Nome: LUCAS DIAS DA SILVA

Matrícula: TIC370100847

Unidade 7|Capítulo 1 - Projeto de Sistema Embarcado

---

1. Escopo do Projeto

Apresentação do Projeto

O projeto "Monitor de Ruído Inteligente" é um sistema embarcado para monitoramento contínuo dos níveis de ruído ambiente, oferecendo feedback visual e sonoro. A proposta inclui, para futuras expansões, a integração com conectividade Wi-Fi para o envio dos dados coletados a um servidor remoto.

Título do Projeto

Sistema Embarcado para Monitoramento de Níveis de Ruído

Objetivos do Projeto

- Capturar e processar os níveis de ruído ambiente em tempo real.
- Exibir informações em um display OLED e fornecer alertas por LED e buzzer.
- Implementar um sistema de autoteste para verificação do microfone.
- Preparar a expansão para comunicação Wi-Fi e envio de dados para um servidor remoto.

Descrição do Funcionamento

O sistema utiliza um microfone de eletreto para capturar os níveis de ruído, processa os dados e exibe os resultados em um display OLED. LEDs indicam o status do ambiente, enquanto um buzzer alerta para ruídos excessivos.

Observação

A funcionalidade de registro dos dados em formato JSON foi descontinuada na versão atual. O envio de dados para um servidor via Wi-Fi foi planejado como expansão futura, permitindo assim maior integração IoT sem aumentar a complexidade do firmware atual.

Justificativa

A justificativa para o desenvolvimento do "Monitor de Ruído Inteligente" baseia-se em vários aspectos críticos que impactam diretamente a saúde, a eficiência operacional e a qualidade de vida em ambientes industriais, corporativos e urbanos:

Impactos na Saúde:

Estudos demonstram que a exposição prolongada a altos níveis de ruído pode ocasionar perda auditiva, estresse crônico, distúrbios do sono e até mesmo doenças cardiovasculares. Esses efeitos não só comprometem o bem-estar dos indivíduos como também aumentam os custos com cuidados de saúde e reduzem a produtividade em ambientes onde o ruído é um fator predominante.

Necessidade de Monitoramento Contínuo:

Os métodos tradicionais de medição de ruído geralmente são pontuais e de alto custo, dificultando a identificação precoce de condições anormais e a implementação de medidas corretivas. Um sistema embarcado que realize o monitoramento em tempo real permite a detecção imediata de níveis de ruído perigosos, possibilitando intervenções rápidas e eficazes.

Integração com Soluções IoT:

A futura expansão do projeto para incluir conectividade Wi-Fi e envio de dados para um servidor remoto alinha a solução com as tendências atuais da Internet das Coisas (IoT). Essa integração permitirá a centralização dos dados, análise de tendências e suporte à tomada de decisão, contribuindo para a implementação de políticas de controle de ruído e manutenção preventiva em larga escala.

Originalidade

Através de uma pesquisa realizada, foram identificados diversos projetos correlatos na área de sistemas embarcados para monitoramento, como:

- **Monitoramento Automotivo:** Coleta de dados como consumo de combustível, velocidade e temperatura para diagnósticos em tempo real.
  Link da pesquisa
- **Monitoramento de Máquinas Elétricas:** Foco na medição de variáveis físicas do rotor, como temperatura e vibração, para previsão de manutenção.
  Link da pesquisa
- **Gerenciamento de Redes Inteligentes de Energia:** Utiliza sistemas embarcados para monitoramento e gerenciamento de redes elétricas,

