UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
BACHARELADO EM ENGENHARIA ELETRÔNICA

LUCAS PIOVEZAN BORTOLLI

SENSORIAMENTO REMOTO PARA MONITORAR RISCO DE RUPTURA DE BARRAGENS

CAMPO MOURÃO
2025

LUCAS PIOVEZAN BORTOLLI

SENSORIAMENTO REMOTO PARA MONITORAR RISCO DE RUPTURA DE BARRAGENS

Remote sensing for monitoring dam failure risk

Trabalho de Conclusão de Curso de Graduação apresentada como requisito para obtenção do título de Bacharel em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientador(a): Prof. Dr. Eduardo Giometti Bertogna.

CAMPO MOURÃO

2025

LUCAS PIOVEZAN BORTOLLI

SENSORIAMENTO REMOTO PARA MONITORAR RISCO DE RUPTURA DE BARRAGENS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 13 de fevereiro de 2025

Eduardo Giometti Bertogna

Doutorado

Universidade Tecnológica Federal do Paraná

Lucas Ricken Garcia

Doutorado

Universidade Tecnológica Federal do Paraná

Leonardo Faria Costa

Doutorado

Universidade Tecnológica Federal do Paraná

CAMPO MOURÃO

2025

AGRADECIMENTOS

Primeiramente, agradeço a Deus por me conceder força e sabedoria ao longo desta jornada. Estendo meus agradecimentos a meus familiares e amigos, cujo apoio incondicional e incentivo foram fundamentais durante toda a minha trajetória acadêmica. Embora não seja possível citar todos, destaco a importância de minha mãe, Margarete Piovezan, que sempre foi meu pilar e acreditou em mim desde o início. Agradeço também à minha esposa, Laís Regina dos Santos, e minha filha, Stella Piovezan Bortolli, que me acompanharam incansavelmente, oferecendo força e motivação nos momentos mais difíceis.

Aos amigos que fiz ao longo desta trajetória, agradeço pela parceria e pelas vivências compartilhadas. Ao meu orientador, Prof. Dr. Eduardo Giometti Bertogna, sou imensamente grato pelas valiosas orientações, dedicação e paciência. Sua expertise e apoio contínuo foram essenciais para o desenvolvimento deste trabalho.

Além disso, gostaria de expressar minha gratidão a todos os professores e membros do corpo docente da Universidade Tecnológica Federal do Paraná (UTFPR), cujos ensinamentos e experiências foram decisivos para o meu crescimento acadêmico e profissional. Por fim, não poderia deixar de agradecer à UTFPR e aos profissionais que forneceram recursos, dados e suporte técnico ao longo de todo o período do curso.

RESUMO

A segurança de barragens se torna uma preocupação crescente no Brasil, especialmente devido aos recentes acidentes que resultaram em graves impactos ambientais e humanos. Neste contexto, surge a necessidade de desenvolver e aplicar novas tecnologias voltadas ao monitoramento dessas estruturas. Este trabalho teve como objetivo projetar e implementar um sistema baseado em uma Unidade de Medida Inercial (IMU), aplicada a um protótipo que simula o rompimento de uma barragem. A tecnologia IMU, reconhecida por sua precisão e eficiência, integra sensores de aceleração, giroscópio e magnetômetro, sendo amplamente empregada em diversas áreas, como acessórios de entretenimento, equipamentos aeroespaciais, sistemas de defesa civil e militar, além de monitoramento de estruturas como barragens, pistas de rolagem, pontes e edificações. O sistema desenvolvido utiliza transceptores de radiofrequência (RF) para transmitir dados de vibração coletados pelo sensor a uma unidade central. Esses dados são, então, armazenados e exibidos em tempo real em um aplicativo móvel, que apresenta uma interface gráfica intuitiva. Através do aplicativo, os limites de segurança, definidos previamente pelo engenheiro responsável com base nas características especificas da barragem, são monitorados, e alertas são acionados quando esses limites são excedidos. Dada a crescente demanda por tecnologias inovadoras na instrumentação geotécnica, o presente trabalho contribui para o avanço de sistemas eletrônicos aplicados à engenharia civis promovendo maior segurança, eficiência na coleta de dados e mitigação de riscos de acidentes.

Palavras-chave: unidade de medida inercial; monitoramento remoto; instrumentação geotécnica;

ABSTRACT

Dam safety is na increasing concern in Brazil, especially due to recent accidents that have caused severe environmental and human impacts. In this context, the need arises to develop and apply new technologies focused on monitoring these structures. This study aimed to design and implement a system based on an Inertial Measurement Unit (IMU), applied to a prototype simulating a dam failure. IMU technology, recognized for its precision and efficiency, integrates acceleration, gyroscope and magnetometer sensors and is widely used in various fields, such as entertainment accessories, aerospace equipment, civil and military defence systems, as well as monitoring structures like dams, rumways, bridges and buildings. The developed system utulizas radiofrequency (RF) transceivers to transmit vibration data collected by the sensor to a central unit. These data are, then, stored and displayed in real time on a mobile application, that presents an intuitive graphical interface. Through the application, the safety limits, defined previously by the responsible engineer with basis on the specific characteristic of the dam, are monitored, and alerts are triggered when these limits are axceeded. Given the growing demand for innovative technologies in geotechnical instrumentation, this work contributes to the advancement of electronic systems applied to civil engineering, promoting greater safety, efficiency in data collection and mitigation of acccident risks.

Keywords: inertial measurement unit; remote monitoring; geotechnical instrumentation;

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

A/D	Analógico/Digital
ADC	Analog to Digital Converter
AM	Amplitude Modulada
BT	Bandwidth-Time
CC	Corrente Continua
CMOS	*Complementary metal-oxide-semiconductor*
CPFSK	*Continuous Phase Frequency Shift Keying*
E/S	Entrada e Saída
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
FCC	*Federal Communicatons Commission*
FIFO	*First in First out*
FM	Frequência Modulada
GFSK	*Gaussian Frequency Shift Keying*
GSM	*Global System for Mobile Communications*
I/O	*Input/output*
I2C	*Inter-Integrated Circuit*
ICOLD	*International Comission on Large Dams*
IMU	*Inertial Measurement Unit*
ISM	*Industrial Scientific Medical*
MEMS	*Micro-Electro-Mechanical Systems*
MISO	*Master Input Slave Output*
MOSI	*Master Output Slave Imput*
ODR	*Output Data Rate*
PSD	*Power Spectral Density*
PWM	*Pulse Width Modulation*
RAM	*Random Access Memory*
RF	Rádiofrequência
RISC	*Reduced Instruction Set Computer*
RX	Receptor
SCL	*Serial Clock*
SCLK	*Serial Clock*
SDA	*Serial Data*

SPI	*Serial Peripheral Interface*
SS	*Slave Select*
TWI	*Two-Wire Interface*
TX	Transmissor
UART	*Universal Asynchronous Receiver/Transmitter*
USART	*Universal Synchronous Asysnchronous Receiver Transmitter*
UTFPR	Universidade Tecnológica Federal do Paraná
WLAN	*Wireless Local Area Network*

LISTA DE SÍMBOLOS

K_b	Constante de Boltzmann
ε_0	Constante de permissividade do vácuo
μg/√Hz	Densidade de ruído
δ	Deslocamento
Q	Fator de qualidade
F	Força de entrada
f_r	Frequência ressonante
g	Aceleração da gravidade
B	Largura de banda
m	Massa
k	Constante de mola
g_0	Tamanho inicial da lacuna
T	Temperatura
V_B	Tensão de polarização

SUMÁRIO

1 INTRODUÇÃO

As barragens representam uma das maiores estruturas geotécnicas construídas pelo homem. Elas têm sido utilizadas à milênios para inúmeros fins, como armazenamento de água, o controle de vazões e a geração de energia. No entanto, com o avanço da exploração mineral e a carência de conservação socioambiental, a construção de barragens de rejeito tem chamado a atenção devido ao seu potencial danoso em casos de falhas (Cardozo, 2016).

De acordo com a Comissão Internacional de Grandes Barragens (ICOLD – *International Comission on Large Dams*), em 2019 existiam mais de 58.000 barragens de grande porte em todo mundo. A definição de barragens de grande porte adotada pela ICOLD inclui aquelas com altura mínima de 15 metros acima da fundação ou com a altura acima de 5 metros e capacidade de armazenamento superior a 3 milhões de metros cúbicos (ICOLD, 2019).

A história da construção de barragens no Brasil remonta a 1880, quando o país enfrentou a Grande Seca no Nordeste. Nessa época, o imperador Dom Pedro II criou uma comissão para buscar soluções para o problema de falta d'água na região, sendo a construção de barragens uma das medidas adotadas. A primeira barragem de armazenamento de água construída no país foi a Cedros, concluída em 1906 no estado do Ceará. Desde então, várias outras barragens foram erguidas, especialmente nas décadas de 1950 e 1960 (CBDB, 2011).

Em janeiro de 2019, ocorreu a tragédia de Brumadinho, que resultou em um grande deslizamento de lama e deixou centenas de pessoas mortas e desaparecidas (Santos, s.d). Em novembro de 2015, a tragédia de Mariana com a Barragem do Fundão causou um volume 62 milhões de m³ de rejeitos de mineração a serem despejados no Rio Doce e nas comunidades próximas. O desastre resultou em danos ambientais e sociais significativos, que incluem perda de vidas humanas, destruição de cidades e vilas, contaminação do rio e do mar (Bezerra, s.d). As tragédias de Brumadinho e Mariana, relacionadas ao rompimento de barragens das mineradoras Vale e Samarco, representam duas das maiores tragédias ambientais e humanas do Brasil

A evolução da instrumentação para monitoramento de barragens teve início no século XIX com a utilização de medições topográficas na França e de piezômetros

na Índia. Desde então, novos instrumentos e técnicas foram desenvolvidos para atender às necessidades específicas de cada tipo de barragem (Silveira, 2006).

Na primeira metade do século XX, surgiram os instrumentos de inclinometria, que permitem a medição de inclinações e movimento de barragens de concreto. Já na segunda metade do século, os piezômetros passaram a ser utilizados com maior frequência em barragens de terra, tornaram-se uma das principais ferramentas para o monitoramento de pressões e níveis d'água (Silveira, 2006)

Com o avanço da tecnologia, novos instrumentos de monitoramento surgiram, como extensômetros elétricos, que possibilitam a medição de deformações em estruturas metálicas, e os medidores de deslocamento por laser, capazes de registrar movimentos com alta precisão (Silveira, 2006).

Atualmente, a instrumentação para monitoramento de barragens é composta por uma variedade de dispositivos, tais como piezômetros, células de pressão, extensômetros, inclinômetros, medidores de nível d'água e de deslocamento, além de sistemas de aquisição de dados e *softwares* para análise de informações (Silveira, 2006).

Baseado no histórico de dano potencial causado por ruptura de barragens e necessidade constante de inovação das instrumentações que auxiliam na segurança da estrutura, o presente trabalho tem como objetivo aprimorar o monitoramento de barragens, por meio da utilização de equipamentos eletrônicos avançados, que permitirão a coleta e análise mais eficiente de informações. Para isso, será empregada a IMU NXP Precision 9DoF Breakout, composta por acelerômetro e magnetômetro de 3 eixos FXOS8700 e giroscópio de 3 eixos FXAS21002 (Adafruit, 2017). Os dados coletados serão transmitidos remotamente pela placa Arduino Nano com um transceptor de RF - RF-NANO, armazenados em nuvem e exibidos em uma interface gráfica num aplicativo móvel, por meio de uma unidade central.

