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
2025

Sumário

Documentação Técnica: Fase 2 – Leitura de Orientação e Telemetria

Projeto: Mesa Labirinto Controlada por ESP32-S3 **Versão do Firmware:** 2.0 (Integração MPU6050) **Data:** 29/11/2025

1. Visão Geral da Fase 2

O objetivo desta etapa foi integrar um sistema de **Unidade de Medida Inercial (IMU)** ao projeto existente de controle de servos. Utilizando o sensor **MPU6050**, o sistema agora é capaz de ler a inclinação da mesa em tempo real e transmitir esses dados, juntamente com o estado dos joysticks e servos, para um computador externo via serial.

2. Arquitetura de Hardware

2.1. Pinagem (ESP32-S3)

A comunicação com o MPU6050 é realizada via protocolo **I²C**. No ESP32-S3, utilizamos os pinos definidos no código para SDA e SCL.

Componente	Pino Componente	Pino ESP32-S3	Função
MPU6050	VCC	3.3V	Alimentação
MPU6050	GND	GND	Referência Comum
MPU6050	**SDA**	**GPIO 8**	Dados I²C
MPU6050	**SCL**	**GPIO 9**	Clock I²C
Joystick	VRx / VRy	GPIO 4 / 5	Entradas Analógicas
Servos	PWM	GPIO 18 / 19	Controle de Atuadores

2.2. O Sensor MPU6050

O MPU6050 combina um giroscópio de 3 eixos e um acelerômetro de 3 eixos no mesmo silício.

- **Endereço I²C:** `0x68` (Padrão com pino AD0 em LOW).
- **Frequência do Barramento:** 100 kHz (Standard Mode).

3. Implementação de Software

O firmware foi desenvolvido em **C puro** utilizando o **ESP-IDF** (Framework de Desenvolvimento IoT da Espressif) sobre o kernel **FreeRTOS**.

3.1. Estrutura de Tarefas (Multitasking)

O sistema opera em tempo real dividido em tarefas (Tasks) independentes, gerenciadas pelo escalonador do FreeRTOS. Isso garante que a leitura do sensor não bloqueie o movimento dos servos.

Task	Prioridade	Frequência	Descrição
`task_joystick_read`	2 (Alta)	50Hz	Lê ADCs, filtra ruído e calcula Duty Cycle dos servos.
`task_servo_control`	3 (Muito Alta)	50Hz	Aplica o sinal PWM aos motores.
`task_mpu_read`	**2 (Alta)**	**20Hz**	**Nova task dedicada à leitura I²C e cálculo de ângulos.**
`task_monitor`	1 (Baixa)	5Hz	Formata e envia os dados via Serial (UART).

3.2. Gerenciamento de Concorrência (Mutex)

Como múltiplas tarefas acessam variáveis globais (ex: ângulo do servo calculado na Task 1 e lido na Task 4, ou ângulo do MPU lido na Task 3 e enviado na Task 4), foi implementado um **Mutex (Semáforo de Exclusão Mútua)** chamado `xDataMutex`.

- **Função:** Garante que dados não sejam lidos pela metade enquanto estão sendo atualizados, prevenindo corrupção de dados na telemetria.

4. Detalhes da Integração do MPU6050

4.1. Inicialização e "Wake Up"

O MPU6050 inicia em modo *Sleep* (baixo consumo) por padrão. A inicialização implementada segue os seguintes passos:

1. **Configuração I²C:** Inicializa o driver I²C do ESP32 na porta 0, pinos 8 e 9.
2. **Reset:** Envia comando para resetar os registradores do sensor.
3. **Wake Up:** Escreve `0x00` no registrador `PWR_MGMT_1 (0x6B)`. **Este passo é crítico**; sem ele, o sensor retorna apenas zeros.

4.2. Leitura em Burst

Para otimizar o barramento I²C, não lemos registrador por registrador. A função `task_mpu_read` realiza uma leitura sequencial de **14 bytes** a partir do endereço `0x3B`:

- Aceleração X (High/Low)
- Aceleração Y (High/Low)
- Aceleração Z (High/Low)
- Temperatura (Ignorado)
- Giroscópio X, Y, Z (High/Low)

4.3. Cálculo Matemático (Pitch e Roll)

Para a Fase 2, focamos na obtenção da orientação através do **Acelerômetro**. O acelerômetro mede a força da gravidade (1g) distribuída nos eixos. Usando trigonometria, calculamos a inclinação:

As fórmulas implementadas no código são:

1. **Pitch (Inclinação no eixo X):**

$$Pitch = arctan\left(\frac{-Ax}{\sqrt{A_y^2 + A_z^2}}\right) \times \frac{180}{\pi}$$

2. **Roll (Inclinação no eixo Y):**

$$Roll = arctan\left(\frac{A_y}{A_z}\right) \times \frac{180}{\pi}$$

Onde Ax, Ay, Az são os valores brutos do acelerômetro.

> **Nota:** Este método fornece uma leitura absoluta da inclinação (não "deriva" com o tempo), sendo ideal para mesas labirinto onde a inclinação é limitada e os movimentos não são extremamente bruscos.

5. Protocolo de Comunicação (Telemetria)

Para atender ao requisito de envio de dados ao computador, foi padronizado o formato **JSON**. Isso facilita a leitura por humanos e permite integração futura com dashboards (Python, Node-RED, Unity).

Formato da String Serial:

JSON

```
{
  "joy_x": 2048,
  "joy_y": 2048,
  "servo_x": 90,
  "servo_y": 90,
```

```
  "pitch": -4.23,
  "roll": 12.50,
  "gyro_x": 120,
  "gyro_y": -45,
  "gyro_z": 10
}
```

- **joy_x/y:** Valor bruto do ADC do joystick (0-4095).
- **servo_x/y:** Ângulo comandado para os servos (0-180).
- **pitch/roll:** Inclinação calculada pelo MPU6050 em graus.
- **gyro_x/y/z:** Velocidade angular bruta (para análise futura).

6. Acesso ao Projeto (Simulação Wokwi)

A implementação completa da **Fase 2**, integrando o controle dos servomotores com a telemetria do sensor inercial MPU6050, está disponível para validação online através da plataforma de simulação Wokwi.

Neste link, é possível visualizar:

1. **Diagrama Esquemático:** Conexões elétricas entre ESP32-S3, Joystick, Servos e MPU6050.
2. **Código Fonte:** Firmware completo em C (FreeRTOS + ESP-IDF).
3. **Monitor Serial:** Visualização em tempo real dos dados JSON gerados (Pitch, Roll e Status).

Link para Simulação:

https://wokwi.com/projects/448970084530347009

7. Conclusão e Resultados

A implementação da Fase 2 foi bem-sucedida ao:

1. Estabelecer comunicação estável I²C com o MPU6050 no ESP32-S3.

2. Implementar o algoritmo de fusão trigonométrica para converter aceleração em graus.
3. Manter a estabilidade do controle dos servos (Fase 1) através do uso correto de FreeRTOS, onde a leitura do sensor (mais lenta) não interfere na resposta do motor (mais rápida).
4. Prover uma interface de depuração rica via Serial JSON.