**1. Título do Projeto**

**Monitor de Ruído Inteligente com Autoteste, Feedback Multissensor e Registro de Dados em JSON**

**2. Apresentação do Projeto**

Este projeto tem como objetivo desenvolver um sistema embarcado utilizando a placa BitDogLab (baseada no Raspberry Pi Pico W) para monitorar os níveis de ruído ambiente. As principais funções do sistema são:

* **Medição do Ruído Ambiente:** Utiliza o microfone de eletreto para captar os níveis sonoros.
* **Feedback Visual e Sonoro:** Exibe os dados medidos em um display OLED e aciona os LEDs (vermelho para alerta, verde para operação normal) e o buzzer para emitir um alerta sonoro.
* **Autoteste Integrado:** Permite que o sistema execute um autoteste, acionado pelo Botão A, onde o buzzer gera um som e o microfone verifica a captação, garantindo a integridade dos sensores.
* **Registro de Dados:** Armazena os dados coletados em um arquivo JSON (exemplo: "noise\_data.json"), permitindo posterior análise e histórico das medições.
* **Expansibilidade:** Possibilidade futura de integração com comunicação Wi-Fi para envio dos dados a um servidor ou dashboard remoto.

**3. Objetivos do Projeto**

* **Monitoramento Contínuo:** Capturar e exibir em tempo real os níveis de ruído ambiente.
* **Alerta Imediato:** Notificar visualmente (LEDs) e sonoramente (buzzer) quando os níveis ultrapassarem um limiar crítico.
* **Verificação do Sistema:** Executar um autoteste (acionado pelo Botão A) para validar a operação do microfone e do buzzer.
* **Registro dos Dados:** Armazenar cada leitura em um arquivo JSON para registro histórico e análises futuras.
* **Potencial de Expansão:** Integrar, futuramente, módulos de comunicação para monitoramento remoto.

**4. Descrição do Funcionamento**

**Fluxo Geral:**

1. **Inicialização:**
   * Configuração dos periféricos: display OLED, microfone (ADC), LEDs, buzzer, Botão A (com pull-up) e, futuramente, módulo Wi-Fi.
2. **Autoteste (Acionado pelo Botão A):**
   * Ao pressionar o Botão A, o sistema mede o nível de ruído ambiente, gera um tom controlado com o buzzer e mede novamente o som captado pelo microfone.
   * Compara os valores para verificar se o som foi captado corretamente, exibindo “Self-Test: OK” ou “Self-Test: FAIL” no OLED.
3. **Monitoramento Contínuo:**
   * Realiza leituras periódicas do microfone.
   * Exibe os níveis de ruído no display OLED.
   * Se o nível ultrapassar o limiar definido, aciona o LED vermelho e o buzzer; caso contrário, mantém o LED verde aceso.
4. **Registro de Dados:**
   * Cada leitura de nível de ruído é armazenada em um arquivo JSON (por exemplo, com o formato:  
     {"reading": 1, "noise": 1523}), possibilitando a análise posterior.
5. **Comunicação Wi-Fi (Futura Expansão):**
   * Implementação opcional para envio dos dados e alertas a um servidor ou dashboard remoto.

**5. Especificação do Hardware**

Utilizando a BitDogLab, os componentes utilizados são:

* **Microfone de Eletreto:**
  + Conectado ao GPIO28 (canal ADC A2); capta o nível de ruído ambiente.
* **Display OLED (SSD1306, 128x64):**
  + Conectado via I2C (GPIO14 para SDA e GPIO15 para SCL); exibe informações e resultados do autoteste.
* **LED RGB:**
  + **Vermelho (GPIO13):** Acionado em caso de alerta.
  + **Verde (GPIO11):** Indica operação normal.
  + **Azul (GPIO12):** Reservado para futuras funcionalidades.
* **Buzzer:**
  + Conectado ao GPIO21; utilizado tanto para emissão do som de alerta quanto para o autoteste.
* **Botão A:**
  + Conectado ao GPIO5 (configurado com pull-up interno); aciona a rotina de autoteste.
* **Módulo Wi-Fi (integrado no Raspberry Pi Pico W):**
  + Para futura implementação de envio remoto dos dados.
* **Sistema de Armazenamento de Dados:**
  + Utiliza um sistema de arquivos (por exemplo, um SD card conectado via SPI ou um sistema de arquivos implementado na memória flash) para armazenar os dados no formato JSON.

