INSTITUTO HARDWARE BR

Dennis Lopes da Silva

Aplicações de Internet das Coisas e Sistemas Embarcados

São Paulo - SP

2025

Sumário

1. Escopo do projeto:

Apresentação do projeto:

Este projeto tem como objetivo o desenvolvimento de um medidor de intensidade sonora que utiliza um microfone conectado à placa Bitdoglab para medir a intensidade de sons ambiente. A intensidade é calculada a partir das leituras do ADC (Conversor Analógico para Digital), e o valor é exibido em um display OLED SSD1306. O medidor de intensidade sonora pode ser utilizado em diversas aplicações, como monitoramento de níveis de ruído e avaliação de ambientes sonoros.

Objetivos do projeto:

- Desenvolver um medidor de intensidade sonora utilizando a placa Bitdoglab.
- Capturar sinais de áudio via um microfone e processar os dados utilizando um ADC.
- Exibir a intensidade sonora no display OLED em uma escala de 0 a 100 (por questões de tempo não foi possível implementar a medida em decibéis)
- Proporcionar uma interface visual simples e acessível para monitoramento dos níveis de intensidade sonora.

Principais requisitos:

- Placa Bitdoglab para processamento e controle.
- Microfone para captura de som.
- Display OLED para exibição dos dados.
- Leitor ADC da placa para conversão dos sinais analógicos em dados digitais.
- DMA (Acesso Direto à Memória) para otimizar a captura de amostras do ADC.
- Computador com Visual Studio e extensão do Raspberry Pi Pico para desenvolvimento e testes do dispositivo.

Descrição do funcionamento:

No projeto, a placa Bitdoglab ao ter o seu botão A acionado, realiza leituras dos sinais analógicos capturados pelo microfone por um tempo de 10 segundos, convertendo-os para valores digitais utilizando o ADC da placa Bitdoglab. As amostras de áudio são processadas e a potência média, bem como a de pico, é calculada. A partir desses valores, a intensidade do som é convertida para uma escala de 0 a 100. Em seguida, este valor é exibido no display OLED. A cada novo ciclo, o medidor exibe tanto o valor de pico quanto o valor médio da intensidade sonora.

Os ciclos são iniciados através de um botão de controle que permite ao usuário ativar a captura de dados, tornando o dispositivo interativo e fácil de usar.

Justificativa:

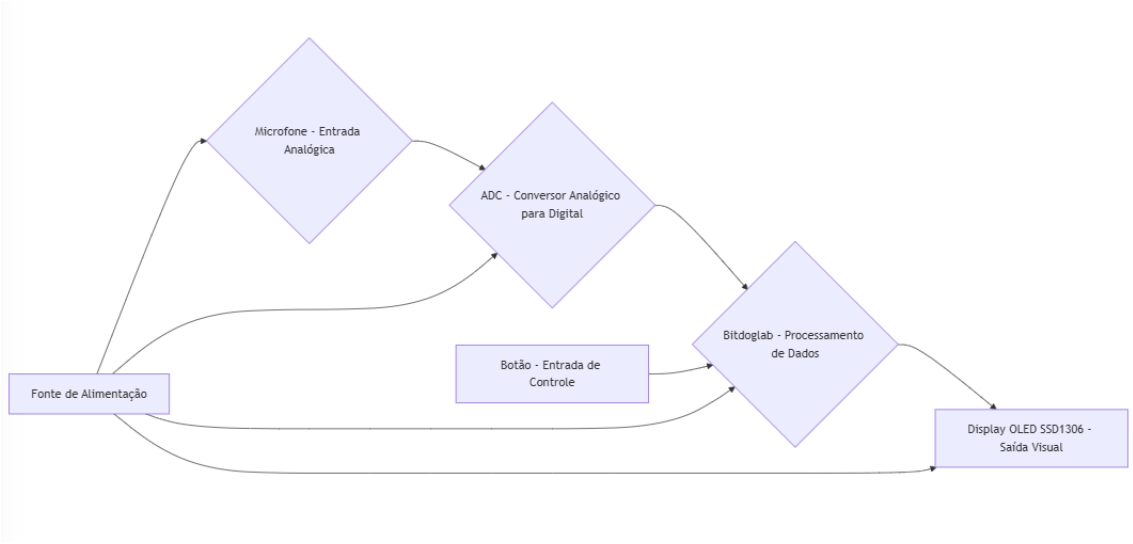
Este projeto se justifica pela crescente necessidade de monitoramento de níveis de ruído, especialmente em ambientes urbanos, indústrias e espaços públicos. A medição da intensidade sonora é essencial para garantir o bem-estar das pessoas, identificar fontes de poluição sonora e, em alguns casos, cumprir regulamentos de níveis de ruído. O uso de uma plataforma como a Bitdoglab torna o desenvolvimento mais acessível e flexível, permitindo facilmente adaptação a diversas necessidades.

