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

Campina Grande
2023

Documentação do Projeto: Controle de Mesa (Ball-on-Plate)

Fase 1: Controle Local Físico (Joystick)

Data: 24/11/2025

Plataforma: ESP32-S3 (ESP-IDF/FreeRTOS)

---

1. Objetivo da Fase

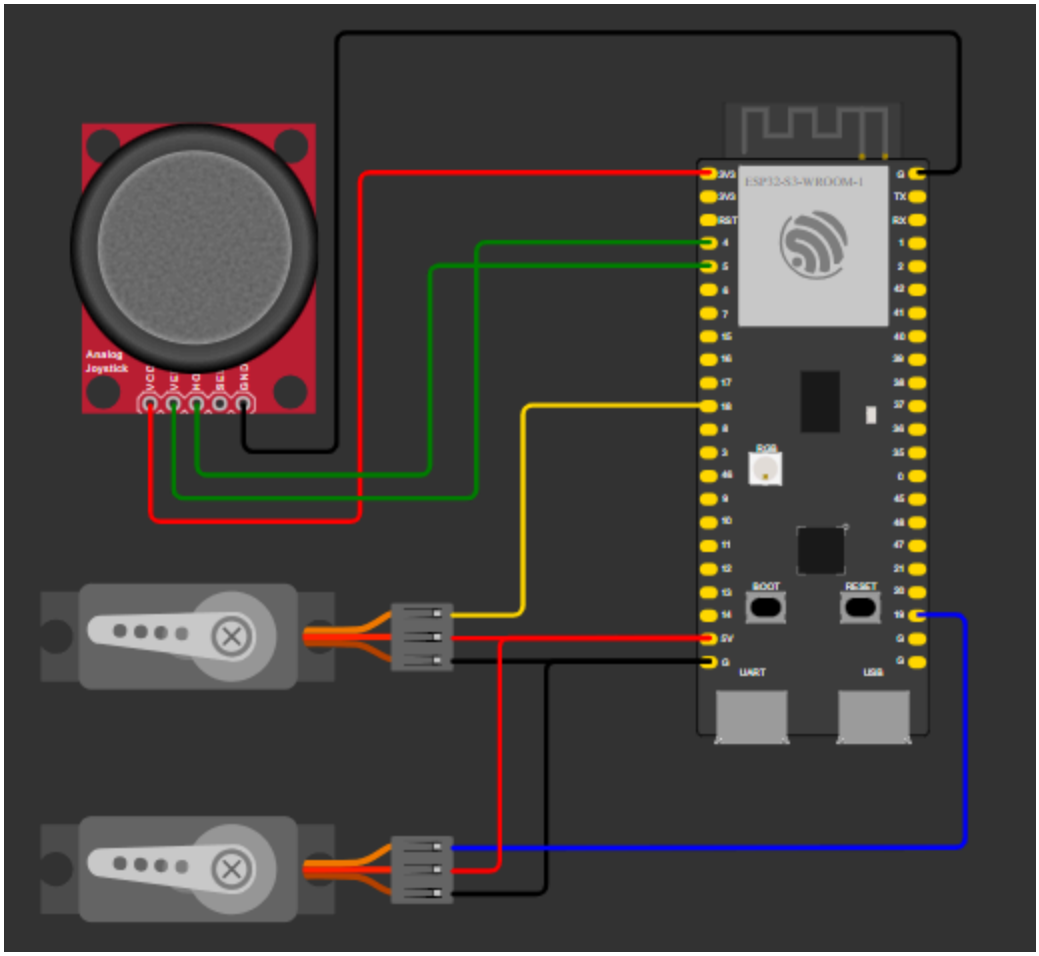
O objetivo desta fase foi desenvolver o firmware base para o controle manual da inclinação da mesa nos eixos X e Y. O sistema deve ler as entradas analógicas de um joystick, processar o sinal para remover ruídos elétricos e acionar dois servomotores com movimento proporcional e suave.

2. Arquitetura de Hardware

O sistema é baseado no microcontrolador **ESP32-S3-WROOM-1**. A escolha dos pinos foi crítica para garantir o uso correto dos canais ADC1, evitando conflitos internos do hardware.

