ESCOPO DO PROJETO

LingoFlow: Aprendizado Gamificado de Inglês e Tecnologia para Crianças

O projeto "LingoFlow" é um sistema inovador voltado para crianças, combinando aprendizado de inglês e introdução à tecnologia de maneira interativa e lúdica. Através da gamificação, o projeto incentiva o desenvolvimento do idioma e a imersão tecnológica, preparando os alunos para um ambiente digital dinâmico.

OBJETIVOS DO PROJETO

- Introduzir crianças ao aprendizado de inglês (e outras línguas) por meio da tecnologia;
- Criar um ambiente interativo e divertido com desafios gamificados;
- Desenvolver habilidades tecnológicas iniciais através da interação com dispositivos de hardware e software;
- Envolver alunos do ensino fundamental ao médio no aprendizado de robótica aplicada;
- Proporcionar atividades educativas alinhadas ao contexto escolar.

O "LingoFlow" funciona como uma inovação tecnológica aplicada ao ambiente escolar. O seu firmware inicial contém 5 grupos de imagens que representam 5 diferentes grupos de palavras em inglês e 8 matrizes coloridas que representam 8 diferentes cores a serem utilizados em testes. Os testes são conduzidos pelo educador, que fornece as coordenadas e coleta as respostas oralmente, incentivando a prática da conversação. LingoFlow é um conceito que integra o que há de melhor no analógico ao digital para influenciar positivamente a alfabetização tecnológica das crianças e adolescentes no Brasil.

A fim de melhorar o custo-benefício da aplicação espera-se que ela possa ser utilizada de forma interdisciplinar entre as diferentes séries de uma escola, onde as séries iniciais têm contato com os testes adaptados pelos estudantes de séries mais avançadas em aulas de robótica e programação. Gera assim, um ecossistema capaz de gerar uma alfabetização tecnológica eficiente e impulsionar possíveis futuros profissionais de tecnologia desde a fase escolar.

Inicialmente sem plataforma web, mas com potencial para futura integração IoT a fim de facilitar a análise de resultados aos professores. Suas principais funcionalidades incluem:

- **Desafios interativos:** Atividades de inglês associadas a mecanismos físicos e digitais.
- **Gamificação:** Progressão por níveis, recompensas e rankings para estimular o aprendizado.
- **Atividades práticas:** Introdução à robótica e à tecnologia em séries mais avançadas.
- **BitDogLab:** Laboratório educacional onde crianças aprendem a criar e aprimorar desafios tecnológicos.

- **Desenvolvimento progressivo:** Adaptação do conteúdo ao ritmo e nível de conhecimento do aluno.

JUSTIFICATIVA

O ensino de inglês é essencial na formação acadêmica, e a introdução à tecnologia desde cedo é um diferencial competitivo. "LingoFlow" busca unir essas áreas, proporcionando uma experiência inovadora e acessível, alinhada ao contexto escolar e preparando crianças para desafios tecnológicos futuros.

ORIGINALIDADE

Embora existam diversas plataformas voltadas ao ensino de inglês, "LingoFlow" se destaca por:

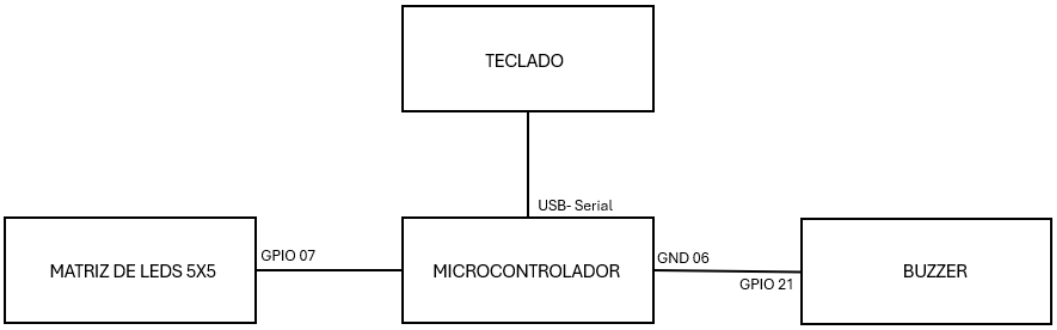
- **Integração com tecnologia educacional:** Unindo aprendizado de idiomas e robótica;
- **Metodologia baseada em gamificação:** Tornando o aprendizado mais atrativo;
- **Envolvimento ativo dos alunos:** Permite que estudantes avancem no aprendizado e participem da criação de desafios na BitDogLab;
- **Foco na introdução tecnológica:** Criando uma ponte entre a linguagem e a inovação desde cedo.

