IFPB - Campus Campina Grande

ENGENHARIA DE COMPUTAÇÃO

Mesa Labirinto Controlada por Joystick

Projeto da disciplina Sistemas Embarcados, ministrada pelo Professor Alexandre Sales. Semestre 2025.1

Equipe:

Eduardo Henrique Lima de Morais

Emilieny de Souza Silva

Victor José Cordeiro de Medeiros

Campina Grande, 15 de dezembro de 2025

Sumário

1. Introdução e Objetivo Geral

Este projeto final tem como objetivo desenvolver um sistema embarcado interativo, utilizando o microcontrolador ESP32-WROOM-32, para controlar uma mesa com labirinto nos eixos X e Y através de dois servomotores. A interface de controle é um joystick analógico. Além do controle físico, o sistema deve capturar a orientação da mesa (inclinação) usando um sensor inercial MPU6050 e criar um gêmeo digital em tempo real, visualizado em um painel Grafana no computador.

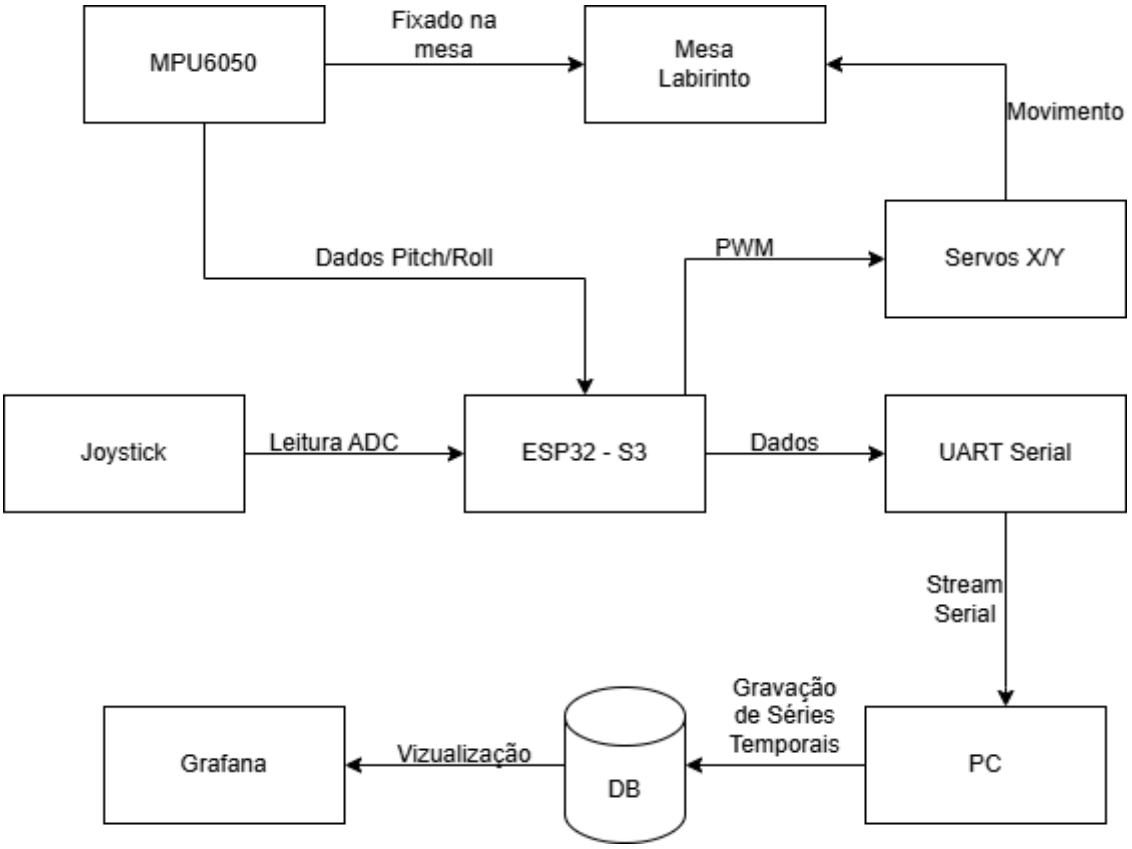
2. Descrição do Sistema e Hardware

O sistema consiste em um ciclo de leitura (joystick e MPU6050), processamento (ESP32) e atuação (servomotores), com comunicação de dados para visualização externa.