**6. Especificação do Firmware**

**Principais Blocos de Software:**

* **Leitura do ADC:**
  + Captura os valores analógicos do microfone e converte para o nível de ruído.
* **Controle do Display OLED:**
  + Exibe as medições de ruído, mensagens do autoteste e alertas.
* **Gerenciamento dos LEDs e Buzzer:**
  + Aciona os LEDs (vermelho e verde) e o buzzer de acordo com os níveis de ruído detectados.
* **Rotina de Autoteste:**
  + Acionada pelo Botão A:
    - Mede o ruído ambiente;
    - Gera um tom controlado com o buzzer;
    - Realiza nova medição;
    - Compara os valores e exibe o resultado no OLED.
* **Registro de Dados em JSON:**
  + Cada leitura é armazenada em um arquivo JSON (exemplo: "noise\_data.json"), com informações como número da leitura e nível de ruído.
* **Módulo de Comunicação Wi-Fi (Opcional/Futuro):**
  + Inicializa a conexão à rede;
  + Cria payloads em JSON para envio dos dados via MQTT ou HTTP.

**7. Metodologia e Desenvolvimento do Projeto**

**Etapas de Execução:**

1. **Planejamento e Definição:**
   * Levantamento dos requisitos funcionais e não funcionais.
   * Pesquisa de projetos correlatos e seleção dos componentes disponíveis na BitDogLab.
2. **Especificação de Hardware e Software:**
   * Elaboração dos diagramas de blocos e circuitos.
   * Definição das funcionalidades do firmware, incluindo autoteste, monitoramento, feedback e registro dos dados.
3. **Implementação:**
   * Desenvolvimento do código em linguagem C para o RP2040, integrando a leitura do ADC, controle do OLED, LEDs, buzzer, Botão A e a função de armazenamento dos dados em JSON.
4. **Testes e Validação:**
   * Execução do autoteste para verificar a integridade dos sensores.
   * Validação do monitoramento e dos alertas (visuais e sonoros).
   * Teste do armazenamento dos dados e verificação do arquivo JSON gerado.
5. **Documentação e Apresentação:**
   * Preparação do relatório final contendo o escopo, diagramas, fluxogramas, código-fonte, resultados dos testes e análise dos dados.
   * Disponibilização do código-fonte em um repositório (por exemplo, GitHub) e produção de um vídeo demonstrativo.

**8. Aplicações Industriais**

* **Segurança Ocupacional:**
  + Monitoramento dos níveis de ruído em ambientes industriais para proteger a saúde dos trabalhadores.
* **Manutenção Preditiva:**
  + Identificação de anomalias sonoras em máquinas, permitindo intervenções preventivas antes de falhas críticas.
* **Controle de Qualidade:**
  + Verificação de padrões sonoros em linhas de produção, auxiliando na detecção precoce de defeitos.
* **Monitoramento Remoto e Registro Histórico:**
  + Com o registro dos dados em JSON e a futura integração Wi-Fi, os dados podem ser enviados e analisados remotamente, permitindo um histórico detalhado das condições sonoras do ambiente.

**9. Possíveis Expansões Futuras**

* **Integração Completa com IoT:**
  + Implementação do módulo Wi-Fi para envio contínuo dos dados a um servidor ou dashboard remoto.
* **Dashboard Interativo:**
  + Desenvolvimento de uma interface web ou mobile para visualização em tempo real e análise histórica dos níveis de ruído.
* **Aprimoramento da Calibração e Registro:**
  + Implementação de algoritmos de calibração automática e armazenamento mais detalhado dos dados (incluindo timestamp, se integrado a um RTC).

**10. Referências**

1. **BitDogLab Datasheet** – Documento técnico fornecido pelo CEPEDI, contendo as especificações e o pinout da placa BitDogLab.
2. **Raspberry Pi Pico W Datasheet** – Informações oficiais sobre o microcontrolador RP2040 e as funcionalidades de rede do Pico W (disponível no site oficial da Raspberry Pi).
3. **CUGNASCA – Metodologia de Projeto em Sistemas Embarcados** – Referência utilizada durante a capacitação, que simplifica os passos do desenvolvimento de sistemas embarcados.
4. Ganssle, J. “The Art of Designing Embedded Systems” – Referência clássica para o desenvolvimento e melhores práticas em sistemas embarcados.
5. Wolf, W. “Embedded Systems Design” – Referência adicional sobre técnicas e ferramentas no desenvolvimento de sistemas embarcados.
6. **Documentação do FATFS e bibliotecas de arquivos em sistemas embarcados** – Referências para a implementação de sistemas de arquivos em microcontroladores, essenciais para o armazenamento de dados em JSON.