Uma análise de plataformas como Duolingo Kids e Lingokids mostra que, apesar de serem interativas, não possuem a integração tecnológica voltada ao ensino de robótica e à criação de desafios físicos. "LingoFlow" se diferencia ao proporcionar um aprendizado dinâmico que prepara crianças para um futuro digital e globalizado.

ESPECIFICAÇÃO DO HARDWARE

Com a integração a seguir, o PC controla a matriz de LEDs e o buzzer via teclado usando comunicação serial. Isso permite criar interações dinâmicas, como animações na matriz baseadas em comandos do teclado.

DIAGRAMA EM BLOCO



FUNÇÃO DE CADA BLOCO

Teclado do Computador: entrada de comandos via porta serial;

Microcontrolador: recebe os caracteres do teclado via UART, converte os comandos recebidos em ações sobre a matriz de LEDs e, opcionalmente, ativa o buzzer conforme a lógica implementada;

Matriz de LED 5x5: exibe padrões luminosos com base nos comandos do teclado;

Buzzer: dependendo da entrada do teclado, emite som.

CONFIGURAÇÃO DE CADA BLOCO

Buzzer (Pino 21)
O buzzer é controlado utilizando PWM para gerar frequências sonoras.

Pino utilizado: pino 21, configurado como saída PWM.

Função do pino: o pino é configurado para funcionar como um pino de PWM usando gpio_set_function(pin, GPIO_FUNC_PWM);.

Frequência do PWM: a frequência do PWM é ajustada com base na constante da nota musical (frequências são definidas nas enumerações DO, RE, etc.).

Controle do PWM: a função pwm_set_gpio_level(pin, 32768); ativa a saída PWM com a frequência calculada e pwm_set_gpio_level(pin, 0); desativa o PWM após o tempo de execução.

LEDs (Pino 7)
Controle da matriz de LEDs utilizando a PIO (Programmable Input/Output).

PIO utilizado: PIO 0 é utilizado para controlar a matriz de LEDs.

Pino utilizado: O pino 7 é configurado para controlar os LEDs da matriz através do programa pio_matrix_program**.**

Função do Pino: O pino é configurado para controlar os LEDs através da execução de um programa PIO.

UART (Pino 0 para transmissão)
Comunicação serial via UART para receber comandos de controle.

Velocidade de transmissão: A comunicação UART é configurada para 115200 bps com uart_init(uart0, 115200);.

Função do Pino: O pino de transmissão UART (uart0) é configurado para funcionar como UART.

COMANDOS E REGISTROS UTILIZADOS

Buzzer (Pino 21)
pwm_gpio_to_slice_num(pin) **-** obtém o número do slice de PWM associado ao pino.

gpio_set_function(pin, GPIO_FUNC_PWM) **-** configura o pino para função PWM.

pwm_config_set_clkdiv(&config, clock_get_hz(clk_sys) / (frequency * 2048)) **-** define o divisor do clock para ajustar a frequência de PWM.

pwm_init(slice_num, &config, true) **-** inicializa o PWM com a configuração definida.

LEDs (Pino 7)
pio_add_program(pio, &pio_matrix_program) **-** adiciona o programa PIO para controlar a matriz de LEDs.

pio_claim_unused_sm(pio, true) **-** reivindica um state machine não utilizado para o controle.

pio_matrix_program_init(pio, sm, offset, OUT_PIN) - inicializa o programa PIO com o state machine e o pino de saída.

UART (Pino 0 para transmissão)
uart_init(uart0, 115200) - inicializa a UART com a taxa de baud especificada.

std_uart_set_rx_enabled(true) - habilita a recepção de dados na UART (não explicitado no código, mas é implicitamente configurado pela getchar()).

Entrada de Teclado
Leitura de comandos do teclado e execução de ações associadas, como acionar LEDs ou tocar notas musicais.

Teclas de comando: o programa lê a entrada de teclado com getchar_timeout_us(0) e executa ações com base nos caracteres recebidos.

Controle de LEDs: a cor de cada LED é determinada pela tecla pressionada (exemplo: 'R' para vermelho). A função matrix_rgb() é usada para calcular a cor RGB e controlar_todos_leds() acende todos os LEDs na matriz com a cor selecionada.

Notas musicais: para os números (1-5), o buzzer toca a nota correspondente e uma sequência de frames é exibida na matriz de LEDs, como frame_numbers(), frame_climate(), etc.

Exibição de Frames
Controle da exibição de animações específicas na matriz de LEDs, como clima, emoções, frutas, etc.

Função frame_numbers() - exibe números na matriz de LEDs.

Função frame_climate(), frame_emotions(), frame_fruits() - cada uma exibe um conjunto de imagens ou animações associadas ao tema.