Mapeamento de Pinos (Pinout):

Componente	**Função**	**Pino Físico (GPIO)**	**Recurso Interno**
Servo X	Saída PWM	GPIO 18	LEDC Channel 0
Servo Y	Saída PWM	GPIO 19	LEDC Channel 1
Joystick Horiz	Entrada Analógica	GPIO 4	ADC1 Channel 3
Joystick Vert	Entrada Analógica	GPIO 5	ADC1 Channel 4



---

3. Arquitetura de Software

O firmware foi desenvolvido em **C** utilizando o framework **ESP-IDF** sobre o sistema operacional de tempo real **FreeRTOS**. A aplicação é dividida em três tarefas (Tasks) independentes que se comunicam através de variáveis globais protegidas por um **Mutex** (Semáforo de Exclusão Mútua).

3.1. Estrutura das Tarefas (Tasks)

1. **Task 1: task_joystick_read (Prioridade 2)**
    - **Função:** Responsável pela aquisição de dados.
    - **Ciclo:** 20ms.
    - **Processamento:** Lê o ADC, aplica o filtro digital (EMA), aplica a zona morta (Deadzone) e converte os valores para Duty Cycle do PWM.
2. **Task 2: task_servo_control (Prioridade 3)**
    - **Função:** Responsável pela atuação física.
    - **Ciclo:** 20ms.

    - **Processamento:** Lê os valores de Duty Cycle calculados e atualiza o periférico LEDC (PWM) para mover os motores. Tem prioridade alta para garantir estabilidade no sinal PWM.
3. **Task 3: task_monitor (Prioridade 1)**
    - **Função:** Responsável pela telemetria e debug.
    - **Ciclo:** 200ms.
    - **Processamento:** Imprime no Serial Monitor os valores filtrados do ADC e os ângulos atuais dos servos.
    - **Nota:** O tamanho da pilha (Stack) foi ajustado para **4096 bytes** para suportar operações de printf com floats.

---

4. Soluções Técnicas Implementadas

Para garantir um movimento "amanteigado" e sem trepidações (jitter), foram implementados algoritmos específicos de processamento de sinal:

4.1. Filtro Exponencial (EMA - Exponential Moving Average)

Como o joystick analógico e o ADC do ESP32 possuem ruído elétrico natural, a leitura bruta oscila. O filtro EMA suaviza essas oscilações sem introduzir o atraso (lag) excessivo de uma média simples.

- **Fórmula:** $S_t = \alpha \cdot Y_t + (1 - \alpha) \cdot S_{t-1}$
- **Configuração:** Alpha (α) definido em 0.10. Isso prioriza a estabilidade, filtrando 90% do ruído de alta frequência.

4.2. Zona Morta (Deadzone)

Para evitar que a mesa se mova sozinha quando o joystick está solto (no centro), foi criada uma zona morta de **50 unidades** ao redor do centro (2048).

- Se |Leitura - 2048| < 50, o sistema força o valor para o centro exato.

4.3. Mapeamento de Alta Resolução

O controle PWM utiliza o periférico **LEDC** com resolução de **13 bits** (valores de 0 a 8191). Isso oferece uma precisão muito superior à biblioteca Servo.h padrão do Arduino (que geralmente usa graus inteiros de 0-180), permitindo micro-ajustes na posição da mesa.

---

5. Parâmetros de Configuração

O comportamento do sistema pode ser ajustado através das macros definidas no início do código (main.c):

```
// Limites Físicos da Mesa
#define MIN_ANGLE 40    // Ângulo mínimo de inclinação
```

```
#define MAX_ANGLE 140   // Ângulo máximo de inclinação

// Ajuste de Sensibilidade
#define FILTER_ALPHA 0.10 // 0.1 (Suave) a 1.0 (Rápido/Ruidoso)
#define JOY_DEADZONE 50   // Zona neutra do joystick

// Pinos (Hardware)
#define PIN_JOY_HORIZ 4
#define PIN_JOY_VERT  5
```

6. Resultados e Validação

O sistema foi validado em ambiente de simulação (Wokwi) apresentando os seguintes resultados:

- **Estabilidade:** O servo permanece imóvel quando o joystick está centralizado (graças à Deadzone).
- **Suavidade:** Movimentos bruscos no joystick são traduzidos em movimentos fluidos nos servos (graças ao Filtro EMA).
- **Confiabilidade:** O sistema opera continuamente sem reinicializações (Stack Overflow corrigido).