
Instituto Federal da Paraíba - IFPB

Engenharia de Computação

Disciplina: Sistemas Embarcados

Professores: Alexandre Sales Vasconcelos

**Alunos:**

Álisson Brenner, Caio Livio, Lucas Alves, Vinicius Rodrigues, Vinícius Gonzaga

# RELATÓRIO TÉCNICO: SISTEMA DE CONTROLE DE MESA LABIRINTO COM MONITORAMENTO IOT

Campina Grande
2025

# RELATÓRIO TÉCNICO: SISTEMA DE CONTROLE DE MESA LABIRINTO COM MONITORAMENTO IOT

# RELATÓRIO TÉCNICO: SISTEMA DE CONTROLE DE MESA LABIRINTO COM MONITORAMENTO IOT

## 1. Introdução

Este projeto consiste no desenvolvimento de uma Mesa Labirinto (Ball and Plate) controlada eletronicamente. O sistema utiliza um microcontrolador ESP32-S3 para processar entradas de um joystick analógico e controlar dois servomotores que alteram a inclinação da mesa nos eixos X (Pitch) e Y (Roll).

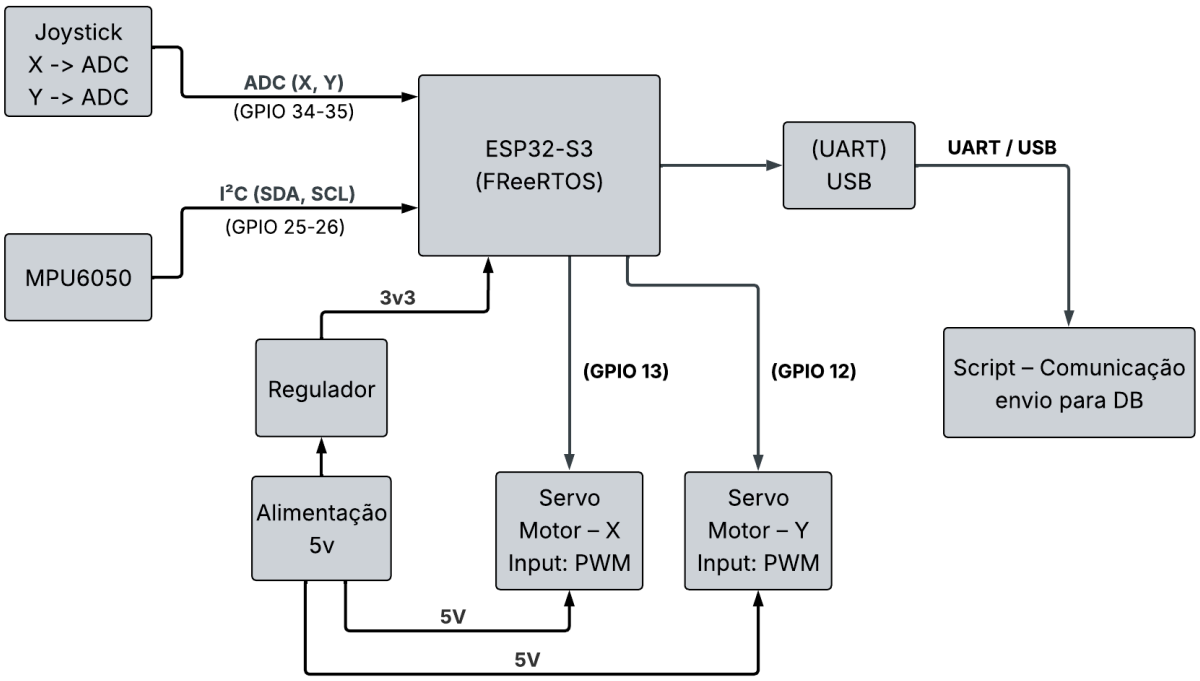
Além do controle mecânico, o sistema integra uma solução de IoT (Internet of Things) local. Um sensor inercial (MPU6050) coleta dados de inclinação em tempo real, que são enviados via Porta Serial para um computador, armazenados em banco de dados temporal (InfluxDB) e visualizados em dashboards interativos (Grafana).

## 2. Diagrama em Blocos do Sistema

O fluxo de informação ocorre da seguinte maneira:

1. **Entrada:** O usuário move o Joystick.
2. **Processamento:** O ESP32 lê o Joystick (ADC) e calcula o ângulo desejado.
3. **Atuação:** O ESP32 envia sinal PWM para os Servos moverem a mesa.
4. **Sensoriamento:** O MPU6050 mede a inclinação real da mesa.
5. **Monitoramento:** O ESP32 envia um pacote JSON via USB -> Python Script -> InfluxDB -> Grafana.

