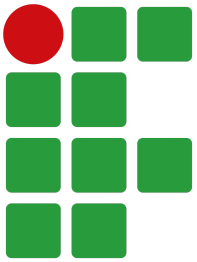

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA - CAMPUS CAMPINA GRANDE COORDENAÇÃO DO CURSO SUPERIOR BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

DANIEL LIMA NETO

GEOVANA STEFANI LOPES BEZERRA

IVYSSON FERNANDES DE QUEIROZ UCHÔA

LUIS FELIPE FERREIRA TAVARES NONATO

Projeto Final - Sistemas Embarcados - 2025.2

Relatório Técnico

Campina Grande - PB

2025

1. Introdução e Visão Geral

1.1. Objetivo Geral:

O objetivo deste projeto é o desenvolvimento de um sistema embarcado interativo para controle e monitoramento de uma mesa labirinto. O sistema, utilizando o microcontrolador ESP32, visa controlar a inclinação da mesa em dois eixos (X e Y) através de servomotores, permitindo que o usuário movimente uma esfera metálica até a saída utilizando um joystick analógico para executar os comandos.

Adicionalmente, o projeto tem como objetivo coletar os dados de movimento da mesa através da leitura de um sensor inercial (MPU6050), e transmitir sua orientação espacial para um dashboard no Grafana, criando uma representação virtual (Gêmeo Digital) que replica os movimentos físicos da mesa.

1.2. Funcionamento do Sistema:

O funcionamento do sistema integra controle mecânico e monitoramento digital, operando conforme a descrição a seguir:

- **Inicialização e Status:** Ao ser energizado, o ESP32 executa as rotinas de configuração dos periféricos. A prontidão do sistema é indicada visualmente através de um LED, sinalizando ao usuário que a calibração foi concluída.
- **Mecanismo de Controle:** A estrutura da mesa é sustentada por um mecanismo acionado por dois servomotores de 90g. A entrada de controle é realizada por um joystick analógico, cujos sinais são processados pelo ESP32 e convertidos em comandos PWM para os motores, alterando a inclinação da superfície para movimentar a esfera através do labirinto.
- **Sensoriamento:** Um acelerômetro e giroscópio (MPU6050) fixado à base da mesa monitora continuamente os ângulos de inclinação (Pitch e Roll). Esses dados são processados e enviados para um computador.
- **Visualização (Gêmeo Digital):** Na interface de supervisão (Grafana), os dados recebidos alimentam um modelo visual que replica a orientação da mesa física em tempo real.
- **Condição de Vitória:** Um sensor indutivo de proximidade npn tl-w5mc1 detecta quando a bolinha metálica chega ao fim do labirinto e sinaliza a vitória através de um LED indicador.

