



**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 Moraes**

**Emilieny de Souza Silva**

**Victor José Cordeiro de Medeiros**

**Campina Grande, 15 de dezembro de 2025**

# Sumário

<b>1. Introdução e Objetivo Geral.....</b>	<b>3</b>
<b>2. Descrição do Sistema e Hardware.....</b>	<b>3</b>
2.1 Lista de componentes de Hardware.....	3
2.2 Diagrama em Blocos do Sistema.....	4
2.3 Esquemático.....	4
<b>3. Estrutura e Funcionamento do Código (FreeRTOS).....</b>	<b>5</b>
3.1 Descrição das Tarefas (Tasks).....	5
3.2 Funcionamento do Sistema.....	6
<b>4. Integração do Gêmeo Digital (Visualização em Grafana).....</b>	<b>6</b>
4.1 Protocolo de Comunicação Serial e Banco de Dados.....	6
4.2 Configuração e Dashboard Grafana.....	7
<b>5. Dificuldades e Soluções Encontradas.....</b>	<b>8</b>
<b>6. Resultados.....</b>	<b>9</b>
6.1 Modo calibração.....	9
6.2 Modo Pitch/Roll.....	9
6.3 Vitória.....	9
6.4 Calibração.....	10
6.5 Jogando.....	10
6.6 Vitória.....	10