Originalidade:

Embora existam projetos e dispositivos comerciais que medem a intensidade sonora, como medidores de nível de pressão sonora (SPL) e medidores de decibéis, este projeto se destaca pela sua implementação de baixo custo utilizando a placa Bitdoglab, permitindo uma solução acessível e personalizável. Além disso, a implementação de um display OLED para mostrar em tempo real os valores de intensidade sonora e a interação via botão adiciona valor à experiência do usuário, tornando seu uso mais intuitivo e funcional.

2. Hardware

Diagrama de Blocos:



Função dos blocos:

1. Microfone (entrada analógica):

Captura o som ambiente e converte-o em sinal elétrico analógico.

3. ADC (Conversor Analógico para Digital):

Converte o sinal analógico do microfone em valor digital, que pode ser processado pela Bitdoglab.

3. Bitdoglab (Processamento de Dados):

Processa os dados digitais provenientes do ADC, realiza cálculos e mapeia os resultados para uma escala de decibéis e controla o display OLED e os processos de leitura de dados. Tudo isso através do microcontrolador Raspberry Pi Pico.

4. Display OLED SSD1306 (saída visual):

Exibe a informações sobre a coleta de dados e os resultados da coleta: valor médio e valor de pico.

5. Botão (entrada de controle):

Permite ao usuário ativar e controlar o processo de medição de intensidade sonora.

6. Fonte de Alimentação:

Fornece energia para o sistema (Bitdoglab, Display OLED, Microfone, etc.).

Configuração de cada bloco:

Microfone: O microfone é conectado ao pino analógico da Bitdoglab. O sinal gerado pelo microfone é enviado ao ADC da Bitdoglab para conversão.

ADC: O ADC da Bitdoglab é configurado para realizar a conversão dos sinais analógicos do microfone para valores digitais. A taxa de amostragem e resolução (12 bits) são ajustadas para garantir a precisão da medição de som.

Bitdoglab: A Bitdoglab é configurada para processar os sinais digitais do ADC. Ela calcula a potência média do sinal, mapeia os valores para uma escala de decibéis e envia os dados para o display OLED.

Display OLED SSD1306: A Bitdoglab controla a exibição de informações no display OLED via interface I2C. O texto mostras os valores de pico/média e também status do dispositivo.

Botão: O botão de controle é conectado a um pino GPIO da Bitdoglab, que é configurado como entrada com pull-up. Ele permite que o usuário inicie ou pare a medição de som.

Fonte de Alimentação: A alimentação do sistema é provida por uma fonte DC que atende aos requisitos de voltagem e corrente para os componentes conectados à Bitdoglab.

Especificações técnicas:

Microfone:

Tipo: Microfone analógico (GY-MAX4466)

Faixa de Frequência: 20 Hz - 20 kHz (para capturar áudio dentro da faixa de audição humana).

Impedância: Depende do modelo do microfone (geralmente entre 100Ω e 10kΩ).

ADC (Conversor Analógico para Digital):

Resolução: 12 bits.

Canal de entrada: Configurado para o pino do microfone.

Taxa de amostragem: Ajustável para captar os dados em tempo real (frequência de amostragem variável, com 200 amostras como exemplo).

Bitdoglab:

Processador: ARM Cortex-M0.

Tensão de Operação: 3.3V.

Memória RAM: 256 KB.

Display OLED SSD1306:

Tamanho: 128x64 pixels.

Interface: I2C.

Tensão de Operação: 3.3V.

Consumo de Energia: Baixo, adequado para sistemas embarcados.

Botão:

Tipo: Push-button.

Tensão de Operação: 3.3V.

Configuração: Entrada digital com pull-up.

Fonte de Alimentação:

Tipo: Fonte DC 5V (por exemplo, adaptador USB ou bateria recarregável).