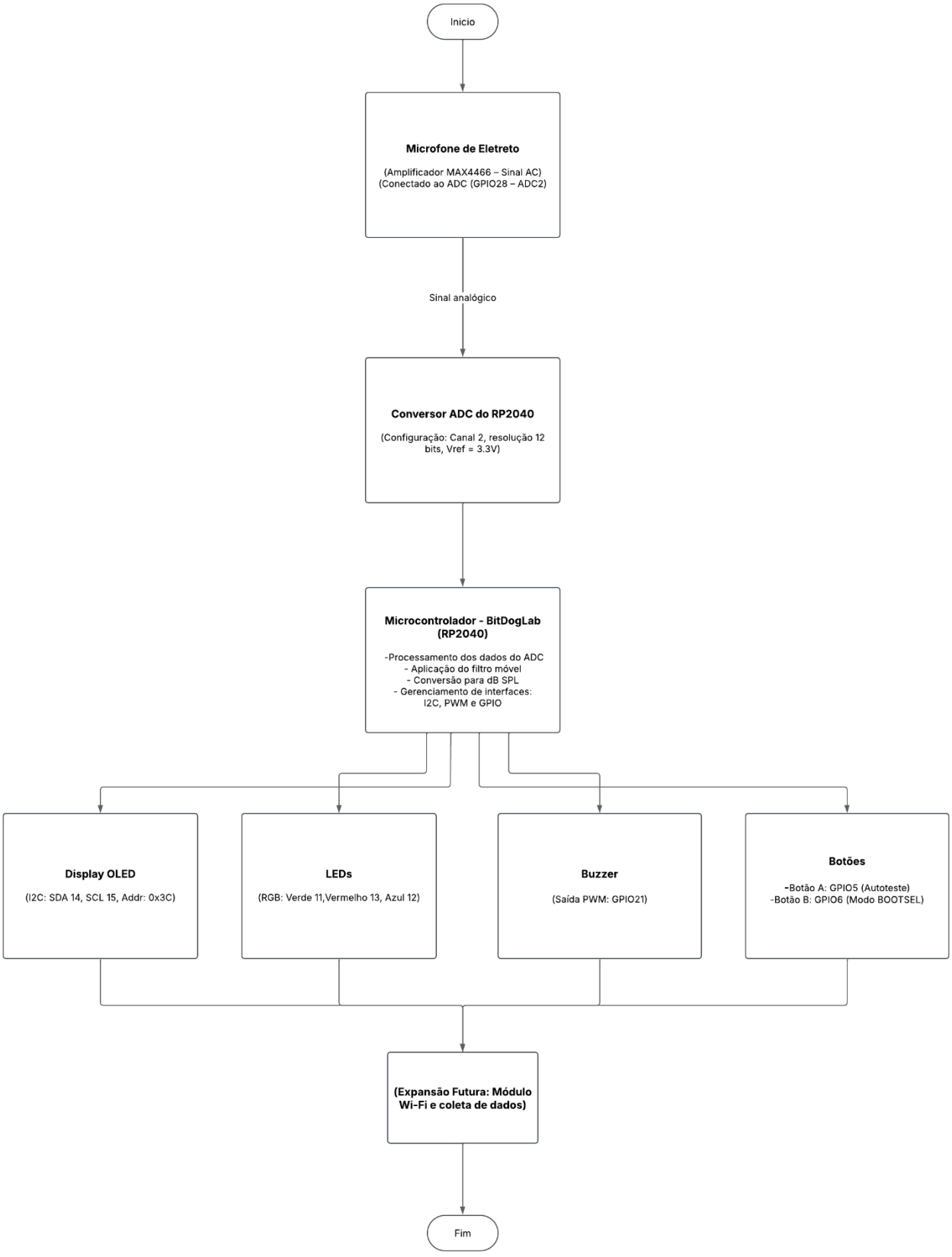
com ênfase em variáveis de operação e segurança. Link da pesquisa

- **Coleta de Dados para Diagnóstico Remoto:** Projetos que capturam dados (como vibração) para análise e detecção de falhas, por exemplo em drones. Link da pesquisa

Embora existam projetos correlatos que monitoram variáveis em ambientes automotivos, industriais ou de redes inteligentes, o **Monitor de Ruído Inteligente** se diferencia pela sua aplicação específica na medição acústica, combinada com um feedback imediato (através de display OLED, LEDs e buzzer) e pela preparação para futuras expansões que possibilitarão a integração com a IoT. Essa combinação única de funcionalidades não só agrega valor ao monitoramento ambiental, como também amplia as possibilidades de aplicação e escalabilidade do sistema.

2. Especificação do Hardware

Diagrama em Bloco



Descrição Detalhada de Cada Bloco

1. Microfone de Eletreto

Função:

- Capturar os sinais acústicos do ambiente e convertê-los em um sinal analógico.

Configuração:

- Utiliza o amplificador MAX4466 para amplificar o sinal do microfone.
- O sinal é direcionado para o canal ADC2 do microcontrolador (GPIO28).

Comandos/Registros Utilizados:

No firmware, é feita a seleção do canal com:

```
// Lê o valor do microfone (valor bruto do ADC: 0 a 4095)
adc_select_input(2);
uint16_t mic_value = 0;
for (int i = 0; i < 10; i++) { // Média de 10 leituras para suavizar
    mic_value += adc_read();
    sleep_us(10); // Reduzido o atraso entre leituras
}
mic_value /= 10;
```

2. Conversor ADC do RP2040

Função:

- Converter o sinal analógico do microfone em um valor digital para que o microcontrolador possa processá-lo.

Configuração:

- Configurado para operar no canal 2 (associado ao GPIO28) com resolução de 12 bits (valores de 0 a 4095).
- Tensão de referência definida como 3.3V.

Comandos/Registros Utilizados:

- Funções de inicialização e leitura do ADC, como:

```
// Inicializa o ADC e o GPIO do microfone
adc_init();
adc_gpio_init(MIC_ADC);
```

3. Microcontrolador – BitDogLab (RP2040)

Função:

- Processar os dados do ADC, aplicar filtros, calcular o nível de ruído em dB SPL e gerenciar a comunicação com os demais periféricos.

Configuração:

- Interfaces configuradas:
  - **ADC:** Leitura do microfone (canal 2, GPIO28).
  - **I2C:** Para comunicação com o display OLED, usando SDA (GPIO14) e SCL (GPIO15).
  - **GPIO:** Para controle dos LEDs, buzzer e botões.
  - **PWM:** Utilizado para controlar a intensidade e frequência do som emitido pelo buzzer.

Comandos/Registros Utilizados:

- Inicialização e configuração das interfaces, por exemplo:

```
// Configura o I2C para o display OLED
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);
```

4. Display OLED

Função:

- Exibir, em tempo real, os níveis de ruído (valor bruto do ADC e o nível em dB SPL) e outros parâmetros, como os limiares definidos.

Configuração:

- Conectado via I2C usando os pinos SDA (GPIO14) e SCL (GPIO15).
- Endereço do display: 0x3C.
- Dimensões configuradas para 128x64 pixels.