## 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.

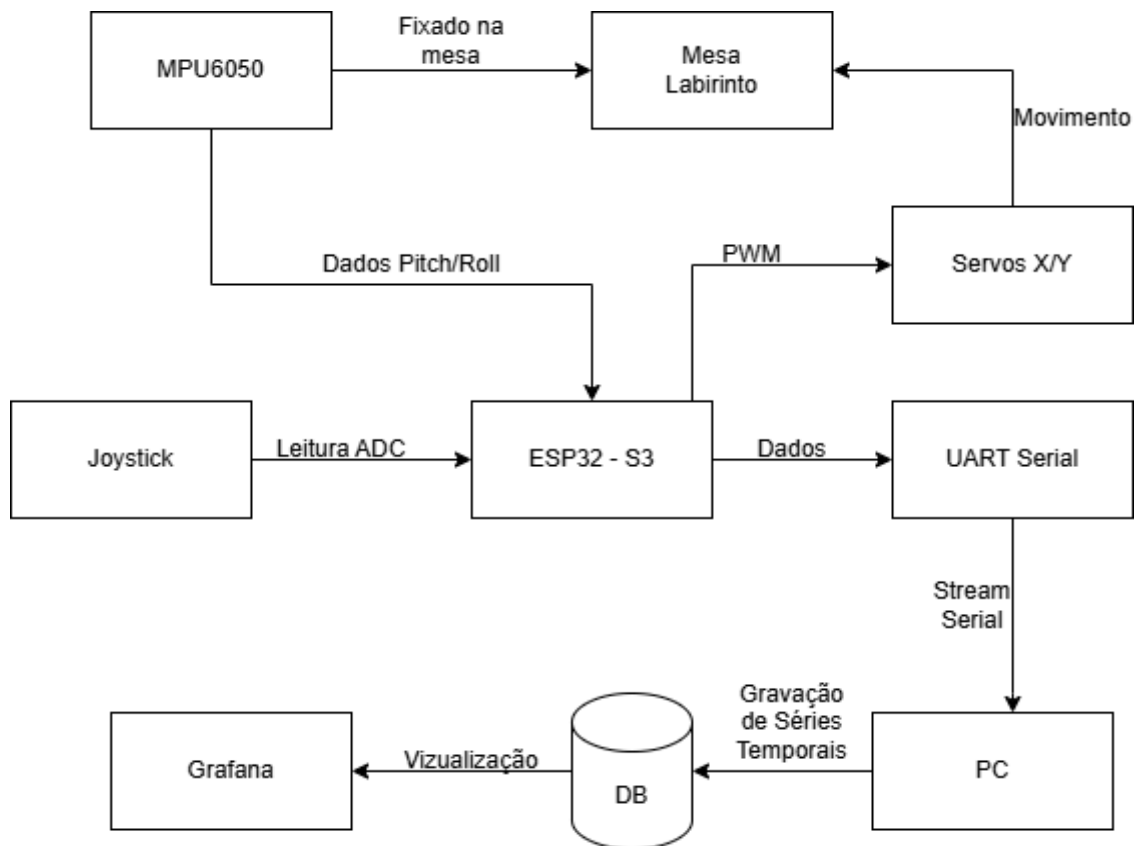
## 2. Descrição do Sistema e Hardware

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

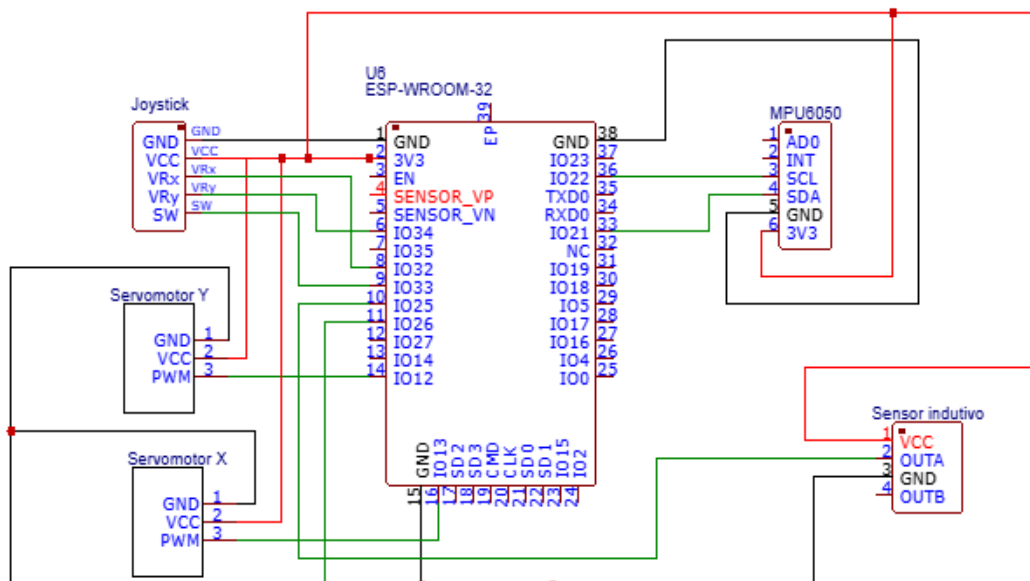
### 2.1 Lista de componentes de Hardware

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)	I <sup>2</sup> C
1	LED	Indicador de sistema pronto (LED_READY)	GPIO
1	Sensor de Vitória	Deteccção 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
<b>Task MPU6050</b> (task_mpu6050)	Leitura do sensor inercial e cálculo de Pitch e Roll.	7 (Mais Alta)	Lê os registradores do MPU6050 via I <sup>2</sup> C e chama compute_pitch_roll_from_accel a cada 50ms.
<b>Task Servo Control</b> (task_servo_control)	Aplicação suave do movimento nos servomotores.	6	A cada 20ms, aplica um <b>filtro de suavização</b> nos ângulos alvo (angleX_target, angleY_target) para evitar movimentos bruscos.
<b>Task Leitura Joystick</b> (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.
<b>Task Touch Sensor</b> (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.

<b>Task</b> <b>Debug/Comunicação</b> (task_debug)	Envio periódico dos dados de orientação.	3 (Mais Baixa)	A cada 500ms, imprime os valores de Pitch e Roll no formato <b>JSON</b> ( <code>{"pitch": %.2f, "roll": %.2f}\n</code> ), 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.

