Documentação Técnica – Protótipo ESP32 (IMU, OLED, microSD) ESP-IDF/FreeRTOS

Projeto: Plataforma de minigames com ESP32, MPU6050, display OLED SSD1306 (I²C), microSD (SPI), botões e buzzer
Autoria: Ialy Sousa, Estevão Holanda, Eduardo Nogueira, Wesley Wilson
Versão: 1.0

---

1. Visão Geral

Projeto de sistema embarcado que integra múltiplos periféricos para criar uma plataforma de **minigames controlados por movimento**. Desenvolvido em **ESP-IDF** com **FreeRTOS**.

Resumo: Leitura da IMU (MPU6050) via I²C, interface gráfica em OLED **SSD1306** (I²C), armazenamento de **recordes** em cartão **microSD via SPI**, feedback sonoro (buzzer) e navegação por botões. Estrutura baseada em **Máquina de Estados** + **multithreading** (tasks FreeRTOS).

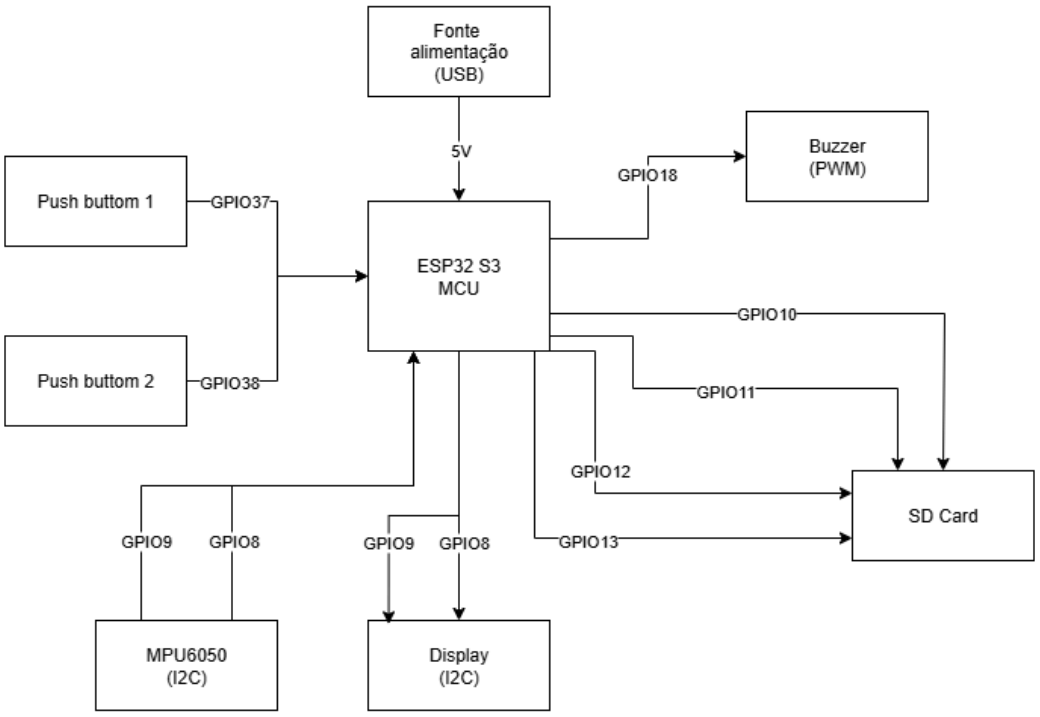
2. Objetivos e Requisitos

- **Funcionais:**
  - Menu com seleção de **4 minigames**.
  - Controle por movimento usando **MPU6050** (acel/girosc.).
  - Exibição de status/placar no **OLED SSD1306**.
  - Persistência de **recordes** no **microSD (SPI)**.
  - Navegação por **2 botões**; **buzzer** para feedback.
- **Não funcionais:**
  - Operação a 3V3.
  - Código modular (drivers + aplicação) em ESP-IDF.
  - Logs via UART/USB para diagnóstico.

- **Critérios de aceitação:** definir taxas de amostragem, FPS do display, latência de input, integridade dos arquivos, e estabilidade do loop de jogo.