1.1 OBJETIVO GERAL

O objetivo deste trabalho é desenvolver um sistema de sensoriamento remoto para monitorar o risco de ruptura de barragens, será utilizado uma IMU para medir a vibração em tempo real e transmitir os dados via sinal RF para um dispositivo de recepção.

1.2 OBJETIVO ESPECÍFICO

Para atingir o objetivo deste trabalho, foram realizadas as seguintes etapas:

- Realizar uma revisão bibliográfica sobre o sensoriamento remoto aplicado ao monitoramento de barragens;
- Estudar as características da IMU e suas possibilidades de aplicação para a medição de vibração em barragens;
- Desenvolver um Sistema de coleta e transmissão de dados com a utilização de um Arduino e módulo *transceiver* de RF;
- Realizar testes funcionais para atestar a eficiência da transferência de dados dos sensores para a nuvem.
- Desenvolver um Aplicativo Móvel utilizando o FlutterFlow para exibir aos usuários os dados obtidos, acompanhado de alertas.

1.3 JUSTIFICATIVA

A monitoração de barragens é uma questão crítica de segurança pública e ambiental. O uso de tecnologias de sensoriamento remoto pode auxiliar na prevenção de acidentes e na mitigação de seus impactos. A aplicação de uma IMU (*Inertial Measurement Unit*) para medir a vibração em barragens e transmitir os dados via sinal RF (radiofrequência) é uma abordagem inovadora que pode proporcionar maior precisão e abrangência no monitoramento. Além disso, a utilização de um dispositivo como o Arduino, pode tornar a tecnologia mais acessível para empresas de menor porte ou até mesmo para governos municipais. Este trabalho justifica-se para o desenvolvimento de tecnologias de monitoramento de barragens mais eficientes, seguras e acessíveis.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo, serão explorados conceitos essenciais e pertinentes para a compreensão do tema tratado e desenvolvido ao longo deste trabalho.

2.1 Monitoramento de barragens

A Lei nº 12.334/2010 estabelece a Política Nacional de Segurança de Barragens no Brasil, com o objetivo de garantir a segurança das barragens e prevenir acidentes. Essa lei estabelece diretrizes gerais para o monitoramento, a fiscalização, a segurança e a prevenção de acidentes em barragens, abrangendo barragens de usos múltiplos, rejeitos de mineração, de resíduos industriais, de água e de energia.

Entre as diretrizes estabelecidas pela lei, destaca-se a obrigatoriedade de planos de segurança de barragens, que devem conter medidas preventivas e corretivas em caso de acidentes, bem como a realização de inspeções periódicas e a apresentação de relatórios sobre a situação das barragens. Além de que, a lei prevê a criação de um sistema nacional de informações sobre segurança de barragens, que inclui o cadastro de todas as barragens existentes no país, com informações técnicas e dados sobre o monitoramento e a segurança das mesmas (Brasil, 2010).

Em 2020, após os graves acidentes ocorridos no Brasil envolvendo barragens, a Lei nº 14.066 foi aprovada, que altera a Lei nº 12.334/2010. A nova lei incluiu novas disposições para o monitoramento e a fiscalização de barragens, que amplia as exigências de segurança e prevenção de acidentes. Dentre as principais alterações, destaca-se a exigência de que as barragens sejam classificadas quanto ao seu risco potencial e ao seu dano potencial associado, permitindo que sejam estabelecidas medidas de segurança e prevenção especificas para cada caso (Brasil, 2020).

Ademais, a Lei nº 14.066/2020 determina que as empresas responsáveis pelas barragens devem disponibilizar informações atualizadas sobre a segurança das mesmas em seus sites na internet. A nova lei prevê também a criação de um fundo nacional para a segurança de barragens, com recursos destinados a ações de prevenção e reparação de danos em caso de acidentes (Brasil, 2020).

2.2 Sensores MEMS

Um dispositivo MEMS (Sistemas Microeletromecânico de silício) é um dispositivo microusinado que é capaz de converter um estímulo físico, como aceleração, força, pressão, temperatura e umidade, em um único sinal elétrico proporcional à magnitude do estímulo. Estes sensores são fabricados com tecnologias de semicondutores a litografia, o que lhes permite ser integrados com outros componentes eletrônicos em um único *chip* (GAD-EL-HAK, 2006).

Os dispositivos MEMS, que incluem sensores inerciais (como acelerômetros e giroscópios), micro espelhos, micro válvulas, micro bombeadores e micro atuadores, podem ser projetados para desempenhar uma ampla variedade de funções. Eles são amplamente utilizados em várias áreas, como eletrônica de consumo, sistemas de navegação, telecomunicações, automotivo, aviação, saúde e defesa. A rápida adoção desses dispositivos é impulsionada pela redução significativa do custo por eixo de detecção, possibilitado pelas tecnologias altamente integradas de sistemas MEMS/CMOS (Metal-óxido Semicondutor Complementar), além da crescente conscientização sobre os benefícios da aplicação do recurso de rastreamento de movimento para aprimorar a interação do usuário com diversos dispositivos (Shaeffer, 2013).

2.2.1 Acelerômetros

Em 1991, a empresa *Analog Devices*, em Norwood, Massachusetts, anunciou o lançamento do primeiro produto comercial baseado em microusinagem de superfície, o ADXL-50, um acelerômetro de 50 g utilizado para ativar o acionamento de airbags em veículos (“*Analog Divices Combine Micromachining with BICMOS*”). Em 2001, a *Analog Devices* produzia mensalmente dois milhões de acelerômetros microusinados de superfície, com um custo de US$ 4 por dispositivo em volume (GAD-EL-HAK, 2006).

Acelerômetros são dispositivos utilizados para medir a aceleração de um objeto em relação a um referencial inercial. Comumente, eles são compostos por um objeto de massa conhecida, suspenso por molas, que responde livremente a uma aceleração. A posição desse objeto é medida por um transdutor capacitivo, baseado na medida de deflexão de um elemento sensível à aceleração, que é uma estrutura suspensa por molas que gera um sinal elétrico proporcional à aceleração (Alexis, s.d).

Além de medir a orientação da gravidade, também respondem à aceleração linear devido ao movimento ou vibração (Shaeffer, 2013). Esses dispositivos podem medir a aceleração em um, dois ou três eixos (Alexis, s.d).

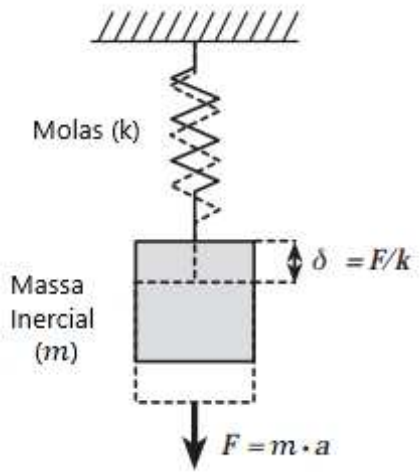
Todos os acelerômetros possuem uma estrutura básica que consiste em uma massa inercial suspensa por uma mola. No entanto, eles diferem na forma como detectam a posição relativa da massa inercial à medida que ela se desloca sob o efeito de uma aceleração externa aplicada. Um método comum de detecção é o capacitivo de duas placas. Para esse método, são necessários circuitos eletrônicos especiais capazes de detectar mudanças mínimas na capacitância ($<10^{-15}$ F) e convertê-las em uma tensão de saída amplificada.

Outro método comum utiliza piezoresistores para detectar o estresse interno induzido na mola. Em um método diferente, a mola é piezoelétrica ou contém um filme fino piezoelétrico, que gera uma tensão diretamente proporcional ao deslocamento. Em alguns casos raros, como em operações em temperaturas elevadas, é necessário utilizar detecção de posição com uma fibra ótica (Maluf, 2000).

As principais especificações de um acelerômetro são o alcance, geralmente dado em g, a aceleração gravitacional da Terra (1g = 9,81 m/s²), a sensibilidade (V/g) a resolução (g), a largura de banda (Hz); a sensibilidade a eixos cruzados e a imunidade a choques. O alcance e a largura de banda necessários variam significativamente em relação à aplicação.

O conceito fundamental de um acelerômetro pode ser compreendido por meio do sistema canônico da Figura 1. Neste sistema, uma massa de prova, representada pela variável m, é suspensa em uma estrutura mecânica por meio de uma mola, k, e responde a uma força de entrada, F, que é representativa da grandeza que se deseja medir. A força de entrada provoca um deslocamento, δ, na massa, que é então medido para detectar a força (Maluf, 2000).

Figura 1 – Esquema eletromecânico de um acelerômetro



Fonte: Adaptado de Maluf (2000).

Frequência ressonante:

$$f_r = \frac{1}{2\pi}\sqrt{\frac{k}{m}} \tag{1}$$

Aceleração equivalente ao ruído:

$$a_{noise} = \sqrt{\frac{8\pi K_b T f_r B}{Q\ m}}\ ; B < f_r \tag{2}$$

$K_b = constante\ de\ Boltzmann$

$T = Temperatura$

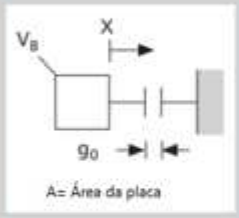
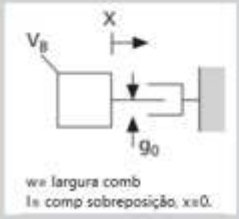
$B = largura\ de\ banda$

$Q = Fator\ de\ qualidade$

Uma análise entre duas geometrias de capacitores amplamente utilizadas, a placa-paralela e o *comb-finger* que se caracteriza por uma estrutura na qual uma microlamina se desloca no interior de uma estrutura na forma de um pente, como mostra a Figura 2(b). A Figura 2 resume os principais resultados obtidos, e demonstra a sensibilidade de carga dos capacitores é diretamente proporcional à tensão de polarização, V_B, e inversamente proporcional ao tamanho inicial da lacuna, g_0, leva em conta os valores de constante de permissividade do vácuo, ε_0. Essas observações são relevantes no contexto do projeto de dispositivos de alta sensibilidade, onde a maximização da carga armazenada é um dos principais objetivos. Em síntese, os resultados apresentados fornecem informações importantes para a seleção adequada

de geometrias de capacitores em aplicações de engenharia elétrica e eletrônica (Shaeffer, 2013).

Figura 2- Transdução de carga, transdução de forças e constante de mola eletrostática para dois tipos comuns de capacitores

		Capacitância	Transdutor de carga	Transdutor de força	Constante de mola
		C(x)	$\Delta Q(\Delta x)$	$\Delta F(\Delta v)$	k_E
a) capacitor de placas paralelas		$\frac{\varepsilon_0 A}{g_0 - x}$	$\frac{\varepsilon_0 A}{g_0^2} V_B \Delta x$	$\frac{\varepsilon_0 A}{g_0^2} V_B \Delta v$	$-\frac{\varepsilon_0 A}{g_0^3} V_B^2$
b) capacitor comb-finger		$\frac{\varepsilon_0 w(l+x)}{g_0}$	$\frac{\varepsilon_0 w}{g_0} V_B \Delta x$	$\frac{\varepsilon_0 w}{g_0} V_B \Delta v$	0

Fonte: Adaptado de Shaeffer (2013).