1.3. Arquitetura do Sistema

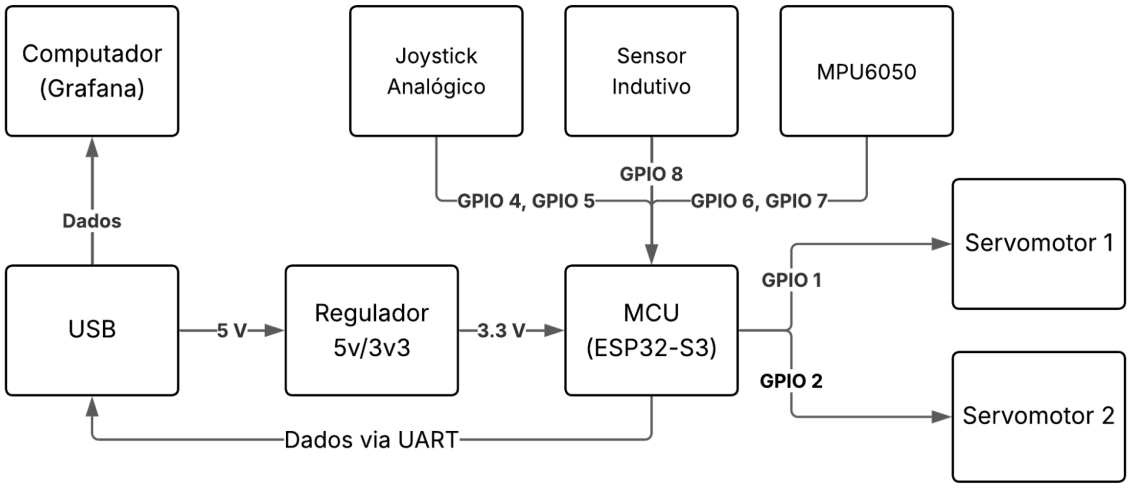

Imagem 1: Diagrama de blocos

2. Hardware e Eletrônica

2.1. Componentes Utilizados

Quant.	Componente	Função
1	ESP32	Microcontrolador principal
1	Joystick analógico	Controle dos servos nos eixos X e Y
2	Servo motor 90G	Movimento da mesa nos eixos X e Y
1	MPU6050	Sensor inercial para orientação (pitch e roll)
1	Cabo USB	Alimentação e envio de dados do ESP32
1	Sensor npn tl-w5mc1	Identificação da condição de vitória
1	LED	Indica estado do sistema (calibração/ jogando/ vitória)