3. Arquitetura do Sistema

- **Diagrama de blocos:**



- **Módulos:**
  - MCU: **ESP32 DevKit** (ESP-IDF/FreeRTOS).
  - Sensor: **MPU6050** (I²C).
  - Display: **OLED SSD1306** (I²C).
  - Armazenamento: **microSD (SPI)**.
  - HMI: **Buzzer** (PWM) e **2 Botões**.
  - Alimentação: 3V3; GND comum.

3.1 Máquina de Estados e Tarefas (FreeRTOS)

- **Estados:** `MENU → GAME_1|GAME_2|GAME_3|GAME_4 → SCORE → MENU`.
- **Tasks sugeridas:**
  - `task_input` (I²C IMU + botões, queue para eventos)
  - `task_render` (SSD1306, double-buffer opcional)

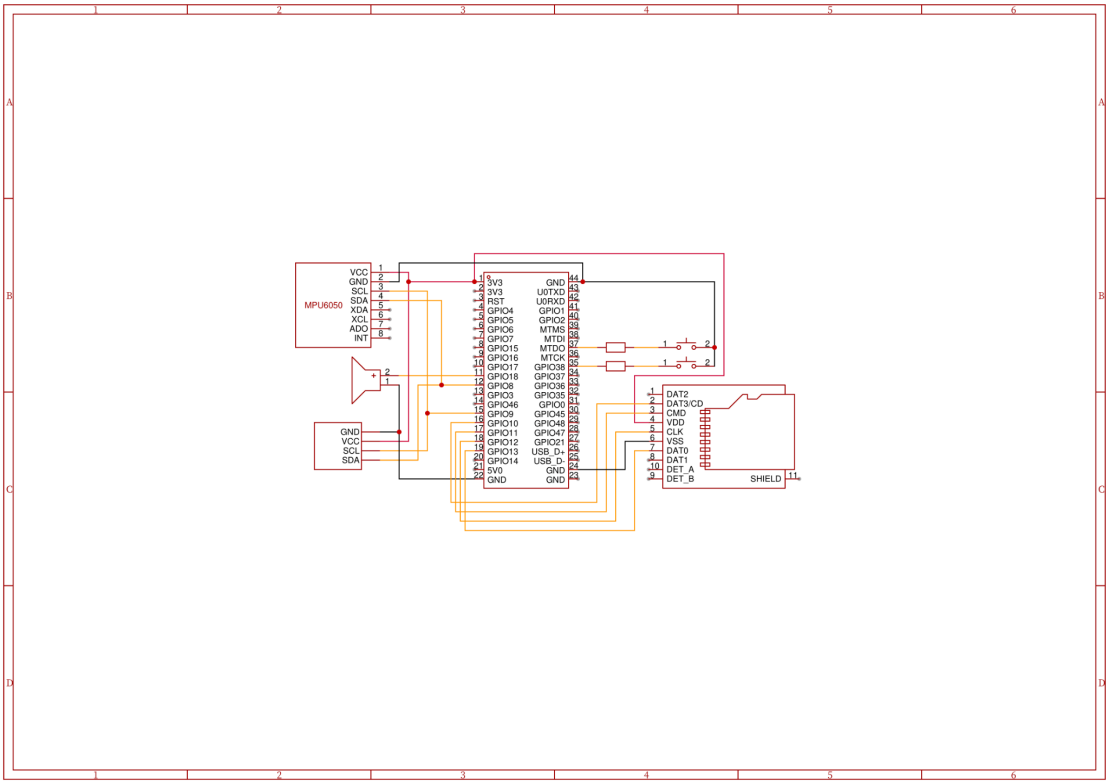
- `task_logic` (FSM do jogo)
- `task_storage` (fila/queue de gravações no SD)

- **Sincronização:** Queues + semáforos; timer para taxa fixa de leitura/atualização.

4. Hardware

4.1 Esquemático

- **Arquivo:**



- **Notas:**

  - **I²C** compartilhado entre MPU6050 e OLED SSD1306 (SDA/SCL em paralelo).
  - **microSD via SPI** (MOSI/MISO/SCLK/CS).
  - Botões com resistor ao 3V3 → usar **pull-down** interno no MCU.