2.2.2 Giroscópios de taxa vibratória

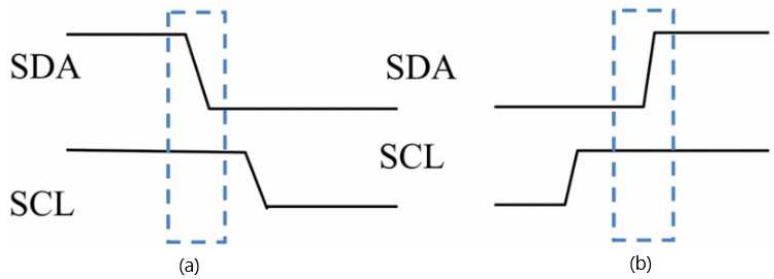
Como os acelerômetros, os giroscópios de taxa vibratória são dispositivos eletromecânicos relativamente complexos que medem a taxa angular de rotação em unidades de graus por segundo (dps). Eles utilizam o princípio da detecção da aceleração de Coriolis que atua em uma massa de prova vibratória, em proporção a taxa de rotação a longo de um eixo ortogonal ao eixo vibratório. Para calcular a taxa de rotação, é necessário conhecer a amplitude da velocidade da massa de prova. Para isso, o giroscópio deve colocar a massa de prova em oscilação ao longo de um eixo, regular a amplitude mecânica para que a massa possua uma velocidade estável e detectar o movimento correspondente da mesma massa ao longo de um eixo ortogonal. A taxa de rotação é proporcional à amplitude do movimento de detecção (Shaeffer, 2013).

2.3 Protocolos de comunicação

O protocolo I2C é utilizado para comunicação entre vários dispositivos em uma mesma rede em barramento, com cada dispositivo identificado por um endereço exclusivo. Os dados são transmitidos por meio de dois fios, SDA (*Serial Data)* e SCL (*Serial Clock)*, e a transferência é iniciada e finalizada por sinais de início e parada, respectivamente. O protocolo requer que um *bit* de reconhecimento ou não reconhecimento seja transmitido após a transmissão de cada *byte*. As mensagens são divididas em quadros de endereços e quadros de dados. O quadro de endereço contém o endereço do dispositivo escravo e a operação (leitura ou escrita) é representado pelo último *bit*. Os quadros de dados contêm vários *bytes* que são transmitidos continuamente até que o sinal de parada seja enviado. O significado de cada *byte* varia de acordo com a aplicação (Liu, 2019).

O barramento I2C é composto por três condições distintas: condição de início, condição de parada e condição de reinicio, que são indicadas pelas linhas SDA e SCL. No modo inativo, ambas as linhas permanecem altas. A condição inicial é dada pelo mestre, em que a linha SDA muda de alto para baixo enquanto a linha SCL permanece alta, conforme ilustrado na Figura 3(a). Após a condição inicial, o barramento I2C entra em modo de transmissão em que o dispositivo transmissor envia os dados após receber a confirmação adequada do dispositivo receptor. Ao término do envio de todos os dados, o mestre emite a condição de parada, em que a linha SDA muda de baixo para alto enquanto a linha SCL permanece alta, como mostrado da Figura 3(b) (Kumari, 2017).

Figura 3 - (a) Condição Inicial, (b) Condição parada



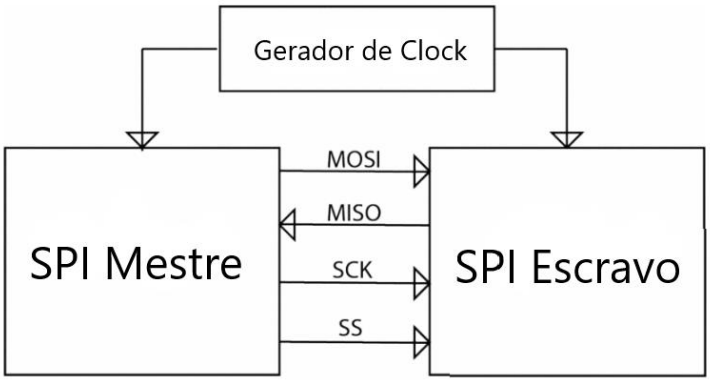
Fonte: Adaptado de Kamuri (2017).

A *Serial Peripheral Interface* (SPI) é um protocolo de comunicação serial síncrona desenvolvida pela Motorola em meados da década de 1980. A arquitetura

SPI consiste em um dispositivo mestre e vários dispositivos escravos, onde o mestre é responsável por gerar o relógio serial (SCLK) e fornecer o sinal apropriado na linha de seleção do escravo (SS). A comunicação SPI é realizada através de quatro linhas de comunicação: MOSI (*Master Output Slave Input*), MISO (*Master Input Slave Output*), SCLK (*Serial Clock*) e SS (*Slave Select*).

O protocolo é projetado para suportar comunicação *full duplex*, permitir a transferência simultânea de dados nos dois sentidos de transmissão. Com sua arquitetura mestre-escravo e interface de quatro fios, o protocolo SPI oferece uma ampla gama de possibilidades de configuração, que permite aos dispositivos se comuniquem de maneira rápida e confiável (Trivedi, 2018).

Figura 4 - Arquitetura SPI



Fonte: Adaptado de Pahlevi (2018).

2.4 Microcontroladores

O interesse em aplicação de controle digital tem crescido rapidamente nas últimas décadas, desde a introdução dos microcontroladores. Um microcontrolador é um computador em um único *chip* que inclui a maioria dos recursos de um computador, mas em dimensões reduzidas. Atualmente, existem centenas de tipos diferentes de microcontroladores, que variam de dispositivos com 8 pinos a dispositivos de 40 pinos, ou até mesmo dispositivos com 64 pinos ou mais (Ibrahim, 2006).

A memória destes dispositivos se divide entre a memória *Flash* que é não-volátil e é utilizada para armazenar o programa do usuário, memória SRAM de dados, e memória EEPROM de dados não-voláteis. A memória *Flash* pode ser apagada e reprogramada eletricamente. A memória EEPROM *(Electrically-Erasable Programmable Read-Only Memory)* é utilizada para armazenar dados do usuário não-

voláteis e pode ser gravada ou lida através do controle do programa. A memória SRAM *(Static Random Access Memory)* é um componente de acesso a memória de dados que permite a leitura e escrita dos conteúdos quando requisitada (Ibrahim, 2006).

Um exemplo de microcontrolador é o ATmega328P fabricado pela *Microchip,* sendo um microcontrolador de 8 bits baseado em arquitetura RISC *(Reduced Instruction Set Computer)* que utiliza um conjunto reduzido de instruções, que proporciona alta performance combinada com memória *Flash* ISP de 32 kB com capacidades de leitura enquanto escreve, 1024 B de EEPROM, 2 kB de SRAM, 23 linhas de E/S (entrada e saída) de propósito geral, 32 registradores de trabalho de propósito geral, três *Timer/Counters* (Temporizador/Contador) flexíveis com modos de comparação e PWM, interrupções internas e externas, USART *(Universal synchronous Asysnchronous Receiver Transmitter)* serial programável, que significa Transmissor/Receptor Universal Síncrono e Assíncrono, uma interface serial I2C ou *Two-Wire Interface* (TWI) orientada por *bytes*, porta serial SPI, um conversor A/D de 10 *bits* com até 8 canais analógicos de entrada, *Watchdog Timer* (recurso de segurança) programável com oscilador interno e cinco modos de economia de energia selecionáveis por *software.* O dispositivo opera entre 1,8 e 5,5 *Volts* (Microchip, s.d).

2.5 Sinais de Radiofrequência

A radiofrequência é uma banda do espectro eletromagnético que vária de 8,3 kHz a 3000 Ghz e é utilizada para a transmissão de sinais de comunicação (ANATEL, 2020), incluindo redes de telefonia móvel, transmissão de televisão e rádio e comunicação via satélite.

Os sinais de radiofrequência possuem características como amplitude, frequência, fase e polarização, que afetam sua transmissão e recepção. A amplitude determina a intensidade do sinal, a frequência sua taxa de oscilação, a fase indica a posição em que o sinal está em relação a uma onda de referência e a polarização se refere à orientação do campo elétrico do sinal (Haykin, 2009).

A transmissão analógica de sinais por meio de Modulação de Amplitude (AM) e Modulação de Frequência (FM) são técnicas comuns para transmissão de sinais de rádio de alcance curto e amplo à sua simplicidade e custo baixo, em comparação com transceptores digitais (Hussain, 2019).

Em 1985, a *Federal Communicatons Commission* (FCC) dos Estados Unidos alocou a banda de frequência de 2,4 Ghz da *Industrial Scientific Medical* (ISM) para ser utilizada em sistemas *Wireless Local Area Network* (WLAN), o que estimulou pesquisa e desenvolvimento na área. De acordo com as regulamentações estabelecidas pela FCC, modulações de espectro espalhado, *frequency hopping* (salto de frequência) ou sequência direta podem ser implementadas na banda ISM.

Em sistema de frequência *hopping,* o sinal do usuário é modulado com a portadora de maneira convencional, mas a frequência da portadora é continuamente alterada. Esse salto de frequência ocorre de forma aleatória e somente os receptores que estejam *"hopin step"* (pulando passo) com o transmissor podem receber o sinal da portadora para extrair as informações do usuário (Tuch, 1993).

A modulação GFSK *(Gaussian Frequency Shift Keying)* é uma técnica de modulação de deslocamento de frequência de fase continua (CPFSK), que apresenta vantagens e *design* de amplificador de baixa potência. Por essa razão, ela é amplamente utilizada em sistemas de entretenimento de transmissão de áudio/vídeo sem fio pessoais de baixa taxa de dados e baixo custo, bem como em aplicações modernas de comunicação, como Bluetooth (Chang, 2006).

Os sistemas de comunicação pessoal estão cada vez mais integrados às aplicações de baixa taxa de dados passadas com base na modulação GFSK. Embora o projeto do circuito analógico para GFSK seja maduro e barato, o *design* digital é atraente por sua capacidade de integração na construção de sistemas de *chip* complexos.

A largura de banda do filtro gaussiano tem um impacto significativo no espectro dos sinais GSFK. Um dos parâmetros mais importantes na concepção da modulação GSFK é o produto largura de banda – B, em 3 dB, e o período do símbolo T, também conhecido como "BT". É importante ressaltar que valores distintos de BT são empregados em diferentes padrões de comunicação sem fio. Por exemplo, o valor de BT adotado no padrão GSM = 0,3, enquanto que para Bluetooth = 0,5. De maneira geral, uma redução no valor de BT leva a uma diminuição na largura de banda espectral do sinal GSFK (Chang, 2006).

2.6 Firebase

O Firebase é uma plataforma desenvolvida pelo Google para criar e gerenciar aplicativos móveis e web. Ele oferece uma série de ferramentas e serviços para

desenvolvimento, hospedagem e gerenciamento de aplicativos, sem que seja necessário criar e manter infraestrutura de servidores. A plataforma é bastante popular por sua facilidade de integração e pelos serviços que oferece para desenvolvimento de aplicativos.