2.2 Diagrama Esquemático

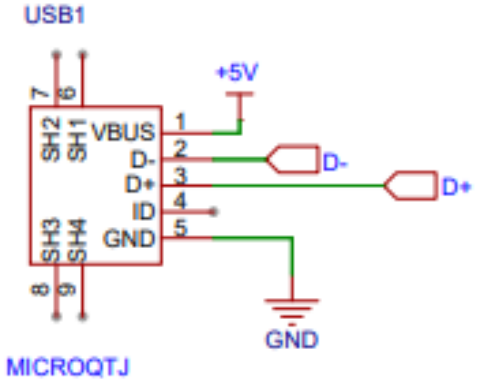

Imagem 2: Esquema do conector USB

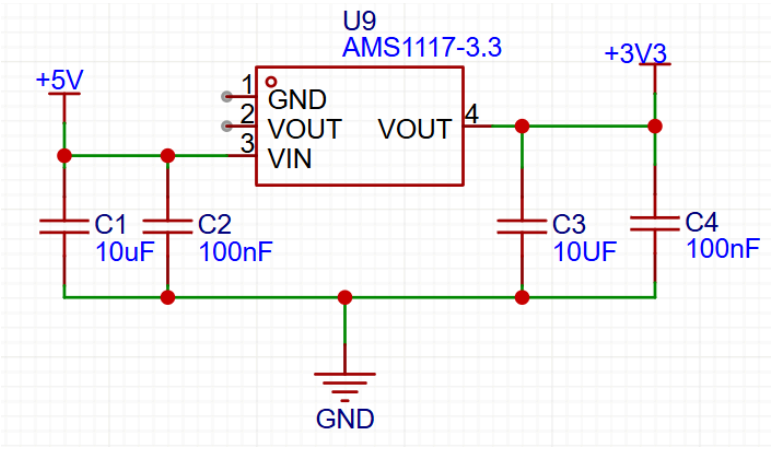

Imagem 3: Esquema do Conversor 5V para 3.3 V

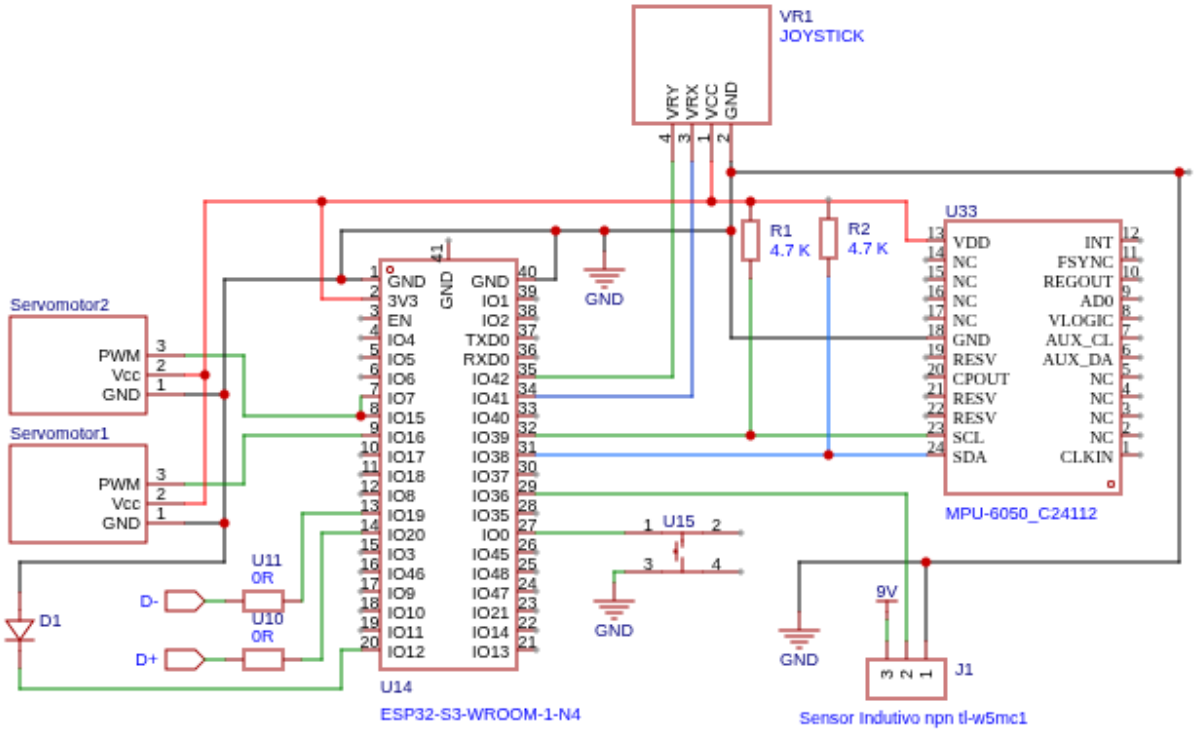

Imagem 4: Esquema da entrada e saída de dados com ESP32

3. Desenvolvimento de Firmware

O firmware foi desenvolvido utilizando o ESP-IDF, priorizando a modularidade e a resposta em tempo real através do FreeRTOS ao separar a leitura de sensores e o controle de atuadores em tarefas (Tasks) independentes.

3.1. Bibliotecas e Drivers Utilizados

Para o desenvolvimento do hardware, foram utilizados os drivers nativos do ESP-IDF, sendo os mais importantes:

- **driver/adc.h:** Responsável pela configuração e leitura dos conversores Analógico-Digitais (ADC1) para a interface do Joystick
- **driver/ledc.h:** Utilizado para a geração de sinais PWM (Pulse Width Modulation) com precisão de 16 bits, controlando o ângulo dos servomotores
- **driver/i2c.h:** Gerencia a comunicação serial síncrona com o sensor inercial MPU6050
- **freertos/semphr.h:** Implementa os mecanismos de sincronização (Mutex) para garantir a integridade dos dados compartilhados entre as tarefas
- **math.h:** Fornece funções trigonométricas (atan2, fabs) utilizadas para o cálculo da orientação espacial da mesa labirinto

3.2. Arquitetura de Tarefas (FreeRTOS)

O sistema foi dividido em cinco tarefas principais, escalonadas com base na sua necessidade de execução