Segue abaixo a imagem do diagrama de blocos:



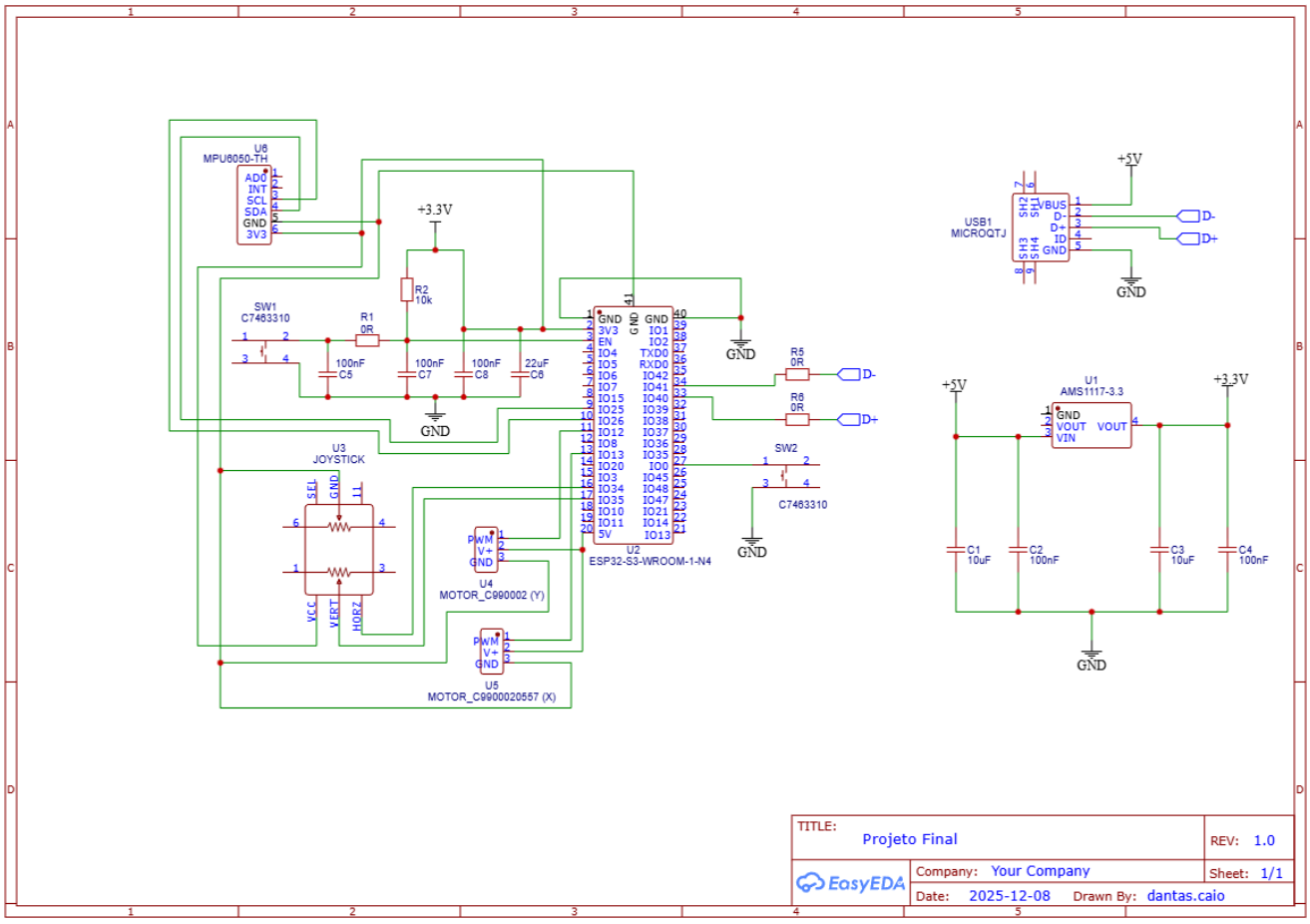

# 3. Esquemático e Conexões (Hardware)

Abaixo estão listadas as conexões físicas realizadas no ESP32-S3.

| Componente | Pino Componente | Pino ESP32 (GPIO) | Função |
|---|---|---|---|
| **Joystick** | VRx (Horiz) | GPIO 4 | Leitura Analógica (ADC) |
| | VRy (Vert) | GPIO 5 | Leitura Analógica (ADC) |
| | SW (Botão) | N/C | Não utilizado |
| **Servos** | Sinal X | GPIO 18 | PWM (LEDC) |
| | Sinal Y | GPIO 19 | PWM (LEDC) |
| **MPU6050** | SDA | GPIO 8 | Comunicação I2C |
| | SCL | GPIO 9 | Comunicação I2C |
| **Alimentação** | VCC | 5V Externo | Alimentação de potência |
| | GND | GND Comum | Referência |

**Nota de Hardware:** Foi utilizada uma fonte de alimentação externa de 5V para os servos, garantindo que a corrente de partida dos motores não reiniciasse o microcontrolador.

