wrap**), GPIO (**gpio_init**), e temporização (**time_us_64**).

7.3 Definição das Principais Variáveis

- **int second, minute, hour:** Contadores de tempo (segundos, minutos, horas), armazenam o estado atual do cronômetro (0-59 para segundos/minutos, 0-31 para horas).
- **uint64_t last_time:** Último tempo registrado em milissegundos (via **time_us_64**), usado para contagem precisa de 1 segundo.
- **uint64_t last_toggle_time:** Último tempo de alternância do texto no OLED no estado pausado, controla a exibição alternada a cada 1 segundo.
- **bool is_running:** Estado do cronômetro (**false** = parado, **true** = rodando).
- **bool display_toggle:** Controla a alternância entre tempo pausado e mensagem no OLED (**true** = tempo, **false** = mensagem).

- **bool is_reset_prompt**: Indica se está no modo de confirmação de reset (**true** = esperando confirmação, **false** = estado normal).
- **npLED_t leds[LED_COUNT]**: Buffer global para os 25 LEDs WS2812, armazena os valores RGB de cada LED.
- **uint8_t ssd[ssd1306_buffer_length]**: Buffer do OLED (1024 bytes para 128x64 pixels), armazena os dados a serem exibidos.
- **PIO np_pio; uint sm**: Variáveis para controle do PIO (instância e máquina de estado) usado pelos LEDs WS2812.

7.5 Inicialização

O processo de inicialização ocorre na função **main** antes do loop principal:

1. Inicialização da Comunicação Serial:

- **stdio_init_all()**: Configura a comunicação USB para saída serial, útil para debug (embora não usada diretamente no código).

2. Inicialização do I2C para o OLED:

- **i2c_init(i2c1, ssd1306_i2c_clock * 1000)**: Configura o I2C1 com clock de 400 kHz (padrão ajustado pela constante **ssd1306_i2c_clock**).
- **gpio_set_function(I2C_SDA, GPIO_FUNC_I2C)** e **gpio_set_function(I2C_SCL, GPIO_FUNC_I2C)**: Define GP14 (SDA) e GP15 (SCL) como pinos I2C.
- **gpio_pull_up(I2C_SDA)** e **gpio_pull_up(I2C_SCL)**: Ativa pull-ups internos para os pinos I2C.
- **ssd1306_init()**: Envia comandos de inicialização ao OLED via I2C (definidos em **ssd1306_i2c.c**).

3. Configuração da Área de Renderização do OLED:

- **struct render_area frame_area**: Define a área total do OLED (128x64, 8 páginas).
- **calculate_render_area_buffer_length(&frame_area)**: Calcula o tamanho do buffer (1024 bytes).
- **uint8_t ssd[ssd1306_buffer_length]**: Aloca o buffer e o zera com **memset**.

4. Inicialização dos LEDs WS2812:

- **npInit(LED_PIN)**: Configura o PIO em GP7 para controlar a matriz WS2812, carregando o programa **ws2818b.pio** e limpando os LEDs.

5. Configuração dos Botões:

- **gpio_init(BUTTON_A)** e **gpio_init(BUTTON_B)**: Inicializa GP5 e GP6.
- **gpio_set_dir(BUTTON_A, GPIO_IN)** e **gpio_set_dir(BUTTON_B, GPIO_IN)**: Define como entradas.
- **gpio_pull_up(BUTTON_A)** e **gpio_pull_up(BUTTON_B)**: Ativa pull-ups internos.

6. Configuração dos Buzzers:

- **gpio_set_function(BUZZER1_PIN, GPIO_FUNC_PWM)** e **gpio_set_function(BUZZER2_PIN, GPIO_FUNC_PWM)**: Define GP21 e GP10 como saídas PWM.
- **pwm_set_wrap** e **pwm_set_clkdiv**: Configuram frequências de 1000 Hz (buzzer 1) e 500 Hz (buzzer 2).
- **pwm_set_enabled**: Ativa os canais PWM.