Nome da Task	Função Principal
servo_task()	Controle dos motores
joystick_task()	Leitura, filtragem e normalização do input
monitor_task()	Log de debug e visualização de dados
mpu_task()	Leitura, cálculo e envio da orientação espacial da mesa labirinto
sensor_task()	faz a leitura do sensor indutiva para a realizar a condição de vitória e congelamento das task joystick, monitor e mpu

3.3. Fluxo de Dados

O funcionamento do firmware baseia-se em processamento concorrente, onde a entrada de comandos, o controle de motores e a determinação de orientação ocorrem simultaneamente, orquestrados pelo scheduler do FreeRTOS. O fluxo de dados do sistema, desde a entrada do usuário até a atuação física e monitoramento, ocorre em quatro estágios principais:

1. **Aquisição e tratamento de entrada (joystick_task)**
    - Esta tarefa é responsável pela interface homem-máquina.
    - Ao ler os valores brutos do ADC, o sistema aplica uma normalização e também aplica uma zona morta (deadzone) de 2% ao redor do centro do joystick para ignorar ruídos
    - Seguido disso, um filtro de Média Móvel Exponencial suaviza as variações bruscas do sinal
    - Por fim, o ângulo calculado não é enviado diretamente aos motores. Ele é escrito em uma estrutura de dados global (g_state)
2. **Sincronização de dados (Mutex)**
    - É utilizado um Semáforo do tipo Mutex (g_state_mutex) para garantir a integridade dos dados durante a leitura e escrita dos dados
    - Quando a joystick_task precisa atualizar a posição desejada, ela solicita o Mutex, bloqueia o acesso à variável, escreve o novo valor e libera o recurso

- Simultaneamente, a servo_task solicita o mesmo Mutex para ler a posição alvo. Assegurando que os servos nunca recebam dados parciais

3. **Atuação suave e controle de hardware (servo_task)**
    - A tarefa compara a posição atual do servo com a posição alvo recebida do joystick
    - Com isso, a posição atual é incrementada ou decrementada em pequenos passos (step_deg), criando uma transição linear e fluida
    - O valor resultante é convertido em Duty Cycle e enviado ao periférico LEDC, gerando o sinal PWM para os servomotores
4. **Coleta de dados independente (mpu_task)**
    - Ocorrendo em paralelo às tarefas de controle, a mpu_task monitora o comportamento físico da mesa
    - Ela comunica-se via I2C com o sensor MPU6050 para obter a aceleração e velocidade angular
    - Aplica-se um Filtro Complementar, com a fusão de dados do acelerômetro e giroscópio, para mitigar o ruído calculando a inclinação real da mesa
    - Os dados processados são enviados via Serial (UART/USB), servindo de base para a visualização do Gêmeo Digital no Grafana
5. **Condição de vitória (sensor_task)**
    - Realiza a leitura periódica do sensor indutivo de proximidade NPN para identificar a presença de objeto metálico no final do labirinto.
    - Detecta a ativação do sensor quando o nível lógico do pino de entrada assume valor baixo (LOW), característica do sensor NPN.
    - Utiliza controle de estado (`last_state`) para identificar transições e evitar múltiplas impressões repetidas no monitor serial
    - Controla a variável global `g_stop_mpu_print`, interrompendo temporariamente a impressão dos dados do MPU6050 durante a ativação do sensor.

4. Montagem da Mesa

A estrutura da mesa foi construída a partir de peças impressas em 3D, totalizando cinco partes individuais, conforme ilustrado na **Imagem 5**.