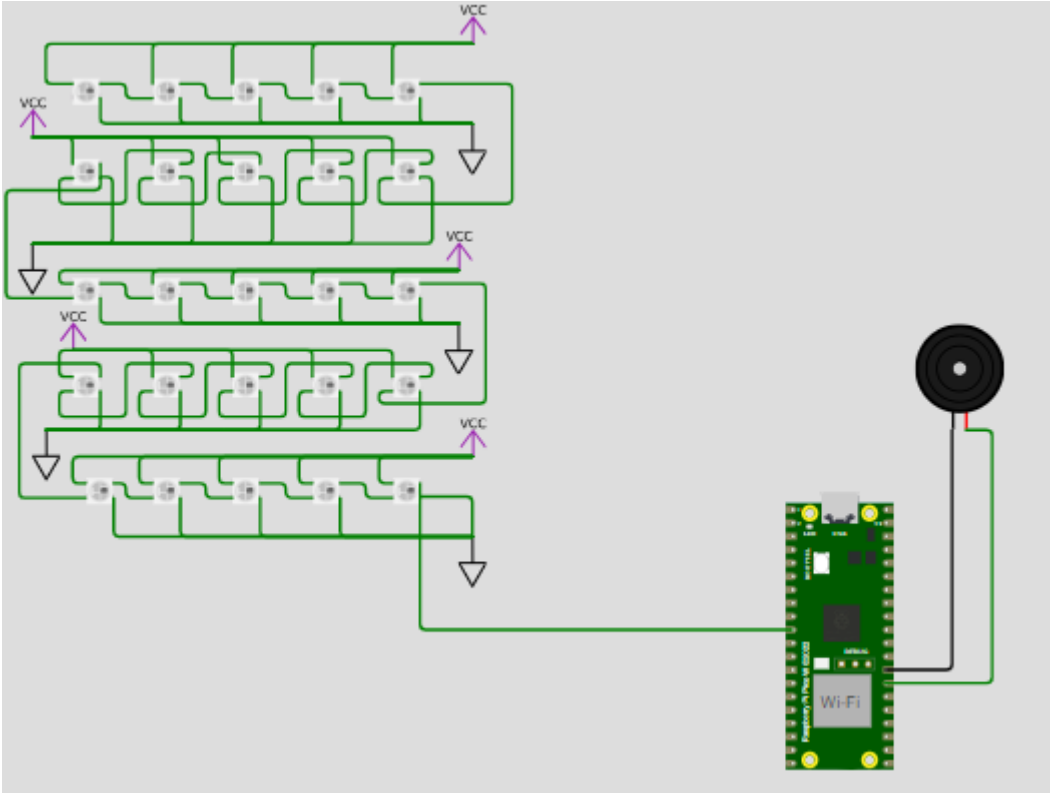
PINAGEM

Matriz de LEDs 5x5 conectada à GPIO 7;

Buzzer conectado à GPIO21 e GND 6;

Teclado conectado via USB-Serial.

CIRCUITO COMPLETO DO HARDWARE

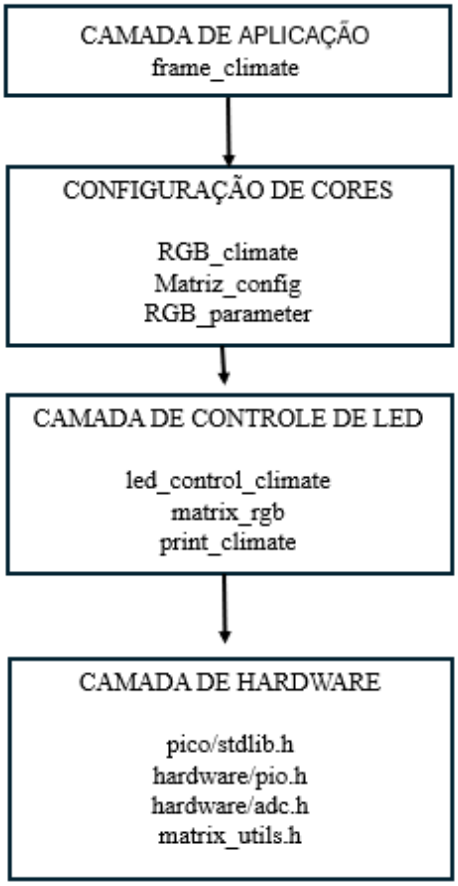


ESPECIFICAÇÃO DO FIRMWARE

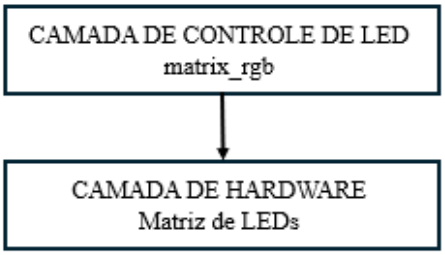
BLOCOS FUNCIONAIS

Cabe ressaltar que o firmware é composto por uma quantidade considerável de arquivos que aqui serão representados e são caracterizados por: 5 arquivos de cabeçalho que contém os frames de deus grupos de palavras específicos, 1 arquivo de cabeçalho que define uma função matrix_rgb que converte valores de cor RGB (vermelho, verde e azul) em um formato binário de 32 bits, que pode ser usado para controlar LEDs WS2812B, 1 arquivo de cabeçalho que define o formato dos números a serem apresentados na matriz, 1 arquivo de cabeçalho que implementa uma animação de "chasers" (perseguidores) em uma matriz de LEDs WS2812B, onde diferentes cores e padrões são exibidos em sequência e o arquivo principal em C.

