INSTITUTO FEDERAL Paraíba Campus Campina Grande	**INSTITUTO FEDERAL DA PARAÍBA - IFPB** **Campus Campina Grande** **Cursos: Engenharia da Computação** **Componente: Sistemas Embarcados** **Professora: Alexandre Sales Vasconcelos** **Estudante(s): Milena Lins Aguiar**

RELATÓRIO TÉCNICO PROJETO FINAL

Mesa Labirinto Controlada por Joystick

1. Diagrama em blocos do sistema.
2. Esquemático.
3. Descrição das tarefas e fluxos FreeRTOS.

Estado Global e Mutex

Os dados do sistema são armazenados em uma estrutura global (system_state_t), protegida por um **mutex**, garantindo acesso seguro entre as tarefas.

task_joystick

- Inicializa e calibra o joystick
- Realiza leituras periódicas
- Atualiza os valores normalizados no estado global

task_servo

- Lê os valores do joystick
- Aplica controle suave aos servos
- Executa a 50 Hz, compatível com PWM de servos

task_mpu6050

- Lê os dados do sensor inercial
- Calcula pitch e roll
- Atualiza o estado global

task_monitor

- Formata os dados do sistema em JSON
- Envia os dados pela porta serial
- Permite integração com InfluxDB e Grafana

app_main

Cria o mutex e inicializa todas as tarefas do sistema, garantindo execução concorrente e organizada.

4. Explicação do funcionamento do sistema.

O sistema desenvolvido consiste em uma **mesa com controle de inclinação em dois eixos**, utilizando um **ESP32-S3**, um **joystick analógico**, **dois servomotores** e um **sensor inercial MPU6050**. O sistema integra **controle físico**, **aquisição de dados**, **processamento em tempo real** e **visualização digital via Grafana**, caracterizando um **gêmeo digital** da mesa.

O funcionamento geral do sistema ocorre da seguinte forma:

1. O **joystick analógico** é utilizado como interface de entrada do usuário, permitindo definir a inclinação desejada da mesa nos eixos X e Y.
2. O ESP32 realiza a **leitura contínua do joystick via ADC**, normalizando os valores para uma faixa padrão de -1 a 1.

3. Esses valores normalizados são utilizados para comandar os **servomotores**, responsáveis pelo movimento físico da mesa.

4. Simultaneamente, o **MPU6050** realiza a medição da aceleração nos três eixos, permitindo o cálculo dos ângulos de **pitch** e **roll** da mesa.

5. Os valores de joystick, pitch e roll são organizados em formato **JSON** e enviados via **porta serial**.

6. Um script em Python lê esses dados da serial e os armazena no **InfluxDB**.

7. O **Grafana** consome os dados do InfluxDB e exibe gráficos em tempo real, permitindo visualizar a sincronização entre o movimento físico da mesa e seu modelo digital.

Todo o sistema embarcado é estruturado utilizando **FreeRTOS**, com tarefas independentes e comunicação segura via **mutex**, garantindo execução determinística e estável.

5. Descrição das bibliotecas/componentes utilizados.

5.1 Componente joystick.c

Este componente é responsável pela **leitura, calibração e normalização** do joystick analógico.

gpio_to_adc(int gpio)

Função auxiliar que converte um número de GPIO válido do ESP32-S3 em um canal do ADC1 correspondente.
Ela garante que apenas pinos compatíveis com ADC sejam utilizados, evitando configurações inválidas.

joystick_init(const joystick_config_t *cfg)

Inicializa o ADC no modo **oneshot**, configurando:

- Unidade ADC1
- Resolução de 12 bits
- Atenuação de 12 dB
  Também associa os GPIOs informados aos canais ADC do eixo X e Y do joystick.

joystick_calibrate(uint32_t ms)

Executa a **calibração dinâmica do joystick** durante um intervalo de tempo definido (ms):

- Detecta os valores mínimos e máximos de cada eixo.
- Calcula o ponto central médio do joystick.
- Aplica mecanismos de segurança caso a variação detectada seja muito pequena.

Essa calibração garante precisão mesmo com variações mecânicas do joystick.

normalize(int raw, int min, int center, int max)

Função interna que:

- Converte o valor bruto do ADC em uma faixa normalizada de -1.0 a +1.0.
- Aplica **zona morta** para eliminar ruído próximo ao centro.
- Garante saturação nos limites.

joystick_read_norm(float *x, float *y)

Realiza a leitura atual do joystick e retorna os valores normalizados dos eixos X e Y, já prontos para uso no controle dos servos.