O *Firebase Realtime Database* é uma solução de banco de dados na nuvem do tipo não relacional (NoSQL), projetada para armazenar dados no formato JSON e sincroniza-los em tempo real com todos os dispositivos conectados. Diferente de bancos de dados tradicionais que utilizam solicitações HTTP pontuais, ele emprega sincronização contínua, o que permite que as alterações nos dados sejam transmitidas instantaneamente para todos os clientes conectados (Google, 2025).

Os desenvolvedores podem acessar o *Realtime Database* diretamente de dispositivos móveis ou navegadores, sem a necessidade de um servidor intermediário. Para garantir segurança, ele oferece uma linguagem de regras baseada em expressões, permitindo definir como os dados são estruturados e controlando quem pode lê-los ou alterá-los. Além disso, é possível integrar o *Firebase Authentication* (Autenticação do *Firebase*) para autenticar usuários e gerenciar permissões (Google, 2025).

O *Cloud Firestore* é uma alternativa avançada, projetada para aplicativos modernos. Ele organiza dados em coleções e documentos, permitindo consultas complexas, escalabilidade e maior flexibilidade na estruturação de informações. Assim como o *Realtime Database*, o *Firestore* suporta sincronização em tempo real e funciona *offline*, mas se destaca por oferecer recursos otimizados para modelos de dados mais sofisticados de grande escala, com suporte abrangente para clientes em plataformas Apple, Android e Web (Google, 2025).

O *Cloud Firestore* oferece uma cota gratuita que permite o uso inicial do banco de dados padrão sem qualquer custo. A cota gratuita inclui os seguintes limites:

- Dados armazenados: até 1GiB.
- Leituras de documentos: 50.000 por dia.
- Gravações de documentos: 20.000 por dia.
- Exclusões de documentos: 20.000 por dia.
- Transferência de dados de saída: até 10GiB por mês (Google, 2025).

2.7 Flutterflow

O Flutterflow é uma ferramenta alternativa ao *Framework* Flutter, o kit de desenvolvimento de interfaces de usuário (UI) desenvolvida pelo Google, projetado para criar aplicativos compilados nativamente para dispositivos móveis (*Android* e *iOS*), *web* e *desktop* (*Windows* e *Mac*) a partir de uma única base de código. A plataforma possibilita que os desenvolvedores projetem e prototipem aplicativos Flutter de maneira visual, considerada *low code*, utilizando componentes baseados em arrastar e soltar, em uma interface intuitiva e semelhante às ferramentas de *design* amplamente utilizadas (A. N. A. e Khalifa, 2024).

A plataforma é ideal para o desenvolvimento de aplicativos devido ao seu *framework* robusto, flexibilidade, curva de aprendizado acessível e desempenho nativo. Um dos destaques da ferramenta é o uso de *widgets*, que são elementos pré-configurados que facilitam a criação de interfaces e funcionalidades (Fadaee, 2024).

A autenticação de usuários é um componente essencial na arquitetura de aplicativos desenvolvidos com FlutterFlow. Esse mecanismo garante que apenas usuários registrados tenham acesso a aplicação. A integração com o Firebase facilita a autenticação por meio de contas de terceiros, como Google, Facebook e Apple, além de oferecer modelos prontos para recursos de segurança essenciais, como verificação de endereço de e-mail, redefinição de senha e validação por SMS. Esses recursos não apenas simplificam a implementação, mas também podem ser personalizados para proporcionar uma experiência do usuário fluida e harmoniosa (Fadaee, 2024).

A plataforma FlutterFlow não oferece um *backend* completo, sendo integrado ao Firebase para fornecer a infraestrutura necessária. O Firebase gerencia dados, autenticações e recursos essenciais, como armazenamento seguro de informações e acesso em tempo real aos dados, garantindo o suporte *backend* para o funcionamento eficiente da aplicação (Xiao et al., 2023). Ademais, o plano gratuito não oferece suporte à integração com o GitHub, ao download do código e APK, nem à publicação na web com domínio personalizado.

3 METODOLOGIA

Neste capitulo, serão descritas as ferramentas e tecnologias utilizadas no desenvolvimento da pesquisa, bem como a proposta de trabalho adotada. Tal descrição permitirá uma compreensão mais aprofundada do processo metodológico empregado, que inclui a identificação e utilização de recursos tecnológicos específicos, que visa a consecução dos objetivos.

3.1 Materiais

Nesta seção, serão expostas as ferramentas e tecnologias empregadas no desenvolvimento do presente estudo.

3.1.1 NXP *Precision* 9DoF *Breakout*

O dispositivo NXP *Precision 9DoF Breakout* é uma IMU que se comunica por meio de um protocolo I2C e combina dois dos melhores sensores do mercado. O acelerômetro e magnetômetro de 3 eixos FXOS8700 e o giroscópio de 3 eixos FXAS21002. A integração desses recursos tecnológicos permite obter resultados mais precisos e confiáveis na medição de movimento, desta forma proporciona um desempenho superior na execução de diversas tarefas.

O dispositivo mais relevante desta IMU é o acelerômetro FXOS8700 que por sua vez tem características interessantes comparadas com outros do mesmo seguimento de sensores MEMS.

- Alimentação de 2 a 3,6 V;
- Faixa de medição *(Measurement range)*: ± 2g (padrão), ± 4g, ± 8g que representa o intervalo de valores que o acelerômetro é capaz de medir;
- Não-linearidade *(Non-linearity)*: ±0,5 % de diferença entre a resposta real do acelerômetro e a resposta esperada, assume-se que ela seja linear;
- Densidade de ruído *(noise density)*: 99 µg/√Hz é a medida de ruído eletrônico adicionado à saída do acelerômetro;
- Taxas de dados de saída (ODR) de 1,563 Hz a 800 Hz, padrão é 100Hz;
- Resolução ADC de 14 bits para medições de aceleração (Adafruit, 2017).

3.1.2 RF NANO

O RF-Nano é um módulo de desenvolvimento de sistemas embarcados sem fio em 2.4 GHz baseado na plataforma Arduino Nano. Ele é equipado com um microcontrolador ATmega328p de 8 *bits* demais circuitos de suporte, como conversor UART para USB, cristal oscilador de 16 MHz e um transceptor de rádio nRF24L01 e antena integrada ao módulo.

Características do módulo RF-Nano:

- Tensão operacional: 5 V recomendado, tensão de entrada de alimentação VIN: 6 V ~ 15 V;
- Pinos de E/S digital: 20 (6 dos quais podem ser usados como saídas PWM);
- Pinos de entrada analógica: 8 (A0~A7);
- Corrente CC dos pinos de E/S: 40 mA;
- Corrente máxima da alimentação de 5 V: 1000 mA quando o VIN está conectado;
- Corrente de 3,3 V Pino: 50 mA;
- Memória *Flash*: 32 kB (com 0,5 kB dedicado ao programa *bootloader*);
- Memória SRAM: 2 kB;
- Memória EEPROM: 1 kB;
- Conversor UART para USB baseado no *chip* CH340G;
- Velocidade do relógio: 16 MHz (Microchip, s.d).

Características do nRF24L01:

- Módulo Transceptor de RF com interface via SPI a 4 pinos de até 8 MHz;
- Operação na Banda ISM *(Industrial Scientic and Medical Band)* de 2,4 GHz com 126 canais de RF;
- Modulação GFSK *(Gaussian Frequency-Shift Keying)* com taxas de transmissão de dados no ar de 250 kbps, 1 Mbps e 2 Mbps;
- Em transmissão a 0 dBm consume 11,3 mA, e em recepção a 2 Mbps 12,3 mA;
- Operação em ultra-baixa potência *(900 nA - power down, e 22 µA - standby);*
- Potências de transmissão programáveis em 0, -6, -12 ou -18 dBm;

- Manipulação de pacotes automática com tamanhos de 1 a 32 *bytes* e 6 *data pipes;*
- Alimentação entre 1,9 e 3,3 V com I/O tolerantes a 5 V;
- 3 FIFOs de 32 *bytes* para TX e RX;
- Alcance de até 1.000 m em campo aberto, 2.000 m com antena *biquad* (Microchip, s.d).

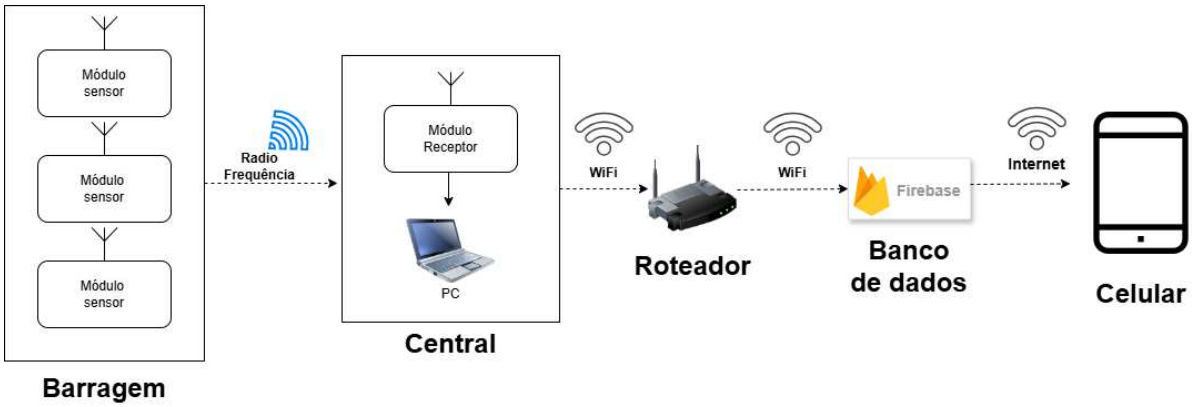
3.2 Módulos de *hardware*

Nesta seção, serão apresentados os componentes do projeto.

3.2.1 Módulos do sistema

A configuração do sistema, representada na Figura 5 é composta pelos seguintes componentes: Módulo Barragem, Módulo Central, banco de dados e um aplicativo. Essa estrutura modular foi projetada para atender à necessidade de monitoramento remoto da situação da barragem.

Figura 5 - Diagrama em blocos do sistema



Fonte: Autoria própria (2025).

Os Módulos Sensores, alimentados por um banco de baterias (power bank) e uma placa solar localizados na barragem, captam dados de vibração e os transmitem por meio de um link de comunicação RF para o Módulo Central. Este último é responsável por realizar a calibração dos dados recebidos, armazená-los no banco de dados em nuvem, e disponibilizá-los ao aplicativo em tempo real. O aplicativo, por sua vez, apresenta as leituras de forma acessível e fornece mecanismos de alerta, permitindo uma resposta rápida a possíveis anomalias.

O Módulo Barragem é composto por um conjunto de três Módulos Sensores, cuja função é realizar a leitura das vibrações presentes na estrutura da barragem. Esses dispositivos são fundamentais para a monitoração contínua das condições de segurança da barragem, permite a detecção precoce de possíveis anomalias e ocorrência de falhas. A disposição estratégica dos Módulos Sensores na estrutura da barragem é essencial para garantir uma leitura precisa e confiável das vibrações presentes no ambiente.