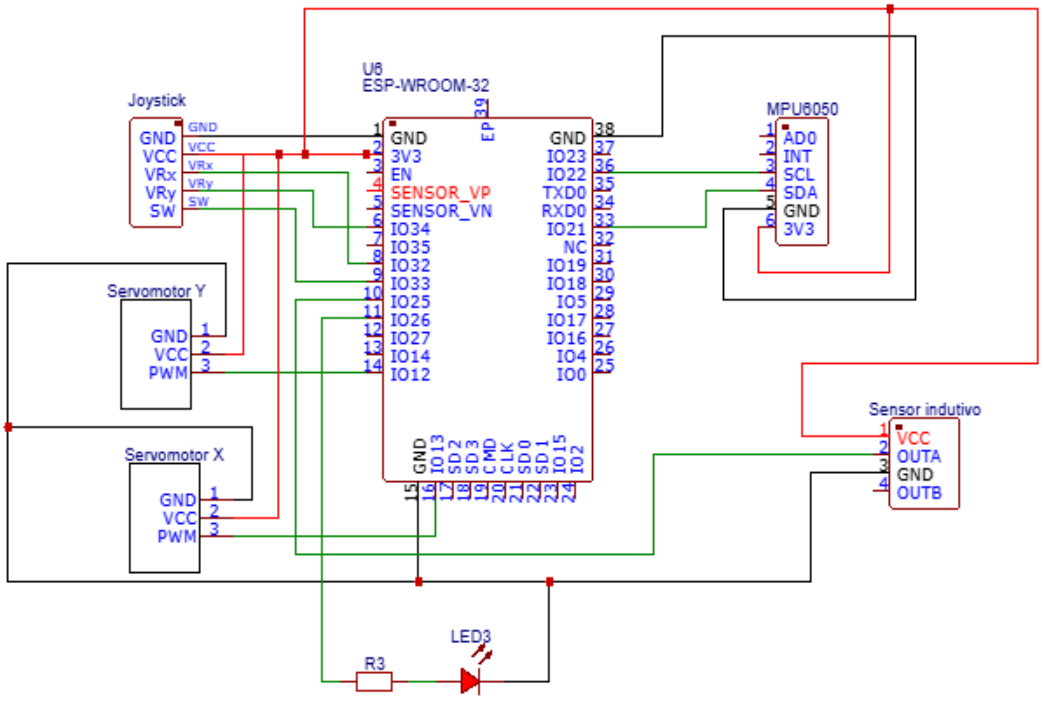
2.1 Lista de componentes de Hardware

Quant.	Componente	Função	Protocolo
1	ESP32-WROOM-32	Microcontrolador principal	-
1	Joystick Analógico	Interface de controle X/Y	ADC
2	Servo motor 90G	Atuação, controle de inclinação	PWM (LEDC)
1	MPU6050	Sensor inercial (Pitch/Roll)	I²C
1	LED	Indicador de sistema pronto (`LED_READY`)	GPIO
1	Sensor de Vitória	Detecção de conclusão do labirinto (`TOUCH_PIN`)	GPIO

2.2 Diagrama em Blocos do Sistema



2.3 Esquemático



3. Estrutura e Funcionamento do Código (FreeRTOS)

O software é baseado no **FreeRTOS** para garantir a execução concorrente e a capacidade de resposta. O sistema utiliza a exclusão mútua (pitch_roll_mutex) para proteger as variáveis de orientação (pitch_angle, roll_angle) contra acesso simultâneo.

