



## CURSO DE REDES DE COMPUTADORES

Dannihercleston Victorino Silva  
Ana Lívia Vicente Torres

### RELATÓRIO FINAL

**Monitoramento de Deslizamento de Encostas, usando  
Microcontrolador, MPU6050, MQTT, Docker,  
InfluxDB e Grafana**

**Dannihercleston Victorino Silva  
Ana Lívia Vicente Torres**

**Monitoramento de Deslizamento de Encostas, usando  
Microcontrolador, MPU6050, MQTT, Docker, InfluxDB e Grafana.**

**Trabalho de Pesquisa da Disciplina do Projeto Integrador**  
Curso Tecnólogo em Redes de Computadores  
Instituto Federal de Ciências e Tecnologia do Rio Grande do Norte, Natal Central  
**Orientador: Moisés Cirilo de Brito Souto**

Natal 2024

# Sumário

<b>1</b>	<b>Introdução</b>	<b>4</b>
1.1	Metodologia . . . . .	5
<b>2</b>	<b>Sprint 1 - Seleção do Sensor Adequado e Aquisição</b>	<b>7</b>
2.1	Objetivo . . . . .	7
2.2	Comparação Entre os Sensores . . . . .	7
2.2.1	Strain gage . . . . .	7
2.2.2	NEO-6 . . . . .	8
2.2.3	MPU-6050 . . . . .	8
2.2.4	TEROS 12 . . . . .	8
2.2.5	Fibra óptica . . . . .	8
2.2.6	Tabela Comparativa . . . . .	9
2.2.7	Sensor Escolhido . . . . .	10
<b>3</b>	<b>Sprint 2 - Configuração do ESP32 e MPU6050</b>	<b>12</b>
3.0.1	Objetivo . . . . .	12
3.0.2	Configuração ESP32 e Broker MQTT . . . . .	12
<b>4</b>	<b>Sprint 3 - Instalação e Configuração do Docker e dos containers telegraf, influxdb e Grafana</b>	<b>14</b>
4.0.1	Configuração do Docker, Containers e Telegraf . . . . .	14
<b>5</b>	<b>Sprint 4 - Configuração do Grafana</b>	<b>16</b>
5.1	Objetivo . . . . .	16
5.1.1	Configuração do Grafana para obter dados do InfluxDB	16
5.1.2	Configuração Para Receber os Alertas . . . . .	17
5.1.3	Materiais Para a Maquete . . . . .	18
5.1.4	Simulação 3D Para Detectar Deslizamento de Encostas	19
5.1.5	MQTT na Rede Local . . . . .	20

<b>6 Sprint 5 - Testes, Simulação e Análise de Resultados via Maqueta</b>	<b>21</b>
6.1 Objetivo . . . . .	21
6.1.1 Construção da Maquete . . . . .	21
6.1.2 Desigualdade de Chebyshev . . . . .	24
6.1.3 Aplicando o Teorema no Monitoramento de Encostas .	25
6.1.4 Python e Chebyshev . . . . .	25
<b>7 Sprint 6 - Relatório Final e Testes</b>	<b>27</b>
7.1 Objetivos . . . . .	27
7.1.1 Resultados com o teorema de Chebyshev . . . . .	27
7.1.2 Filtro de Kalman . . . . .	28
7.2 Teste com a Maqueta . . . . .	30
7.2.1 Alertas Gerados Durantes os Testes . . . . .	31
7.3 Cenários de Monitoramento Usando Processing . . . . .	34
<b>8 Documentação</b>	<b>36</b>
<b>9 Commits</b>	<b>37</b>
<b>10 Resultados</b>	<b>38</b>
<b>11 Conclusão</b>	<b>40</b>

# Capítulo 1

## Introdução

O movimentos de massa, como deslizamentos e escorregamentos, envolvem o deslocamento de solos e rochas sob a gravidade, intensificados pela água. O Brasil é especialmente vulnerável devido às chuvas intensas em áreas montanhosas e à ação humana, como desmatamentos e ocupações irregulares em encostas. Esses movimentos são classificados em quedas, deslizamentos, fluxos de lama e detritos, e subsidências. Dentre eles, os deslizamentos são os mais monitorados pelo CEMADEN. Os fluxos de lama são rápidos e destrutivos, enquanto subsidências resultam no afundamento do solo causado por colapsos subterrâneos [1]. Assim, este projeto planeja monitorar e detectar, antecipadamente, movimentos de massa, contribuindo para a preservação de vidas em casos de desastres naturais. Para alcançar essa detecção precoce, utiliza-se o sensor MPU6050, que integra um acelerômetro de 3 eixos e um giroscópio de 3 eixos em um único dispositivo. O MPU6050 é amplamente utilizado para monitoramento de movimentos devido à sua precisão, baixo consumo de energia, tamanho compacto e medidor de temperatura integrado, características que o tornam ideal para este tipo de projeto [15]. Além do sensor, o microcontrolador ESP32 é utilizado para uma comunicação sem fio. Para otimizar essa comunicação, o protocolo MQTT será responsável por enviar os dados do sensor para um tópico. Um ponto importante deste projeto é seu baixo custo, desde os dispositivos de medição até a etapa final de alerta. Nesse sentido, o banco de dados utilizado é o InfluxDB, que utiliza o sistema próprio de carregamento de dados, o Telegraf, como agente para receber os tópicos MQTT e enviá-los ao banco. Além disso, o software Grafana é usado para visualização dos dados em dashboards, enquanto os alertas serão recebidos no Telegram por meio de um bot, que processa informações em tempo real através de um código desenvolvido em Python. Todos esses sistemas são provisionados por meio de um docker compose, o qual mantém todos operando localmente. Todo o projeto foi desenvolvido e testado ao

longo de seis semanas, no qual cada semana envolveu a entrega de sprints pré-estabelecido, no qual são: Sprint 1: Seleção de Sensores Adequados e Aquisição, Sprint 2: Configuração do ESP32 e MPU6050, Sprint 3: Instalação e Configuração do Docker e dos containers telegraf, influxdb e Grafana, Sprint 4: Configuração do Grafana, Sprint 5: Testes, Simulação e Análise de Resultados via Maquete e Sprint 6: Relatório Final. Todas as informações, código e documentação foram centralizadas em um repositório do Github [13].

Para comprovar a eficácia do projeto, foi desenvolvida uma metodologia de teste experimental, no qual envolveu a implementação prática de um cenário específico, usando uma maquete para avaliar o funcionamento e resposta do sistema. Apesar de algumas limitações ao longo do desenvolvimento do projeto, os resultados obtidos foram satisfatórios indicando que ele é viável e pode ser usado para prevenir destastres naturais relacionados a deslizamentos de terra.

## 1.1 Metodologia

A metodologia utilizada foi do tipo experimental e qualitativa. Experimental porque foram realizados testes numa maquete, que buscavam simular um cenário real de deslizamento de terra, e qualitativa, pois se propôs a avaliar o funcionamento e a eficácia do sistema de monitoramento de deslizamento de encostas em um ambiente controlado. O experimento foi realizado na casa de um dos integrantes do grupo, localizado em Lagoa Seca, Natal-RN. Onde foi montada a maquete que simulava um cenário real de deslizamento de terra. Esta maquete se trata de um recipiente retangular que mede 27 cm por 55 cm por 39 cm. Além disso, o sensor, microcontrolador e uma bateria de 9V foram colocados numa caixa de passagem de câmera posicionada na extremidade do recipiente retangular, apoiada por um cano de PVC medindo 15 cm.

O sensor MPU6050 foi integrado ao ESP32, que se comunicava com o sistema sem fio via protocolo MQTT. A comunicação entre os componentes foi realizada por meio de uma rede Wi-Fi local. O Telegraf foi configurado para captar os dados enviados pelo ESP32 e encaminhá-los para um bucket no InfluxDB, enquanto o Grafana foi utilizado para a visualização em dashboards. Ademais, um código Python monitorava os dados em tempo real e enviava alertas para um grupo do Telegram sempre que um desvio padrão fora do estipulado fosse identificado, seguindo a regra de Chebyshev. Esses dados enviados para o bucket já estavam com o filtro de Kalman.

Os testes foram realizados em condições controladas, em ambiente fe-

chado, garantindo a estabilidade da conexão sem interferências externas. O cenário de teste incluiu a simulação de um deslizamento de terra ao distribuir dois litros de água como se fosse chuva, próximo ao suporte do sensor, no qual foi feito com uma garrafa de plástico de dois litros, que possuía furos na tampa, reproduzindo um cenário de chuva real. O desempenho do sistema foi avaliado com base no recebimento dos alertas enviados ao Telegram, na resposta do sensor MPU6050 à detecção de movimentos anômalos.

Esse procedimento experimental foi conduzido ao longo de duas semanas, no qual se iniciou no sprint V5, mas só foi finalizada na V6. Algumas limitações observadas durante os testes mostraram que o trabalho precisa de mais testes para se obter um monitoramento sem falhas, contudo, foi obtido um bom desempenho no que o projeto propôs.