Tensão de Operação: 5V (regulada para 3.3V para os componentes da Bitdoglab).

Descrição da Pinagem Utilizada:

Componente	Pino na Bitdoglab	Função
Microfone	Canal ADC 2 (Pino 26)	Entrada analógica para captura de áudio
Display OLED (I2C)	SDA: GPIO 14	Comunicação I2C (Dados - SDA)
	SCL: GPIO 15	Comunicação I2C (Clock - SCL)
Botão	GPIO 5	Entrada digital (com pull-up interno)
Fonte de Alimentação	Conector DC 5V	Alimentação do circuito

4. Software

Blocos Funcionais do Software:

Bloco Funcional	Função	Tarefas
Leitura do Microfone	Captura de dados do microfone via ADC	Iniciar DMA, capturar amostras, Desligar ADC após captura
Processamento do Sinal	Cálculo de potência e intensidade do sinal	Calcular RMS, ajustar valores, Calcular intensidade
Conversão de escala	Conversão da intensidade para escala de 0 a 100	Conversão da intensidade para escala de 0 a 100
Controle do Display OLED	Exibição das informações no display OLED	Atualizar valores no display OLED
Leitura do Botão	Controle manual para iniciar e parar medições	Detectar pressionamento do botão, iniciar o processo de medição
Gerenciamento de Fluxo (Loop Principal)	Controle do fluxo do programa	Alternar entre capturar dados e exibir resultados
Delay e Temporização	Implementar delays para controle de temporização	Realizar pauses entre amostras e atualizações de display

Descrição das funcionalidades:

Leitura do Microfone (Amostragem ADC)

Função: Captura as amostras do microfone através do ADC. O DMA é utilizado para transferir as amostras diretamente para a memória, permitindo que o processador se concentre em outras tarefas.

Tarefas:

Iniciar o ADC para leitura contínua.

Usar DMA para coletar as amostras de áudio e armazená-las no buffer adc_buffer.

Esperar a conclusão do DMA e desligar o ADC após a captura.

Processamento do Sinal (Cálculo de Potência e Intensidade)

Função: Processa as amostras de áudio para calcular a potência média e a intensidade do som. A intensidade é calculada com base na potência média ajustada para a escala de 0 a 100.

Tarefas:

Calcular a potência média das amostras (mic_power()).

Ajustar os valores para a escala de 0 a 3.3V com a função ADC_ADJUST().

Calcular a intensidade a partir da potência média usando a função get_intensity().

Conversão de escala

Função: Converte a intensidade medida (em unidades proporcionais) para uma escala de 0 a 100

Tarefas:

Aplicar a fórmula de conversão de intensidade.

Controle do Display OLED (Interface Gráfica)

Função: Exibe as informações sobre a intensidade sonora no display OLED. O display é atualizado com as medições de potência e a intensidade sonora.

Tarefas:

Desenhar strings com valores no display OLED para mostrar os resultados das medições.

Controlar o fluxo de exibição, atualizando o display após cada nova medição.

Leitura do Botão (Controle Manual)

Função: Permite ao usuário iniciar ou parar o processo de medição pressionando o botão.

Tarefas:

Detectar quando o botão é pressionado (com pull-up).

Iniciar o processo de amostragem quando o botão for pressionado.

Gerenciamento de Fluxo (Loop Principal)

Função: Gerencia o fluxo do programa, controlando as ações de captura de dados e exibição dos resultados no display OLED.

Tarefas:

Controlar a sequência de amostragem e exibição com base no estado do botão.

Aguardar intervalos de tempo para garantir que o display seja atualizado e o processo de medição ocorra de forma eficiente.