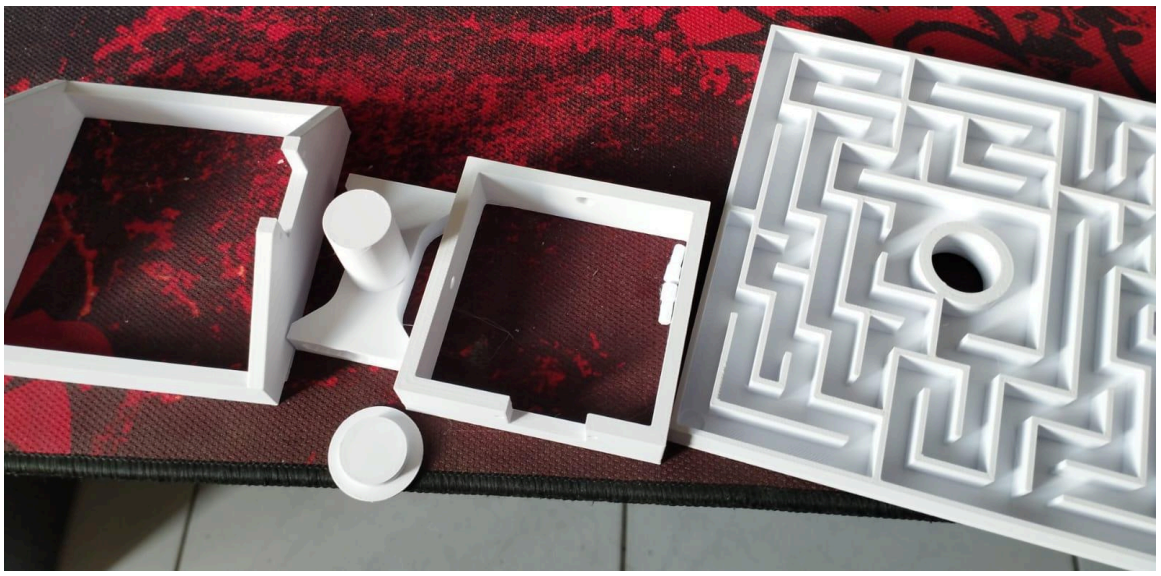

Imagem 5: Mesa labirinto desmontada

Para o funcionamento do sistema, as peças são encaixadas entre si ou acopladas por meio dos próprios componentes eletrônicos, como ocorre na fixação do sensor MPU6050 e no uso do eixo dos servomotores para a sustentação da mesa. Destacam-se como exceções o uso de parafusos para a fixação de cada servomotor em sua respectiva base, garantindo sua estabilidade, bem como a utilização de cola quente para a fixação do sensor indutivo abaixo do labirinto, conforme apresentado na **imagem 6**.