# **Capítulo 2**

## **Sprint 1 - Seleção do Sensor Adequado e Aquisição**

### **2.1 Objetivo**

O objetivo principal deste sprint é identificar e avaliar os sensores disponibilizados no mercado. No qual será selecionado o mais adequado para um sistema de monitoramento de encostas. Para alcançar esse objetivo, são realizadas atividades planejadas, como pesquisa, comparação de diferentes sensores e análise de suas especificações técnicas. Ao final, espera-se ter definido o melhor sensor para a proposta do projeto.

### **2.2 Comparação Entre os Sensores**

#### **2.2.1 Strain gage**

O sensor é utilizado para medir deformações em objetos com base na variação da resistência elétrica. O sensor é altamente preciso, com tolerâncias de resistência que variam de  $\pm 0,15\%$  a  $\pm 0,5\%$  e um fator de medição sensível à deformação. Ele é projetado para ser durável e robusto, operando em temperaturas entre  $-75^{\circ}\text{C}$  e  $200^{\circ}\text{C}$ . A instalação é facilitada por terminais de solda pré-cabeados e fitas de cobre estanhado, e o sensor é compatível com diversos materiais, o que permite sua integração em diferentes sistemas de medição [2].

## **2.2.2 NEO-6**

O NEO-6 é um módulo GPS desenvolvido pela u-blox, conhecido por sua alta precisão e sensibilidade em navegação e comunicação sem fio. Ele oferece precisão horizontal de até 2 metros, ou menos de 1 metro com SBAS + PPP em condições ideais. A sensibilidade de rastreamento pode chegar a -162 dBm, e opera em temperaturas de -40°C a 85°C, mostrando sua robustez. O NEO-6 é configurável via UART, USB, SPI, e DDC, com suporte para múltiplos protocolos (NMEA, UBX, RTCM) e compatibilidade com antenas passivas e ativas, facilitando a integração com diversos sistemas [3].

## **2.2.3 MPU-6050**

O MPU-6050 é um sensor de movimento de 6 eixos, combinando um giroscópio e um acelerômetro de três eixos em um único chip, amplamente usado em eletrônica e robótica. Ele mede simultaneamente a velocidade angular e a aceleração em três dimensões. A sensibilidade do acelerômetro varia de acordo com a faixa de escala, com opções de  $\pm 2g$  a  $\pm 16g$ , enquanto a sensibilidade do giroscópio varia de  $\pm 250^{\circ}/s$  a  $\pm 2000^{\circ}/s$ . O sensor também mede a temperatura com precisão e opera em uma ampla faixa térmica. A interface I2C facilita a leitura e manutenção dos dados dos sensores, com suporte para diagnósticos e auto-teste [4].

## **2.2.4 TEROS 12**

O sensor TEROS 12 é um dispositivo avançado para medir a umidade e temperatura do solo, utilizando tecnologia capacitiva para garantir medições precisas. Ele possui agulhas de aço inoxidável robustas e um sensor de temperatura estrategicamente posicionado para leituras exatas. A calibração do sensor reduz a variabilidade entre unidades para menos de 1%, assegurando consistência. Construído com materiais duráveis, o TEROS 12 é projetado para operar em condições adversas, com uma vida útil de até 10 anos. O sensor é compacto, fácil de instalar e suporta comunicação SDI-12, permitindo integração com diversos sistemas e monitoramento remoto em tempo real[5].

## **2.2.5 Fibra óptica**

Um sensor de força de fibra óptica opera baseado na interferência modulada, onde a aplicação de força deforma a fibra e altera a transmissão da luz. Fabricado pela SICK, o sensor possui ajuste de sensibilidade com 10 voltas e

uma escala de 270°, com distâncias de comutação de 0 a 4.000 mm (unidirecional) e 0 a 160 mm (varredura). Ele é robusto, operando em temperaturas de -25 °C a +55 °C e sendo armazenado de -40 °C a +70 °C. A instalação é facilitada por um cabo de 3 fios, uma cantoneira de fixação, e uma tampa de proteção rebatível, tornando-o prático para aplicações industriais [6].

### 2.2.6 Tabela Comparativa

Tipo de Sensor	STRAIN GAGE	NEO-6	MPU6050	TEROS 12	Fibra Óptica
Precisão e Sensibilidade	±0.15% a ±0.5% - 2.0 ±5% e 2.10 ±10%	-162 dBm e reacquisition de até -160 dBm	Acelerometro: ±2g: 16384 LSB/g a ±16g: 2048 LSB/g giroscópio: ±250 °/s: a ±2000 °/s: 16.4 LSB/°/s	±0.03 m3/m3 typical in mineral soils that have solution EC <8 dS/ me	10 voltas e escala de 270
Durabilidade e Robustez	-75 a 200°C	-40°C a 85°C	-96.85 °C a 98.99 °C.	-40°C a 60°C	25 °C a +55 °C
Facilidade de Instalação e Manutenção	terminais de solda pré-cabeados e fitas de cobre estanhado chato	ART, USB, SPI, DDC	Interfaces I2C	ferramenta Borehole	cantoneira de fixação
Conectividade e Integração	terminais de solda ou fios de fitametálicas para a conexão dos cabos de medição	USS e protocolos (NMEA, UBX, RTCM)	barramento I2C	SDI-12 - logger ZL6	cabo de 3 fios com 2 metros
Custo	baixo	baixo	baixo	baixo	alto

## 2.2.7 Sensor Escolhido

Após a comparação da tabela entre os sensores disponíveis, o selecionado foi o MPU6050. Devido à sua versatilidade, alta precisão, robustez, facilidade de instalação e manutenção, baixo custo e ampla documentação.

As principais observações para a escolha são: precisão, robustez e custo. O MPU6050 oferece alta precisão com: Acelerômetro: Sensibilidade ajustável entre  $\pm 2g$  e  $\pm 16g$ , com variações de 16384 LSB/g a 2048 LSB/g. Giroscópio: 9 Sensibilidade ajustável entre  $\pm 250^{\circ}/s$  e  $\pm 2000^{\circ}/s$ , com variações de 131 LSB/ $^{\circ}/s$  a 16.4 LSB/ $^{\circ}/s$ . Comparado ao sensor de fibra óptica, que também possui alta precisão e sensibilidade, o MPU6050 se destaca pela sua capacidade de detectar movimentos e inclinações, medindo aceleração linear em três direções (X, Y e Z) e velocidade angular em três eixos (pitch, roll e yaw).

Embora o NEO-6 ofereça alta precisão para rastreamento de posição, ele não mede movimentos e inclinações, limitando sua utilidade para o monitoramento de deslizamentos. O TEROS 12 é especializado em medições de umidade e temperatura do solo, e não é adequado para detectar movimentos de massa. Além disso, o MPU6050 suporta uma faixa de temperatura operacional de -96.85 °C a 98.99 °C, que é significativamente mais ampla do que o intervalo do sensor de fibra óptica (-25 °C a +55 °C). Isso o torna adequado para uma ampla gama de condições ambientais, comparável ao NEO-6 e ao TEROS 12, que operam em faixas de temperatura de -40°C a 85°C e -40°C a 60°C, respectivamente.

O custo do MPU6050 é baixo, tornando-o uma opção economicamente viável para projetos, especialmente quando comparado ao sensor de fibra óptica, que possui um custo mais elevado. Embora o TEROS 12 e o NEO-6 também tenham custos baixos, o MPU6050 oferece uma combinação vantajosa de custo e funcionalidade ao integrar medição de aceleração e velocidade angular em um único chip. Por fim, o MPU6050 combina baixo custo, alta precisão e uma ampla faixa de temperatura, tornando-o uma escolha superior para o monitoramento de deslizamentos de massa. Sua capacidade de medir simultaneamente aceleração e velocidade angular supera as capacidades dos sensores NEO-6 e TEROS 12 para essa aplicação. Comparado ao sensor de fibra óptica, o MPU6050 oferece uma solução mais econômica e versátil, adequada para monitorar a estabilidade e prever deslizamentos em diversas condições ambientais.

Além disso, o microcontrolador ESP32 será utilizado para realizar a comunicação sem fio com o MPU6050. O ESP32 é altamente integrado com conectividade Wi-Fi e Bluetooth, muito usado em aplicações IoT. Ele possui um processador dual-core de 32 bits, frequência de até 240 MHz e interfaces de comunicação, como, por exemplo: I2C, SPI e UART, o que facilita

a integração com o MPU6050 [7]. O ESP32 também suporta uma faixa de temperatura operacional de -40°C a 125°C, garantindo robustez e confiabilidade em diversas condições ambientais. Com essas características, o ESP32 se torna ideal para transmitir dados coletados pelo MPU6050, facilitando a comunicação e a transmissão dos dados.

# Capítulo 3

## Sprint 2 - Configuração do ESP32 e MPU6050

### 3.0.1 Objetivo

O objetivo principal deste sprint foi a configuração do ESP32 para coletar os dados do MPU6050 e enviá-los a um tópico MQTT. O processo envolveu as seguintes etapas: Criação do Código “.ino” que será carregado no software Arduino Uno. Este código gerenciará a comunicação do sensor MPU6050 que fornece dados de acelerômetro e giroscópio, para o ESP32 e estabelecerá a conexão com uma rede Wi-Fi. Essa etapa é muito importante para que o ESP32 possa se comunicar corretamente com o tópico MQTT e receber os dados enviados pelo sensor. Isso inclui a definição do servidor MQTT, porta e tópicos para a comunicação dos dados.