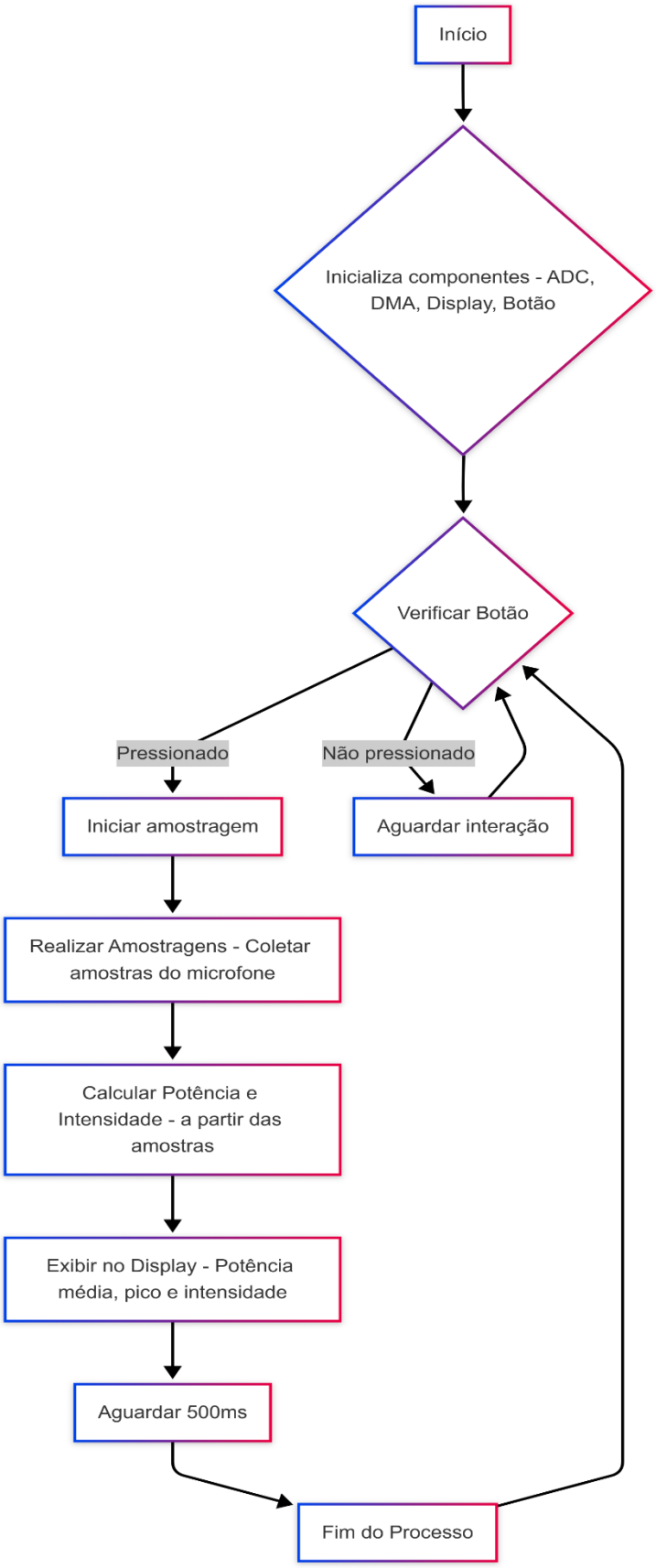
Delay e Temporização

Função: Controla o tempo entre as amostras e as atualizações do display.

Tarefas:

Realizar pausas entre as amostras para evitar sobrecarga no sistema e garantir a atualização eficiente do display.

Fluxograma:


Início
Inicializa componentes - ADC, DMA, Display, Botão
Verificar Botão
Pressionado
Não pressionado
Iniciar amostragem
Aguardar interação
Realizar Amostragens - Coletar amostras do microfone
Calcular Potência e Intensidade - a partir das amostras
Exibir no Display - Potência média, pico e intensidade
Aguardar 500ms
Fim do Processo

Etapas do Fluxograma:

1. **Início:**

Inicializa os componentes do sistema (ADC, DMA, Display, Botão).

2. **Verificar Botão**

Se o botão for pressionado: Iniciar o processo de amostragem.

3. **Realizar Amostragens**

Coletar amostras do microfone utilizando ADC.

4. **Calcular Potência e Intensidade**

Calcular a potência média e a intensidade a partir das amostras.

5. **Exibir no Display**

Mostrar os resultados (potência média, pico e intensidade) no display OLED.

6. **Esperar**

Aguardar um intervalo de tempo (500 ms).

7. **Fim do Processo**

Voltar para o início e aguardar nova interação do botão.