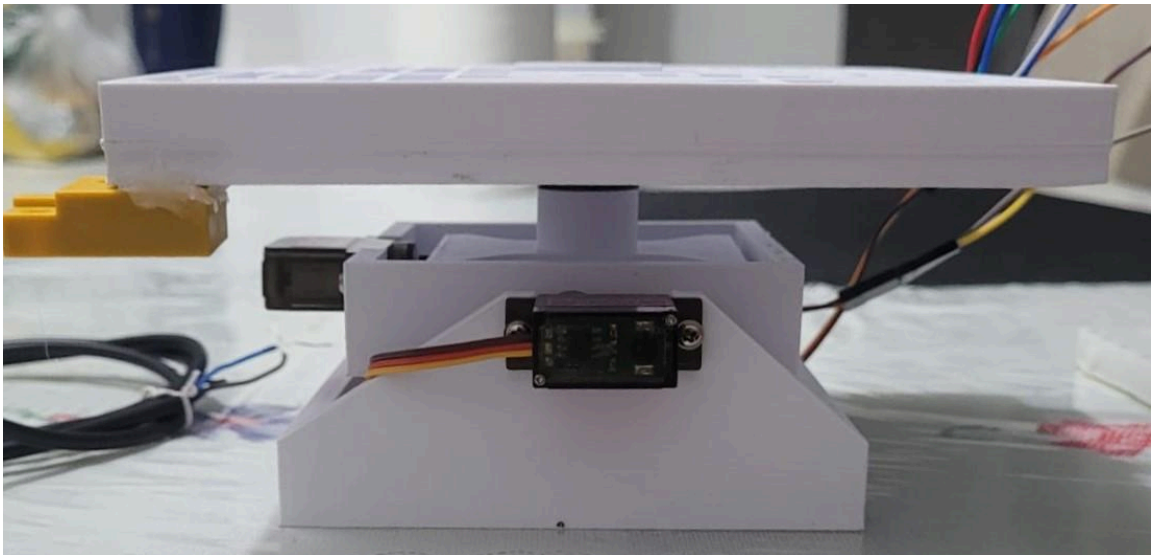

Imagem 6: Vista lateral da mesa labirinto montada

A definição dos pontos de início e término do desafio da mesa labirinto é realizada por meio de duas marcações localizadas em extremidades opostas da estrutura. O ponto de largada corresponde à posição inicial da bolinha metálica, indicado pela seta vermelha, enquanto o ponto de chegada é caracterizado por um orifício que permite a queda da bolinha e o acionamento do sensor indutivo posicionado logo abaixo, indicado pela seta verde, conforme apresentado na **Imagem 7**, que mostra a vista superior da mesa.

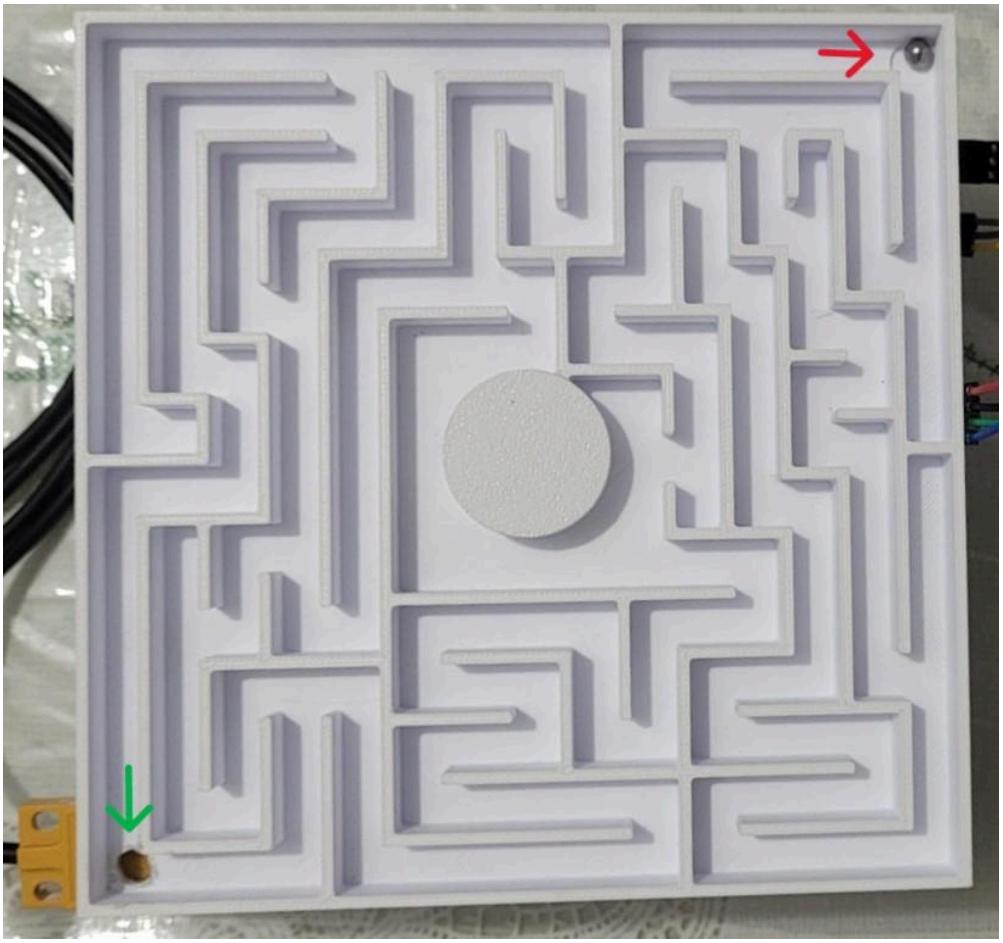

Imagem 7: Início e fim do percurso do labirinto.