7.6 Configuração dos Registros

Funções de Configurações dos Registros

- **I2C (OLED):**
 - **i2c_init(i2c1, ...)**: Configura registradores do I2C1 (ex.: baudrate) para 400 kHz.
 - **ssd1306_init()**: Envia comandos ao OLED via I2C (ex.: **ssd1306_set_display**, **ssd1306_set_mux_ratio**), ajustando registradores internos do SSD1306 (definidos em `ssd1306_i2c.c`).
- **PIO (LEDs):**
 - **pio_add_program**: Carrega o programa PIO (**ws2818b.pio**) nos registradores do PIO0 ou PIO1.
 - **ws2818b_program_init**: Configura registradores do PIO (ex.: pino de saída GP7, clock) para controlar os LEDs WS2812 a 800 kHz.
- **GPIO (Botões):**
 - **gpio_set_function**: Configura registradores de função dos pinos GP14/GP15 (I2C) e GP5/GP6 (entrada).
 - **gpio_pull_up**: Ativa pull-ups nos registradores de controle dos pinos GP5 e GP6.
- **PWM (Buzzers):**
 - **gpio_set_function**: Define GP21 e GP10 como saídas PWM nos registradores de função.
 - **pwm_set_wrap**: Define o valor de wrap nos registradores do PWM para ajustar a frequência (2499 para 1000 Hz, 4999 para 500 Hz).
 - **pwm_set_clkdiv**: Ajusta o divisor de clock nos registradores PWM para atingir as frequências desejadas.
 - **pwm_set_enabled**: Ativa os canais PWM nos registradores correspondentes.

7.7 Estrutura e Formato dos Dados

Dados Específicos Usados no Software

- **npLED_t leds[LED_COUNT]:**
 - Estrutura: Array de 25 elementos do tipo **struct pixel_t** (3 bytes por elemento: G, R, B).
 - Formato: Cada elemento contém valores de 0 a 255 para verde (G), vermelho (R) e azul (B).
 - Tamanho: $25 \times 3 = 75$ bytes.
- **uint8_t ssd[ssd1306_buffer_length]:**

- Estrutura: Array de 1024 bytes (128 colunas × 8 páginas).
- Formato: Cada byte representa 8 pixels verticais (1 bit por pixel, 0 = apagado, 1 = aceso).
- Tamanho: 1024 bytes.
- **char time_str[9]:**
 - Estrutura: String de 9 caracteres (incluindo terminador nulo \0).
 - Formato: "HH:MM:SS\0" (ex.: "01:23:45"), onde cada par de dígitos é formatado com zeros à esquerda via **sprintf**.
 - Tamanho: 9 bytes.

7.8 Organização da Memória

Descrição dos Endereços de Memória Utilizados

- **Memória SRAM (RP2040 tem 264 KB):**
 - **leds[LED_COUNT]:** 75 bytes na SRAM para o buffer dos LEDs. Endereço exato depende da alocação pelo compilador (variável global).
 - **ssd[ssd1306_buffer_length]:** 1024 bytes na SRAM para o buffer do OLED. Alocado na pilha ou como global, dependendo do compilador.
 - **time_str[9]:** 9 bytes na pilha dentro de **main**.
 - Variáveis de estado: **second**, **minute**, **hour** ($3 \times 4 = 12$ bytes), **last_time**, **last_toggle_time** ($2 \times 8 = 16$ bytes), **is_running**, **display_toggle**, **is_reset_prompt** ($3 \times 1 = 3$ bytes), totalizando ~31 bytes na pilha ou SRAM.
- **Memória Flash (2 MB):**
 - Código do programa (incluindo funções como **main**, **npSetLED**, etc.) e constantes (ex.: **positions[]** em **secToLed**) são armazenados na Flash.
 - Programa PIO (**ws2818b.pio**) é carregado na Flash e copiado para o PIO durante **nplnit**.
- **Registradores do RP2040:**
 - GPIO: Configurações dos pinos (ex.: GP5, GP6, GP7, GP14, GP15, GP21, GP10) em endereços como **0x40014000** (base do GPIO).
 - I2C1: Registradores em **0x40048000** (base do I2C1).
 - PIO0/PIO1: Registradores em **0x50200000** (PIO0) ou **0x50300000** (PIO1).
 - PWM: Registradores em **0x40050000** (base do PWM).