Principais variáveis:

- ***adc_buffer[SAMPLES]:*** Buffer onde as amostras do ADC são armazenadas.
- ***avg:*** Armazena o valor da potência média calculada.
- ***avg_sum:*** Acumula os valores de potência média para cálculo da média final.
- ***peak:*** Armazena o valor de pico da intensidade medida.
- ***intensity:*** Armazena a intensidade do som calculada.
- ***text[]:*** Array que contém as strings a serem exibidas no display OLED.
- ***dma_channel:*** Canal utilizado para a configuração do DMA.
- ***dma_cfg:*** Configuração do canal DMA.
- ***ssd:*** Buffer de dados do display OLED.
- ***frame_area:*** Estrutura que define a área de renderização no display OLED.
- ***y:*** Variável que controla a posição vertical do texto no display.

Estrutura e formato dos dados:

O buffer ***adc_buffer[]*** armazena as amostras do ADC, que são 16 bits (uint16_t) para cada amostra. O valor de intensidade é calculado com base nas amostras de áudio.

A intensidade calculada é um valor entre 0 e 100, representando a intensidade do som. O valor de potência é um valor numérico em float, representando a potência média das amostras.

Protocolo de comunicação:

A comunicação com o display OLED é feita via I2C. Os comandos de controle e os dados são enviados para o display utilizando o protocolo I2C.

5. Execução do projeto

Metodologia:

1. **Realização de pesquisas:**

    a) **Pesquisa sobre Microfone e ADC:** Inicialmente, foi necessário entender como capturar sinais de áudio usando um microfone. Buscou-se informações sobre o uso de ADC (Conversor Analógico para Digital) para leitura das amostras de som e como configurar o ADC na placa Bitdoglab.
    b) **Pesquisas sobre Display OLED e I2C:** Como o projeto inclui uma interface gráfica para exibir os dados, foi necessário estudar como controlar o display OLED utilizando a comunicação I2C. A pesquisa foi focada em bibliotecas de controle do display SSD1306, que é amplamente utilizado em projetos embarcados.
    c) **Escala de Intensidade e Conversão para dB:** Outra pesquisa envolveu entender como a intensidade do som é medida, e como converter a intensidade calculada em valores mais familiares como decibéis (dB). Devido ao tempo exíguo do projeto, foi decidido que usaríamos uma escala de 0 a 100 dentro dos limites físicos do microfone da placa Bitdoglab.

2. **Implementação no Hardware do projeto Embarcatech:**

    a) **Placa Bitdoglab:** A escolha da Bitdoglab foi feita devido ao requisito do projeto Embarcatech e a facilidade de ter atuadores e sensores já disponíveis em uma placa de prototipação.
    b) **Microfone (canal do ADC):** Utilizou-se o microfone conectado ao canal 2 do ADC da Bitdoglab, garantindo boa captação de sinais analógicos com resolução suficiente para capturar variações de intensidade sonora.
    c) **Display OLED SSD1306:** O display SSD1306 foi escolhido por ser amplamente utilizado em projetos de microcontroladores embarcados e por sua interface simples via I2C, o que facilita a integração com a placa Bitdoglab.

3. **Definição das Funcionalidades do Software:**

    O software foi projetado para realizar a captura das amostras de áudio, processá-las para calcular a potência média e intensidade, e exibir os resultados no display OLED. Além disso, foi necessário implementar o controle do botão para iniciar o processo de medição e configurar a comunicação I2C com o display OLED.

4. **Instalação da IDE**

    A IDE utilizada foi o **Visual Studio Code** com a extensão do **Pico SDK**. A configuração inicial envolveu a instalação do SDK e a definição do ambiente de desenvolvimento para o microcontrolador RP2040. A instalação do compilador GCC também foi realizada.

5. **Programação na IDE**

    Durante a programação, as bibliotecas necessárias para o controle do ADC, DMA e I2C foram incluídas. A lógica de aquisição e processamento dos dados foi implementada, além da exibição dos resultados no display OLED. Testes Iterativos: Durante o desenvolvimento, as funções foram testadas iterativamente. Primeiro, o ADC foi testado separadamente para garantir que as leituras estavam corretas, e, em seguida, a comunicação I2C com o display foi testada para garantir que as informações fossem exibidas corretamente.

6. **Depuração**

   A depuração foi feita utilizando o terminal serial para verificar os valores das variáveis durante a execução. Também foram feitos testes com o display para garantir que as informações aparecessem corretamente, ajustando a posição e o formato do texto exibido.