Comandos/Registros Utilizados:

- Inicialização e atualização do display com comandos como:

```
// Inicializa e configura o display OLED
ssd1306_init(&ssd, WIDTH, HEIGHT, false, DISPLAY_ADDR, I2C_PORT);
ssd1306_config(&ssd);
ssd1306_fill(&ssd, false);
ssd1306_send_data(&ssd);
```

5. LEDs (RGB)

Função:

- Fornecer indicação visual do estado do sistema com base no nível de ruído detectado (ex.: acionar LED verde para níveis normais, vermelho para alerta e azul para níveis intermediários).

Configuração:

- Conectados aos pinos:
    - Verde: GPIO11
    - Vermelho: GPIO13
    - Azul: GPIO12
- Configurados como saídas digitais, com possibilidade de utilização de PWM para controle de brilho, se necessário.

Comandos/Registros Utilizados:

- Para definir o estado de um LED, por exemplo:

```
// Controle dos LEDs e buzzer com base no valor bruto (você pode também usar o dB SPL)
if (mic_value > limiar_1 && mic_value < limiar_2) {
    gpio_put(LED_BLUE, true);
    gpio_put(LED_RED, false);
    gpio_put(LED_GREEN, false);
    if (!buzzer_ligado) {
        parar_som_buzzer(BUZZER_A);
        parar_som_buzzer(BUZZER_B);
    }
}
```

6. Buzzer

Função:

- Emitir alerta sonoro quando o nível de ruído ultrapassa um limiar definido.

Configuração:

- Conectado ao pino GPIO21.

- Configurado para operar com PWM, permitindo a geração de um som com frequência (por exemplo, aproximadamente 2000 Hz).

Comandos/Registros Utilizados:

- Para ativar o buzzer:

```
emitir_som_buzzer(BUZZER_A);
emitir_som_buzzer(BUZZER_B);
```

(A função configura o pino para PWM, define a divisão de clock, wrap e nível de canal.)

- Para desativar

```
parar_som_buzzer(BUZZER_A);
parar_som_buzzer(BUZZER_B);
```

7. Botões

Botão A (GPIO5 – Pull-up):

Função:

- Aciona o autoteste do sistema (por exemplo, alterna o estado do buzzer).

Configuração:

- Configurado como entrada digital com resistor de pull-up.

Comandos/Registros Utilizados:

- Leitura do estado:

```
if (!gpio_get(BUTTON_A)) {
    debounce_delay();
```

Botão B (GPIO6 – Pull-up):

Função:

- Aciona o modo BOOTSEL, permitindo a entrada no modo de programação (reset para bootloader).

  Configuração:

- Configurado como entrada digital com resistor de pull-up.

  Comandos/Registros Utilizados:

- Para entrar no modo BOOTSEL:

```c
// Botão B: entra no modo BOOTSEL
if (!gpio_get(BUTTON_B)) {
    debounce_delay();
    printf("[BOTÃO B] Pressionado! Entrando em modo BOOTSEL.\n");
    reset_usb_boot(0, 0);
}
```

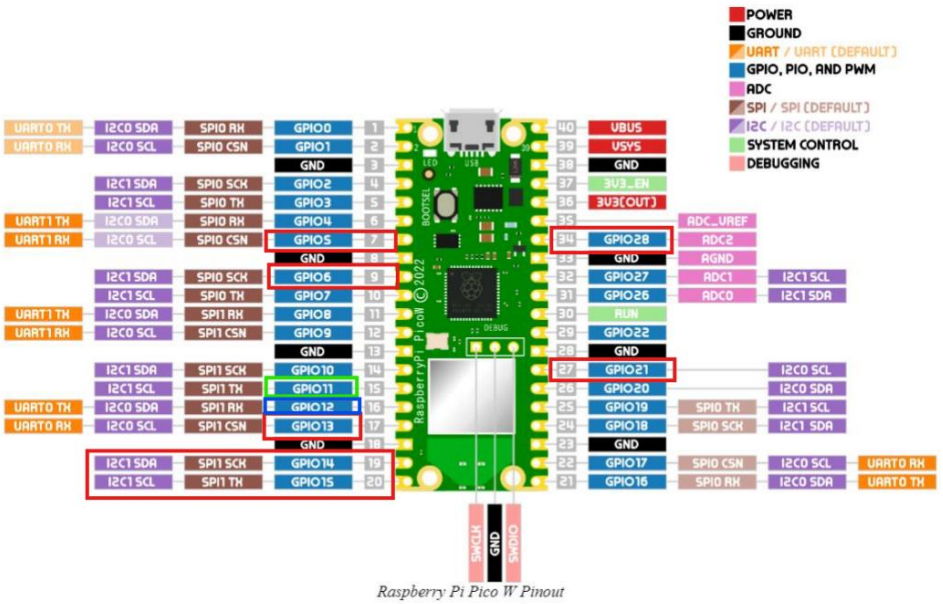
Componentes Relevantes

A placa BitDogLab é baseada no **Raspberry Pi Pico W** e possui diversos circuitos auxiliares que facilitam sua integração com sensores e atuadores. Entre os principais componentes:

- **Amplificador de Microfone (MAX4466)**
- **LEDs WS2812B**
- **Reguladores de Tensão (3.3V)**
- **Gerenciamento de Energia com suporte à bateria**

