SoundAlert

Victor André de Moraes

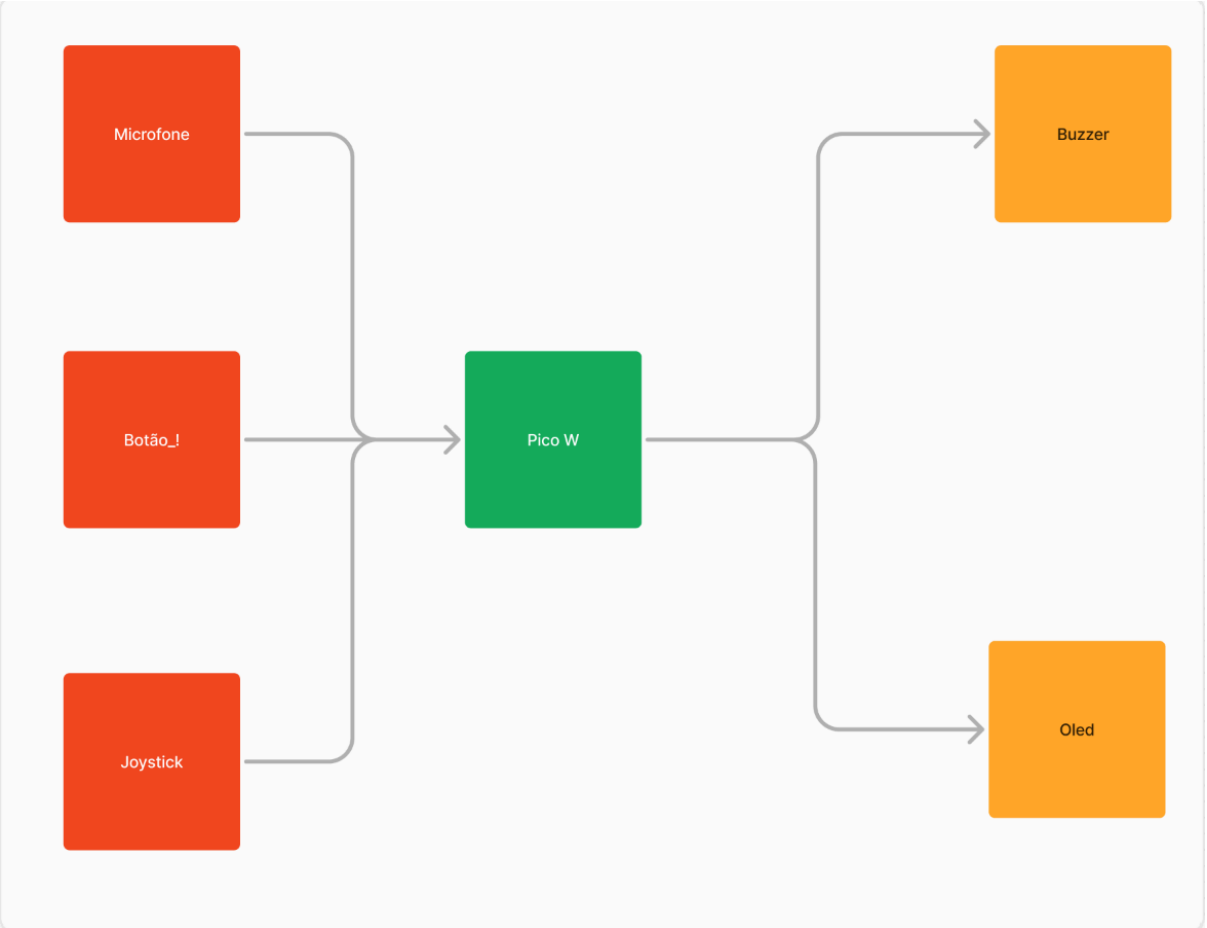
1) Escopo do Projeto

- **Apresentação do Projeto**
  - O projeto consiste em um sistema embarcado baseado na placa BitDogLab (Raspberry Pi Pico W) que monitora o nível de ruído ambiental. Quando o ruído fica abaixo de um limiar pré-definido, um alarme sonoro é acionado via buzzer. O usuário pode ajustar o volume do alarme com um joystick e visualizar informações em uma tela OLED.
- **Título do Projeto**
  - Sistema de Monitoramento Acústico com Alerta Adaptável (Sound Alert)
- **Objetivos do Projeto**
  - Detectar variações de ruído em tempo real.
  - Alertar o usuário quando o ruído estiver abaixo de um limiar configurável.
  - Permitir ajuste intuitivo do volume do alarme via joystick.
  - Exibir informações de interface (volume, status) em uma tela OLED.
- **Principais Requisitos**
  - Leitura contínua do nível de ruído via microfone.
  - Configuração do limiar de ruído com um botão (70% do valor atual).
  - Controle de volume do buzzer em incrementos de 10%.
  - Interface gráfica simples mas funcional.
  - Baixo consumo de recursos computacionais.
- **Descrição do Funcionamento**
  - O microfone capta o ruído ambiente, convertido em sinal analógico.
  - O Raspberry Pi Pico W processa o sinal e compara com o limiar definido.
  - Se o ruído ficar abaixo do limiar, os buzzers são acionados com volume ajustável.
  - O joystick regula o volume (0–100%), exibido na tela OLED.
  - Um botão salva o limiar como 70% do ruído atual.
- **Justificativa**
  - O projeto justifica-se pela necessidade de monitorar ambientes onde variações sonoras críticas podem indicar falhas em equipamentos (ex: máquinas industriais) ou eventos específicos (ex: término de operação de eletrodomésticos).
- **Projetos Semelhantes**
  - Embora existam sistemas similares baseados em sensores de vibração com IA (ex: semeq, sensor de manutenção preditiva [referência 1]), este projeto destaca-se por:
    - Custo reduzido: Uso de componentes simples (microfone analógico, buzzer).

- Simplicidade: Detecção baseada em limiares fixos, sem necessidade de algoritmos complexos.
- Aplicabilidade imediata: Funcionalidade direta para cenários como monitoramento de máquinas ou aviso sonoro em residências (ex: alerta quando uma fritadeira elétrica desliga).