7.9 Protocolo de Comunicação

Descrição do Protocolo

- **I2C (OLED):**
 - Protocolo: I2C com endereço 0x3C (definido em **ssd1306_i2c.h**), clock de 400 kHz.
 - Uso: Comunicação mestre-escravo entre o RP2040 (mestre) e o OLED SSD1306 (escravo).
 - Funções: **i2c_write_blocking** envia comandos e dados ao OLED.
- **PIO (LEDs WS2812):**

- Protocolo: Protocolo proprietário WS2812, implementado via PIO em **ws2818b.pio**.
- Uso: Comunicação unidirecional do RP2040 (GP7) para os LEDs, enviando pulsos de 800 kHz (codificados como 0 ou 1 por duração).
- Funções: **npWrite** envia o buffer RGB via PIO.
- **Sem Comunicação Externa:** Não há protocolo de comunicação com dispositivos externos (ex.: USB ou Wi-Fi) neste código.

8. Metodologia

Descrição da Execução das Etapas do Projeto

O desenvolvimento do "Cronômetro com BitDogLab" seguiu uma metodologia estruturada típica de projetos em sistemas embarcados, com etapas que envolveram pesquisa, seleção de hardware, definição de funcionalidades do software, configuração da IDE, programação e depuração. Abaixo, cada etapa é detalhada com base na construção do projeto.

1. Pesquisas Realizadas

- **Objetivo Inicial:** O projeto começou com a ideia de criar um cronômetro interativo utilizando a placa BitDogLab, aproveitando seus periféricos (botões, LEDs, OLED, buzzers).
- **Fontes Consultadas:**
 - i. Documentação do Raspberry Pi Pico SDK (versão 1.5.1) para entender o uso de PIO, I2C, PWM e GPIO.
 - ii. Datasheets do OLED SSD1306 e LEDs WS2812 para compreender os protocolos I2C e de temporização, respectivamente.
 - iii. Exemplos de código do Pico SDK (**pico-examples**) para controle de LEDs WS2812 via PIO e comunicação I2C.
 - iv. Tutoriais online sobre programação embarcada com RP2040 (ex.: Raspberry Pi Foundation e fóruns da comunidade).
- **Resultados:** Identificação do suporte da placa BitDogLab para botões (GP5, GP6), LEDs WS2812 (GP7), OLED (GP14, GP15), e buzzers (GP21, GP10), além das bibliotecas necessárias.

2. Escolha do Hardware

- **Placa:** A BitDogLab, baseada no Raspberry Pi Pico W RP2040, foi escolhida por sua versatilidade e recursos embarcados:
 - i. **Microcontrolador RP2040:** 264 KB SRAM, 2 MB Flash, suporte a PIO, I2C, PWM e GPIO.
 - ii. **Matriz de LEDs WS2812 (5x5):** Ideal para exibição binária de tempo (17 LEDs usados: 6 para segundos, 6 para minutos, 5 para horas).
 - iii. **Display OLED SSD1306 (128x64):** Compatível com I2C, perfeito para exibir "HH:MM:SS".
 - iv. **Botões:** Dois botões físicos (A e B) para controle interativo.

- v. **Buzzers:** Dois buzzers para feedback sonoro distinto.
- o **Justificativa:** A placa já possuía todos os periféricos necessários, eliminando a necessidade de hardware adicional, e o RP2040 oferecia desempenho suficiente para o cronômetro.

3. Definição das Funcionalidades do Software

- o **Requisitos Iniciais:**
 - i. Iniciar/pausar o cronômetro com um botão.
 - ii. Exibir o tempo no OLED em "HH:MM:SS".
 - iii. Mostrar segundos, minutos e horas em binário nos LEDs.
- o **Evolução:**
 - i. Adição de reset com botão B, mantendo LEDs acesos na pausa e apagando ao zerar.
 - ii. Feedback sonoro com buzzers (1000 Hz para botão A, 500 Hz para botão B).
 - iii. Alternância de texto no OLED ("Paused" e tempo pausado).
- o **Decisões:**
 - i. Uso de PIO para LEDs WS2812 devido à precisão de temporização.
 - ii. I2C para o OLED por ser um protocolo padrão suportado.
 - iii. PWM para buzzers para gerar tons distintos.
- o **Resultado:** Um cronômetro interativo com controle por botões, exibição visual e sonora.