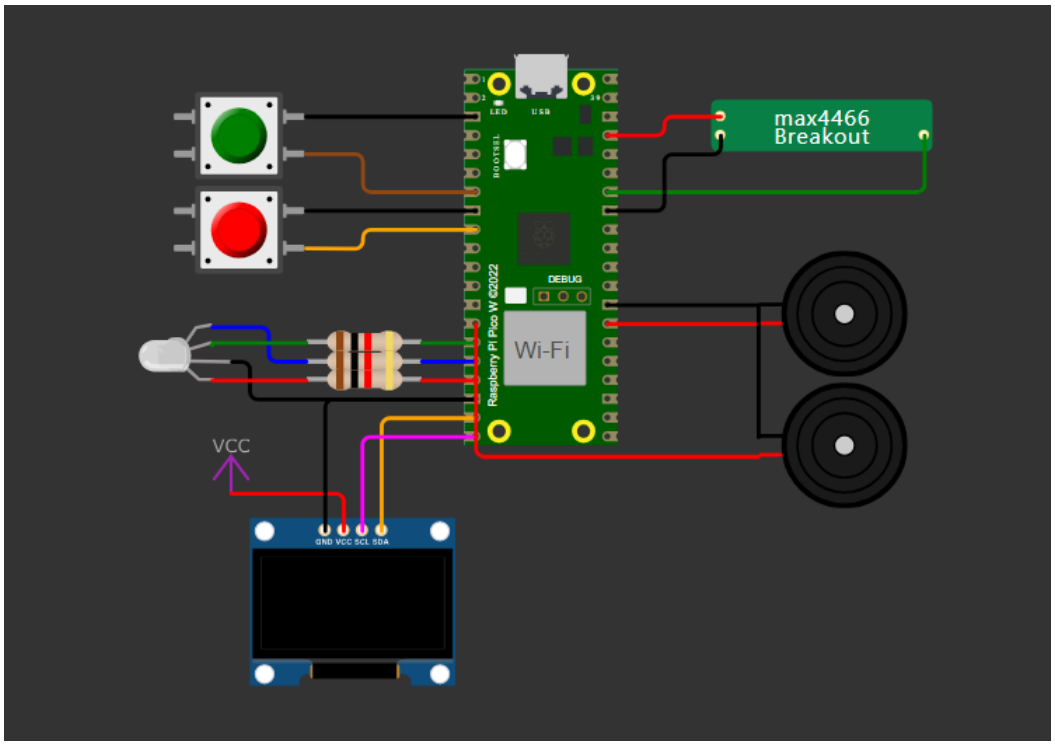
Descrição da Pinagem

- **Microfone:** GPIO28 (ADC2)
- **OLED (I2C):** SDA (GPIO14), SCL (GPIO15)
- **LEDs:** Vermelho (GPIO13), Verde (GPIO11), Azul (GPIO12)
- **Buzzer:** GPIO21
- **Botão A:** GPIO5 (pull-up)
- **Botão B:** GPIO6 (pull-up)