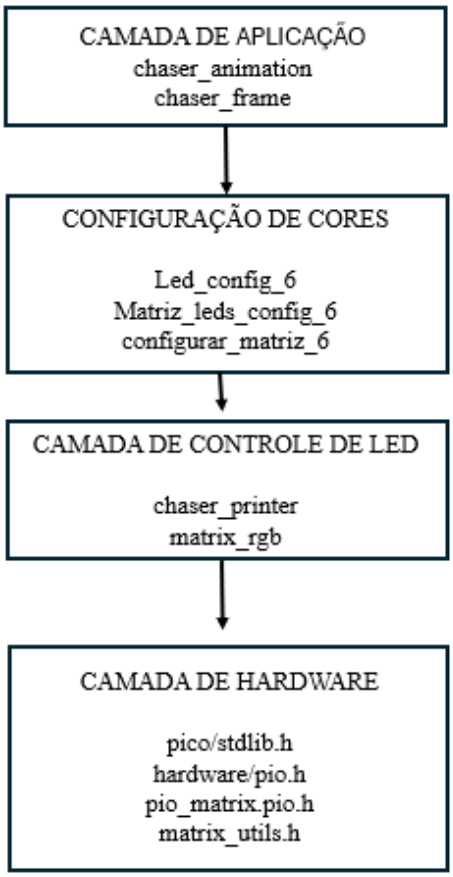
PADRÃO DOS ARQUIVOS DE FRAMES
(frame_fruits.h e cia)