5. Interface e Monitoramento (Grafana)

Com o objetivo de permitir a comunicação entre o sistema físico e o computador, bem como o monitoramento em tempo real da posição da mesa, foi desenvolvido um “gêmeo virtual” utilizando em conjunto a plataforma Grafana, o banco de dados InfluxDB e a linguagem Python. Essa abordagem permite replicar virtualmente o comportamento dinâmico da mesa física, exibindo em tempo real os ângulos de X e Y medidos pelo sistema embarcado.

5.1 Configuração do banco de dados

Responsável pela persistência e armazenamento dos dados coletados pelo ESP32 a partir do sensor MPU6050, foi utilizado o banco de dados InfluxDB em ambiente local. O InfluxDB é um banco de dados de séries temporais (TSDB) de código aberto, otimizado para a ingestão, armazenamento e consulta de dados que variam ao longo do tempo, como métricas e sinais de sensores, o que o torna adequado para esta aplicação.

Para facilitar a implementação e permitir a reprodução dos resultados, o InfluxDB foi instalado e configurado por meio de uma imagem Docker, possibilitando seu acesso em ambiente local via “`localhost:8086`”. Após a inicialização do serviço, foi criado um bucket denominado mesa_labirinto, destinado ao armazenamento dos valores de pitch e roll obtidos a partir do sensor inercial.

5.2 Configuração Grafana

O Grafana é uma plataforma de código aberto voltada à visualização e análise de dados, permitindo a criação de painéis interativos e gráficos em tempo real a partir de diferentes fontes de dados, incluindo o InfluxDB. Assim como o banco de dados, o Grafana foi instalado em ambiente local utilizando uma imagem Docker, o que simplifica sua configuração e uso.

Após a instalação e configuração das credenciais de acesso, foi estabelecida a comunicação entre o Grafana e o InfluxDB por meio da rede local. Com essa conexão, foi possível criar um dashboard dedicado ao projeto, responsável por filtrar e exibir, em tempo real, os dados de pitch e roll escritos no bucket “mesa_labirinto”.