4. Inicialização da IDE

- o **IDE Escolhida:** Visual Studio Code (VS Code) com extensões para sistemas embarcados.
- o **Procedimento Padrão para Projetos Embarcados no VS Code:**
 - i. **Requisitos:**
 - 1. Instalar o Pico SDK (ex.: `/home/rod/pico-sdk`).
 - 2. Instalar ferramentas: CMake, GCC ARM (`arm-none-eabi-gcc`), Ninja (opcional).
 - ii. **Instalação do VS Code:**
 - 1. Baixar e instalar o VS Code a partir do site oficial.
 - iii. **Configuração do Ambiente:**
 - 1. Definir a variável de ambiente **PICO_SDK_PATH** (ex.: `export PICO_SDK_PATH=/home/rod/pico-sdk`).
 - 2. Adicionar ao **PATH** o compilador ARM (ex.: `/usr/bin/arm-none-eabi-gcc`).
 - iv. **Extensões do VS Code:**
 - 1. Instalar "C/C++" (Microsoft) para suporte a IntelliSense.
 - 2. Instalar "CMake Tools" (Microsoft) para integração com CMake.
 - 3. Instalar "Pico-W-Go" ou "Cortex-Debug" (opcional) para suporte ao Pico.
 - v. **Criação do Projeto:**
 - 1. Criar um diretório (ex.: **BinaryClock-BitDogLab**).

2. Adicionar **main.c**, **CMakeLists.txt**, e **pico_sdk_import.cmake** (copiado do SDK).
- vi. **Configuração do CMake:**
 1. Abrir o VS Code no diretório do projeto.
 2. Configurar o CMake via "CMake Tools" (selecionar kit ARM GCC).
 3. Gerar o build com **cmake ..** em um diretório **build**.
- vii. **Aplicação ao Projeto:**
 1. Usei o VS Code com o Pico SDK configurado em **/home/rod/pico-sdk**.
 2. Criei o projeto com **main.c**, bibliotecas OLED (**inc/**), e **ws2818b.pio**.
 3. Configurei o **CMakeLists.txt** para incluir **hardware_pwm**.
- **Programação na IDE**
 - i. **Desenvolvimento:**
 1. **Estrutura Inicial:** Código básico com contagem de tempo e exibição no OLED.
 2. **Iterações:**
 - a. Adição de controle dos LEDs WS2812 via PIO (**nplnit**, **npWrite**, **secToLed**, etc.).
 - b. Implementação de botões A e B (**gpio_get**) com estados (**is_running**, **is_reset_prompt**).
 - c. Integração de buzzers com PWM (**buzzer1_beep**, **buzzer2_beep**).
 - d. Ajuste da alternância de texto no OLED no estado pausado.
 3. Ferramentas do VS Code:
 - a. IntelliSense para autocomplete e verificação de sintaxe.
 - b. Terminal integrado para executar **cmake ..** e **make**.
 - ii. **Abordagem:** Desenvolvimento incremental, testando cada funcionalidade antes de adicionar a próxima.
- **Depuração**
 - i. **Métodos Utilizados:**
 1. **Saída Serial:** **stdio_init_all()** permitiu depuração via USB (embora não usado ativamente, disponível para **printf**).
 2. Testes Manuais:
 - a. Pressionei os botões A e B para verificar estados (**is_running**, **is_reset_prompt**).
 - b. Observei o OLED para confirmar a exibição de "HH:MM:SS" e alternância de texto.
 - c. Verifiquei os LEDs para assegurar a representação binária correta.
 - d. Testei os buzzers para validar os tons (1000 Hz e 500 Hz).
 3. **Compilação e Carregamento:**

- a. Após cada alteração, compilei com **make** no diretório **build** e carreguei o **.uf2** na placa via USB.
- b. Usei o recurso de "drag-and-drop" do Pico para upload rápido.

ii. **Correções:**

1. Inicialmente, esqueci de incluir **hardware_pwm** no **CMakeLists.txt**, corrigido após erro de compilação.
2. Ajustei temporizações (ex.: alternância de 500 ms para 1000 ms) com base em testes visuais.