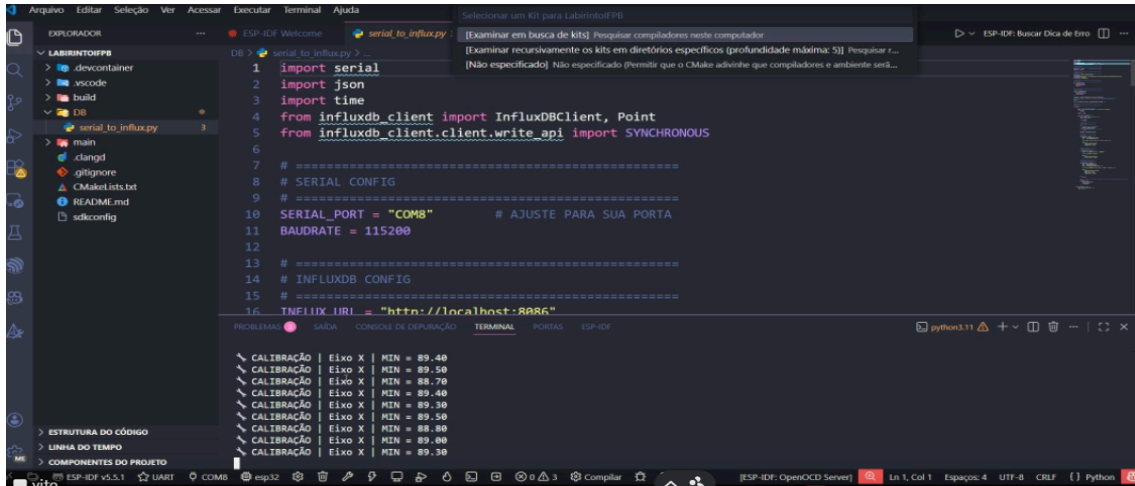
Dificuldade Encontrada	Solução Implementada
<b>Instabilidade no Movimento (Jitter):</b> O movimento dos servos era abrupto devido à leitura direta e ruidosa do ADC.	<b>Solução:</b> Implementação de um <b>Filtro de Suavização</b> na <code>task_servo_control</code> , tornando o movimento proporcional e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

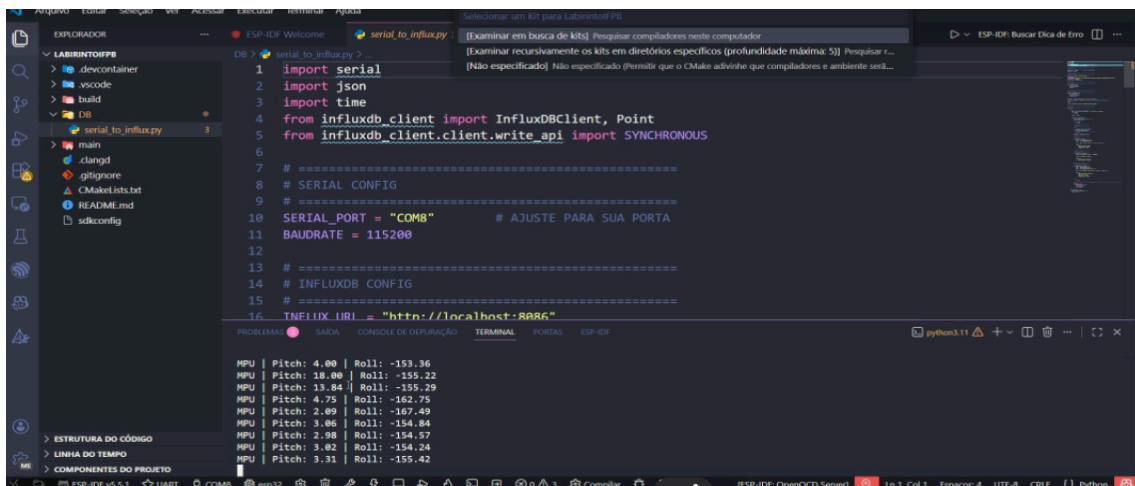


```
1 import serial
2 import json
3 import time
4 from influxdb_client import InfluxDBClient, Point
5 from influxdb_client.client.write_api import SYNCHRONOUS
6
7 # =====
8 # SERIAL CONFIG
9 # =====
10 SERIAL_PORT = "COM8"      # AJUSTE PARA SUA PORTA
11 BAUDRATE = 115200
12
13 # =====
14 # INFLUXDB CONFIG
15 # =====
16 INFLUX_URI = "http://localhost:8086"
```