### 3.0.2 Configuração ESP32 e Broker MQTT

A configuração do ESP32 para coletar dados do sensor MPU6050 e enviá-los para um tópico MQTT foi realizada utilizando a IDE Arduino. O processo começa com a criação de uma instância do objeto mpu, fornecido pela biblioteca Adafruit MPU6050, que facilita a leitura dos dados do acelerômetro, giroscópio e temperatura. Em seguida, variáveis são definidas para armazenar os valores lidos de cada sensor.

No bloco `setup()`, a comunicação serial e a interface I2C são inicializadas, permitindo que o ESP32 se comunique adequadamente com o sensor MPU6050. O sensor é configurado para iniciar as leituras e, em intervalos definidos no código, os valores de aceleração, giroscópio e temperatura são capturados e armazenados nas variáveis apropriadas. Esse processo garante a coleta contínua de dados sensoriais para o envio.

Além disso, foi implementada a lógica necessária para conectar o ESP32 a uma rede Wi-Fi e a um broker MQTT. Esse passo é essencial para transmitir os dados do sensor em tempo real para um tópico MQTT específico. O código inclui a definição do servidor MQTT, da porta de comunicação e dos tópicos para onde os dados serão enviados, garantindo que o ESP32 funcione como um nó de monitoramento eficiente numa rede IoT. Deste modo, os objetivos propostos foram alcançados com sucesso, e os dados do MPU6050 estão sendo enviados para um tópico MQTT intitulado “Encostas Dados” através do Broker MQTT “test.mosquitto.org” na porta “1883”.

# Capítulo 4

## Sprint 3 - Instalação e Configuração do Docker e dos containers telegraf, influxdb e Grafana

Neste sprint, o foco principal é a instalação do Docker para suportar os containers do Telegraf, InfluxDB e Grafana. Além disso, o Telegraf será configurado para receber dados via MQTT e enviá-los ao banco de dados InfluxDB. Para comprovar a conclusão dos objetivos, foram produzidos vídeos e fotos ao longo do processo.

### 4.0.1 Configuração do Docker, Containers e Telegraf

Após a instalação do Docker, é criado um código “.yml” para codificação dos containers do InfluxDB, Telegraf e Grafana. Após isso, é necessário apenas provisionar os recursos e testar o acesso por meio do localhost e da porta definida no arquivo docker-compose. O vídeo disponível no Youtube [14] demonstra todo o processo, desde a criação dos containers até a confirmação de que todos os sistemas estão online e funcionando corretamente. Na imagem 4.1 é possível visualizar os containers online.

Para configurar o Telegraf e permitir que ele carregue os dados do protocolo MQTT, é necessária a criação de um código com a extensão “.conf”, que será responsável por coletar, processar e enviar os dados para o banco de dados do InfluxDB. Além do código, é necessário criar um token de API no InfluxDB e adicioná-lo ao código do Telegraf, assim, permitindo a comunicação entre o sistema e o agente. Também um vídeo [17] foi enviado para o YouTube a fim de permitir o fácil acesso à visualização do processo. Os

resultados do sprint V3 também foram como esperados, sem muitos problemas os containers foram instanciados e o InfluxDB recebeu e armazenou os dados do sensor, por meio do agente Telegraf.

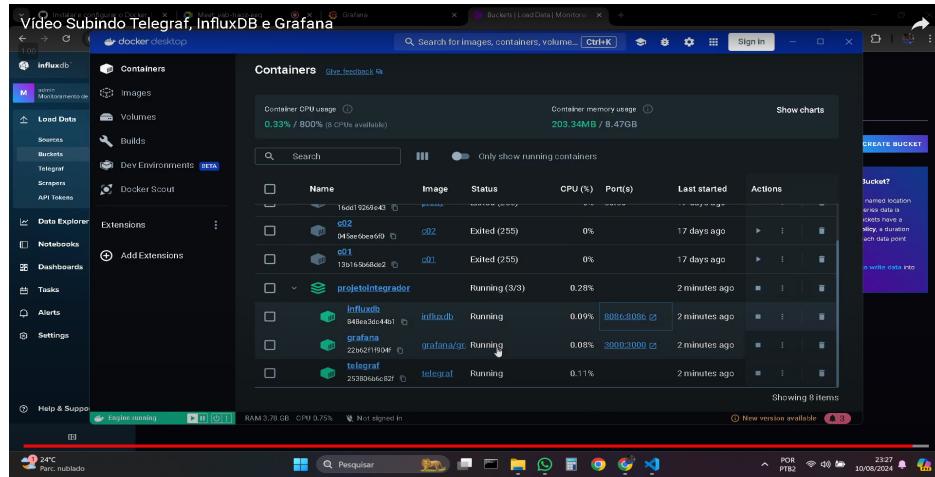


Figura 4.1: Containers Online - Fonte: Elaboração Própria

# Capítulo 5

## Sprint 4 - Configuração do Grafana

### 5.1 Objetivo

O objetivo para o sprint V4 é configurar o Grafana para receber dados do InfluxDB e criar alertas para serem disparados ao haver qualquer variação nos dados do sensor. A ideia central é que, ao detectar qualquer movimento, o Grafana envie uma notificação de alerta para o Telegram, em um grupo da equipe. Além disso, os materiais para a construção da maquete serão adquiridos. Também, será criado um container para manter o MQTT local e será feito um protótipo para simular detecção do movimento de terra em 3D usando Processing.

#### 5.1.1 Configuração do Grafana para obter dados do InfluxDB

Após garantir que o InfluxDB está recebendo os dados do sensor, o próximo passo é adicionar um novo data source no Grafana. Após selecionar o InfluxDB, é necessário selecionar o “Flux” como query language. Depois, adicionar o endereço e a porta do banco de dados, desmarcar o Basic auth, e por fim adicionar o nome da organização e usar um token de API para autenticação. Para visualizar as informações no Dashboard, basta clicar em adicionar novo, selecionar o data source, adicionar o query que o influxDB gera, selecionar o modo de visualização que deseja dos dados e salvar. Ademais, todo processo de criação foi gravado e encontra-se disponível no YouTube [8].

A imagem demonstra o dashboard criado no Grafana para exibir os dados recebidos em tempo real do sensor.5.1

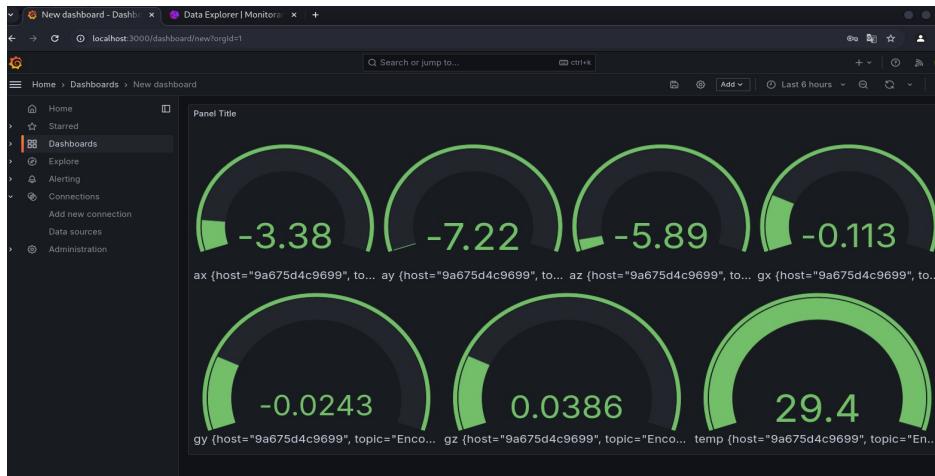


Figura 5.1: Dashboard Grafana - Fonte: Elaboração Própria

### 5.1.2 Configuração Para Receber os Alertas

Antes de criar as regras de alerta, é necessário definir o meio pelo qual as notificações serão recebidas. Optamos por receber as notificações pelo Telegram. Assim, o primeiro passo é a criação de um bot usando o “Bot Father”, o qual recebeu o nome de “Grafana-encostas”. Após isso, é preciso configurar um novo ponto de contato no Grafana e então adicionar um nome para o contato, inserir o token gerado pelo bot e o ID do contato, que pode ser obtido após adicionar o bot a um grupo.

Ademais, um dos desafios deste sprint foi a implementação de alerta no Grafana, que não ocorreu como esperado. Embora a ideia fosse criar alerta baseados em regras definidas, as opções disponíveis no Grafana mostraram-se limitadas. As regras testadas não geraram resultados satisfatórios, principalmente por não permitirem a criação de alertas com base no desvio padrão identificado, o que era um dos critérios essenciais para o monitoramento pretendido. Como solução, foi desenvolvido um código python que fará a notificação de alertas, o qual se mostrou promissor e será discutido mais adiante.

A imagem 5.2 exibe como a configuração do contato que receberá os alertas deve ser preenchido.