Segue abaixo a imagem do Diagrama Esquemático:




# 4. Descrição das Tarefas e Fluxos FreeRTOS

O firmware foi desenvolvido utilizando o **FreeRTOS** (Real-Time Operating System) para garantir determinismo e multitarefa. O sistema foi dividido em 4 tarefas principais e uma rotina de inicialização, protegidas por um Semáforo (Mutex) para acesso seguro às variáveis globais.

### 4.1. Rotina de Inicialização (Main)

Antes de iniciar as tarefas, o sistema executa a função `calibrar_inicial()`.

- **Ação:** Solicita ao usuário que não toque no joystick (mede o centro) e depois que rotacione o joystick (mede os extremos).
- **Resultado:** Gera um mapeamento personalizado para corrigir zonas mortas e desvios mecânicos do joystick.

### 4.2. Tarefas (Tasks)

| Tarefa | Prioridade | Frequência | Descrição |
|---|---|---|---|
| **task_joystick_read** | Alta (2) | 50Hz (20ms) | Lê os valores ADC do joystick, aplica filtro exponencial (suavização) e mapeia os valores brutos para ângulos alvo (70º a 110º) usando a calibração. |
| **task_servo_control** | Muito Alta (3) | 100Hz (10ms) | Responsável por mover os motores. Implementa uma rampa de aceleração suave para evitar movimentos bruscos ("trancos") na mesa. |
| **task_mpu_read** | Média (2) | 20Hz (50ms) | Lê acelerômetro e giroscópio via I2C. Aplica um **Filtro Complementar**(98% Giroscópio, 2% Acelerômetro) para obter um ângulo estável sem ruído e sem deriva (drift). |
| **task_monitor** | Baixa (1) | 5Hz (200ms) | Coleta o "snapshot" atual de todas as variáveis (Joystick, Servos, MPU), formata em uma string **JSON** e envia pela porta Serial (USB). |

# 5. Explicação do Funcionamento do Software

## 5.1. Mapeamento Inteligente

Diferente da função `map()` padrão, foi implementada uma função assimétrica. Ela divide o eixo do joystick em duas partes (Centro até Mínimo e Centro até Máximo). Isso garante que, mesmo que o potenciômetro do joystick esteja descentralizado fisicamente, o servo motor permanecerá exatamente em 90º quando a alavanca estiver solta.

## 5.2. Ponte de Dados (Python Bridge)

Como o Grafana não lê portas USB diretamente, foi desenvolvido um script em **Python** (`serial_to_influx.py`).

1. O script escuta a porta COM/TTY.
2. Captura a string JSON enviada pelo ESP32: `{"joy_x": 2048, "pitch": 5.2, ...}`.
3. Usa a biblioteca `influxdb_client` para injetar esses dados no banco de dados com a tag `hardware_real`.