2) Hardware

- **Diagrama em Blocos**



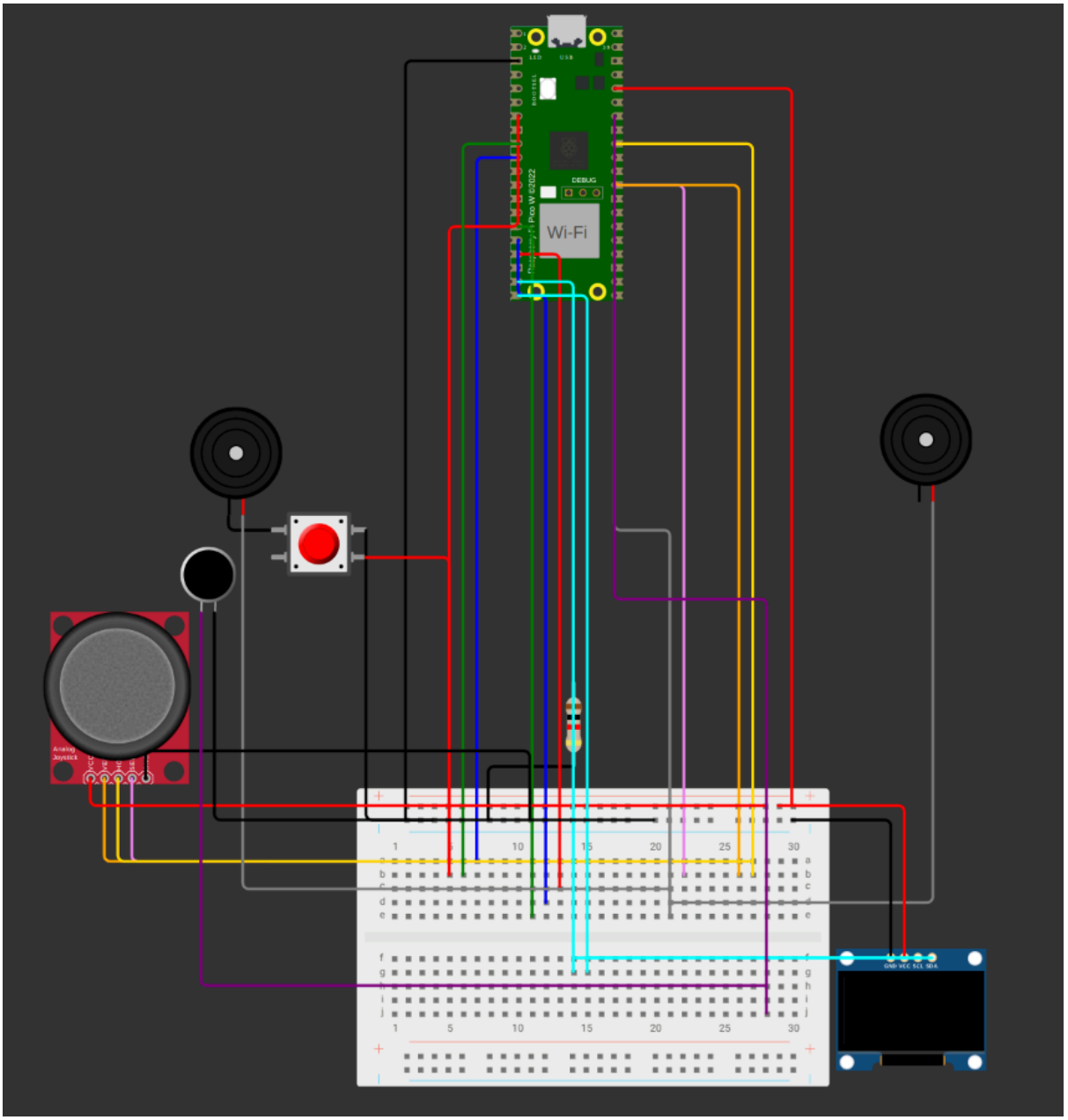
- **Função de Cada Bloco**
  - **Pico W**: Processamento central e comunicação com periféricos.
  - **Microfone** Conversão de ruído ambiental em sinal elétrico.
  - **Joystick** Ajuste de volume e interação com o sistema.
  - **Buzzer** Geração de alertas sonoros.
  - **OLED** Exibição do volume e status do sistema.
  - **Botão** Definição do limiar de ruído.
- **Configuração de Cada Bloco**
  - **Microfone**: Conectado ao ADC0 (GP28) para leitura analógica.