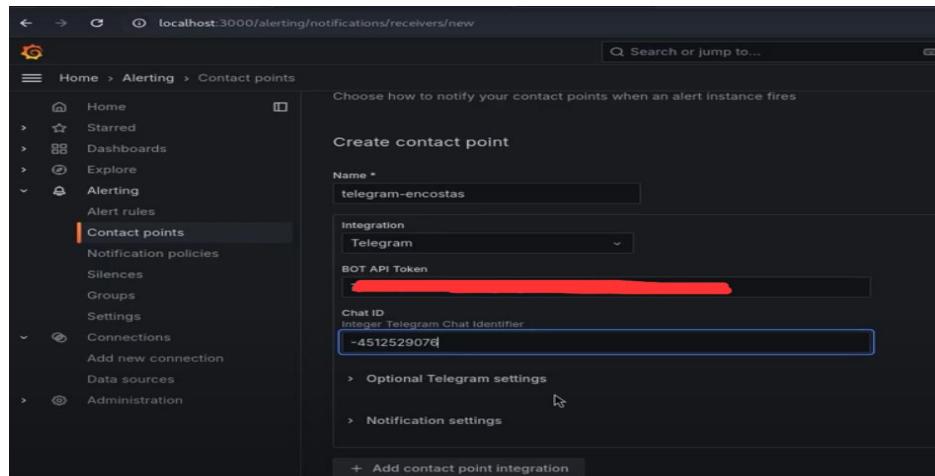


Figura 5.2: Contato Grafana - Fonte: Elaboração Própria

### 5.1.3 Materiais Para a Maquete

Para a maquete, foi adquirido um recipiente retangular nas dimensões de 27 cm por 55 cm por 39 cm, na cor preta. Para apoiar o sensor MPU6050, será utilizado um tubo de PVC com 15 cm de altura. Para suportar o sensor, será usada uma caixa de passagem de câmera.



Figura 5.3: Recipiente Retangular - Fonte: Elaboração Própria

Os demais materiais adquiridos podem ser visualizados na imagem 5.4

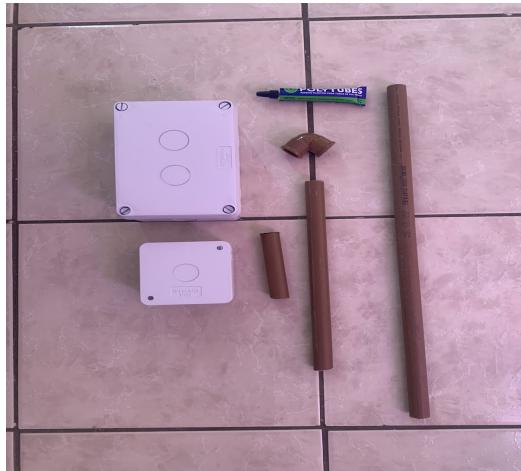


Figura 5.4: Materiais - Fonte: Elaboração Própria

#### 5.1.4 Simulação 3D Para Detectar Deslizamento de Encostas

Para obter uma perspectiva diferente sobre o deslizamento de terras, foi desenvolvida uma visualização 3D usando o software Processing. As delimitações para esta entrega incluíram: uma encosta em movimento, um sensor fixo que detecta os movimentos do giroscópio (inclinação) e do acelerômetro. Assim, foi criado um código com a extensão “.pde” usando a linguagem Java. Quando um sketch (código) é escrito no software, ele é compilado como um programa Java. Nesse sentido, os objetivos foram concluídos, o que é possível verificar na imagem 5.5.



Figura 5.5: Simulação 3D - Fonte: Elaboração Própria

### 5.1.5 MQTT na Rede Local

Anteriormente, o projeto utilizava o protocolo MQTT de forma online, através do “mosquitto.org”, entretanto, para evitar problemas de conectividade e manter todos os sistemas localmente, como já ocorre com InfluxDB, Grafana e Telegraf, foi acordado implementar o MQTT usando localmente o docker compose. Isso inclui a criação das configurações necessárias para garantir a comunicação adequada com os demais sistemas. Deste modo, foi necessário apenas adicionar o código para criação do container mosquitto e criar uma rede para ele se comunicar com as demais aplicações.

A imagem 5.6 é uma captura de tela, que demonstra o container mosquitto em execução.

```
root@kali: /home/ana/Desktop/integrador/Monitoramento-de-Deslizamento-de-Ecostas-usando-Microcontrolador-MPU6050-MQTT-e-Grafana./docker]# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
d6bfd86529e6      telegraf           "/entrypoint.sh tele..."   15 minutes ago    Up 10 minute
s     8092/udp, 8125/udp, 8094/tcp          telegraf
6aec7e11fe6f      grafana/grafana    "/run.sh"
s     0.0.0.0:3000→3000/tcp, :::3000→3000/tcp      grafana
a97c9cb7e0e4      eclipse-mosquitto  "/docker-entrypoint..."
s     0.0.0.0:1883→1883/tcp, :::1883→1883/tcp      mosquito
b8acd6157d78      influxdb           "/entrypoint.sh infl..."
s     0.0.0.0:8086→8086/tcp, :::8086→8086/tcp      influxdb
```

Figura 5.6: MQTT Local - Fonte: Elaboração Própria

Assim, os objetivos alcançados foram: A criação de um dashboard para exibir os dados do sensor recebidos pelo InfluxDB e configurado o tipo de contato para alerta. Também foi desenvolvida uma simulação 3D utilizando Processing para detectar o movimento de terra e configurado o MQTT local.

# **Capítulo 6**

## **Sprint 5 - Testes, Simulação e Análise de Resultados via Maquete**

### **6.1 Objetivo**

Na Sprint V5, o foco central estará nos testes, simulação e análise de resultados via Maquete. Nesta fase, será construída uma maquete detalhada para simular as condições reais de deslocamento de encostas, visando analisar os dados coletados e gerar insights valiosos. Esses resultados fornecerão uma base sólida para a tomada de decisões informadas, especialmente no que diz respeito à segurança e à prevenção de desastres naturais. A simulação na maquete permitirá observar o comportamento do terreno em situações parecidas de desabamento de encostas, facilitando a identificação de padrões de risco e a proposição de medidas preventivas eficazes.

#### **6.1.1 Construção da Maquete**

De modo a construir a maquete com um embasamento teórico, foi usado como base o estudo de Arthur Semione et al., intitulado “Desenvolvimento de Módulos para Monitoramento de Encostas com o Uso de Sensoriamento Wireless” [16], no qual os autores desenvolveram um sistema de identificação de movimento de encostas utilizando o MPU6050, com algumas diferenças em relação ao nosso trabalho. O sistema de monitoramento é composto por um sistema embarcado com funcionalidades específicas, incluindo o MPU6050 para aquisição de dados, um datalogger para armazenar as informações, e um filtro de Kalman, que combina os dados do acelerômetro e do giroscópio

para oferecer uma medição mais precisa do ângulo, reduzindo a influência de ruídos. Além disso, o sistema utiliza um cartão micro SD para memória e um Real Time Clock para registrar a data e hora das medições. A comunicação sem fio é realizada por meio de um módulo XBee, que utiliza o protocolo Zigbee para a transmissão dos dados. Para os testes, os autores usam três módulos, conforme imagem abaixo.



Figura 6.1: Maquete Semione - Fonte: Adaptado de Semione A. et al. 2020

Inicialmente, foi construído dois protótipos para posicionar o sensor, conforme ilustrado nas imagens 6.2 e 6.4. Entretanto, salienta-se que ambos os protótipos 1 e 2 não foram usados no teste final com água, mas apenas o protótipo 3. Ademais, a imagem mostra a caixa aberta, mas ela foi fechada durante os testes para evitar contato dos eletrônicos com água.

O primeiro protótipo apresentando para armazenar o MPU6050, foi apenas para começar a testar o sensor sem o uso de água na maquete.



Figura 6.2: Protótipo 1 - Fonte: Elaboração Própria

O segundo também não foi testado com água, e foi notado que ele não traria bons resultados devido ao suporte usado, um conduto de 12 cm, que se mostrou fragil e não suportava bem o peso da caixa de passagem.



Figura 6.3: Protótipo 2 - Fonte: Elaboração Própria

Por fim, o melhor resultado foi no protótipo 3, no qual armazenou bem além do MPU6050, o ESP32 e a bateria de 9V.



Figura 6.4: Protótipo 3 - Fonte: Elaboração Própria

### 6.1.2 Desigualdade de Chebyshev

Para identificar e criar uma métrica válida que gere alertas preciso quando uma encosta ou talude está para desmoronar, os testes com a maquete serão baseados na desigualdade de Chebyshev, no qual diz o seguinte:

É um princípio estatístico utilizado para estimar a probabilidade de que o valor de uma variável aleatória se encontre a uma determinada distância de sua média [11].

O Teorema de Chebyshev afirma que, para qualquer distribuição de probabilidade com média  $\mu$  e variância  $\sigma^2$ , a probabilidade de que o valor de uma variável aleatória  $X$  esteja a uma distância maior do que  $k$  desvios padrão da média é, no máximo,  $\frac{1}{k^2}$ . A formula é:

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}, \quad \text{para } k > 0.$$

No qual:

$P(\cdot)$  é a probabilidade;

$|X - \mu|$  é a distância entre  $X$  e a média  $\mu$ ;

$\sigma$  é o desvio padrão;

$k$  é um valor positivo que indica o número de desvios padrão.