Testes de validação:

Testes de Leitura do Microfone (ADC):

O primeiro teste realizado foi verificar se o microfone estava sendo corretamente amostrado pelo ADC. Foram feitas leituras diretas dos valores de ADC e os resultados foram analisados com um osciloscópio virtual, para garantir que os sinais capturados correspondiam a sons de diferentes intensidades.

Testes de Potência e Intensidade:

Foi realizada uma série de medições com diferentes níveis de som (sons baixos, médios e altos) para garantir que o cálculo da potência e intensidade estava sendo realizado corretamente. Os valores calculados foram validados em comparação com o esperado para as diferentes intensidades.

Testes de Exibição no Display OLED:

Foram feitos testes para garantir que os valores de potência e intensidade estivessem sendo exibidos corretamente no display. Isso incluiu testar a atualização das informações e verificar a clareza e precisão dos números mostrados.

Testes de Estabilidade do Sistema:

O sistema foi deixado em funcionamento contínuo por longos períodos para garantir que não ocorressem falhas ou travamentos. Isso incluiu verificar a resposta do sistema quando o botão era pressionado para reiniciar a medição.

Testes com Diferentes Níveis de Ruído:

Testes foram realizados em ambientes com diferentes níveis de ruído para verificar se o sistema respondia adequadamente e se o cálculo da intensidade estava sensível às variações de volume.

Discussão dos resultados:

Confiabilidade dos Resultados:

Os testes realizados mostraram que o sistema é confiável para medir a intensidade sonora em ambientes de diferentes níveis de ruído. As medições de potência média e pico foram precisas, e a exibição das informações no display OLED foi clara e consistente.

A precisão da conversão de potência para a escala de intensidade (0 a 100) foi considerada satisfatória, proporcionando ao usuário uma medida útil de volume.

Desempenho:

O sistema teve bom desempenho no que diz respeito à captura e processamento das amostras em tempo real. O uso do DMA garantiu uma boa eficiência de processamento, permitindo que o microcontrolador se concentrasse em outras tarefas enquanto as amostras eram capturadas e armazenadas.

Aplicabilidade:

O projeto tem alta aplicabilidade para diversas situações que exigem medições de intensidade sonora em tempo real, como monitoramento de níveis de ruído em ambientes urbanos ou industriais, ou mesmo em situações educativas onde a medição da intensidade sonora pode ser usada para demonstrar conceitos de física relacionados ao som.

O sistema poderia ser expandido para incluir funções adicionais, como a detecção de picos de ruído e alertas, ou a integração com outros sistemas de monitoramento de ambientes.

Limitações:

A principal limitação observada foi a necessidade de calibração manual para converter a leitura de intensidade para dB.

A precisão da medição também depende da qualidade do microfone utilizado, sendo que um microfone de maior qualidade resultaria em medições mais precisas.

Conclusão:

O projeto atingiu seus objetivos de medir a intensidade sonora e exibir os resultados em tempo real de forma clara. A escolha do hardware e a implementação do software se mostraram eficazes para a realização da tarefa proposta. A conclusão é que o sistema é confiável, aplicável e pode ser expandido para outras funcionalidades de medição e monitoramento sonoro.

Vídeo no YouTube:

https://youtu.be/AeRc5TTNsi4

Projeto no GitHub:

https://github.com/dennislopes/embarcatech

REFERÊNCIAS

1. **SMART KITS**. Guia de como usar o ADC no ESP32. Disponível em: https://blog.smartkits.com.br/adc-no-esp32-como-usar/. Acesso em: 10/02/2024.
2. **Documentação do RP2040 (Pico SDK)**. Disponível em: https://raspberrypi.github.io/pico-sdk-doxygen/. Acesso em: 10/02/2024.
3. **Microfone Analógico**. Como usar microfones com conversores analógicos para digitais. Disponível em: https://www.circuitdigest.com/article/how-to-interface-an-analog-microphone-with-arduino. Acesso em: 10/02/2024.
4. **SSD1306 OLED Display**. Guia completo para uso do SSD1306 com microcontroladores. Disponível em: https://www.electronicwings.com/nodemcu/ssd1306-oled-display-interfacing-with-nodemcu. Acesso em: 10/02/2024.
5. **Pico SDK Documentation**. Documentação oficial sobre DMA e ADC no RP2040. Disponível em: https://raspberrypi.github.io/pico-sdk-doxygen/. Acesso em: 10/02/2024.
6. **Introdução à acústica e medição de níveis de som**. Princípios fundamentais da acústica e como medir intensidade sonora. Disponível em: https://www.acoustics.org/press/2012/03/18/acoustical-sound-measurement-fundamentals/. Acesso em: 10/02/2024.
7. **Arduino Sound Meter**. Como criar um medidor de som com o Arduino. Disponível em: https://www.arduino.cc/en/Tutorial/AudioSensor. Acesso em: 10/02/2024.
8. **Digital Sound Level Meter Using Microcontroller**. Projeto de medidor de nível de som digital utilizando microcontroladores. Disponível em: https://www.circuitstoday.com/digital-sound-level-meter-using-microcontroller. Acesso em: 10/02/2024.
9. **Biblioteca SSD1306 I2C para C**. Implementação e exemplos de uso. Disponível em: https://github.com/greiman/SSD1306. Acesso em: 10/02/2024.
10. **BitDogLab-C**. Repositório oficial do BitDogLab. Disponível em: https://github.com/BitDogLab/BitDogLab-C. Acesso em: 10/02/2024.
11. **BitDogLab - Uma jornada educativa com eletrônica embarcados e IA**. Disponível em: https://embarcados.com.br/bitdoglab-uma-jornada-educativa-com-eletronica-embarcados-e-ia/. Acesso em: 10/02/2024.

Apêndice: Código Fonte

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "pico/stdlib.h"
#include "hardware/adc.h"
#include "hardware/dma.h"
#include "hardware/pio.h"
#include "pico/binary_info.h"
#include "inc/ssd1306.h"
#include "inc/ssd1306_i2c.c"
#include "hardware/i2c.h"

#define MIC_CHANNEL 2                    // Canal do microfone no ADC.
#define MIC_PIN (26 + MIC_CHANNEL)       // Pino do microfone no ADC.
#define ADC_CLOCK_DIV 96.f               // Parâmetros e macros do ADC.
#define SAMPLES 200                      // Número de amostras que serão feitas
do ADC.
#define ADC_ADJUST(x) (x * 3.3f / (1 << 12u) - 1.65f) // Ajuste do valor do ADC
para Volts.
#define ADC_MAX 3.3f
#define ADC_STEP (3.3f / 5.f)            // Intervalos de volume do microfone.
#define abs(x) ((x < 0) ? (-x) : (x))
#define BUTTON_PIN 5     // Pino do botão A

const uint I2C_SDA = 14;
const uint I2C_SCL = 15;
ssd1306_t oled;

// Canal e configurações do DMA
uint dma_channel;
dma_channel_config dma_cfg;

// Buffer de amostras do ADC.
uint16_t adc_buffer[SAMPLES];
```

```
void sample_mic();
float mic_power();
uint8_t get_intensity(float v);

/**
 * Realiza as leituras do ADC e armazena os valores no buffer.
 */
void sample_mic()
{
  adc_fifo_drain(); // Limpa o FIFO do ADC.
  adc_run(false);   // Desliga o ADC (se estiver ligado) para configurar o DMA.

  dma_channel_configure(dma_channel, &dma_cfg,
                adc_buffer,     // Escreve no buffer.
                &(adc_hw->fifo), // Lê do ADC.
                SAMPLES,        // Faz "SAMPLES" amostras.
                true            // Liga o DMA.
  );

  // Liga o ADC e espera acabar a leitura.
  adc_run(true);
  dma_channel_wait_for_finish_blocking(dma_channel);

  // Acabou a leitura, desliga o ADC de novo.
  adc_run(false);
}

/**
 * Calcula a potência média das leituras do ADC. (Valor RMS)
 */
float mic_power()
{
  float avg = 0.f;

  for (uint i = 0; i < SAMPLES; ++i)
    avg += adc_buffer[i] * adc_buffer[i];

  avg /= SAMPLES;
  return sqrt(avg);
}
```

```
/**
 * Calcula a intensidade do volume registrado no microfone, usando a tensão.
 */
uint8_t get_intensity(float v)
{
  uint count = 0;

  while ((v -= ADC_STEP / 20) > 0.f)
    ++count;

  return count;
}

/**
 *  Programa principal
 */
int main()
{
  stdio_init_all();