# 6. Componentes e Bibliotecas Utilizados

## Hardware

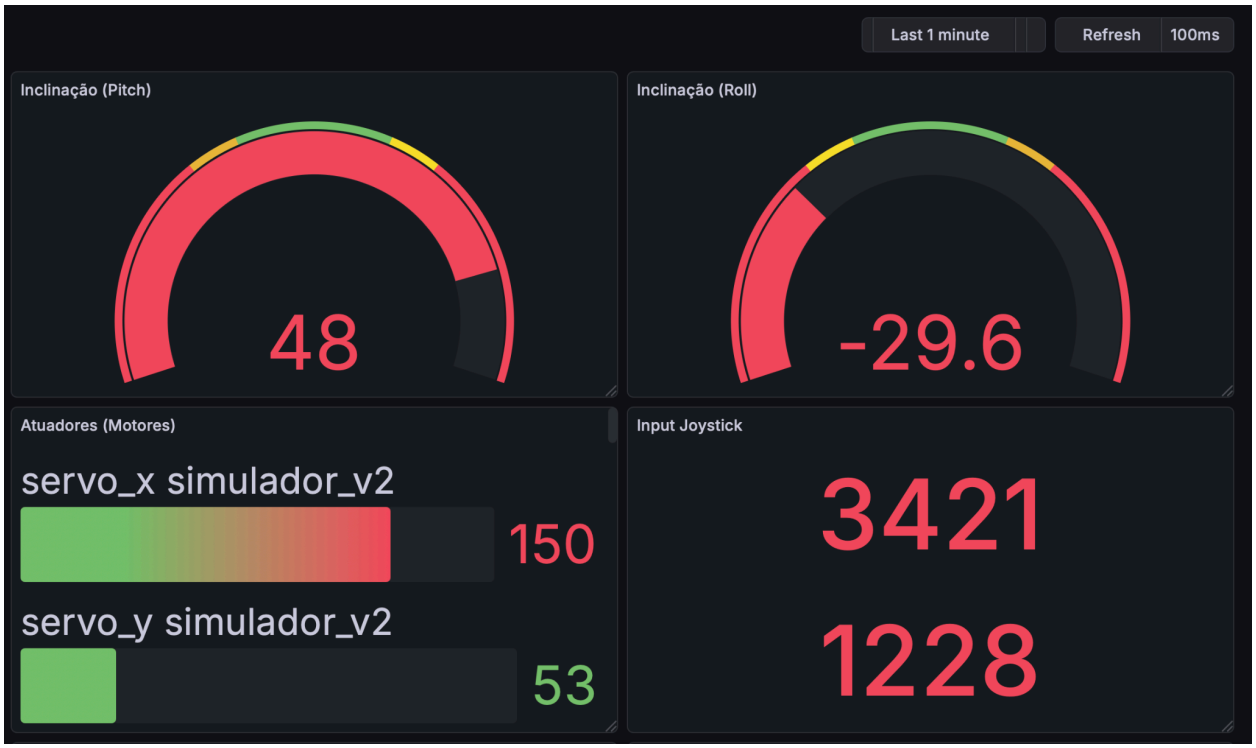
- **Microcontrolador:** ESP32-S3 (Espressif).
- **Atuadores:** Servomotores Padrão (ex: MG996R ou SG90).
- **Sensores:** Joystick Analógico KY-023, IMU MPU6050 (6 eixos).

## Software & Bibliotecas

- **Firmware:** ESP-IDF framework.
    - `driver/adc`: Leitura OneShot de alta precisão.
    - `driver/ledc`: Geração de PWM via hardware.
    - `driver/i2c`: Comunicação com MPU6050.
    - `freertos/semphr`: Gestão de concorrência (Mutex).
- **Backend:**
    - Python 3.10+ (`pyserial`, `influxdb-client`).

    - InfluxDB v2 (Time Series Database).
- **Frontend:**
    - Grafana (Visualização de dados).