### 6.1.3 Aplicando o Teorema no Monitoramento de Encostas

Foram coletados dados durante 6 horas, com o sensor posicionado em um ambiente sem qualquer interferência, seja humana ou de vento, para analisar o desvio padrão dos dados em condições normais. O cálculo da média e do desvio padrão será realizado para os eixos ax, ay, az, gx, gy, gz, e temp. Esses cálculos permitirão estabelecer um intervalo de confiança para a variação natural dos dados em repouso. Após essa análise, será realizado um novo teste, aplicando “chuva” artificial para simular um deslizamento de encosta. Os dados obtidos nesse cenário serão comparados com os resultados em condições normais, permitindo identificar uma métrica segura para a geração de alertas em casos de instabilidade.

### 6.1.4 Python e Chebyshev

Os dados são extraídos do banco de dados InfluxDB e são pouco intuitivos devido ao grande volume de informações. Para resolver esse problema, foi desenvolvido um código em Python que utiliza a biblioteca Pandas para filtrar e processar os dados de maneira eficiente. O código está disponível no repositório [13].

A imagem 7.1 mostra os dados filtrados, após o uso do código python. Após essa filtragem, será mais fácil analisar os dados obtidos. A fim de

A	B	C	D	E
start	stop	time	value	field
dateTime:RFC3339	dateTime:RFC3339	dateTime:RFC3339	double	string
start	stop	time	value	field
2024-08-27T21:36:55.03304434	2024-08-28T09:36:55.03304434	2024-08-28T03:02:00Z	0.033703703703703715	ax
2024-08-27T21:36:55.03304434	2024-08-28T09:36:55.03304434	2024-08-28T03:04:00Z	0.03112500000000003	ax
2024-08-27T21:36:55.03304434	2024-08-28T09:36:55.03304434	2024-08-28T03:06:00Z	0.03662499999999999	ax
2024-08-27T21:36:55.03304434	2024-08-28T09:36:55.03304434	2024-08-28T03:08:00Z	0.03887500000000001	ax
2024-08-27T21:36:55.03304434	2024-08-28T09:36:55.03304434	2024-08-28T03:10:00Z	0.02425000000000008	ax
2024-08-27T21:36:55.03304434	2024-08-28T09:36:55.03304434	2024-08-28T03:12:00Z	0.034000000000000016	ax
2024-08-27T21:36:55.03304434	2024-08-28T09:36:55.03304434	2024-08-28T03:14:00Z	0.025500000000000016	ax
2024-08-27T21:36:55.03304434	2024-08-28T09:36:55.03304434	2024-08-28T03:16:00Z	0.01987500000000001	ax

Figura 6.5: Dados Filtrados - Fonte: Elaboração Própria

ter um melhor resultado na coleta das informações ao usar o teorema de Chebyshev, outro código python foi desenvolvido para auxiliar nessa tarefa.

O código extrai os valores dos eixos, faz alguns tratamentos de renomeação nas colunas para uma melhor visualização, calcula a média e desvio padrão, calcula o intervalo e coleta os valores que estão fora do intervalo, por fim, salva o resultado em um arquivo CSV. O código também encontra-se disponível no repositório juntamente com os arquivos CSV. Mas para ilustrar o que este estudo propõe, a imagem 5.1 demonstra as informações obtidas após a aplicação do código Python com uso da regra de Chebyshev nos dados do sensor após 6h em execução.

A	B	C	D
eixo	média	desvio padrão	proporção mínima dentro de 2
ax	0.0135618141109914	0.007394905171477051	0.75
ay	9.868135800349087	0.003608091223540239	0.75
az	0.2300333908339901	0.018238015690990435	0.75
gx	-0.1171495375035090	0.001145477590096498	0.75
gy	-0.0206429030270608	0.000275577346093929	0.75
gz	0.0358058993545404	0.000504268934637947	0.75
temp	29.653051353630058	0.4097263002023016	0.75

Figura 6.6: Desvio Padrão em Dados de seis horas - Fonte: Elaboração Própria

O objetivo deste experimento é identificar o desvio padrão para cada eixo, exceto a temperatura, a fim de determinar o intervalo correto para a geração de alertas. Por exemplo, se o desvio padrão do eixo “AX” é de 0,0073 (arredondado), o alerta será configurado para ser acionado sempre que houver uma variação fora desse intervalo.

Os resultados para a sprint V5 não foram como esperados, problemas na construção da maquete e com a ligação da filha 9V com o protoboard trouxeram atrasos significativos. Assim, os testes com a maquete e o uso do teorema de Chebyshev foram executados apenas na sprint 6.

# Capítulo 7

## Sprint 6 - Relatório Final e Testes

### 7.1 Objetivos

Como mencionado, os testes com a maquete foram iniciados na quinta semana, Sprint V5, e finalizados na sexta semana, Sprint V6. O objetivo desta seção é descrever com detalhes os testes realizados, além de apresentar dois cenários que simulam deslizamento de massa em 3D usando processing. Ademais, esta seção fornece os resultados e comparações em relação à aplicação do teorema de Chebyshev e o filtro de kalman.

#### 7.1.1 Resultados com o teorema de Chebyshev

Dada a alta sensibilidade do sensor MPU6050, optou-se por realizar a coleta de dados durante um período contínuo de 48 horas, visando garantir maior precisão no cálculo do desvio padrão. Além disso, para assegurar a integridade dos dados, o sensor foi mantido em repouso e estático, sem interferência humana ou de fatores externos, como vento. Também foi garantido que o sensor permanecesse ligado ininterruptamente ao longo de todo o experimento. O desvio padrão calculado, usando um código python, pode ser visualizado na imagem 7.1.

Nesse contexto, o desvio padrão de cada eixo foi incorporado ao código Python responsável pelo envio de alertas. Conforme já mencionado, o código coleta dados em tempo real do bucket no InfluxDB a cada trinta segundos, calcula o intervalo baseado na desigualdade de Chebyshev e compara o desvio padrão atual com o valor obtido na análise dos dados das 48 horas anteriores. Se houver discrepância, um alerta é emitido.

A	B	C	D
eixo	média	desvio padrão	proporção mínima dentro de 2 d
ax	-0.2535516532194273	0.012517734385558758	0.75
ay	9.706129172599272	0.013716388835661371	0.75
az	1.4799545166903971	0.018326056408730772	0.75
gx	-0.11802839740547377	0.001633486516904124	0.75
gy	-0.02056082898275587	0.000896470303540645	0.75
gz	0.03571033064388542	0.002196115796697022	0.75
temp	30.1109832305015	0.11313377450955028	0.75

Figura 7.1: Desvio Padrão Calculado Após 48h - Fonte: Elaboração Própria

### 7.1.2 Filtro de Kalman

Os sistemas modernos utilizam sensores para estimar variáveis ocultas, como localização e velocidade, baseadas em medições. No caso do GPS, essas estimativas são influenciadas por incertezas de fatores externos, como ruído térmico e efeitos atmosféricos. O Filtro de Kalman é um algoritmo amplamente utilizado para produzir estimativas precisas dessas variáveis, mesmo com medições imprecisas. Ele também prevê o estado futuro do sistema com base em dados anteriores. O filtro, desenvolvido por Rudolf E. Kalman em 1960, é aplicado em diversas áreas, incluindo rastreamento por radar, sistemas de navegação e controle [12].

O estudo relacionado a este trabalho, conduzido por Arthur Semione et al. e intitulado “Desenvolvimento de Módulos para Monitoramento de Encostas com o Uso de Sensor Wireless” [16], utilizou o filtro de Kalman para reduzir os ruídos do sensor MPU6050. Diante disso, os autores deste trabalho ficaram curiosos sobre a eficácia desse filtro e decidiram realizar um teste comparando os dados coletados pelo sensor, com e sem o filtro, para avaliar se seria vantajoso incorporá-lo neste estudo.

Assim, foi realizada uma comparação entre os dados brutos e os dados processados pelo filtro de Kalman. Ambos os conjuntos de dados foram coletados utilizando as seguintes métricas: sensor em repouso absoluto, sem interferências de vento ou movimento, durante um período de duas horas.

Os primeiros dados coletados do sensor foram obtidos sem o uso do filtro de Kalman. O gráfico 7.2 foi gerado após a execução de um código Python, que plotou os dados coletados pelo sensor durante um período de duas horas, em 27 de agosto de 2024. Esses dados, originalmente utilizados para testar a regra de Chebyshev, foram reaproveitados neste experimento. É importante destacar que os dados estão íntegros, e não houve interferência humana ou de fatores externos.