  // Inicialização do i2c
  i2c_init(i2c1, ssd1306_i2c_clock * 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);

  // Processo de inicialização completo do OLED SSD1306
  ssd1306_init();

  // Preparar área de renderização para o display (ssd1306_width pixels por
ssd1306_n_pages páginas)
  struct render_area frame_area = {
    start_column : 0,
    end_column : ssd1306_width - 1,
    start_page : 0,
    end_page : ssd1306_n_pages - 1
  };

  calculate_render_area_buffer_length(&frame_area);
```

```
 // zera o display inteiro
 uint8_t ssd[ssd1306_buffer_length];
 memset(ssd, 0, ssd1306_buffer_length);
 render_on_display(ssd, &frame_area);

 adc_gpio_init(MIC_PIN);
 adc_init();
 adc_select_input(MIC_CHANNEL);

 adc_fifo_setup(
   true,  // Habilitar FIFO
   true,  // Habilitar request de dados do DMA
   1,    // Threshold para ativar request DMA é 1 leitura do ADC
   false, // Não usar bit de erro
   false  // Não fazer downscale das amostras para 8-bits, manter 12-bits.
 );

 adc_set_clkdiv(ADC_CLOCK_DIV);

 // Tomando posse de canal do DMA.
 dma_channel = dma_claim_unused_channel(true);

 // Configurações do DMA.
 dma_cfg = dma_channel_get_default_config(dma_channel);
 channel_config_set_transfer_data_size(&dma_cfg, DMA_SIZE_16); // Tamanho
da transferência é 16-bits (usamos uint16_t para armazenar valores do ADC)
 channel_config_set_read_increment(&dma_cfg, false);         // Desabilita
incremento do ponteiro de leitura (lemos de um único registrador)
 channel_config_set_write_increment(&dma_cfg, true);         // Habilita
incremento do ponteiro de escrita (escrevemos em um array/buffer)
 channel_config_set_dreq(&dma_cfg, DREQ_ADC);               // Usamos a
requisição de dados do ADC

 // Configuração do GPIO do Botão A como entrada com pull-up interno
 gpio_init(BUTTON_PIN);
 gpio_set_dir(BUTTON_PIN, GPIO_IN);
 gpio_pull_up(BUTTON_PIN);


 while (true)
 {
```

```
  if (gpio_get(BUTTON_PIN) == 0){
  float avg;
  float avg_sum = 0;
  float peak = 0;
  int count = 0;
  uint intensity = 0;

  char *text[] = {
   "Realizando    ",
   "Amostragem  ",
   "            ",
   "            ",
   "            ",
   "            "};

 int y = 0;
 for (uint i = 0; i < count_of(text); i++)
 {
  ssd1306_draw_string(ssd, 5, y, text[i]);
  y += 8;
 }
 render_on_display(ssd, &frame_area);

  for (int i = 0; i < 20; i++)
  {
   // Realiza uma amostragem do microfone.
   sample_mic();
   // Pega a potência média da amostragem do microfone.
   avg = mic_power();
   avg = 2.f * abs(ADC_ADJUST(avg)); // Ajusta para intervalo de 0 a 3.3V.
(apenas magnitude, sem sinal)
   avg_sum += avg;
   if (avg > peak)
   {
    peak = avg;
   }
   count++;
   sleep_ms(500);
  }
```

```
   float avg_final = avg_sum / count;

   for (int i = 0; i < 10; i++)
   {
    uint intensity = get_intensity(avg_final); // Calcula intensidade a ser
mostrada na matriz de LEDs.
    uint peak_intensity = get_intensity(peak); // Calcula intensidade a ser
mostrada na matriz de LEDs.
    char intensity_str[10];
    sprintf(intensity_str, "%d", intensity);
    char peak_str[10];
    sprintf(peak_str, "%d", peak_intensity); // Converte o valor de pico para
string com 2 casas decimais

    char *text[] = {
      " Decibelimetro  ",
      "            ",
      "Valor Medio    ",
      intensity_str,
      "Valor de Pico    ",
      peak_str};

    int y = 0;
    for (uint i = 0; i < count_of(text); i++)
    {
     ssd1306_draw_string(ssd, 5, y, text[i]);
     y += 8;
    }
    render_on_display(ssd, &frame_area);

    sleep_ms(500);
   }
  }
}
}
```