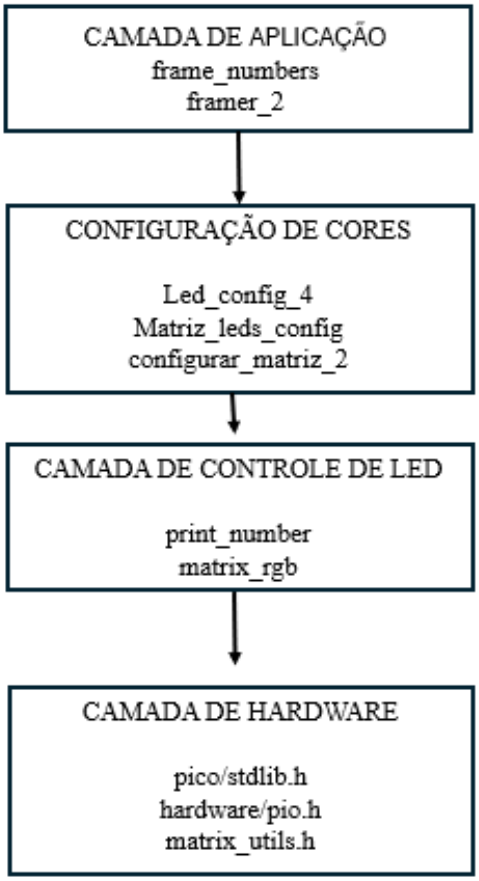
matrix_utils.h

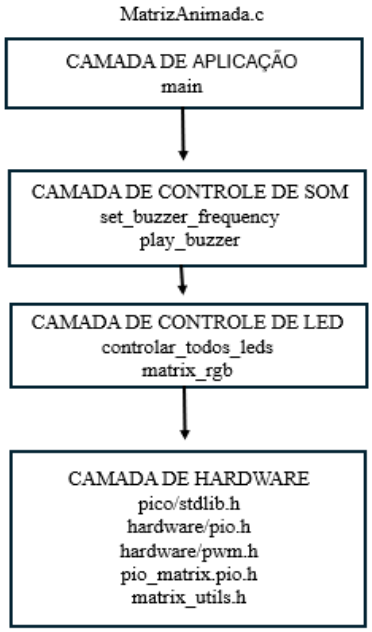


frame_color_chaser.h

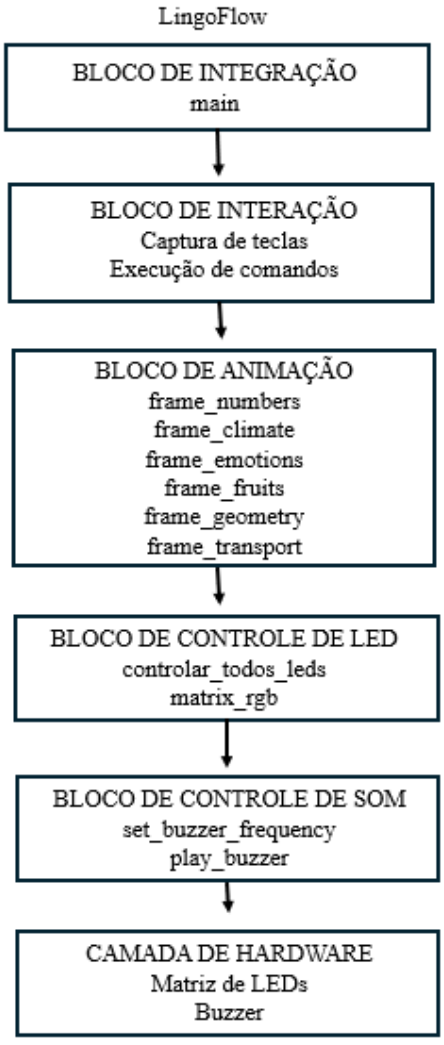


frame_numbers.h





O bloco funcional do firmware está representado a seguir:



DESCRIÇÃO DAS FUNCIONALIDADES

O **bloco de integração** integra todos os blocos funcionais e gerencia o fluxo do programa.

Funcionalidades
main: função principal que orquestra a execução dos blocos funcionais, como a leitura de entradas do usuário, exibição de animações e reprodução de sons.

Arquivos relacionados: código principal.

O **bloco de interação com o usuário** captura entradas do usuário (via teclado) e toma decisões com base nelas.

Funcionalidades
Leitura de teclas: captura as teclas pressionadas pelo usuário usando getchar_timeout_us.
Execução de comandos: interpreta as teclas pressionadas e executa as ações correspondentes, como exibir cores, animações ou tocar sons.

Arquivos relacionados: código principal (onde a lógica de interação está implementada).

O **bloco de animação** gerencia as animações exibidas na matriz de LEDs.

Funcionalidades
frame_numbers: exibe números de 0 a 9 na matriz de LEDs.
frame_climate: exibe animações relacionadas ao clima (chuva, sol, nublado, etc.).
frame_emotions: exibe animações relacionadas a emoções (feliz, triste, surpreso, etc.).
frame_fruits: exibe animações relacionadas a frutas (maçã, banana, laranja, etc.).
frame_geometry: exibe animações relacionadas a formas geométricas (círculo, quadrado, triângulo, etc.).
frame_transport: exibe animações relacionadas a meios de transporte (carro, avião, barco, etc.).

Arquivos relacionados: frame_numbers.h, frame_climate.h, frame_emotions.h, frame_fruits.h, frame_geometry.h, frame_transport.h.

O **bloco de controle de LEDs** gerencia a matriz de LEDs WS2812B, controlando as cores e animações exibidas.

Funcionalidades
matrix_rgb: converte valores de cor RGB (vermelho, verde, azul) em um formato binário de 32 bits, que é compreendido pelos LEDs.
controlar_todos_leds: envia uma cor específica para todos os LEDs da matriz simultaneamente.
Funções de animação: controlam a exibição de padrões específicos na matriz de LEDs, como números, emoções, frutas, formas geométricas, etc.

Arquivos relacionados: matrix_utils.h e pio_matrix.pio.h

O **bloco de controle de som** gerencia o buzzer para reproduzir sons e notas musicais.

Funcionalidades
set_buzzer_frequency: configura a frequência do buzzer para tocar uma nota específica.
play_buzzer: toca o buzzer com uma frequência e duração definidas, permitindo a reprodução de notas musicais.

Arquivos relacionados: código principal (onde as funções do buzzer estão definidas).

O **bloco de hardware** interage diretamente com o hardware do Raspberry Pi Pico.

Funcionalidades
Inicialização do sistema: configura o clock, inicializa a comunicação serial e o PIO.
Controle do PIO: gerencia o envio de dados para a matriz de LEDs.
Controle do PWM: configura o PWM para controlar o buzzer.