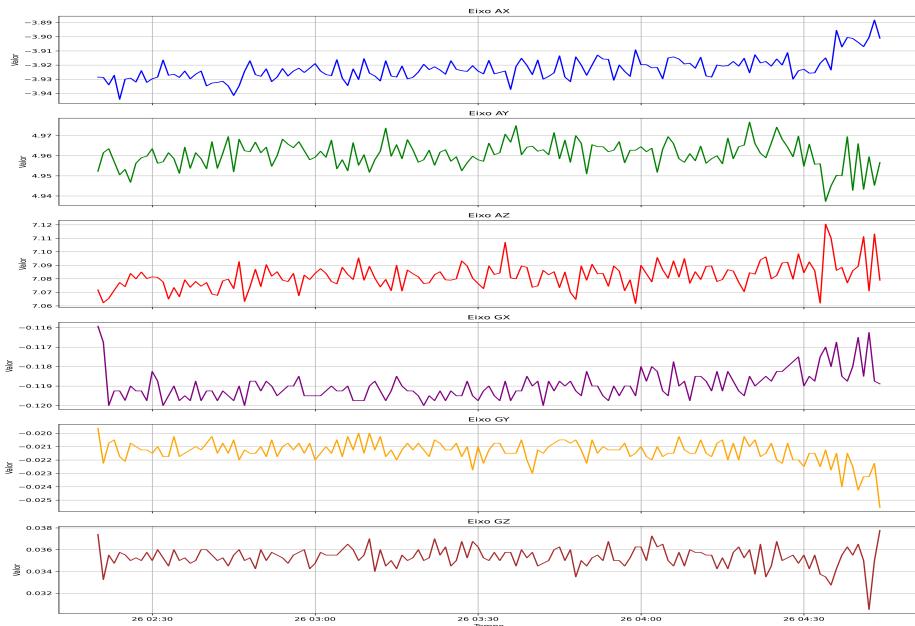


Figura 7.2: Dados Sem Filtro de Kalman - Fonte: Elaboração Própria

Por fim, o gráfico abaixo 7.3 apresenta os dados coletados após a implementação do filtro de Kalman, inserido no código “.ino” que faz a comunicação com o ESP32. Esses dados foram obtidos nas mesmas condições e localização dos dados sem filtro, com a mesma duração de duas horas. A única diferença foi a data de coleta, que ocorreu em 04 de setembro de 2024.

Ao comparar os dois gráficos, percebe-se claramente uma discrepância significativa na variação de cada eixo do sensor. Nos dados sem o filtro, há grandes oscilações, enquanto, após a aplicação do filtro, os valores se mantiveram quase estáveis durante a maioria do tempo, resultando em linhas mais retas e com poucas variações.



Figura 7.3: Dados Com Filtro de Kalman - Fonte: Elaboração Própria

Conclui-se, portanto, que a aplicação do filtro de Kalman pode aumentar a confiabilidade no monitoramento e na geração de alertas. Dessa forma, é importante destacar alguns pontos relevantes. Primeiro, tanto os dados com e sem filtro serão exibidos no dashboard do Grafana. Segundo, os alertas serão gerados apenas com base nos dados filtrados, uma vez que o desvio padrão foi calculado a partir desses dados. Por fim, a utilização do filtro trouxe maior confiança aos dados monitorados, garantindo maior precisão nos alertas.

## 7.2 Teste com a Maquete

Foi realizado seguinte teste com a maquete. A caixa de passagem de câmera foi posicionada na extremidade do recipiente retangular, apoiado por um cano de PVC. O ESP32 estava conectado na bateria e assim foi feita uma comunicação sem fio. O vídeo “Teste 01” [9] demonstra a caixa de câmera

sedendo ao receber água e assim gerando alertas recebidos no Telegram. A imagem 7.4 mostra a maquete pronta para os testes.



Figura 7.4: Maquete Para Testes - Fonte: Elaboração Própria

Ademais, foi realizado outros dois testes, um no mesmo cenário mencionado anteriormente, que encontra-se disponível em [10]. E outro cenário com o uso de código python que será detalhado mais a frente.

### 7.2.1 Alertas Gerados Durantes os Testes

Para gerar alerta através do Grafana, inicialmente pretendia-se utilizar o desvio padrão como métrica. No entanto, devido às limitações da plataforma, não foi possível configurar essa funcionalidade diretamente. Assim, optou-se por usar um valor estático para o eixo, configurando um alerta para qualquer variação desse valor. Por exemplo, se o valor estático do eixo AX for 3.00, qualquer variação resultará na geração de um alerta.

A imagem 7.5 mostra como esse alerta é recebido no Telegram.

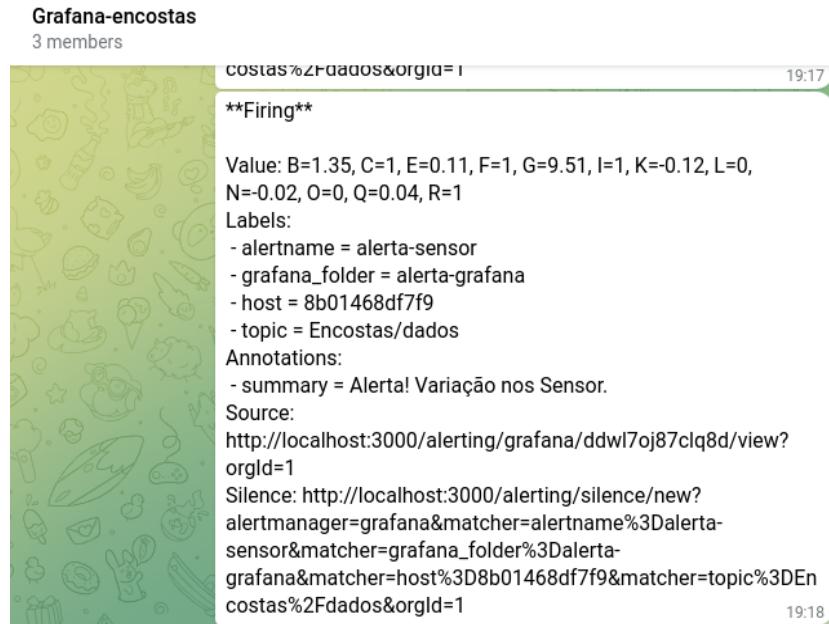


Figura 7.5: Alertas Recebidos do Grafana - Fonte: Elaboração Própria

Para aprimorar o sistema de alerta, foi desenvolvido um código Python que busca dados em tempo real no bucket do InfluxDB, calcula o desvio padrão e compara o valor atual com o desvio padrão filtrado. Caso seja identificado um desvio relevante, um alerta é gerado. Com essa abordagem, observou-se uma comunicação mais eficiente e a possibilidade de utilizar o desvio padrão calculado anteriormente. Este desvio padrão fixo foi calculado com base em dados de 48 horas, utilizando a regra de Chebyshev e um filtro de Kalman.

A imagem 7.7 demonstra o desvio padrão para os eixos, após uso do filtro de Kalman em dados de 48h. Se comparado com a imagem 6.6 no qual foi obtida antes de inserir o filtro, nota-se uma mudança alta entre os desvios identificados.

A	B	C	
eixo	média	desvio padrão	proporção
ax	0.1700000000000387	2.3840437043484105e-19	0.75
ay	9.870000000001742	1.615464016800653e-19	0.75
az	-1.0000000000001643	9.912347992235866e-19	0.75
gx	-0.12000000000002768	1.406326764427561e-19	0.75
gy	-0.02000000000000429	3.64039894836553e-16	0.75
gz	0.04000000000000858	7.28079789673106e-16	0.75
temp	30.33499964699319	0.012158451085089013	0.75

Figura 7.6: Desvio Padrão Pós Filtro de Kalman - Fonte: Elaboração Própria

Abaixo a imagem demonstra como é recebido o alerta no Telegram, usando o código desenvolvido.

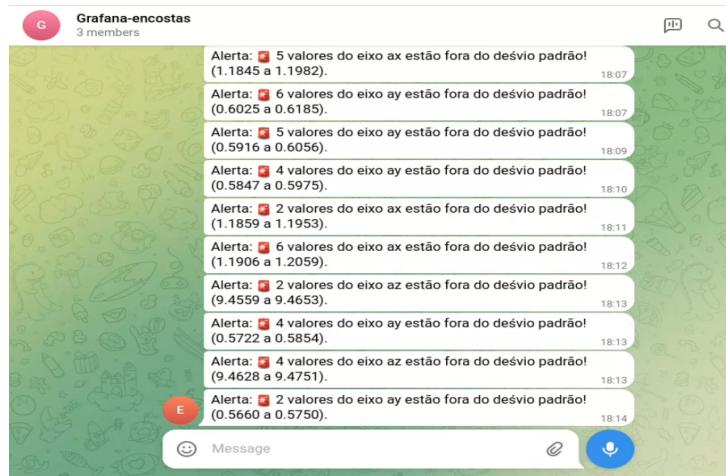


Figura 7.7: Alerta Usando Python - Fonte: Elaboração Própria

## 7.3 Cenários de Monitoramento Usando Processing

De modo a obter outras visualizações da proposta do projeto, foi desenvolvido um modelo 3D de detecção de movimento, no qual se usa o software processing. Essa abordagem foi melhor desenvolvida na sprint 4. Para a sprint 6, adicionou-se o cenário de chuva, e o protocolo MQTT para envio dos dados para um tópico.