## 7. Capturas de Tela do Grafana

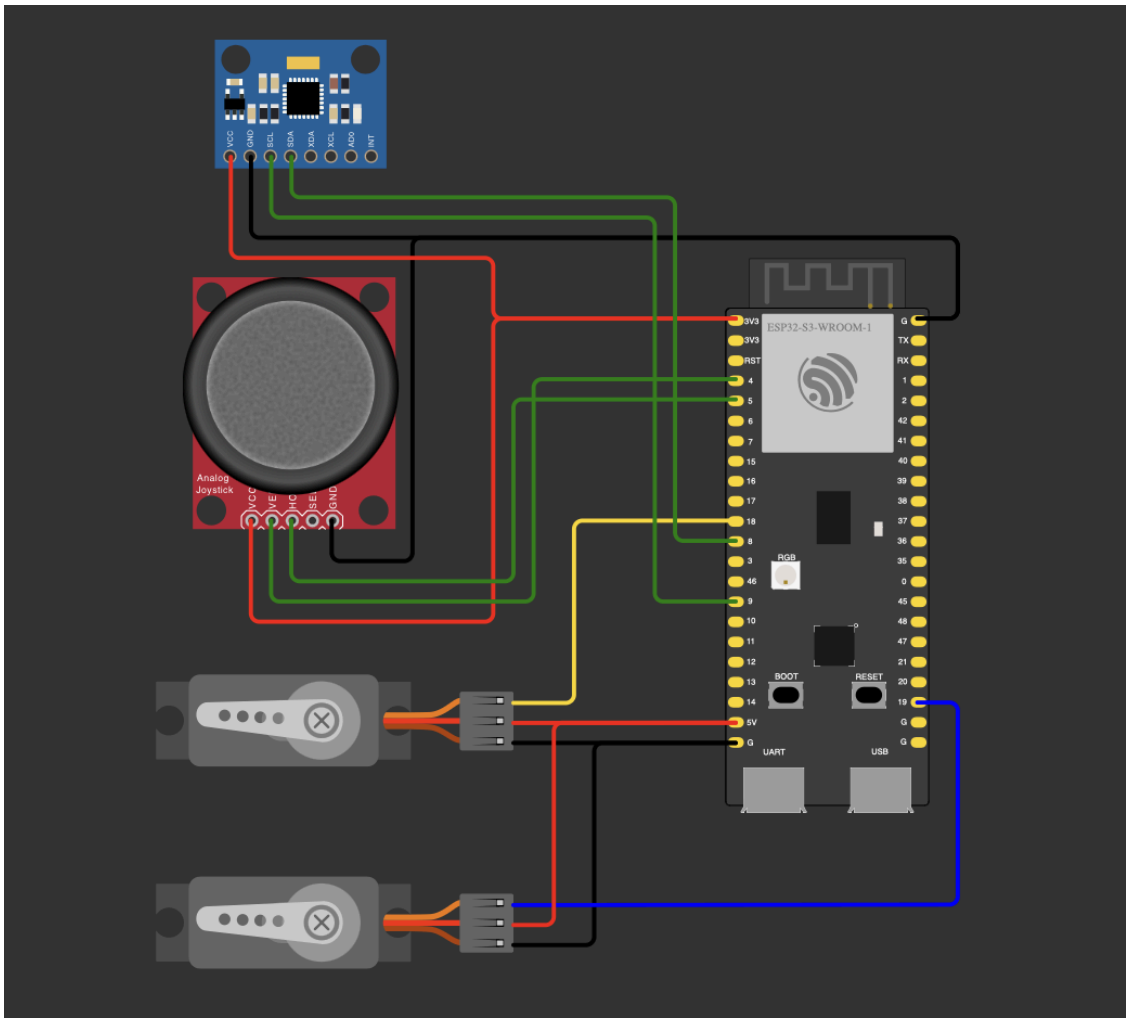



**Painel 1: Horizonte Artificial (Gauge)** *Mostra a inclinação real da mesa (Pitch e Roll) lida pelo MPU6050. Configurado com zonas de cor (Verde = Nivelado, Vermelho = Inclinado).*

**Painel 2: Monitor de Atuadores (Bar Gauge)** *Barras horizontais que indicam a força/posição aplicada aos motores X e Y em tempo real.*

**Painel 3: Histórico de Comandos (Time Series)** *Gráfico de linhas mostrando a correlação entre o comando do Joystick (entrada) e a resposta da Mesa (saída) ao longo do tempo.*

# 8. Fotos da Mesa em Operação




*(Espaço reservado para fotos do seu projeto)*

- **Foto 1:** Simulação no Wokwi.
- **Foto 2:** Detalhe da eletrônica (ESP32 e conexões).
- **Foto 3:** Visão geral do sistema (Mesa ao lado do Monitor com Grafana).

# 9. Dificuldades e Soluções Encontradas

Durante o desenvolvimento, os seguintes desafios técnicos foram superados:

**1. Zona Morta e Descentralização do Joystick**

- *Problema:* O joystick físico não retornava exatamente para o valor 2048, fazendo a mesa inclinar sozinha.
- *Solução:* Implementação de uma rotina de calibração no boot e criação de uma função de mapeamento assimétrico que considera uma "Deadzone" de software.

**2. Ruído nas Leituras do Acelerômetro**

- *Problema:* A vibração dos servos causava ruído na leitura do ângulo pelo MPU6050.
- *Solução:* Implementação de um **Filtro Complementar**. O filtro confia 98% na integração do giroscópio (preciso, mas tem deriva) e 2% no acelerômetro (ruidoso, mas não tem deriva), resultando em um sinal limpo.

**3. Movimento Brusco dos Servos**

- *Problema:* Mover o joystick rapidamente causava trancos na estrutura mecânica.
- *Solução:* Aplicação de suavização via software (Filtro Exponencial com fator 0.10) na task de controle dos servos, criando uma aceleração fluida.

**4. Latência na Visualização de Dados**

- *Problema:* Enviar muitos dados via Serial travava o processador.
- *Solução:* A tarefa de monitoramento (`task_monitor`) foi configurada com prioridade baixa e executa apenas a cada 200ms, enviando dados formatados em JSON leve, garantindo que o controle dos motores (10ms) nunca seja interrompido.