  - **Joystick**: Eixo Y no ADC1 (GP26), botão interno em GP22.
  - **Buzzer**: Dois canais em GP10 e GP21, controlados por PIO para geração de PWM.
  - **OLED**: Comunicação I²C via GP14 (SDA) e GP15 (SCL).

- **Botão de Configuração**: Entrada digital em GP5 com pull-up.

Especificações Técnicas

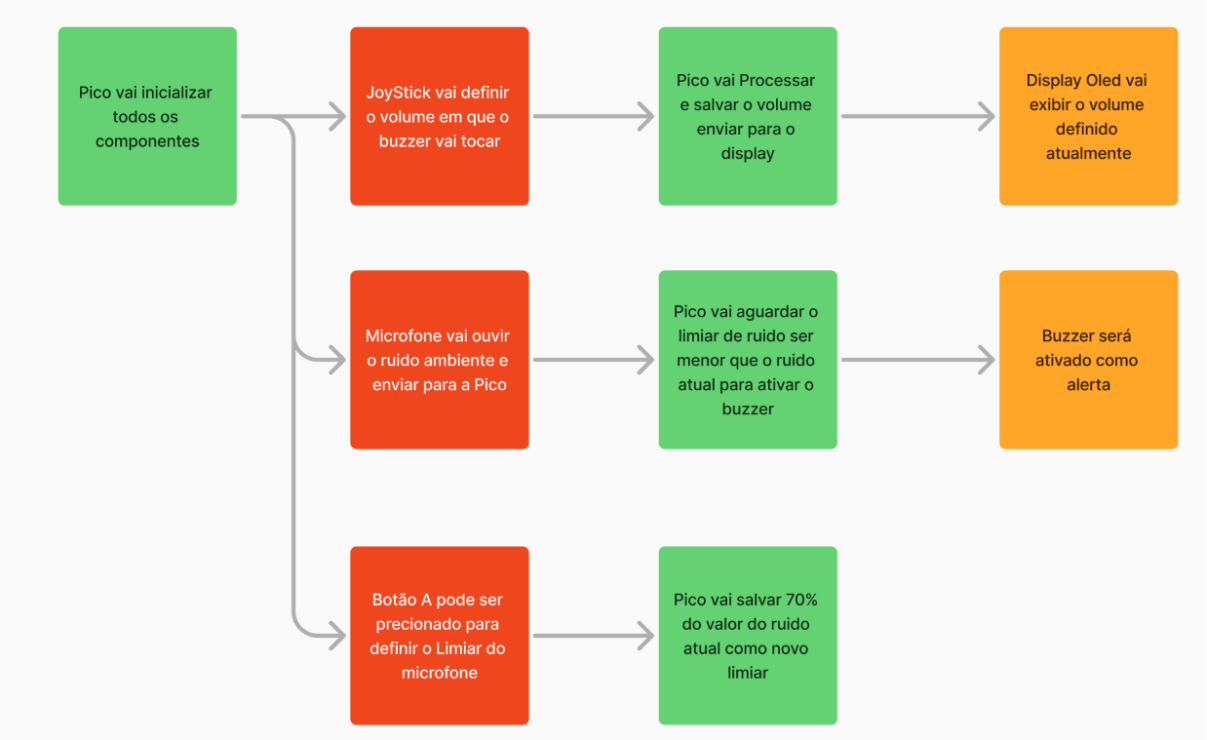
- **Microfone**: Resolução de 12 bits (ADC), faixa de 48 dB.
- **Buzzer**: Frequência ajustável via PIO, volume controlado por duty cycle.
- **OLED**: Display 128x64 pixels, protocolo I²C.