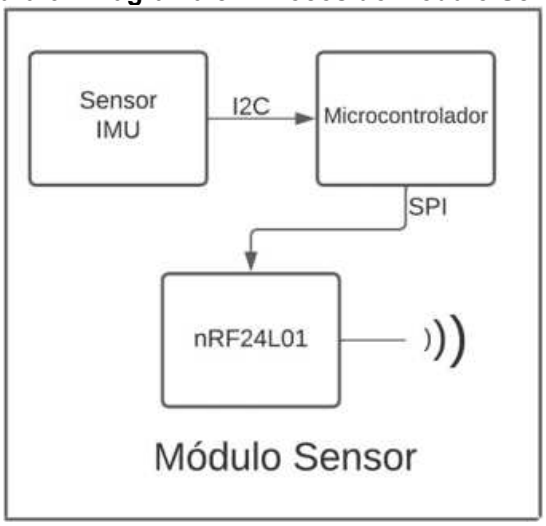
O Módulo Central é composto por um Módulo Receptor e um computador pessoal (PC), responsável pela integração de um código Python com o banco de dados do *Cloud Firestore*. Esse processo possibilita a conexão ao aplicativo desenvolvido na plataforma FlutterFlow, que exibe os dados de vibração da barragem e aciona alarmes sempre que condições perigosas são detectadas. Para garantir a eficiência e confiabilidade na transferência dos dados por meio de radiofrequência, o Módulo Central deve ser posicionado a uma distância estratégica e adequada em relação ao módulo Barragem.

Esses dados são interpretados e armazenados no *Cloud Firestore*, permitindo uma análise detalhada e precisa das informações obtidas pelos Módulos Sensores. A comunicação sem fio entre os dispositivos permite uma maior flexibilidade e mobilidade na monitoração das condições de segurança da barragem, que contribuem para a prevenção de eventuais acidentes. A interface com o usuário, na forma de um aplicativo multiplataforma, proporciona uma visualização detalhada e precisa das informações obtidas. Isso facilita a interpretação dos dados pelos usuários do sistema e permite a produção de alarmes, quando necessário.

3.2.2 Desenvolvimento do Módulo Sensor

O Módulo Sensor desempenha a função de adquirir dados por meio do sensor IMU (Unidade de Medição Inercial), e transmití-los ao Módulo Receptor. Para sua implementação, foi utilizado um microcontrolador ATmega328 integrado ao transceptor nRF24L01, configurado como um módulo RF-Nano, em conjunto com o dispositivo NXP *Precision 9DoF Breakout* conforme ilustrado no diagrama em blocos da Figura 6.

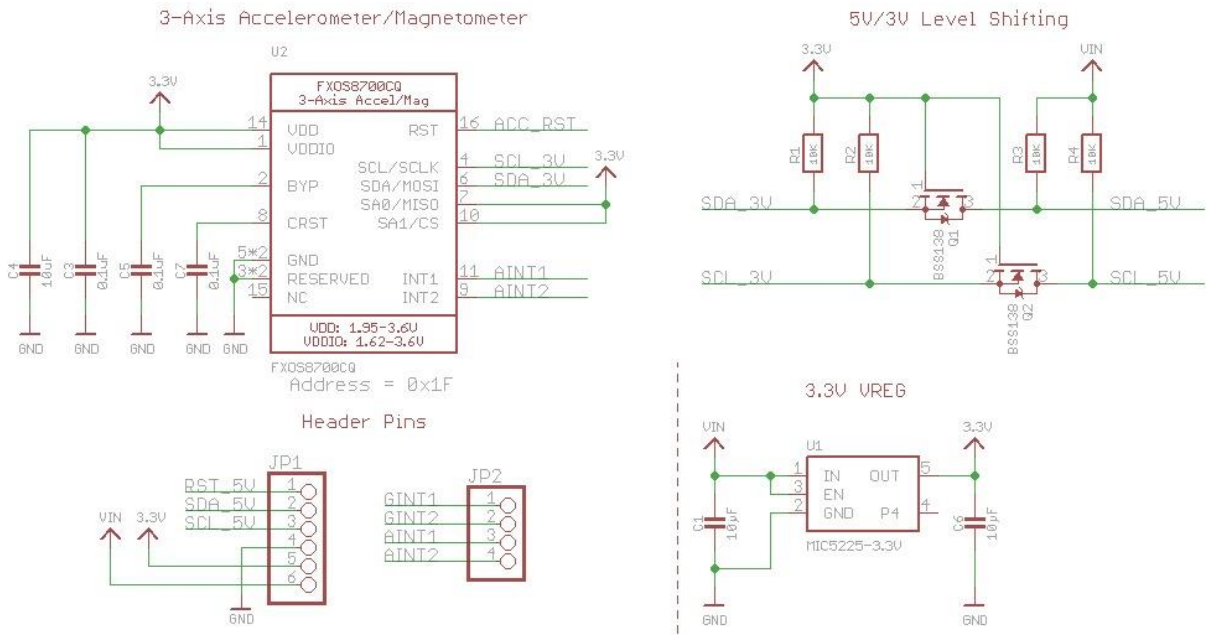
Figura 6 - Diagrama em Blocos do Módulo Sensor



Fonte: Autoria própria (2023).

O esquemático do sensor FXOS8700, apresentado na Figura 7, inclui um conversor de nível (*level shifting*), que é um circuito responsável por ajustar a tensão dos sinais de comunicação entre dispositivos que operam com diferentes níveis de tensão, garantindo compatibilidade e evitando danos. Esse recurso é essencial para a comunicação entre o sensor, que opera com 3,3V, e o microcontrolador, que pode utilizar 5V. A figura detalha todos os componentes e circuitos associados ao FXOS8700, incluindo a implementação do conversor de nível, crucial para a operação do IMU.

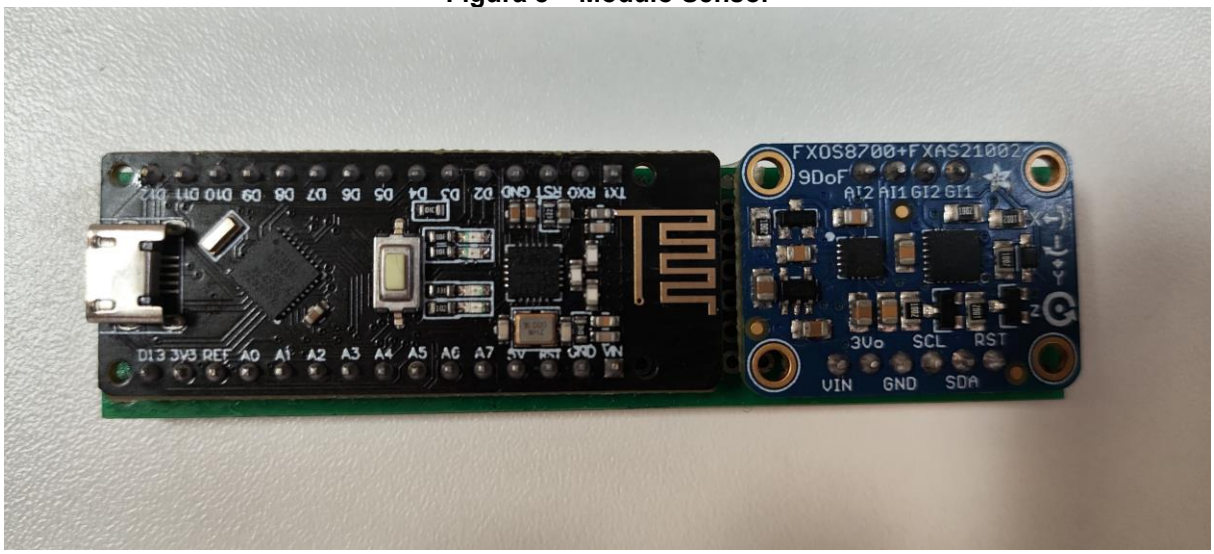
Figura 7– Esquemático do FXOS8700



Fonte: Adaptado de NXP Precision 9DoF Breakout (2024, p. 20).

Os elementos integrados no Módulo Sensor mostrado na Figura 8 enviam dados de forma confiável ao Módulo Central. Estes elementos são essenciais para o correto funcionamento do sistema de monitoramento de vibrações e permitem a aquisição, processamento e transmissão dos dados de forma integrada.

Figura 8 – Módulo Sensor



Fonte: Autoria própria (2025).

O código do Módulo Sensor é responsável por coletar dados de aceleração captados pelo sensor FXOS8700, configurado com o alcance de 2G (gravidade) e taxa de saída de dados padrão de 100 Hz. Esses dados são transmitidos ao Módulo Receptor utilizando o transceptor RF nRF24L01, que opera na banda ISM *(Industrial Scientic and Medical Band)* de 2,4 GHz. Para viabilizar essa comunicação, são utilizadas bibliotecas especificas personalizadas no Arduino, conforme ilustrado na Figura 9, que gerenciam a comunicação I²C do acelerômetro e a interface de radiofrequência do transceptor.

Figura 9 – Bibliotecas Arduino

```
1  #include <Wire.h>
2  #include <Adafruit_Sensor.h>
3  #include <Adafruit_FXOS8700.h>
4  #include <RF24.h>
5
```

Fonte: Autoria própria (2025).

No início do programa, o acelerômetro é iniciado, verificando-se sua conexão e funcionamento. Caso ocorra algum erro, o sistema exibe uma mensagem no monitor serial e interrompe sua execução. De forma similar, o módulo RF é configurado com um endereço de comunicação específico e ajustado para transmitir dados.

Durante a execução, o programa coleta continuamente os valores de aceleração nos eixos X, Y e Z por meio do sensor. Esses dados são organizados em um *array* e enviados via radiofrequência para o Módulo Receptor. Um atraso de 500 *ms* é introduzido entre as leituras para manter uma taxa de amostragem estável e evitar sobrecargas do sistema.

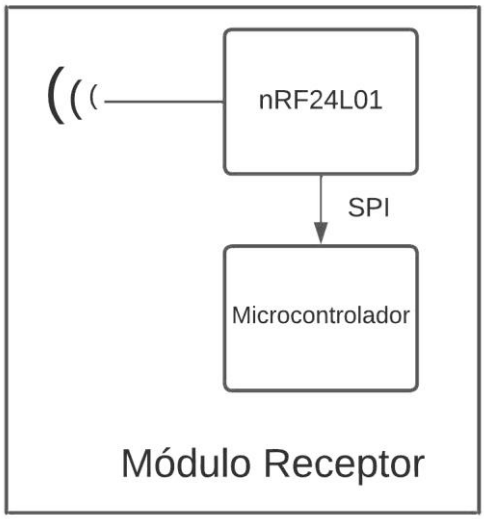
3.2.3 Desenvolvimento do Módulo Receptor

O Módulo Receptor é o dispositivo responsável pela recepção dos dados adquiridos pelos Módulos Sensores, composto por um módulo RF-Nano, que contém o microcontrolador ATmega328 e o transceptor nRF24L01 (Figura 10). A integração dos Módulos Sensores com o Módulo Receptor permite a aquisição e processamento dos dados de forma eficiente.

No início do programa, o modulo RF é configurado, incluindo a inicialização do endereço de leitura e a definição de seus níveis de potência. Caso ocorra falha na inicialização, o sistema exibe uma mensagem no monitor serial e interrompe sua execução. Após a configuração, o transceptor é ajustado para o modo de recepção contínua.

Durante a execução, o programa verifica constantemente a disponibilidade de dados no módulo RF. Quando há dados recebidos, eles são armazenados em um *array* que contém os valores de aceleração dos eixos X, Y, Z. antes de retransmitir esses valores para o computador via comunicação serial, o código substitui o valor da gravidade (9,81 m/s²) do eixo Z para ajustar as medições. Os valores restantes são enviados como uma *string*, com os eixos separados por vírgulas, no formato X, Y, Z.