REFERÊNCIAS

RASPBERRY PI FOUNDATION. Raspberry Pi Pico C/C++ SDK. Versão 1.5.1. [S.l.]: Raspberry Pi Foundation, 2023. Disponível em: <https://www.raspberrypi.com/documentation/pico-sdk/>. Acesso em: 23 fev. 2025.

RASPBERRY PI FOUNDATION. RP2040 Datasheet. [S.l.]: Raspberry Pi Foundation, 2021. Disponível em: <https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf>. Acesso em: 23 fev. 2025.

ADAFRUIT INDUSTRIES. SSD1306 OLED Displays Datasheet. [S.l.]: Adafruit Industries, [s.d.]. Disponível em: <https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf>. Acesso em: 23 fev. 2025.

WORLDSEMI. WS2812B Intelligent Control LED Datasheet. Versão 1.4. [S.l.]: Worldsemi Co., Limited, 2016. Disponível em: <http://www.world-semi.com/DownLoadFile/36>. Acesso em: 23 fev. 2025.

RASPBERRY PI FOUNDATION. Pico Examples: PIO WS2812. [S.l.]: Raspberry Pi Foundation, 2023. Disponível em: <https://github.com/raspberrypi/pico-examples/tree/master/pio/ws2812>. Acesso em: 23 fev. 2025.

RASPBERRY PI FOUNDATION. Pico Examples: I2C SSD1306. [S.l.]: Raspberry Pi Foundation, 2023. Disponível em: <https://github.com/raspberrypi/pico-examples/tree/master/i2c/ssd1306>. Acesso em: 23 fev. 2025.

TOM'S HARDWARE. Stopwatch Using Raspberry Pi Pico. [S.l.]: Tom's Hardware, 2022. Disponível em: <https://www.tomshardware.com/how-to/raspberry-pi-pico-stopwatch>. Acesso em: 23 fev. 2025.

GITHUB. LED-Matrix-Clock. Autor: [nome do autor, se disponível]. [S.l.]: GitHub, [s.d.]. Disponível em: [https://github.com/\[autor\]/LED-Matrix-Clock](https://github.com/[autor]/LED-Matrix-Clock). Acesso em: 23 fev. 2025.

Nota: Projeto correlato de relógio com matriz LED, citado no texto (ajuste o link e autor se disponível).

HACKSTER.IO. Striking Binary Clock. Autor: [nome do autor, se disponível]. [S.I.]: Hackster.io, [s.d.]. Disponível em: [https://www.hackster.io/\[projeto\]/striking-binary-clock](https://www.hackster.io/[projeto]/striking-binary-clock). Acesso em: 23 fev. 2025.

GITHUB. Pico Binary Clock. Autor: [nome do autor, se disponível]. [S.I.]: GitHub, [s.d.]. Disponível em: [https://github.com/\[autor\]/pico-binary-clock](https://github.com/[autor]/pico-binary-clock). Acesso em: 23 fev. 2025.

INSTRUCTABLES. Electronic Clock for Raspberry Pi Pico. Autor: [nome do autor, se disponível]. [S.I.]: Instructables, [s.d.]. Disponível em: <https://www.instructables.com/Electronic-Clock-for-Raspberry-Pi-Pico/>. Acesso em: 23 fev. 2025.

Outras Referências Úteis

ARM LIMITED. ARM Cortex-M0+ Technical Reference Manual. [S.I.]: ARM Limited, 2017. Disponível em: <https://developer.arm.com/documentation/100166/0001>. Acesso em: 23 fev. 2025.

MICROCHIP TECHNOLOGY. Introduction to I2C Protocol. [S.I.]: Microchip Technology, 2019. Disponível em: <https://www.microchip.com/en-us/application-notes/introduction-to-i2c>. Acesso em: 23 fev. 2025.

O'REILLY, Chris. Programming the Raspberry Pi Pico in C. [S.I.]: O'Reilly Media, 2021. Disponível em: <https://www.oreilly.com/library/view/programming-the-raspberry/9781492080992/>. Acesso em: 23 fev. 2025.