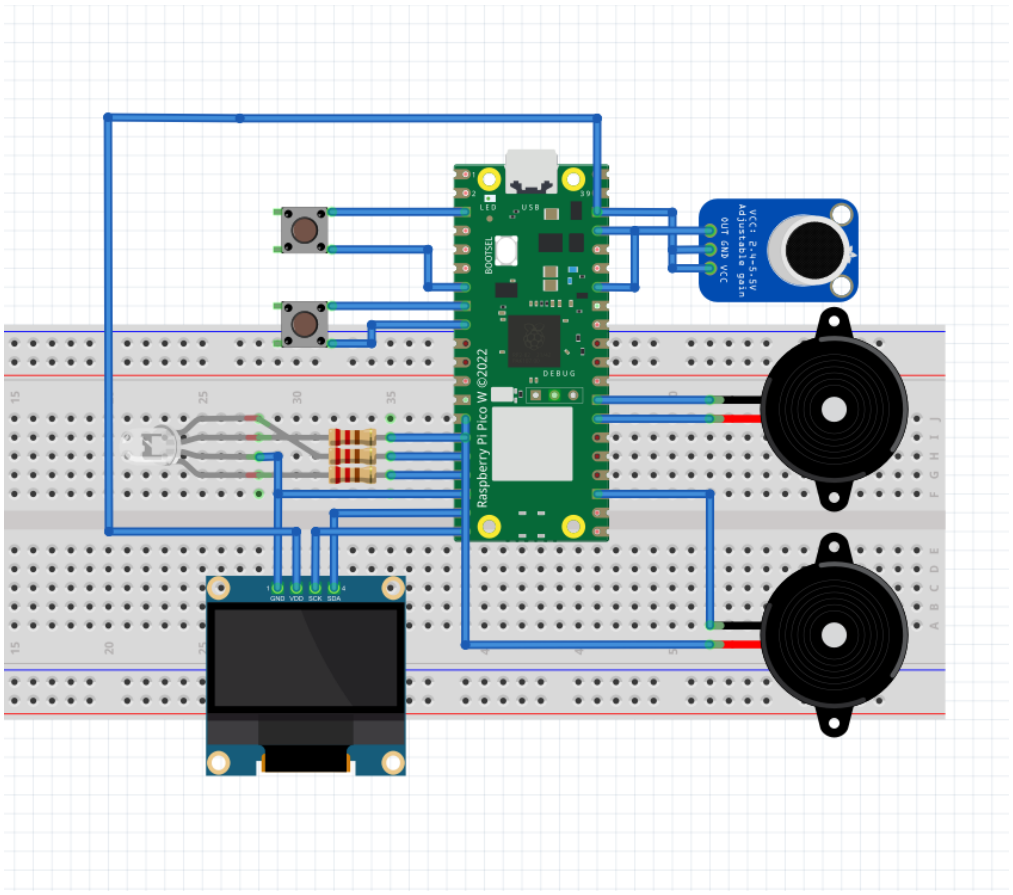


Raspberry Pi Pico W Pinout

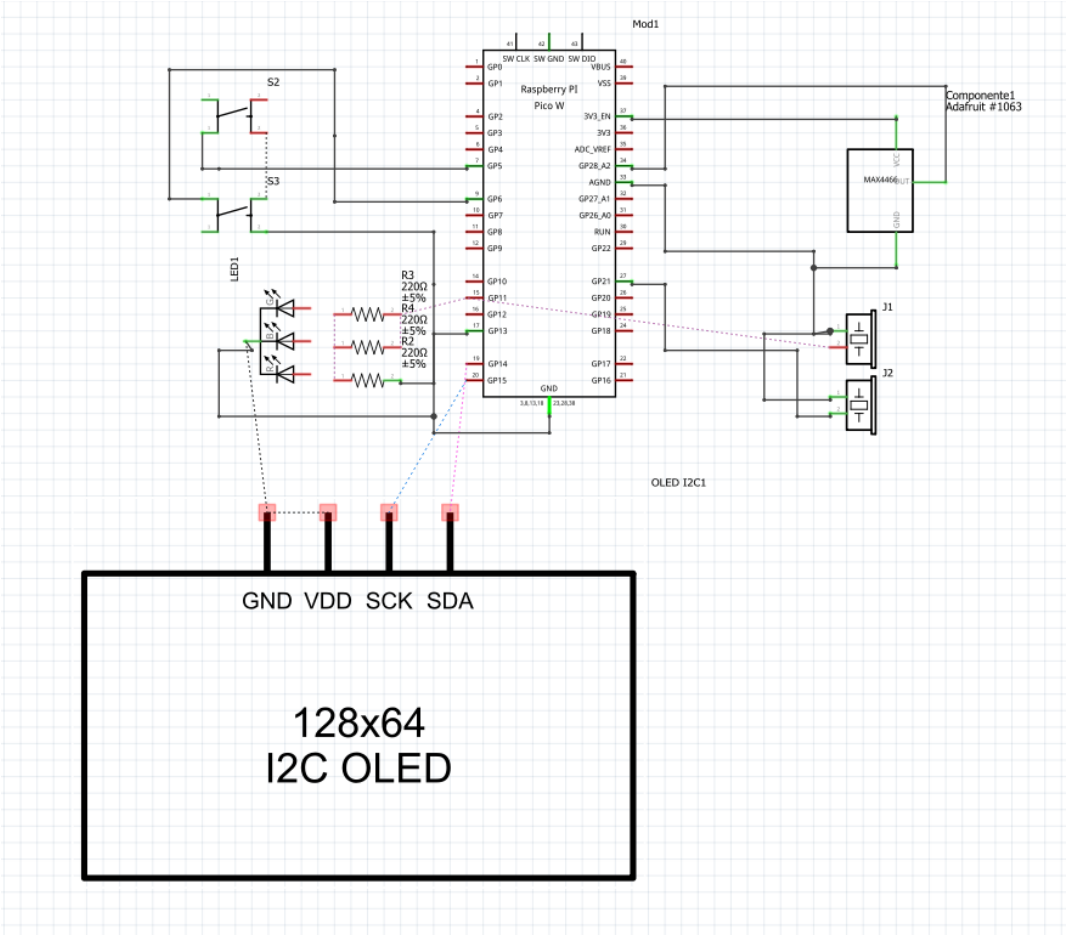
Circuito completo do hardware



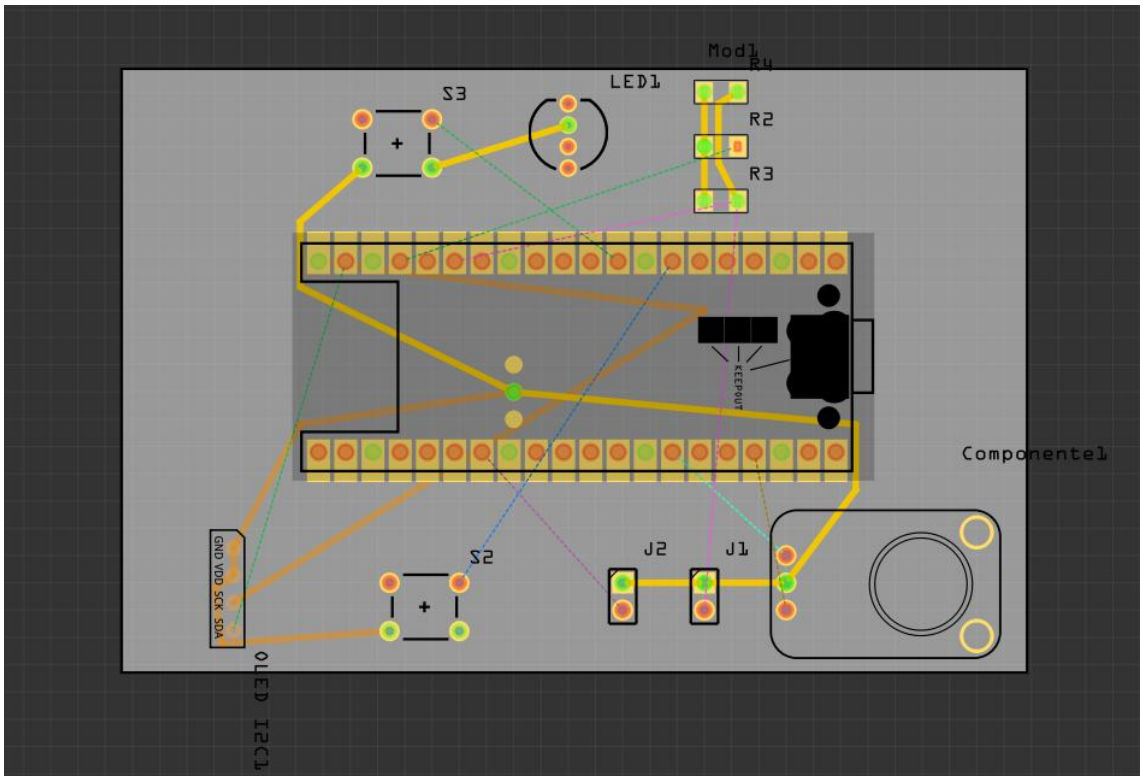
(Desenho do circuito completo do hardware no simulador **Wokwi**)



(Desenho do circuito completo no **Fritzing**)



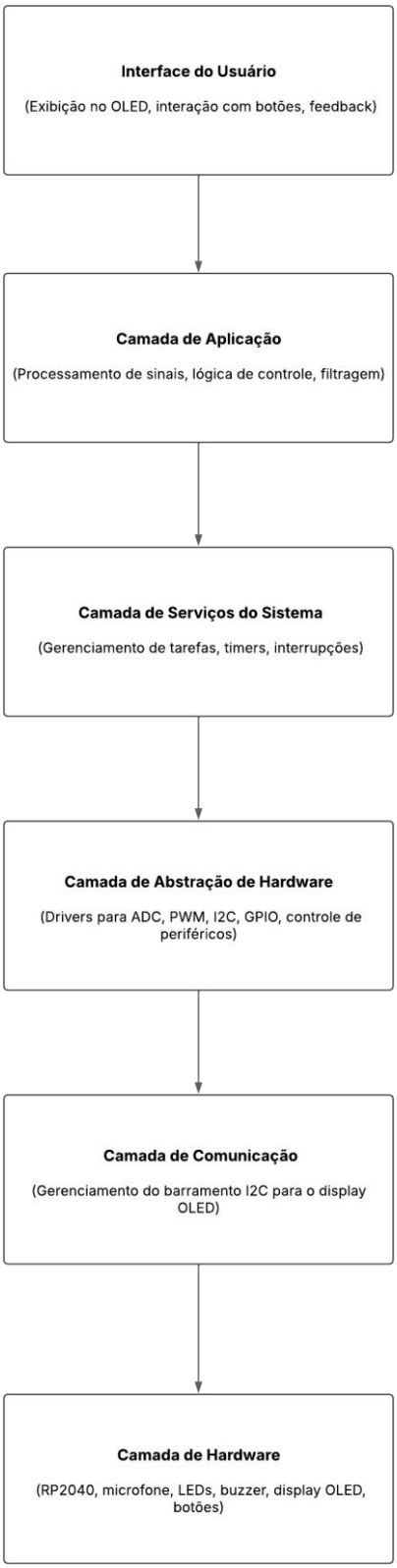
(Desenho do circuito esquemático no **Fritzing**)



(Desenho do circuito PCB no **Fritzing**)

3. Especificação do Firmware

Blocos Funcionais



Descrição das Funcionalidades

- **Interface do Usuário:**
  - Exibe os níveis de ruído no display OLED.
  - Permite interação via Botões A (autoteste) e B (modo BOOTSEL).
  - Fornece feedback visual e sonoro.
- **Camada de Aplicação:**
  - Processa sinais do microfone.
  - Aplica filtro móvel para suavização.
  - Converte sinais em dB SPL.
  - Controla LEDs e buzzer conforme limiares.
- **Camada de Serviços do Sistema:**
  - Gerencia timers para controle de tempo e debouncing de botões.
  - Lida com interrupções do ADC e GPIO.
  - Coordena as tarefas de leitura e exibição de dados.
- **Camada de Abstração de Hardware:**
  - Gerencia leitura do ADC.
  - Configura PWM para controle do buzzer.
  - Controla GPIOs para LEDs e botões.
  - Gerencia comunicação I2C com o display OLED.
- **Camada de Comunicação:**
  - Configura e gerencia o barramento I2C.
  - Controla o fluxo de dados para o display OLED.
- **Camada de Hardware:**
  - Inclui o microcontrolador RP2040 e periféricos (microfone, LEDs, buzzer, display OLED).