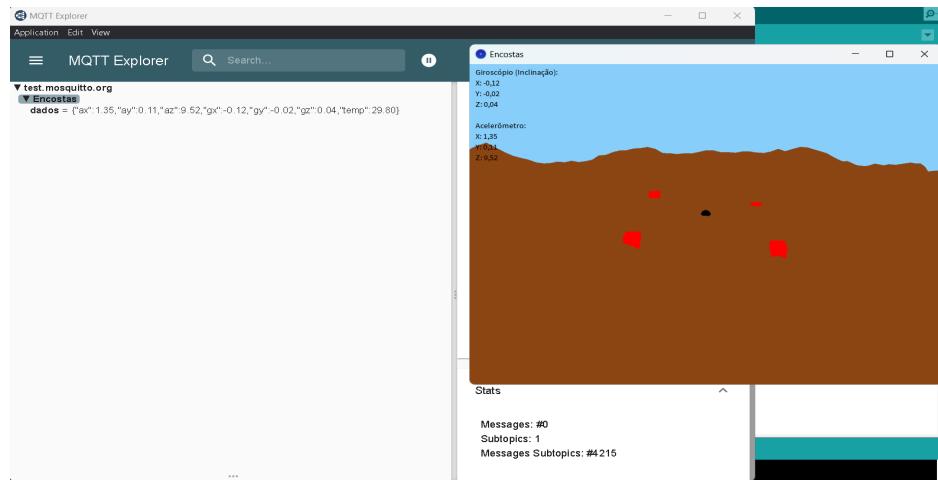


Figura 7.8: Processing Enviando dados para o Tópico MQTT - Fonte: Elaboração Própria

Além disso, o cenário com chuva pode ser visualizado na imagem 9.1. No entanto, há um vídeo disponível no repositório do GitHub, no Milestones, sprint 6, que apresenta o cenário proposto com mais detalhes.

Por fim, a proposta inicial para a modelagem 3D era simular o processo de monitoramento de deslizamento de massa, replicando todos os sistemas, regras e alertas definidos no projeto. No entanto, devido a limitações de tempo e ao conhecimento restrito no uso do software e na linguagem Java, a simulação apresentou algumas limitações.

A sprint V6 foi muito produtiva, com a realização dos testes na maquete, que permitiram identificar pontos de melhoria, os quais já estão sendo abordados para a apresentação do Pitch Deck. Houve também um aprimoramento no tempo de envio de alertas via Grafana, embora ele não será utilizado para esse fim, pois o código em Python demonstrou maior eficiência. O Grafana será mantido como ferramenta de visualização em dashboards. O filtro de Kalman foi implementado com sucesso, e o uso do teorema de Chebyshev provou ser eficaz na identificação de anomalias nos dados, gerando alertas

em caso de movimentação de terra. Por fim, o relatório final, que incluirá todas as informações e detalhes do projeto, está em desenvolvimento e será apresentado na próxima semana.

Abaixo, imagem representando o cenário de chuva, no software Processing.

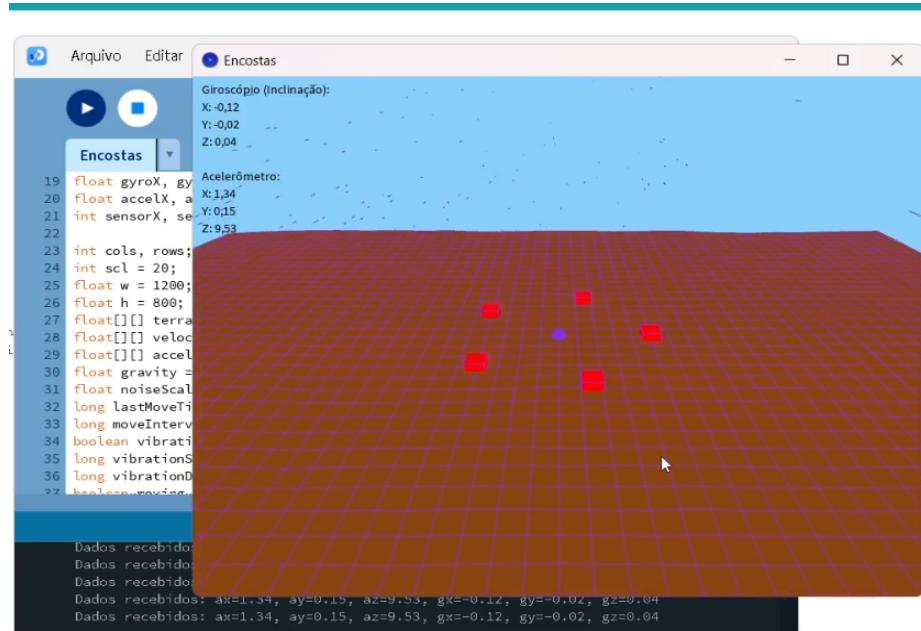


Figura 7.9: Cenário com Chuva - Fonte: Elaboração Própria

# Capítulo 8

## Documentação

Toda a documentação do projeto está disponível no repositório da equipe no GitHub [13], onde é possível acessar todos os códigos, dados do sensor utilizados nos testes e informações adicionais. O repositório também conta com uma wiki detalhada e milestones, que fornecem mais detalhes sobre o progresso e desenvolvimento das diferentes etapas.

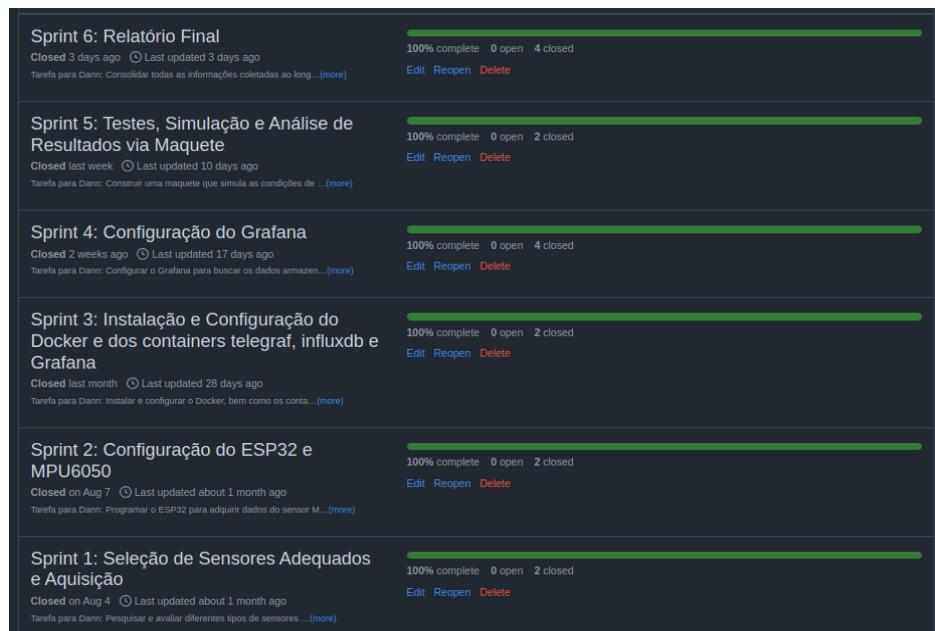


Figura 8.1: Milestones - Fonte: Elaboração Própria

# Capítulo 9

## Commits

Toda a equipe contribuiu de forma igualitária ao longo do projeto. A cada nova implementação ou alteração, as informações foram constantemente atualizadas no GitHub, garantindo a centralização e organização de todos os dados e registros.

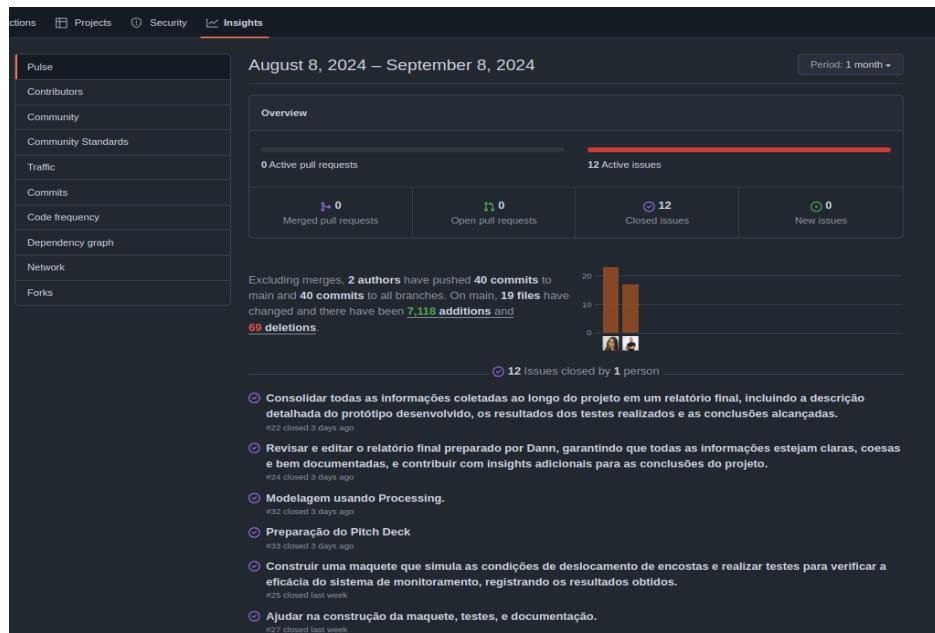


Figura 9.1: Commits - Fonte: Elaboração Própria

# Capítulo 10

## Resultados

Durante seis semanas foi buscado ser desenvolvido um sistema de monitoramento de encostas, no qual às primeiras 4 semanas incluíram a configuração, pesquisa e implementação dos sistemas propostos, e então nas últimas duas semanas foram realizados testes práticos utilizando uma maquete que simulava um deslizamento de terra em condições controladas.