3.1 Descrição das Tarefas (Tasks)

.As seguintes tarefas (Tasks) foram criadas, utilizando xTaskCreate, com prioridades definidas para garantir o fluxo de controle adequado (Prioridade 7 sendo a mais alta, e 3 a mais baixa):

Tarefa (Task)	Função Principal	Prioridade	Implementação no Código
Task MPU6050 (task_mpu6050)	Leitura do sensor inercial e cálculo de Pitch e Roll.	7 (Mais Alta)	Lê os registradores do MPU6050 via I²C e chama compute_pitch_roll_from_accel a cada 50ms.
Task Servo Control (task_servo_control)	Aplicação suave do movimento nos servomotores.	6	A cada 20ms, aplica um **filtro de suavização** nos ângulos alvo (angleX_target, angleY_target) para evitar movimentos bruscos.
Task Leitura Joystick (task_read_joystick)	Leitura e mapeamento da posição do joystick.	5	A cada 15ms, lê os valores brutos do ADC do joystick e os converte para ângulos alvo (0° a 180°), aplicando os limites definidos na calibração.
Task Touch Sensor (task_touch_sensor)	Detecção de "Vitória".	4	Monitora o estado do pino TOUCH_PIN (assumido como sensor de vitória) e imprime "PARABENS, VOCE GANHOU" quando acionado, desligando o LED de status.

Task Debug/Comunicação (task_debug)	Envio periódico dos dados de orientação.	3 (Mais Baixa)	A cada 500ms, imprime os valores de Pitch e Roll no formato **JSON** ({"pitch": %.2f, "roll": %.2f}\n), atendendo ao requisito de envio periódico dos dados via serial.

3.2 Funcionamento do Sistema

1. **Inicialização:** O app_main inicializa o hardware e executa o **Modo de Calibração** (run_calibration_mode). Este modo usa o JOY_BUTTON para definir os limites mecânicos dos servos e o centro do joystick, enviando dados de calibração para o Grafana.

2. **Controle (Lógica de Servos):** A task_read_joystick lê os valores brutos do ADC e os mapeia para ângulos alvo (angleX_target, angleY_target). A task_servo_control aplica um **Filtro de Suavização** sobre esses alvos, eliminando o movimento brusco (*jitter*) e aplicando o sinal PWM (50Hz) aos Servos.

3. **Sensoriamento e Gêmeo Digital:** A task_mpu6050 lê os dados de aceleração e calcula os ângulos Pitch e Roll. A task_debug lê esses valores (protegidos por **Mutex**) e os formata em JSON para envio contínuo via UART.

4. Integração do Gêmeo Digital (Visualização em Grafana)

A comunicação é estabelecida via um pipeline de software robusto, conforme implementado no serial_to_influx.py

4.1 Protocolo de Comunicação Serial e Banco de Dados

O ESP32 envia mensagens JSON com diferentes tags, que são processadas de forma distinta pelo script Python, garantindo a organização dos dados no InfluxDB.

Tipo de Mensagem	Baseado na Tag JSON	Measurement (Tabela) no InfluxDB	Conteúdo Registrado
Telemetria	"type":"mpu"	lab_angle	Valores de Pitch e Roll em tempo real.
Eventos	"type":"event"	evento_labirinto	Status do sistema (CALIBRATING, RUNNING, WIN).
Calibração	"type":"calib"	calibracao_joystick	Limites brutos do joystick definidos pelo usuário.

4.2 Configuração e Dashboard Grafana

O **Grafana** está configurado para interrogar o InfluxDB em tempo real, fornecendo a visualização do Gêmeo Digital. O dashboard obrigatório deve conter:

- **Gráficos em Tempo Real:** Visualização da série temporal de Pitch e Roll (dados da lab_angle).
- **Representação Visual:** Um *Gauge* ou painel que replique a orientação da mesa.
- **Sincronização:** O critério de sucesso é a demonstração da sincronização imediata entre o movimento físico da mesa (MPU6050) e o modelo virtual no Grafana.