Definição das Variáveis

- uint16_t ruido_base: Valor médio do ruído ambiente.

```
uint16_t ruido_base = 0;
```

- bool buzzer_ligado: Estado atual do buzzer.

```
bool buzzer_ligado = false;
```

- uint16_t mic_value: Valor lido do ADC.

```
uint16_t mic_value = 0;
```

- float noiseFiltered: Valor filtrado da amplitude do sinal.

```
float noiseFiltered = 0.0f;
```

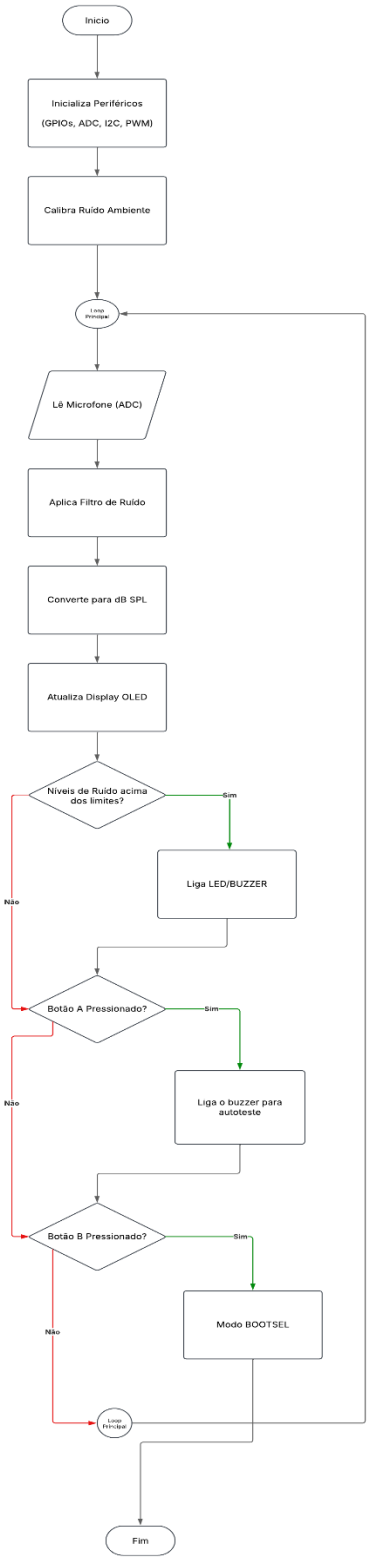
- float noise_dBSPL: Valor convertido em dB SPL.

```
float noise_dBSPL = 0.0f;
```

- uint16_t limiar_1, limiar_2, limiar_3: Limiar para controle dos LEDs e buzzer.

```
// Defina limiares para controle dos LEDs e buzzer (esses valores podem ser ajustados)
uint16_t limiar_1 = ruido_base + 100;   // Por exemplo, para acionar LED azul
uint16_t limiar_2 = 3000; // Para acionar LED vermelho
uint16_t limiar_3 = 4000;             // Para acionar LED vermelho e buzzer
```

Fluxograma



Expansão Futura:

Está prevista a integração com conectividade Wi-Fi e o desenvolvimento de um módulo para envio dos dados para um servidor remoto.

Inicialização

- Configuração de GPIOs para LEDs, botões e buzzer.

```
// Definições de pinos
#define BUZZER_A 21   // Buzzer A no GPIO21
#define BUZZER_B 10   // Buzzer B no GPIO10
#define BUTTON_A 5    // Botão A no GPIO5
#define BUTTON_B 6    // Botão B no GPIO6

const uint LED_RED   = 13;
const uint LED_BLUE  = 12;
const uint LED_GREEN = 11;
const uint MIC_ADC   = 28; // ADC do microfone (GPIO28 -> ADC2)
const uint NUM_AMOSTRAS = 100;

#define I2C_PORT    i2c1
#define I2C_SDA     14
#define I2C_SCL     15
#define DISPLAY_ADDR 0x3C
#define WIDTH  128
#define HEIGHT 64
```

- Inicialização do ADC para leitura do microfone.
- Configuração do I2C para o display OLED.
- Calibração inicial do ruído ambiente.

Configurações dos Registros

- **ADC:**
  - Canal 2 (GPIO28) selecionado via `adc_select_input(2)`.
- **GPIO:**
  - Configurado como saída para LEDs e buzzer.
  - Configurado como entrada com pull-up para botões.
- **PWM:**
  - Configurado para controlar frequência e duty cycle do buzzer.
- **I2C:**
  - Configuração em i2c1 com pinos SDA (GPIO14) e SCL (GPIO15).

Estrutura e Formato dos Dados

- **Dados do ADC: Valores de 0 a 4095.**
    - Os valores lidos pelo ADC variam entre 0 e 4095 (resolução de 12 bits) e representam a amplitude do sinal analógico capturado pelo microfone.
    - Esses valores são usados para medir a intensidade do som ambiente.
- **Dados Filtrados:**
    - Aplicação de um filtro móvel simples que calcula a média dos últimos 10 valores do ADC.
    - Esse filtro suaviza as variações abruptas e reduz o ruído, resultando em uma leitura mais estável.
- **Dados convertidos:**
    - Os valores filtrados são convertidos para **dB SPL** (Sound Pressure Level) é uma unidade que mede o nível de pressão sonora em decibéis (dB), em relação a um valor de referência padrão de 20 µPa (micropascais), que representa o menor som audível pelo ouvido humano. Ele é usado para quantificar o volume do som no ambiente.