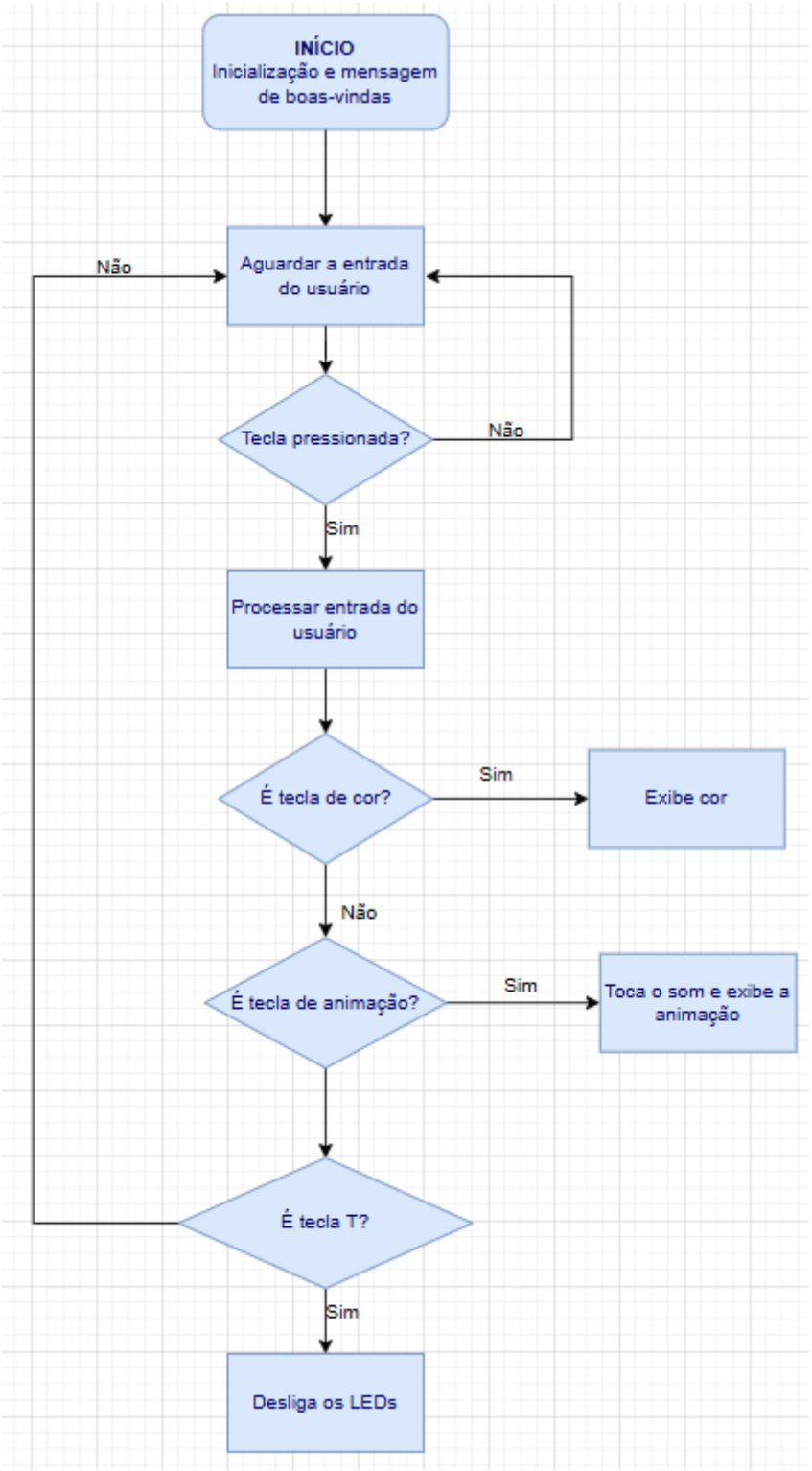
Arquivos relacionados: pico/stdlib.h, hardware/pio.h, hardware/pwm.h, pio_matrix.pio.h.

DEFINIÇÃO DAS VARIÁVEIS

Bloco Funcional	Variável	Tipo	Descrição
Controle de LEDs	PIO pio	PIO	Controlador PIO para gerenciar a matriz de LEDs.
	uint sm	uint	Máquina de estado do PIO para enviar dados aos LEDs.
	uint32_t cor	uint32_t	Cor no formato binário de 32 bits.
	float itnsty	float	Intensidade da cor (0.0 a 1.0).
Controle de Som	uint pin	uint	Pino GPIO conectado ao buzzer.
	uint frequency	uint	Frequência da nota musical.
	uint duration_ms	uint	Duração do som em milissegundos.
Interação com o Usuário	char tecla	char	Tecla pressionada pelo usuário.
	char buffer	char	Buffer para armazenar temporariamente a entrada do usuário.
	char comando[10]	char[]	Comando completo digitado pelo usuário.