4.2 Tabela de Pinagem (ESP32)

COMPONENTES	PINOS
BUZZER	GPIO-18
BOTÕES	GPIO-40 GPIO-38
SDCARD	DO: GPIO-13 SCK: GPIO-12 DI: GPIO-11 CS: GPIO-10
DISPLAY	SCL: GPIO-9 SDA: GPIO-8
MPU6050	SCL: GPIO-9 SDA: GPIO-8

4.3 Especificações Elétricas

- Tensão: 3V3 para todos os módulos.
- Pull-ups I²C: 2.2k–10k efetivos; garantir valor equivalente adequado.

4.4 Lista de Materiais

Item	Quant.	Descrição	Link	Observações
1	1	ESP32 DevKit	ESP32-S3	I²C 3V3
2	1	MPU6050 (breakout)	MPU6050	I²C 3V3
3	1	Módulo microSD (3V3)	SD Card	SPI
4	1	OLED SSD1306 128×64	SSD1306	I²C
5	2	Botão tátil	Botão	SPST
6	1	Buzzer passivo	Buzzer	PWM
7	2	Resistores/fios	Resistores	10kΩ cada resistor

4.5 Bibliotecas

Bibliotecas Padrão da Linguagem C

- stdio.h
- string.h
- stdlib.h
- math.h
- stdbool.h

Bibliotecas do FreeRTOS

- freertos/FreeRTOS.h
- freertos/task.h
- freertos/queue.h

Bibliotecas de Drivers do ESP-IDF

- driver/i2c.h
- driver/gpio.h
- driver/ledc.h
- driver/sdmmc_host.h

Bibliotecas de Sistema e Componentes do ESP-IDF

- esp_log.h
- esp_vfs_fat.h
- sdmmc_cmd.h

Bibliotecas de Componentes Personalizados

- mpu6050.h
- ssd1306.h

5. Estrutura de Códigos

```
├── build
├── main
│   ├── CMakeLists.txt
│   ├── hello_world_main.c
│   └── idf_component.yml
├── managed_components
│   ├── espressif__mpu6050
│   └── espressif__ssd1306
├── CMakeLists.txt
├── README.md
├── dependencies.lock
├── pytest_hello_world.py
├── sdkconfig
└── sdkconfig.ci
```

Fonte: https://github.com/ialysousa/projeto_embarcados_grupo3/tree/main

6. Metodologia

O desenvolvimento foi dividido em etapas para garantir organização e eficiência. Inicialmente, foram escolhidos os componentes de hardware e definidos os protocolos de comunicação adequados. Em seguida, implementou-se a base do sistema no ESP-IDF, criando bibliotecas próprias para lidar com cada módulo, como leitura de dados do MPU6050, controle do display SSD1306 e gerenciamento de arquivos no SD Card. Essa abordagem permitiu facilitar a manutenção e a escalabilidade do código.

Na segunda etapa, foi criada a estrutura lógica do sistema, composta por um menu inicial controlado por botões físicos e quatro minigames independentes. Cada jogo foi desenvolvido para responder de forma responsiva à inclinação do dispositivo, proporcionando interatividade e desafios variados. O buzzer foi integrado para fornecer feedback sonoro durante eventos importantes do jogo, como pontuações ou colisões.

Por fim, implementou-se o sistema de persistência de dados, garantindo que os recordes fossem salvos e recuperados automaticamente a cada inicialização. Foram realizados testes no hardware real para validação de desempenho, usabilidade e confiabilidade da comunicação entre os módulos. O trabalho foi concluído com documentação, diagrama em blocos e vídeos demonstrativos para melhor apresentação do resultado final.

7. Resultados

O sistema embarcado atendeu a todos os requisitos especificados, apresentando funcionamento estável e responsivo. Os quatro minigames foram executados corretamente, com controle preciso via acelerômetro e interface gráfica clara e intuitiva. O menu inicial permitiu a navegação sem erros, porém a gravação de dados no SD Card não obteve êxito por causa do cartão SD corrompido. A modularização do código e a criação de bibliotecas próprias contribuíram para uma arquitetura limpa e de fácil manutenção. Além disso, o uso do buzzer agregou uma camada extra de imersão na experiência do usuário, tornando o sistema não apenas funcional, mas também envolvente e atrativo.