TERMINAL

```
✓ CALIBRAÇÃO | Eixo X | MIN = 89.40
✓ CALIBRAÇÃO | Eixo X | MIN = 89.50
✓ CALIBRAÇÃO | Eixo X | MIN = 88.70
✓ CALIBRAÇÃO | Eixo X | MIN = 89.40
✓ CALIBRAÇÃO | Eixo X | MIN = 89.30
✓ CALIBRAÇÃO | Eixo X | MIN = 89.50
✓ CALIBRAÇÃO | Eixo X | MIN = 88.80
✓ CALIBRAÇÃO | Eixo X | MIN = 89.00
✓ CALIBRAÇÃO | Eixo X | MIN = 89.30
```

### 6.2 Modo Pitch/Roll

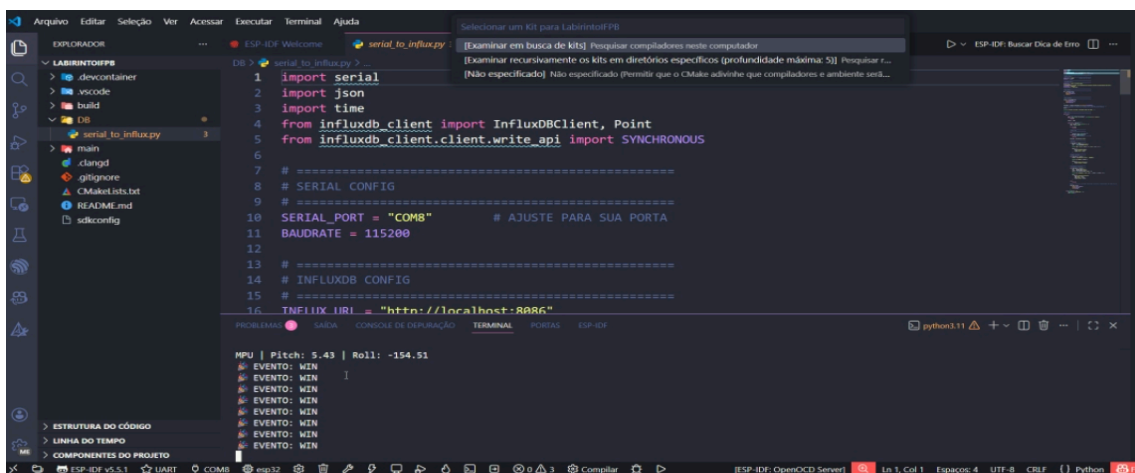


```
1 import serial
2 import json
3 import time
4 from influxdb_client import InfluxDBClient, Point
5 from influxdb_client.client.write_api import SYNCHRONOUS
6
7 # =====
8 # SERIAL CONFIG
9 # =====
10 SERIAL_PORT = "COM8"      # AJUSTE PARA SUA PORTA
11 BAUDRATE = 115200
12
13 # =====
14 # INFLUXDB CONFIG
15 # =====
16 INFLUX_URI = "http://localhost:8086"
```

TERMINAL

```
MPU | Pitch: 4.00 | Roll: -153.36
MPU | Pitch: 18.00 | Roll: -155.22
MPU | Pitch: 13.84 | Roll: -155.29
MPU | Pitch: 4.75 | Roll: -162.75
MPU | Pitch: 2.09 | Roll: -167.49
MPU | Pitch: 3.06 | Roll: -154.84
MPU | Pitch: 2.98 | Roll: -154.57
MPU | Pitch: 3.82 | Roll: -154.24
MPU | Pitch: 3.31 | Roll: -155.42
```