5.2 Componente servo.c

Este componente controla os **servomotores** utilizando PWM via periférico **LEDC** do ESP32.

norm_to_duty(float v)

Converte um valor normalizado (-1 a +1) em um **duty cycle PWM**, correspondente ao pulso necessário para posicionar o servo entre seus limites mecânicos.

servo_init(const servo_config_t *cfg)

Configura:

- Temporizador PWM a 50 Hz (padrão de servos)
- Resolução de 13 bits
- Dois canais PWM independentes (eixo X e Y)

Inicializa os servos na posição central.

servo_set_norm_x(float v)

servo_set_norm_y(float v)

Funções que aplicam diretamente o valor normalizado ao respectivo servo, atualizando o duty cycle do PWM.

servo_set_smooth(float x, float y)

Implementa **suavização de movimento**, limitando a variação máxima por ciclo.
Isso evita movimentos bruscos, vibração e estresse mecânico nos servos, tornando o movimento da mesa mais estável e realista.

5.3 Componente mpu6050.c

Este componente realiza a comunicação com o **MPU6050** via **I²C** e calcula a orientação da mesa.

mpu6050_init(void)

Inicializa o barramento I²C:

- Configura pinos SDA e SCL
- Define frequência de comunicação
- Retira o MPU6050 do modo sleep

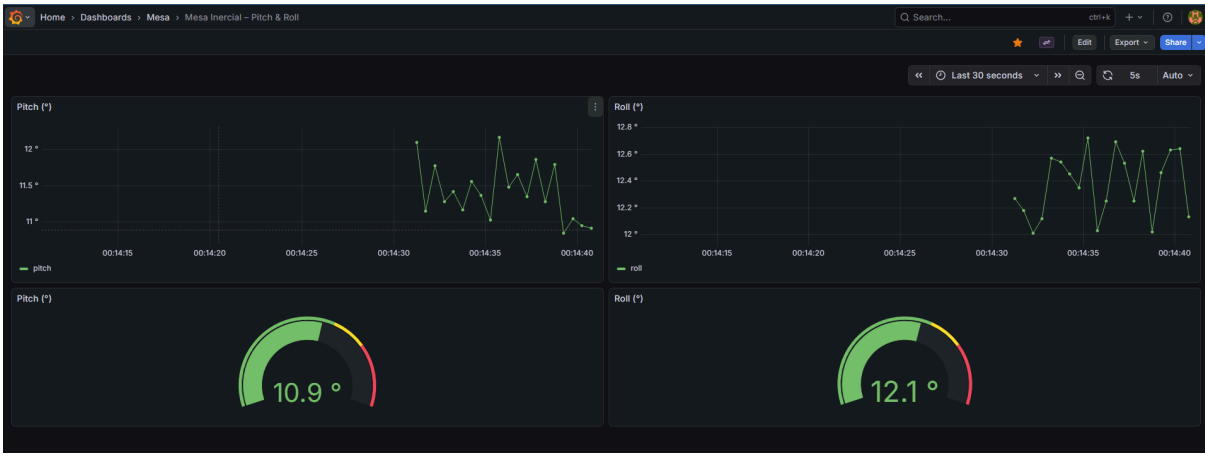
mpu6050_read(mpu6050_t *imu)

Realiza:

- Leitura dos registradores de aceleração
- Conversão dos valores brutos para unidades físicas
- Cálculo dos ângulos:
    - **Roll**: inclinação lateral
    - **Pitch**: inclinação frontal

Utiliza trigonometria baseada nos vetores de aceleração.

6. Capturas de tela do Grafana



7. Dificuldades e soluções encontradas.

Problema 1 — Ruído e saturação do joystick

Solução:

Implementação de calibração e normalização dos valores analógicos.

Problema 2 — Servos não responderam fisicamente

Solução:

O sistema foi validado logicamente via PWM e dados do MPU6050, garantindo o funcionamento do gêmeo digital mesmo sem atuação mecânica.

Problema 3 — Logs interferindo no parser JSON

Solução:

Padronização da saída serial para envio exclusivo de mensagens JSON.

Problema 4 — Integração Grafana + InfluxDB

Solução:

Uso de Docker Compose para padronizar o ambiente e facilitar a execução.