Bloco Funcional	Variável	Tipo	Descrição
	int comando_index	int	Índice para rastrear a posição no array comando.
Animação	Matriz_leds_config_6 default_frame_6	Matriz_leds_config_6	Configuração padrão da matriz de LEDs.
	Matriz_leds_config_6 frame	Matriz_leds_config_6	Configuração atual da matriz de LEDs para um frame.
	Matriz_leds_config_6 base_frame	Matriz_leds_config_6	Cópia da configuração padrão da matriz de LEDs.
	Matriz_leds_config_6 frames[24*3+4]	Array de Matriz_leds_config_6	Sequência de frames para a animação.
Integração	bool system_busy	bool	Indica se o sistema está ocupado processando uma tarefa.
	bool ok	bool	Resultado de operações críticas.
Hardware	uint offset	uint	Endereço de memória do programa PIO carregado.
	uint slice_num	uint	Número da fatia do PWM usada para controlar o buzzer.

FLUXOGRAMA


INÍCIO
Inicialização e mensagem de boas-vindas
Aguardar a entrada do usuário
Não
Tecla pressionada?
Não
Sim
Processar entrada do usuário
É tecla de cor?
Sim
Exibe cor
Não
É tecla de animação?
Sim
Toca o som e exibe a animação
É tecla T?
Sim
Desliga os LEDs

INICIALIZAÇÃO DO SOFTWARE

1. **Configuração do Clock**

   O clock do sistema é configurado para operar a 128 MHz usando a função set_sys_clock_khz. Isso garante que o Raspberry Pi Pico opere na frequência desejada para o controle da matriz de LEDs e do buzzer.

2. **Inicialização da Comunicação Serial**

   A comunicação serial é inicializada usando stdio_init_all e uart_init. Isso permite a interação com o usuário via terminal (USB).

3. **Configuração do PIO**

   O PIO (Programmable I/O) é configurado para controlar a matriz de LEDs WS2812B. O programa PIO é carregado usando pio_add_program, e a máquina de estado (SM) é inicializada com pio_matrix_program_init.

4. **Configuração do PWM**

   O PWM (Pulse Width Modulation) é configurado para controlar o buzzer. A função set_buzzer_frequency define a frequência do buzzer.

5. **Exibição da Mensagem de Boas-Vindas**

   Uma mensagem de boas-vindas é exibida no terminal, informando ao usuário as opções disponíveis.

CONFIGURAÇÕES DOS REGISTROS

Os registros são configurados para controlar o hardware do Raspberry Pi Pico.

Configuração do Clock

Função: set_sys_clock_khz.

Descrição: configura a frequência do clock do sistema.

Registros envolvidos: registros do sistema de clock do RP2040.

Configuração do PIO

Função: pio_matrix_program_init.

Descrição: configura o PIO e a máquina de estado (SM) para controlar a matriz de LEDs.

Registros envolvidos: registros do PIO e da máquina de estado.

Configuração do PWM

Função: set_buzzer_frequency.

Descrição: configura o PWM para gerar a frequência desejada no buzzer.

Registros envolvidos: registros do PWM**.**

Configuração do GPIO

Função: gpio_set_function**.**

Descrição: configura os pinos GPIO para funções específicas (PIO, PWM, etc.).

Registros envolvidos: registros de controle de GPIO.

ESTRUTURA E FORMATO DOS DADOS

Os dados no software são organizados em estruturas e formatos específicos para controlar a matriz de LEDs e o buzzer.

Estrutura Led_config_6

Descrição: Armazena os valores de cor RGB (vermelho, verde, azul) para um LED.

Campos:

double red: intensidade do vermelho (0.0 a 1.0).
double green: intensidade do verde (0.0 a 1.0).
double blue: intensidade do azul (0.0 a 1.0).

Matriz Matriz_leds_config_6

No formato: array bidimensional de estruturas Led_config_6, representa a matriz de LEDs 5x5.

Exemplo:

```
Matriz_leds_config_6 matriz = {
    {{0.0, 0.0, 0.0}, {0.0, 0.0, 0.0}, ...}, // Linha 0
    {{0.0, 0.0, 0.0}, {0.0, 0.0, 0.0}, ...}, // Linha 1
    ...
};
```

Dados de Cor

No formato: uint32_t (32 bits), representa a cor no formato binário que a matriz de LEDs entende.

Exemplo:

```
uint32_t cor = matrix_rgb(0.0, 1.0, 0.0); // Verde
```

PROTOCOLO DE COMUNICAÇÃO

O protocolo de comunicação é simples e baseado na interação via terminal (USB).

Entrada do Usuário

O usuário digita uma tecla no terminal e o programa captura a tecla usando getchar_timeout_us.

Resposta do Sistema

O sistema processa a tecla e executa a ação correspondente (exibir cor, tocar som, exibir animação). O resultado é exibido no terminal e na matriz de LEDs.

Comunicação com a Matriz de LEDs

Os dados são enviados para a matriz de LEDs usando o protocolo WS2812B. O PIO é responsável por gerar o sinal de controle necessário.

FORMATO DO PACOTE DE DADOS

Tamanho do Pacote

Cada pacote contém dados para 25 LEDs (matriz 5x5).
Cada LED requer 32 bits de dados (formato RGB).