### 6.3 Vitória

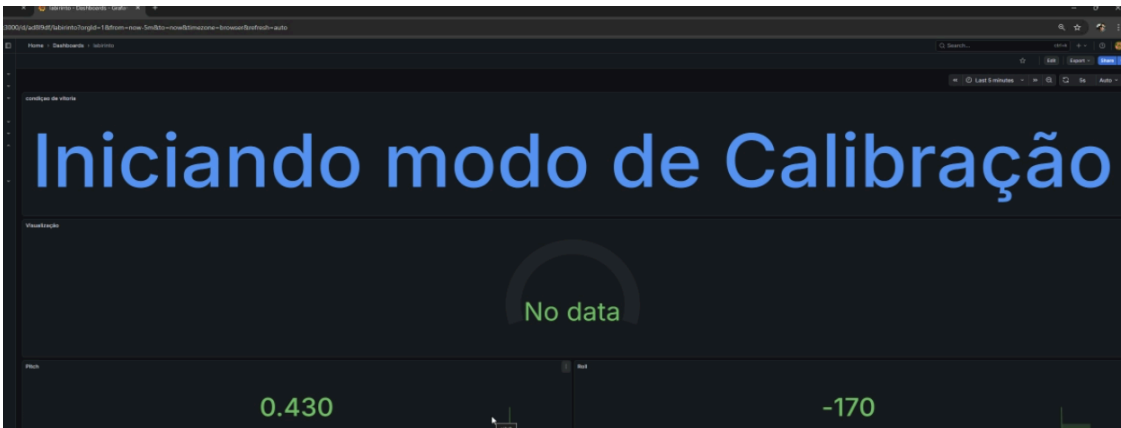


```
1 import serial
2 import json
3 import time
4 from influxdb_client import InfluxDBClient, Point
5 from influxdb_client.client.write_api import SYNCHRONOUS
6
7 # =====
8 # SERIAL CONFIG
9 # =====
10 SERIAL_PORT = "COM8"      # AJUSTE PARA SUA PORTA
11 BAUDRATE = 115200
12
13 # =====
14 # INFLUXDB CONFIG
15 # =====
16 INFLUX_URI = "http://localhost:8086"
```

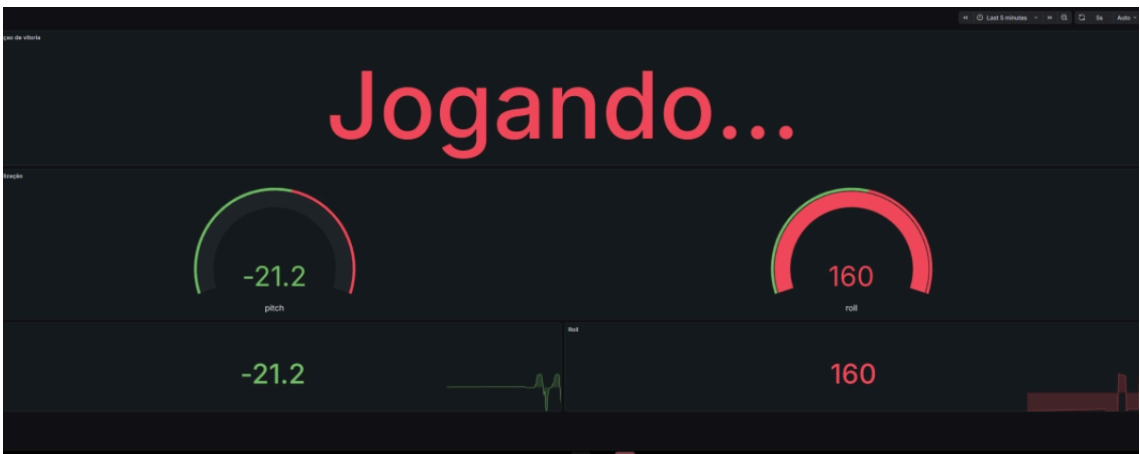
TERMINAL

```
MPU | Pitch: 5.43 | Roll: -154.51
EVENTO: WIN
EVENTO: WIN
EVENTO: WIN
EVENTO: WIN
EVENTO: WIN
EVENTO: WIN
EVENTO: WIN
```

## 6.4 Calibração



## 6.5 Jogando



## 6.6 Vitória