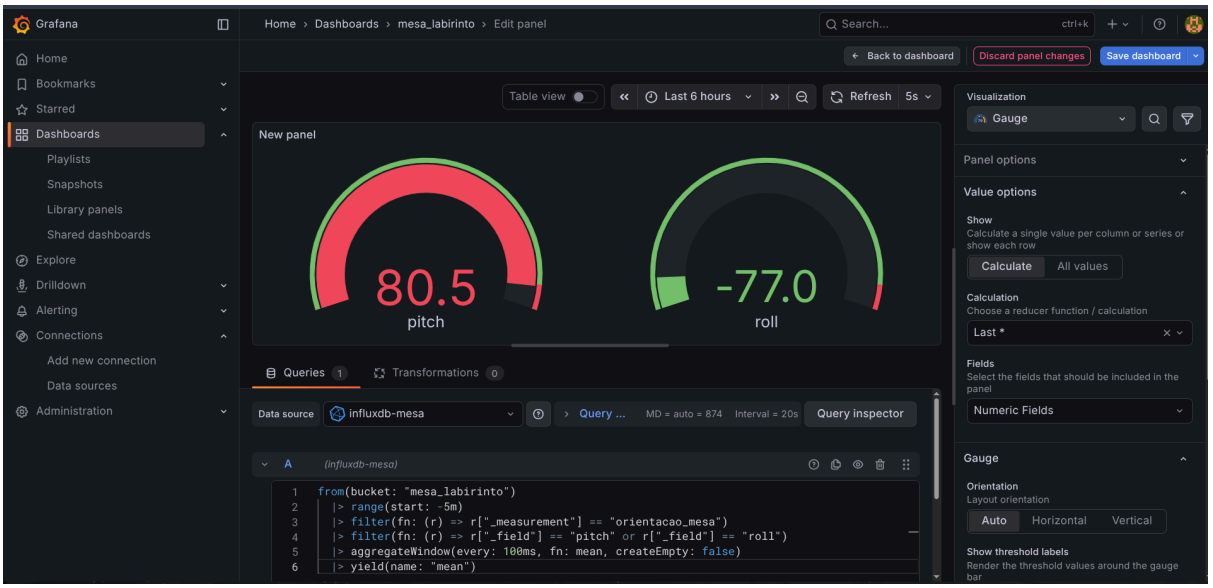

Imagem 8: Visualização do pitch e roll pelo Grafana

5.3 Gateway de coleta e gravação de dados

Como não é possível estabelecer uma comunicação direta entre o ESP32 e o Grafana ou entre o ESP32 e o InfluxDB, foi desenvolvido um script em linguagem Python para atuar como intermediário na coleta e no encaminhamento dos dados. Esse script realiza a leitura dos dados transmitidos pelo ESP32 via UART, por meio de uma conexão USB, no formato JSON.

Após a recepção, os dados são tratados, validados e então gravados no banco de dados InfluxDB, sendo inseridos no bucket “mesa_labirinto”. Dessa forma, o script funciona como um gateway de aquisição de dados, garantindo a integração entre o sistema embarcado e a interface de monitoramento em tempo real.

6. Análise de Desafios

Durante o ciclo de desenvolvimento e integração do sistema, foram identificados desafios técnicos que impactaram o desempenho inicial do protótipo. Abaixo, detalham-se os obstáculos encontrados e as soluções aplicadas para solucioná-los.

6.1 Suavização de controle

Nos testes iniciais de acoplamento entre o joystick e os atuadores, foi observado que a resposta dos servomotores era muito brusca, tendendo a movimentos extremos. Esse

comportamento dificultava o controle preciso da esfera metálica e comprometia a jogabilidade e a estabilidade da mesa.

Solução: Foi implementada no firmware uma lógica de rampa de aceleração. Em vez de aplicar o ângulo alvo instantaneamente, o algoritmo incrementa ou decrementa a posição dos servos em passos fixos e temporizados. Essa alteração resultou em movimentos mais lentos e fluidos, garantindo a precisão necessária para a navegação no labirinto.

6.2 Ruído na Leitura no Sensor Inercial

A tentativa inicial de calcular a orientação da mesa utilizando apenas os dados brutos do giroscópio (MPU6050) revelou-se bastante impreciso, especificamente devido ao fenômeno de drift (deriva), onde o sensor registrava rotação mesmo com a mesa estática.

Solução: Implementou-se um Filtro Complementar para fusão de sensores. Esta técnica combina a estabilidade estática do acelerômetro com a resposta dinâmica do giroscópio para calcular os ângulos X e Y. O filtro eliminou as oscilações inexistentes e estabilizou a leitura enviada ao dashboard de monitoramento.

6.3 Detecção Automática de Vitória

O projeto original possui apenas cavidades simples para marcar o início e o fim do trajeto. No entanto, houve a necessidade de criar um método automático para verificar a conclusão do desafio, indo além de apenas depositar a esfera na cavidade final.

Solução: Realizou-se uma alteração estrutural no ponto de chegada, criando um furo com tamanho maior que o da esfera. Logo abaixo desta abertura, foi fixado um sensor indutivo. Aproveitando as propriedades metálicas da esfera, o sensor detecta a sua presença e envia um sinal digital ao ESP32, que aciona um indicador visual (LED) e congela as outras Task ativas, sinalizando vitória do usuário.