Circuito Completo



3) Software

- ## Blocos Funcionais
  - **Inicialização:** Configuração de periféricos (ADC, I²C, PIO).
  - **Leitura de Sensores:** Amostragem do microfone e joystick.
  - **Processamento:** Cálculo do limiar e comparação com o ruído atual.
  - **Interface Gráfica:** Atualização da tela OLED.
  - **Controle do Buzzer:** Geração de sinais PWM conforme o volume.
- ## Descrição das Funcionalidades
  - **Função update_display()**: Atualiza a tela com o volume atual.
  - **Função buzzer_beep():** Controla a duração e intensidade do alarme.
  - **Função get_sound_level():** Calcula a média de 1000 amostras do ADC.
- ## Definição das Variáveis
  - **volume:** Armazena o volume do buzzer (0–100%).
  - **sound_threshold**: Limiar de ruído para acionar o alarme.
  - **alarm_active**: Flag para estado do alarme (ativo/inativo).
- ## Fluxograma



- ## Inicialização
  - Configuração do **I²C** para a tela OLED.
  - Inicialização do ADC e DMA para leitura do microfone.
  - Carregamento do programa PIO para controle dos buzzers.
- ## Configurações dos Registros
  - **ADC:** Taxa de amostragem de 48 kHz, canal 0 (microfone) e 1 (joystick).
  - **PIO**: Máquinas de estado configuradas para gerar sinais PWM independentes.

- **Protocolo de Comunicação**
    - **I²C:** Comunicação a 400 kHz para a tela OLED (endereço 0x3C).
    - **PIO:** Protocolo customizado para controle de duty cycle nos buzzers.

4) Software

- **Metodologia**
- **Testes de Validação**
- **Discussão dos Resultados**

5) Conclusão

- A solução atende ao escopo proposto, oferecendo uma alternativa econômica para monitoramento sonoro simples, com potencial para expansão em cenários que não exigem análise complexa de áudio.

6) Referencias

- Repositório do projeto: https://github.com/victorandre957/Projeto_Final_Embarcatech
- Link do projeto no simulador: https://wokwi.com/projects/423722585478205441
- Link do Video de demonstração: https://youtu.be/jsb-wMCOq_M
- Semeq - Sensor wireless para manutenção preventiva (https://semeq.com/pt/blog/sensor-wireless-para-manutencao-preditiva/)
- Materiais Embarcatech
- Diagrama da BitdogLab (https://wokwi.com/projects/420545548816358401)

OBS: Durante a gravação do video, minha BitdogLab parou de funcionar, todos os componentes pararam de responder, apenas a pico funciona, por esse motivo, a demonstração do projeto foi feita no simulador.