      A fórmula para calcular o dB SPL é:

      $$dBSPL = 20 \times log_{10}\left(\frac{P}{Pref}\right)$$

        - P é a pressão sonora medida.
        - $P_{(}ref_{)}$ é a pressão de referência (20 µPa).

      No meu projeto, o sinal analógico capturado pelo microfone é convertido em valores digitais pelo **ADC** e depois convertido em **dB SPL** para representar o nível de ruído ambiente em uma escala reconhecível.

    - A conversão permite representar os níveis de ruído em unidades padronizadas de pressão sonora, facilitando a interpretação.

Protocolo de Comunicação

I2C:

- Comunicação entre o RP2040 e o display OLED.
- Endereço padrão do display: 0x3C.

Formato do Pacote de Dados

I2C (para OLED):

- Pacotes enviados contendo comandos e dados para atualização da tela.
- Exemplo: Comando para desenhar texto e enviar buffer de dados gráficos.

4. Execução do Projeto

Metodologia

- **Pesquisas Realizadas:**
    - Estudo de sistemas de monitoramento de ruído.
    - Análise de microcontroladores compatíveis.
- **Escolha do Hardware:**
    - Seleção do microcontrolador RP2040 pela versatilidade e suporte a múltiplos periféricos.
    - Escolha de microfone com amplificador, LEDs, buzzer e display OLED.
- **Definição das Funcionalidades do Software:**
    - Leitura de sinais de áudio.
    - Conversão para dB SPL.
    - Indicação visual e sonora.
    - Interface de usuário via display e botões.
- **Inicialização da IDE:**
    - Configuração da IDE Pico SDK.
    - Instalação de bibliotecas para controle de periféricos.
- **Programação na IDE:**
    - Implementação das funções de leitura de ADC.
    - Controle de LEDs, buzzer e display.
    - Desenvolvimento do algoritmo de filtragem.
- **Depuração:**
    - Testes em bancada com logs seriais.
    - Ajustes de sensibilidade e limiares.

Testes de Validação

- Verificação da calibração do microfone.

- Testes de resposta dos LEDs e buzzer.
- Checagem da exibição correta dos dados no display.
- Testes de ruído ambiente em diferentes níveis.
- Validação do autoteste e modo BOOTSEL.

Discussão dos Resultados

- O sistema apresentou medições consistentes e precisas em diversos ambientes.
- A filtragem implementada reduziu significativamente o ruído de fundo.
- A interface de usuário foi intuitiva, com respostas rápidas.
- O autoteste funcionou conforme o esperado, assegurando a integridade dos componentes.
- O projeto demonstrou alta confiabilidade, com margem de melhoria para futura adição de conectividade sem fio.

5. Referências

1. **BitDogLab Datasheet – Especificação da placa**. Disponível em: https://github.com/BitDogLab/BitDogLab. Acesso em: 23 fev. 2025.
2. **Raspberry Pi Pico W Datasheet – Documentação do microcontrolador.** Disponível em: https://datasheets.raspberrypi.com/rp2040/rp2040-datasheet.pdf. Acesso em: 23 fev. 2025.
3. **Raspberry Pi Foundation.** *Pico series documentation*. Raspberry Pi. Disponível em: https://www.raspberrypi.com/documentation/microcontrollers/pico-series.html. Acesso em: 23 fev. 2025.
4. **CUGNASCA**, **Carlos Eduardo**. *Metodologia de Sistemas Embarcados*. Disponível em: https://integra.univesp.br/courses/2710/pages/texto-base-projetos-de-sistemas-embarcados-%7C-carlos-eduardo-cugnasca. Acesso em: 23 fev. 2025.
5. **GANSSLE**, Jack. *The Art of Designing Embedded Systems*.
6. **WOLF**, Wayne. *Embedded Systems Design*.
7. **Projetos correlatos consultados**.

   Monitoramento Automotivo: TCC – Sistema Embarcado para Monitoramento Automotivo em Tempo Real

   Monitoramento de Variáveis Físicas: Tese – Sistema Embarcado para Monitoramento de Variáveis Físicas do Rotor de Máquinas Elétricas (UFRN)

   Gerenciamento de Redes Inteligentes de Energia: Projeto – Sistema Embarcado para Gerenciamento de Redes Inteligentes de Energia

   Coleta de Dados para Diagnóstico Remoto: TCC – Desenvolvimento de um Sistema Embarcado para Coleta de Dados em Drones (UFPB)

   Desafios no Desenvolvimento de Sistemas Embarcados: Artigo – Projeto de Sistema Embarcado: Desafios do Desenvolvimento de Hardware e Software

   Arquitetura Multi-Placa em Sistemas Embarcados: Artigo – Arquitetura de Sistemas Embarcados: Quando Seu Produto Possui Múltiplas PCBs

6.Entrega

- **Código-fonte:** Repositório GitHub.

  https://github.com/LucaScripts/Projeto_Emcarcado_Ruido.git

- **Vídeo de Demonstração:** Link no Google Drive.

  https://drive.google.com/file/d/1NCCcU1p1Ax-v9K4eb8CkOadhHfP7VtG6/view?usp=sharing