Figura 10 - Diagrama em Blocos do Módulo Receptor



Fonte: Autoria própria (2023).

A comunicação entre Módulo Receptor e o computador (PC) é realizada utilizando um cabo mini-USB conectado ao RF-Nano. Esse cabo serve tanto para alimentar o módulo quanto para transmitir os dados via serial para o computador. No lado do computador, um código Python lê os dados recebidos pela porta serial e os processa, armazenando as informações no banco de dados *Cloud Firestore*. Esse método de comunicação é simples e eficaz, garantindo a transferência confiável dos dados para posterior análise e visualização.

Para a realização dos testes, o Módulo Sensor foi alimentado por uma fonte de 5V conectada a um cabo mini-USB. O Módulo Receptor foi posicionado em um ponto elevado, conectado ao computador para a transmissão de dados, e as simulações de vibrações ocorreram em um ambiente de laboratório, com o Módulo Sensor posicionado em uma bancada estável, simulando uma interação em condições reais.

3.3 Módulos de *software*

Nesta seção, será explicado como foi desenvolvido os protocolos de calibração do sensor e envio das informações para o banco de dados.

3.3.1 Calibração do sensor NXP *Precision 9DoF Breakout*

Embora o alinhamento ortogonal dos três acelerômetros seja o ideal, a metodologia proposta não presume essa condição (DA SILVA, 2010), uma vez que

desvios podem ocorrer durante o processo de fabricação, afetando a orientação precisa dos sensores.

Seguindo essa premissa, a calibração do sensor é essencial antes de seu uso, garantindo que as correções para desvios de ortogonalidade sejam adequadamente realizadas. Entre as bibliotecas personalizadas do Arduino para o sensor de aceleração FXOS8700, destaca-se uma que prometeu realizar a calibração de forma precisa. Contudo, durante o desenvolvimento do protótipo, mesmo seguindo todas as etapas recomendadas, a calibração não foi bem-sucedida ao utilizar as bibliotecas disponíveis.

De modo alternativo o vídeo intitulado "*How to Cabilibrate an Accelerometer*" do canal MicWro Engr, Michael Wrona demonstra um método de calibração desenvolvido por ele para calibrar o acelerômetro de seu quadricóptero, utilizando ferramentas especificas e códigos personalizados para alcançar o resultado desejado (Wrona, 2011).

Utilizando o programa Magneto v1.2, é possível realizar a calibração de sensores de acelerômetro e magnetômetro. O *Software* processa medições brutas fornecidas em um arquivo texto delimitado por tabulação, considerando a norma ideal do campo gravitacional, que representa a magnitude esperada das medições do acelerômetro. Para sensores que fornecem dados em unidades de G (gravidade), por exemplo, a norma/magnitude seria igual a 1. Após gerar o arquivo texto com as medições, por meio do código *record-data.py*, o arquivo foi carregado no Magneto, onde a calibração foi realizada automaticamente (Wrona, 2011).

De forma semelhante ao método descrito por Wrona, a calibração do sensor FXOS8700 neste trabalho seguiu princípios similares, mas com algumas adaptações especificas ao protótipo desenvolvido. Essas modificações foram realizadas para alinhar o processo às particularidades do sistema implementado, mantendo a essência do método original, mas ajustando detalhes para garantir precisão e compatibilidade com o *hardware* utilizado.

Para a coleta das medições brutas destinadas à calibração, foi desenvolvido o código *IMU-calibracao.ino*, executado na plataforma Arduino IDE. Esse código estabeleceu comunicação direta via porta serial com código *record-data.*py, responsável por calcular e armazenar a média de 25 leituras para cada linha de medição em um arquivo texto denominado *acceldata.txt* apresentado na Figura 11. O processo foi realizado em várias posições de cada eixo, resultando em um arquivo

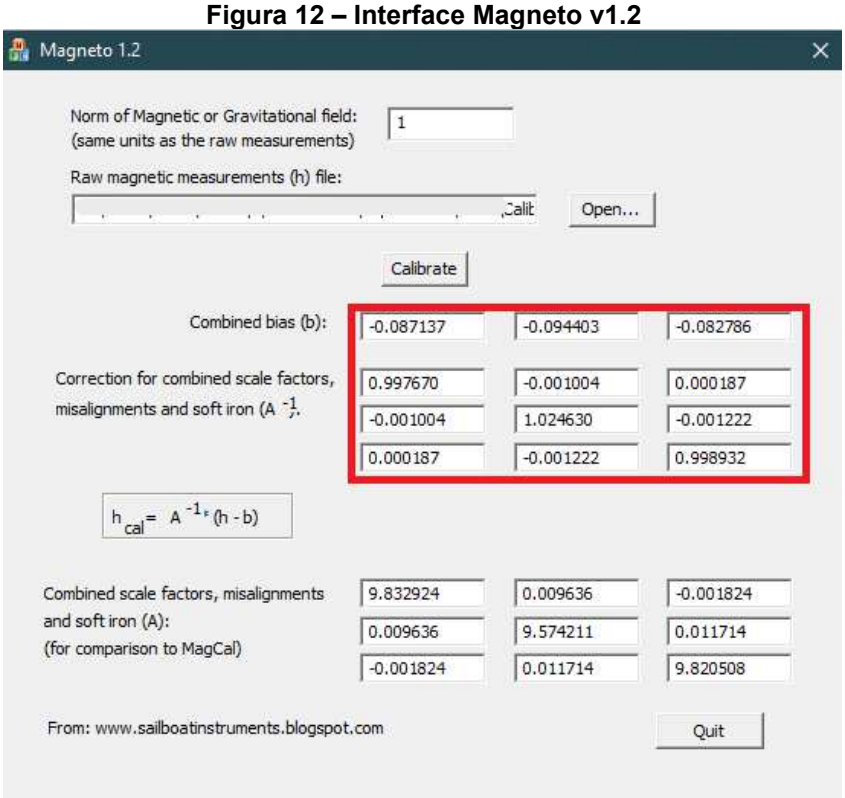
contendo em média, 150 leituras. Esse conjunto de dados foi então preparado para ser utilizado no *software* Magneto v1.2 para calibração exibido na Figura 12.

Figura 11 - Dados de leituras para calibração

```
0.19554143999999998     -0.11667404     9.73419428
1.2021540399999997      -0.11121831999999998    9.653508559999999
3.2126993600000002      -0.18156744000000002    9.17073252
5.03564736      -0.17764332     8.317739040000003
6.026371640000001       -0.15639488000000001    7.60065748
6.84241992      -0.15304492000000003    6.8677838
7.447708399999999       -0.13763524     6.205259120000001
8.03911884      -0.14242084     5.467504080000001
8.447334360000003       -0.13361524     4.80143792
8.781180919999999       -0.13399803999999998    4.17509272
9.171019600000001       -0.12270396     3.145030479999999
9.4475342       -0.12126831999999997    2.30074692
9.62441164      -0.12653259999999997    1.44631764
9.770469599999998       -0.12835096000000001    -0.11370692000000002
-0.26780468     -0.12078971999999999    9.732854399999997
-1.30456684     -0.11408979999999999    9.660208359999999
-3.2308849200000003     -0.058480599999999994   9.216387600000001
```

Fonte: Autoria própria (2025).

Figura 12 – Interface Magneto v1.2

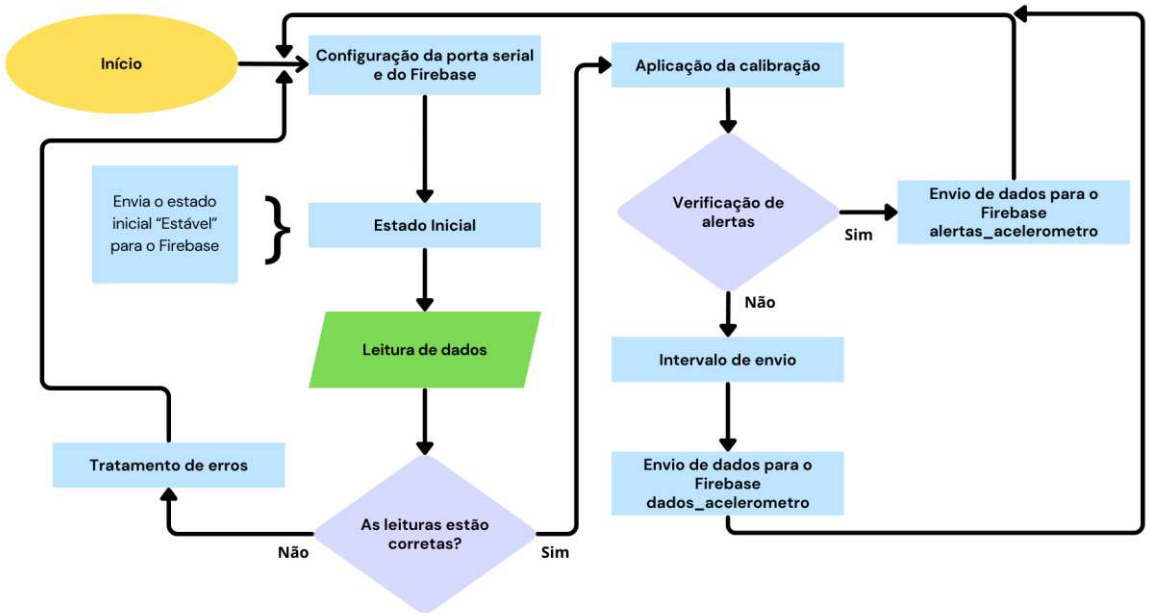


Fonte: Autoria própria (2025).

A partir das correções geradas pelo Magneto v1.2 se obteve uma matriz de calibração (A) e um vetor de *bias* (b) destacado na Figura 12 serão utilizados no código central do protótipo. O procedimento de coleta e calibração é necessário somente uma vez para cada IMU utilizada.

O código principal em conjunto com o Módulo Barragem, tem como função integrar todos os dispositivos envolvidos no processo de monitoramento, com o Módulo Central desempenhando o papel de intermediário entre os componentes do sistema. O código desenvolvido denominado *IntegraçãoPFAlertas.py*, também chamado de código principal é responsável por coletar os dados de aceleração provenientes do sensor IMU conectado ao Módulo Receptor, processar essas informações, realizar a triagem dos alertas e transmiti-las ao *Cloud Firestore* do Firebase. O fluxograma de funcionamento do código principal é exibido na Figura 13.