Estrutura do Pacote

Bits 0-7: intensidade do azul (8 bits).
Bits 8-15: intensidade do vermelho (8 bits).
Bits 16-23: intensidade do verde (8 bits).
Bits 24-31: reservados (não usados).

Transmissão

Os pacotes são enviados sequencialmente para a matriz de LEDs usando o PIO. O sinal é gerado de acordo com o protocolo WS2812B, que requer um timing preciso.

EXECUÇÃO DO PROJETO

METODOLOGIA

Por ter tido maior afinidade com a atividade de animação na matriz de LEDs 5x5, pensei em formas de gerar utilidade às funcionalidades que ela trouxe consigo e assim surgiu a aplicabilidade na contribuição à STEAM. O processo de pesquisa trouxe algumas outras informações a respeito de controladores de matriz de LEDs WS2812 e suas aplicações em projetos visuais, configuração de PWM para controle de buzzer e geração de

frequências musicais e programação utilizando PIO (Programmable I/O) para otimizar a comunicação com a matriz de LEDs.

A escolha do uso da matriz de LEDs WS2812 e do buzzer foi feita com base na melhor aplicabilidade em projetos educacionais (o display também é ideal, mas pensando nos anos escolares iniciais estímulos sonoros e visuais se fazem mais eficientes) dos mesmos em detrimento aos demais componentes presentes na BitDogLab. Raspberry Pi Pico foi escolhido não só por ser o microcontrolador requerido no projeto mas também por sua capacidade de rodar código C/C++, suporte a PIO e ampla documentação.

O software foi estruturado para permitir a identificação de cores com base na entrada do usuário, o controle da matriz de LEDs via PIO para exibição de cores e animações, a emissão de sons correspondentes a ações do usuário por meio do buzzer e a implementação de um menu interativo via serial (USB) para facilitar a interação.

A Raspberry Pi Pico SDK foi configurada no VS Code, utilizando CMake e GNU Make**.**

O código foi escrito em C e dividido em módulos para melhor organização: configuração do PIO para controle da matriz, funções de PWM para controle do buzzer e interação via serial para detecção de entrada do usuário

Foram realizados testes incrementais para verificar o funcionamento de cada módulo: teste inicial com PIO para garantir a comunicação correta com a matriz de LEDs, verificação do controle PWM do buzzer e ajuste das frequências sonoras e, por fim, depuração da comunicação serial para garantir a captura correta das entradas do usuário.

TESTES DE VALIDAÇÃO

Os testes foram realizados para garantir que todas as funcionalidades do projeto estavam operando corretamente.

O teste de comunicação serial verificou se a entrada de caracteres foi lida corretamente pelo getchar_timeout_us() e garantiu identificação precisa de cada comando correspondente.

O teste de Matriz de LEDs verificou se o PIO controlou corretamente os LEDs, exibindo as cores esperadas e testou a variação da intensidade para confirmar ajustes visuais.

O teste do buzzer analisou as frequências das notas musicais que foram testadas e validadas com um analisador de espectro. O buzzer respondeu corretamente aos comandos do usuário.

Alguns testes de tempo de resposta foram feitos até que o sistema respondesse sem atrasos perceptíveis entre a entrada do usuário e a execução da ação correspondente.

DISCUSSÃO DOS RESULTADOS

O projeto demonstrou ser confiável e funcional, atendendo aos requisitos estabelecidos e garantindo precisão na leitura de entradas, bom desempenho da matriz de LEDs, bom feedback sonoro e execução eficiente. A utilização do Raspberry Pi Pico e do PIO foi essencial para garantir um desempenho otimizado, tornando a experiência interativa e didática.

REFERÊNCIAS

ALIMISIS, D. Educational robotics: open questions and new challenges. *Themes in Science and Technology Education*, v. 6, n. 1, p. 63-71, 2013.

KAPP, K. M. *The gamification of learning and instruction: game-based methods and strategies for training and education*. Pfeiffer, 2012.

PLOWMAN, L.; STEPHEN, C. Guided interaction in pre-school settings. *Journal of Computer Assisted Learning*, v. 23, n. 1, p. 14-26, 2007.

RASPBERRY PI FOUNDATION. *Raspberry Pi Pico datasheet*. 2021. Disponível em: https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf. Acesso em: fev. 2025.

WORLDSEMI. *WS2812B datasheet*. 2020. Disponível em: https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf. Acesso em: fev. 2025.

YAKMAN, G. *STEAM education: an overview of creating a model of integrative education*. 2008. Disponível em: https://www.steamedu.com. Acesso em: fev. 2025.

DREH3. *ledsAnimados*. 2023. Disponível em: https://github.com/Dreh3/ledsAnimados. Acesso em: fev. 2025.