5. Dificuldades e Soluções Encontradas

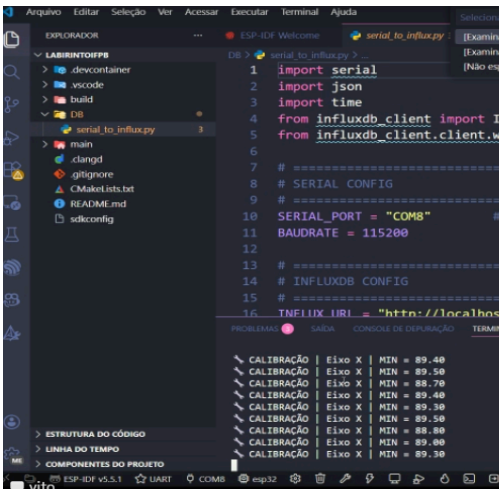
Esta seção detalha os desafios de engenharia e as soluções aplicadas, cruciais para a estabilidade final do sistema.

Dificuldade Encontrada	Solução Implementada
Instabilidade no Movimento (Jitter): O movimento dos servos era abrupto devido à leitura direta e ruidosa do ADC.	**Solução:** Implementação de um **Filtro de Suavização** na `task_servo_control`, tornando o movimento proporcional e suave.
Robustez da Comunicação: Risco de corrupção de dados e JSON incompleto ao enviar via UART.	**Solução:** O código no ESP32 garante a terminação com `\n`, e o script Python (`serial_to_influx.py`) usa `ser.readline()` e tratamento de exceção (`try/except`) para descartar pacotes incompletos ou inválidos, mantendo o sistema rodando.
Variação na Leitura do Joystick: O ponto de "centro" e os limites de SsSalcance do joystick variam entre unidades e a montagem mecânica.	**Solução:** Implementação do **Modo de Calibração** no `app_main` e envio dos valores de `centerH/V` para inicialização.
Leitura Imprecisa do MPU6050: A solução atual calcula Pitch/Roll apenas a partir do acelerômetro, sendo sensível a vibrações.	**Solução Futura:** Aprimorar a `task_mpu6050` com um **Filtro Complementar ou Filtro de Kalman** para integrar o giroscópio e o acelerômetro, alcançando maior estabilidade angular.

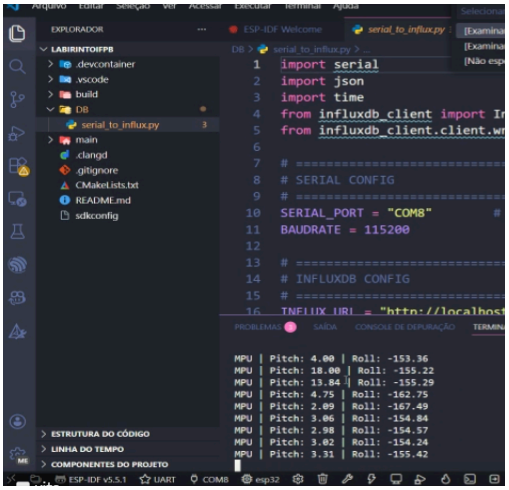
6. Resultados

Abaixo estão alguns prints de resultados do projeto:

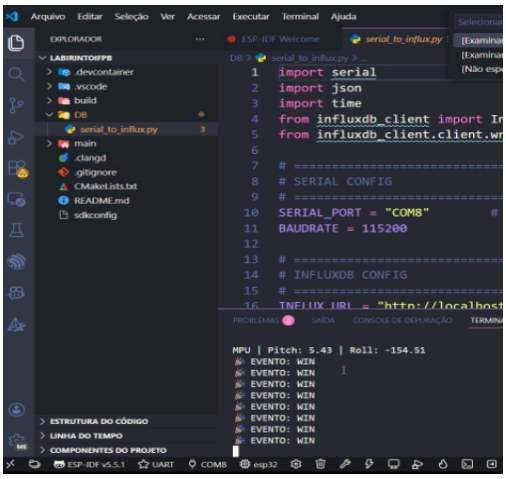
6.1 Modo calibração



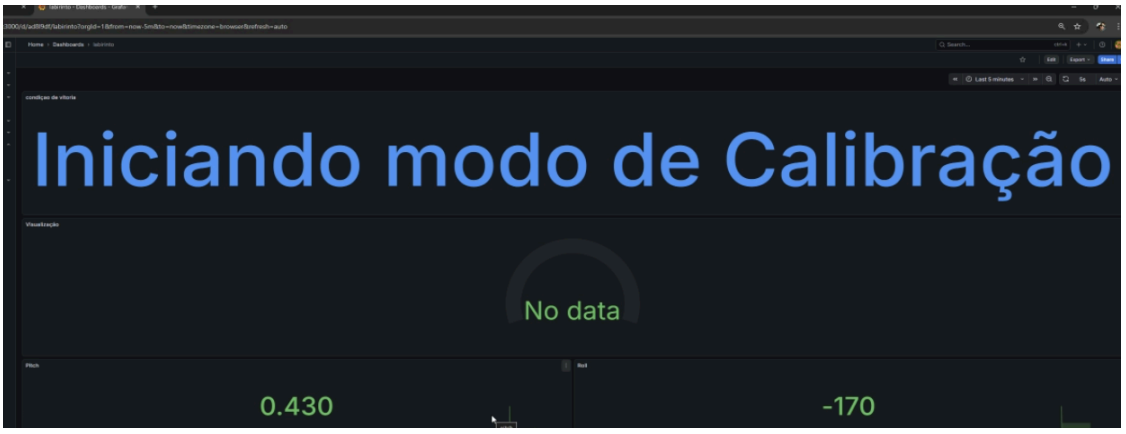
6.2 Modo Pitch/Roll



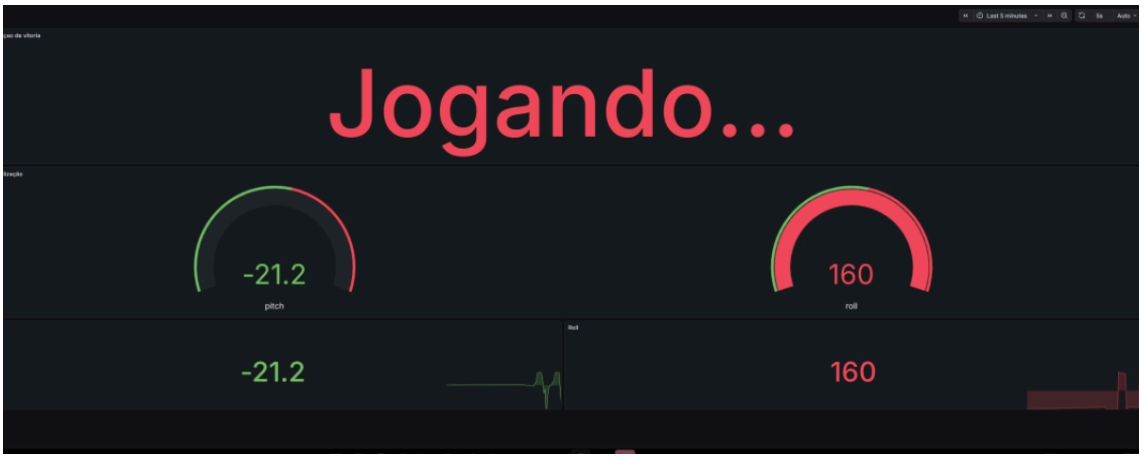
6.3 Vitória



6.4 Calibração



6.5 Jogando



6.6 Vitória