Figura 13 – Fluxograma do código principal



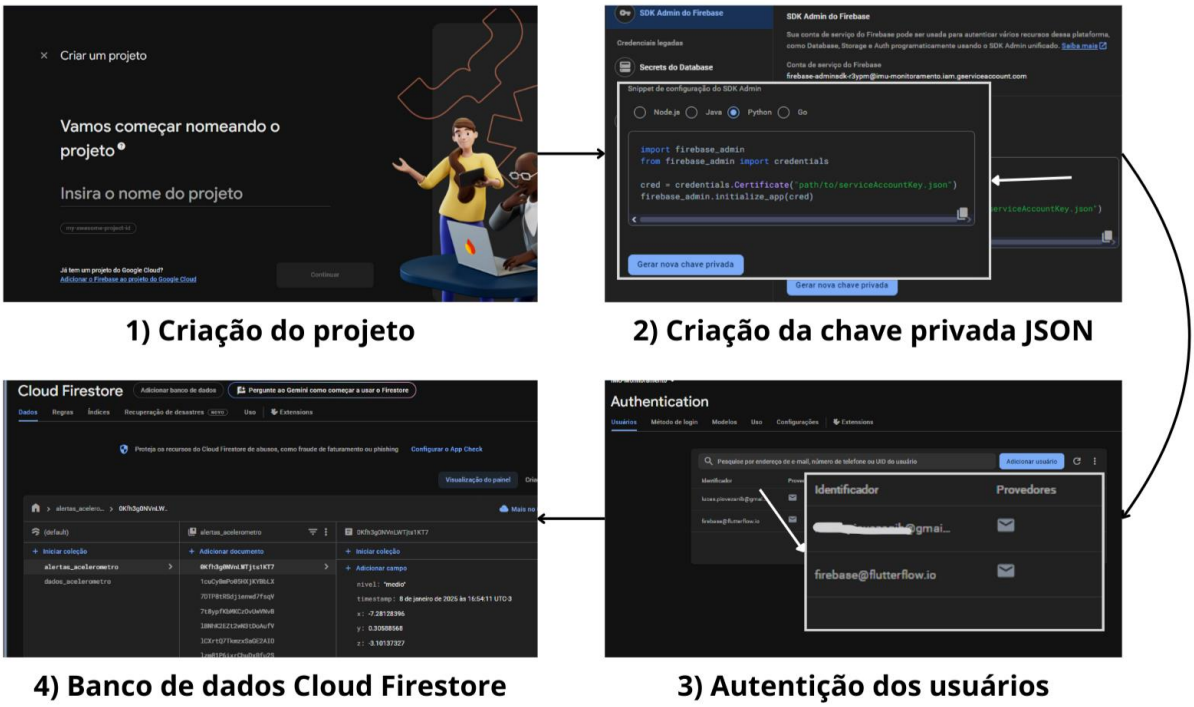
Fonte: Autoria própria (2025).

Na Figura 14, é apresentado o processo de criação de um novo projeto na plataforma, com a seleção do *Cloud Firestore* como banco de dados. Também foi realizada a configuração do *SDK Admin* do Firebase, essencial para autenticação e inicialização do cliente, permitindo o acesso ao banco de dados. Essa configuração inclui a criação de uma chave privada no formato JSON, que viabiliza chamadas API para integrar o código principal ao Firebase.

Adicionalmente, a figura destaca o processo de autenticação, responsável por definir quais usuários rem permissão para acessar e modificar o banco de dado. Entre os usuários cadastrados, um especifico *firebase@fluttherflow.io* é inserido. Essa integração, devido à compatibilidade entre Firebase e FlutterFlow, facilita

significativamente a comunicação entre o banco de dados e o aplicativo, otimizando o fluxo de dados entre as plataformas.

Figura 14 – Procedimento de conexão com Firebase



Fonte: Autoria própria (2025).

Neste contexto, a integração com o Firebase utiliza uma biblioteca especifica que permite configurar as credenciais de autenticação, acessar o banco de dados e inicializa-lo adequadamente. O arquivo de credenciais JSON, gerado pelo console do Firebase, contém as chaves de autenticação e permissões necessárias para o projeto configurado, e seu caminho completo é especificado no código principal. Além disso, o sistema cria um cliente para o banco de dados, possibilitando a execução de operações de leitura e gravação. Isso viabiliza o monitoramento em tempo real das condições do sistema e a geração de alertas, conforme ilustrado posteriormente na Figura 15.

Figura 15 – Inicialização do Cloud Firestore do Firebase

```python
import serial
import time
import json
import numpy as np
from firebase_admin import credentials, firestore, initialize_app #Biblioteca que permite interação com Firebase

# Configurações da porta serial
PORTA_SERIAL = 'COM4'  # Porta serial
BAUD_RATE = 9600
TIMEOUT = 1            # Timeout para leitura da porta serial (em segundos)

# Inicialização do Firebase
CAMINHO_CREDENCIAIS = 'C:/Users/
cred = credentials.Certificate(CAMINHO_CREDENCIAIS) # Autenticação de credencial da chave
initialize_app(cred) # Incializa o Firebase no programa Python
db = firestore.client() # Cria um cliente para acessar o banco de dados Firestore

```

Fonte: Autoria própria (2025).

A comunicação entre o Módulo Receptor é estabelecida por meio da porta serial, configurada para receber dados brutos do sensor nas direções x, y e z. Esses dados são corrigidos por um processo de calibração, que aplica uma matriz de transformação (A) e um vetor de *bias* (b) previamente ajustados para compensar erros e desvios do sensor, exibidas na Figura 16.

Com base nos valores calibrados, o sistema identifica variações de aceleração que ultrapassam limites predefinidos, categorizados por baixo, médio ou alto. Quando isso ocorre, alertas são registrados no *Cloud Firestore*, detalhando a intensidade da vibração e as coordenadas associadas. Além disso, o sistema armazena periodicamente as leituras calibradas em sua determinada coleção no *Cloud Firestore* de acordo com o tempo de leitura que for necessária, mostrado na Figura 16.

Figura 16 – Coleções, parâmetros de calibração e limites de alerta

```python
# Nome da coleção no Firebase
NOME_COLECAO = "dados_acelerometro"
NOME_COLECAO_ALERTAS = "alertas_acelerometro"

# Parâmetros de calibração ajustados
A = np.array([[0.997670, -0.001004, 0.000187],  # Matriz de correção
              [-0.001004, 1.024630, -0.001222],
              [0.000187, -0.001222, 0.998932]])
b = np.array([-0.087137, -0.094403, -0.082786])  # Vetor de correção de bias

# Limites de alerta para vibração
LIMITES_ALERTA = {
    "baixo": 3,
    "medio": 5,
    "alto": 7
}
```

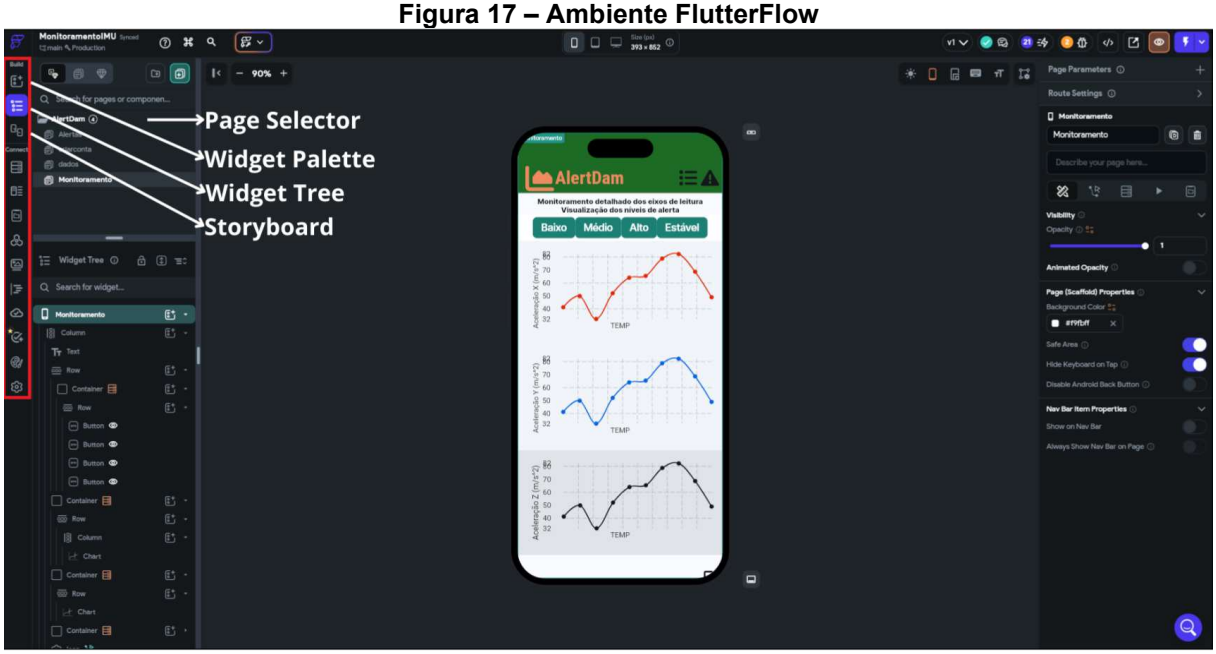
Fonte: Autoria própria (2025).

O código principal incorpora mecanismos para inicializar corretamente o estado do sistema, garantindo que o estado “Estável” seja comunicado ao Cloud Firestore antes do início das coletas também possui rotinas que tratam erros de comunicação serial e asseguram o envio de dados formatados corretamente.

3.4 Desenvolvimento do aplicativo

O FlutterFlow, o menu centraliza e organiza ferramentas essenciais para o desenvolvimento de aplicativos, como o *Widget Palette*, que contém componentes visuais, o *Page Selector*, para gerenciar páginas, e o *Widget Tree*, que exibe a hierarquia e conexões dos elementos. Além disso, o *Storyboard* oferece uma visão geral do fluxo do aplicativo, facilitando ajustes e garantindo uma navegação intuitiva, ilustrada na Figura 17. Essa estrutura torna o processo mais acessível e produtivo, mesmo para usuários com pouca experiencia em programação.

Figura 17 – Ambiente FlutterFlow



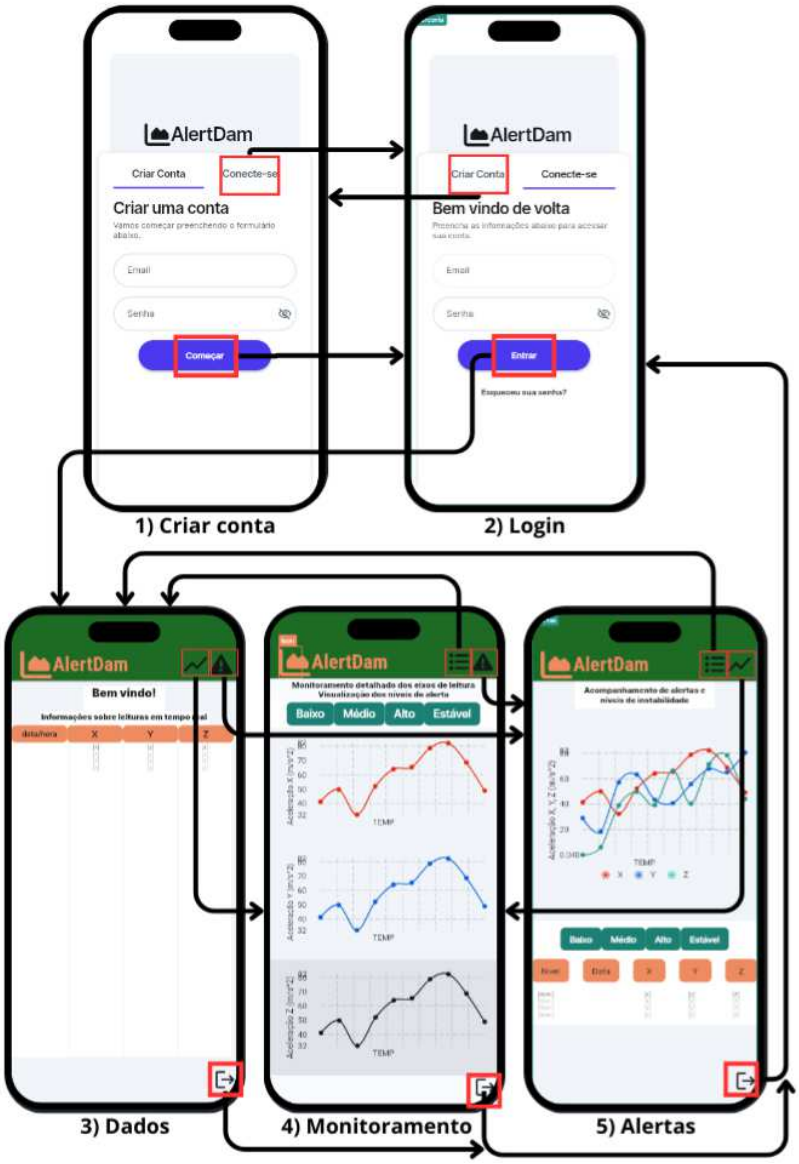
Fonte: Autoria própria (2025).