Durante o experimento, o sensor MPU6050, integrado ao microcontrolador ESP32, demonstrou alta variação e sensibilidade na detecção de movimentos, desse modo, foi implementado o filtro de Kalman, que melhorou significativamente a precisão dos dados coletados. Após isso, foi possível ter alertas mais precisos com base no desvio padrão calculado.

Os resultados dos testes indicam que o sistema foi eficaz na coleta de dados relevantes sobre a movimentação da terra e na identificação de padrões irregulares, seguindo a regra de Chebyshev para o cálculo do desvio padrão.

Ademais, inicialmente o objetivo era receber alertas no Telegram gerados pelo Grafana, contudo, devido à falta de flexibilidade do sistema para envio dos alertas usando o desvio padrão calculado, optou-se por um script em python que notifica no Telegram desvios fora do padrão.

Em relação à simulação com chuva, o sistema respondeu adequadamente ao acréscimo de água próximo ao sensor. Foram observadas variações nos eixos do sensor, que refletiram no aumento dos valores do desvio padrão, desencadeando os alertas automáticos. As visualizações geradas no Grafana também permitiram uma análise dos dados no dashboards, apresentando gráficos em tempo real que ilustravam as variações nos eixos do sensor antes, durante e após a simulação de deslizamento e com os dados filtrados e brutos do sensor.

Apesar dos resultados positivos, algumas limitações foram identificadas. O container MQTT local antes funcionando, parou de enviar os dados para o tópico MQTT e a comunicação não foi reestabelecida, assim foi preciso voltar

a usar o broker do Mosquitto online. Além disso, não foi possível implementar o sistema de monitoramento na simulação 3D completo, foi entregue apenas o cenário de chuva e sem chuva e o envio dos dados coletados do sensor ao tópico MQTT.

No entanto, tais problemas não comprometeram a eficácia do sistema, que demonstrou ser uma solução viável para o monitoramento de deslizamentos de terra.

Em resumo, o sistema conseguiu detectar, monitorar e alertar movimentos anômalos com eficiência, atingindo os principais objetivos da proposta do projeto. A integração entre o sensor, o microcontrolador ESP32 e os componentes de software como o Telegraf, InfluxDB e Grafana se mostrou robusta e eficaz, garantindo uma boa comunicação e visualização dos dados. A metodologia proposta, aliada ao uso do filtro de Kalman e da regra de Chebyshev, trouxe resultados satisfatórios, demonstrando que o projeto tem potencial para ser expandido e utilizado em cenários reais de monitoramento de encostas.

# Capítulo 11

## Conclusão

Este projeto alcançou com sucesso o objetivo principal de desenvolver um sistema de monitoramento capaz de detectar movimentos de massa, como deslizamentos de terra, antecipadamente. A integração do sensor MPU6050 com o microcontrolador ESP32 e o uso do protocolo MQTT para a transmissão dos dados se mostrou eficiente, permitindo a coleta e análise em tempo real das informações de deslocamento. Além disso, o filtro de Kalman desempenhou um papel crucial ao reduzir ruídos nos dados, o que contribuiu para uma maior precisão na detecção de eventos anômalos.

Os testes realizados com a maquete indicaram que o sistema pôde identificar variações significativas nos dados do sensor, e os alertas foram gerados com base no desvio padrão calculado, garantindo uma resposta adequada em situações simuladas de deslizamento. Ademais, a integração com o InfluxDB e Grafana foi eficiente na visualização clara dos dados, enquanto o envio de notificações por meio de um script python ao Telegram garantiu uma comunicação eficiente dos alertas em tempo real.

No entanto, algumas limitações foram identificadas, no qual o container local do protocolo MQTT, que estava funcionando, parou de enviar os dados para o tópico e a comunicação não foi reestabelecida, mesmo após várias tentativas. Assim, foi preciso voltar a usar o broker do Mosquitto online. Outro ponto é que não foi possível concluir o sistema de monitoramento com a simulação 3D do Processing, sendo assim, foi entregue apenas o cenário com e sem chuva e o envio dos dados coletados do sensor ao tópico MQTT.

Para melhorias futuras, sugeriu-se a implementação de mais sensores, como de umidade. Além de melhorias nos tipos de alertas enviados, por exemplo, adequando os alertas a eventos mais precisos, como tempestades ou chuvas fracas.

Portanto, o projeto mostrou-se viável e eficaz, oferecendo uma solução de baixo custo e escalável para o monitoramento de deslizamentos de terra.

Com as melhorias propostas e a realização de novos testes em condições reais, ele tem o potencial de contribuir significativamente para a prevenção de desastres naturais e a preservação de vidas.

# Bibliografia

- [1] CEMADEN. *Deslizamentos*. Accessed: 2024-07-31. 2024. URL: <http://www2.cemaden.gov.br/deslizamentos/>.
- [2] Omega Engineering. *Omega Engineering*. Accessed: 2024-07-29. n.d. URL: [https://www.omega.co.uk/techref/pdf/STRAIN\\_GAGE\\_TECHNICAL\\_DATA.pdf](https://www.omega.co.uk/techref/pdf/STRAIN_GAGE_TECHNICAL_DATA.pdf).
- [3] u-blox. *NEO-6 GPS Modules Data Sheet*. Accessed: 2024-07-29. n.d. URL: [https://content.u-blox.com/sites/default/files/products/documents/NEO-6\\_DataSheet\\_%28GPS.G6-HW-09005%29.pdf](https://content.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf).
- [4] InvenSense. *MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.2*. Accessed: 2024-07-29. 2013. URL: <https://pdf1.alldatasheet.com/datasheet-pdf/view/1132807/TDK/MPU-6050.html>.
- [5] Campbell Scientific. *The 9.4-cm Sensor that delivers 1 L. Volume Of Influence*. Accessed: 2024-07-30. 2019. URL: <https://s.campbellsci.com/documents/ca/product-brochures/teros12-br.pdf>.
- [6] SICK AG. *SENSORES DE FIBRA ÓPTICA*. Accessed: 2024-07-30. 2009. URL: [https://cdn.sick.com/media/pdf/7/87/287/dataSheet\\_WLL170-2P132\\_6029511\\_pt.pdf](https://cdn.sick.com/media/pdf/7/87/287/dataSheet_WLL170-2P132_6029511_pt.pdf).
- [7] ElectronicsHub. *Getting Started with ESP32 / Introduction to ESP32*. Accessed: 2024-07-31. 2024. URL: <https://www.electronicshub.org/getting-started-with-esp32/>.
- [8] Victorino D. *Criação do Dashboard do Grafana*. Accessed: 2024-09-01. 2024. URL: <https://www.youtube.com/watch?v=owK0cidSZgE>.
- [9] Torres A. *Teste com a Maquete 01 - Alertas via Grafana*. Accessed: 2024-09-02. 2024. URL: <https://www.youtube.com/watch?v=IN810A2WRaI>.
- [10] Torres A. *Teste com a Maquete 02 - Alertas via Grafana*. Accessed: 2024-09-02. 2024. URL: <https://www.youtube.com/watch?v=t1mhQ8is4VE>.

- [11] Dr. benjamim anderson. *Teorema de chebyshev*. Accessed: 2024-08-26. 2023. URL: [https://statorial.org/pt/teorema-de-chebyshev/#google\\_vignette](https://statorial.org/pt/teorema-de-chebyshev/#google_vignette).
- [12] Alex Becker. *Sobre o Filtro de Kalman*. Accessed: 2024-09-02. URL: [https://kalmanfilter.net/pt/default\\_pt.aspx](https://kalmanfilter.net/pt/default_pt.aspx).
- [13] Victorino D. *Monitoramento de Deslizamento de Encostas*. Accessed: 2024-07-31. 2024. URL: <https://github.com/Dannihercleston/Monitoramento-de-Deslizamento-de-Encostas-usando-Microcontrolador-MPU6050-MQTT-e-Grafana./issues/15>.
- [14] Victorino D. *Vídeo Subindo Telegraf, InfluxDB e Grafana*. Accessed: 2024-09-02. 2024. URL: <https://www.youtube.com/watch?v=0IfCnkvEwgs>.
- [15] Sriraam Natarajan et al. “Motion Sensing Mechanism Using the Lily-pad Arduino and IMU Sensors: A Pilot Study”. Em: *Trends in Health Informatics* 1.1 (2024), pp. 23–30.
- [16] Arthur Semione et al. “Desenvolvimento de Módulos para Monitoramento de Encostas com o Uso de Sensoriamento Wireless”. Em: () .
- [17] Livia T. *Projeto Integrador - Sprint V3*. Accessed: 2024-09-01. 2024. URL: <https://www.youtube.com/watch?v=S7QvTfpfP9s>.