A Figura 18 representa as cinco interfaces desenvolvidas para o aplicativo, juntamente com o fluxo de navegação projetada para atender às diferentes necessidades do sistema. A primeira interface é destinada ao cadastro de usuários, permitindo o registro de novos perfis devidamente liberados, enquanto a segunda interface é voltada para o login, possibilitando a verificação no banco de dados para autenticar usuários previamente cadastrados.

As três interfaces subsequentes apresentam os dados coletados durante as medições e gráficos de linhas que correlacionam aceleração com tempo, facilitando a análise das informações. A interface Dados exibe leituras periódicas contendo data, hora e os valores de aceleração nos três eixos (X, Y, Z). Esses dados são apresentados em tempo real, com base no intervalo determinado configurado no coleção *dados_acelerometro*.

A interface “Monitoramento” oferece gráficos individuais que mostram o comportamento de cada eixo separadamente, incluindo um alerta de nível destacado na parte superior para maior clareza. Por fim a interface “Alertas” é dedicada aos dados de alertas. Nela, a parte superior exibe um gráfico de linhas com o comportamento dos três eixos simultaneamente. A parte inferior fornece informações detalhadas sobre os níveis de alerta, incluindo data, hora e as leituras registradas no momento exato em que o alerta foi acionado, conforme registrado na coleção *alertas_acelerometro*.

Figura 18 - Interfaces do aplicativo e seu fluxo de navegação



Fonte: Autoria própria (2025).

4 RESULTADOS E DISCUSSÃO

Os resultados obtidos deste trabalho envolveram tanto a implementação prática do sistema de sensoriamento remoto para monitorar o risco de ruptura de barragens quanto a análise dos dados obtidos e visualização dos mesmos.

Foi realizada a implementação funcional do sistema de coleta e transmissão de dados, que permitiu a medição em tempo real das vibrações com uso da IMU. Adicionalmente, foi possível estabelecer uma transmissão ágil dos dados por meio do sinal Radiofrequência (RF) para um dispositivo receptor, bem como a apresentação dos mesmos por meio de um aplicativo móvel, concebido com o propósito de fornecer ao usuário dados adquiridos de maneira intuitiva.

Com todos os componentes energizados e em funcionamento, o código principal é iniciado para verificar as comunicações serial e com o banco de dados. Nesse momento, o sistema enviou o primeiro alerta padrão de “Estável”, indicando uma coordenada inicial zero em todos os eixos. Após essa inicialização, o código principal passou a exibir os dados recebidos pelo Módulo Sensor, já calibrados, além de confirmar o envio dessas informações ao banco de dados e a transmissão dos alertas ao *Cloud Firestore*, conforme ilustrado na Figura 19 .

Figura 19 – Output de dados do código *IntegraçãoPFAlertas.py*

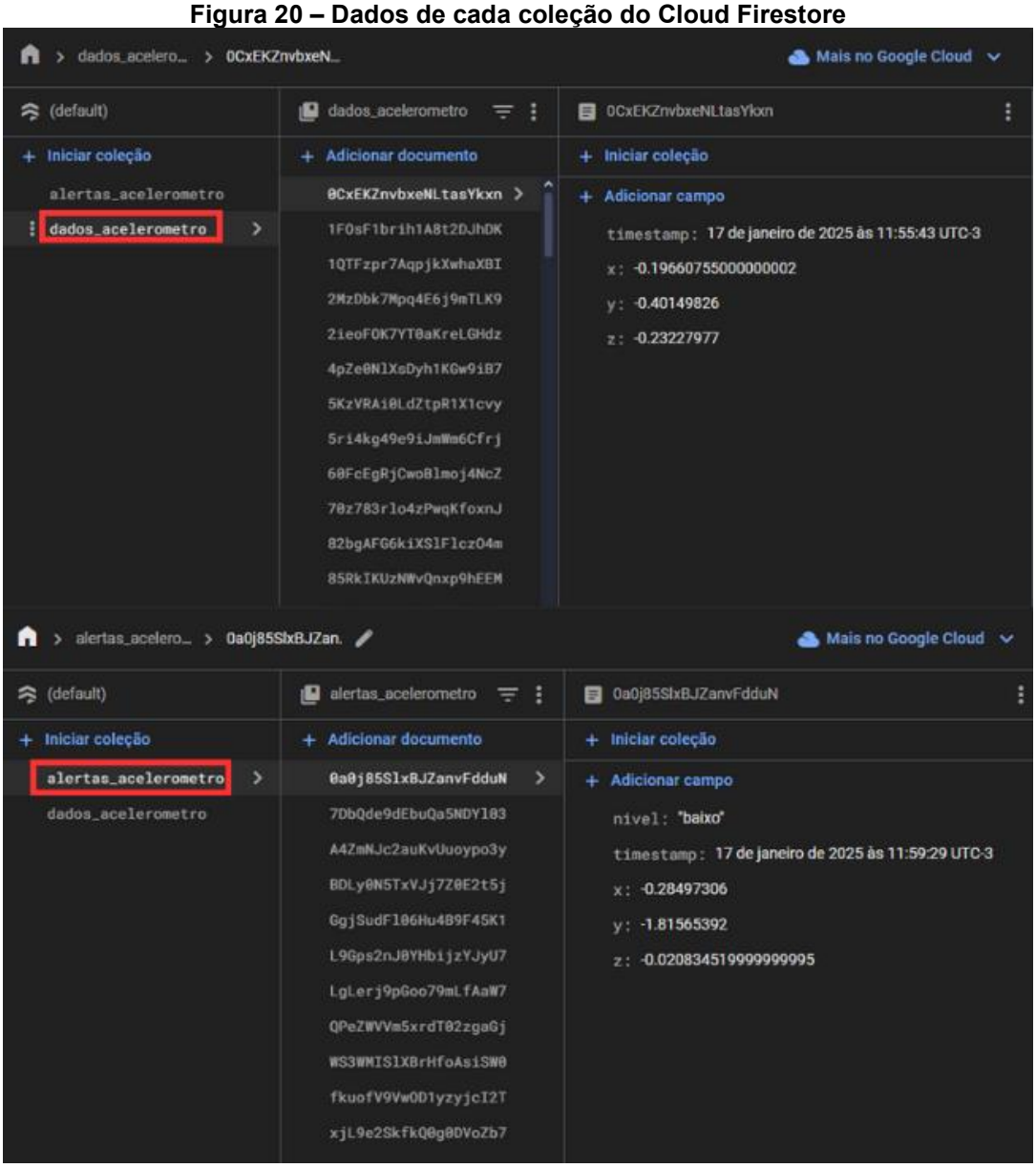
```
Dados após calibração: [-0.15668697 -0.41180916 -0.21228143]
Dados após calibração: [-0.16667115 -0.41175024 -0.25224058]
Dados após calibração: [-0.17666045 -0.39129648 -0.21230961]
Dados após calibração: [-0.15669071 -0.41178472 -0.23226007]
Dados após calibração: [-0.17664854 -0.401555   -0.20230807]
Dados após calibração: [-0.16669192 -0.38107244 -0.20233064]
Dados após calibração: [-0.1866026  -0.43223498 -0.24223056]
Dados após calibração: [-0.17665041 -0.40154278 -0.21229739]
Dados após calibração: [-0.17664037 -0.41178908 -0.21228517]
Dados após calibração: [-0.16665363 -0.42204542 -0.21227108]
Dados após calibração: [-0.18661894 -0.41176682 -0.22227636]
Dados após calibração: [-0.17662846 -0.4220476  -0.20228363]
Dados após calibração: [-0.16666367 -0.41179912 -0.2122833 ]
Dados após calibração: [-0.17661025 -0.44252798 -0.21224851]
Dados após calibração: [-0.15671896 -0.38105804 -0.22230741]
Dados após calibração: [-0.16666554 -0.4117869  -0.22227262]
Dados após calibração: [-0.25500863 -1.63265594  1.12772293]
Alerta de nível 'baixo' enviado ao Firebase: {'x': -0.25500863, 'y': -1.6326559399999998, 'z': 1.12772293, 'nivel': 'baix
o', 'timestamp': Sentinel: Value used to set a document field to the server timestamp.}
Dados do alerta também enviados ao Firebase na coleção dados_acelerometro: {'x': -0.25500863, 'y': -1.6326559399999998, '
z': 1.12772293, 'timestamp': Sentinel: Value used to set a document field to the server timestamp.}
Dados após calibração: [-0.16668936 -0.39126986 -0.2422757 ]
Dados após calibração: [-0.15672083 -0.38104582 -0.23229673]
Dados após calibração: [-0.16662981 -0.44256246 -0.192268  ]
Dados calibrados enviados ao Firebase: {'x': -0.16662981, 'y': -0.44256246, 'z': -0.192268, 'timestamp': Sentinel: Value
used to set a document field to the server timestamp.}
Dados após calibração: [-0.15668067 -0.42203102 -0.23224785]
```

Fonte: Autoria própria (2025).

Os dados coletados pelo sensor, transmitidos ao Módulo Central, são enviados com um intervalo de 500 *ms* e exibidos no terminal de saída já calibrados. Esta configuração foi adotada considerando a possibilidade de inatividade dos dispositivos em casos de intervalos muito longos de transmissões. É importante destacar que, em aplicações reais, as leituras serão enviadas ao banco de dados em intervalos poderão ser ajustados no código principal, conforme a necessidade dos engenheiros responsáveis. Durante os testes, os intervalos foram reduzidos para facilitar a compreensão e a análise do comportamento do sistema.

A Figura 20 apresenta a estrutura do banco de dados *Cloud Firestore*, que organiza as informações enviadas para a nuvem com base no intervalo de tempo e nos limites estabelecidos. Os dados são armazenados em duas coleções principais: *dados_acelerometro*, responsável por registrar periodicamente as medições de aceleração essenciais para a construção de gráficos no aplicativo; e *alertas_acelerometro*, dedicada ao armazenamento e envio de informações de alerta para o aplicativo, garantindo o monitoramento em tempo real.

Figura 20 – Dados de cada coleção do Cloud Firestore



Fonte: Autoria própria (2025).

Com os dados armazenados no *Cloud Firestore*, o aplicativo pode ser iniciado, aproveitando a integração direta e efetiva entre Firebase e FlutterFlow. Essa integração simplifica a utilização das funcionalidades dentro da interface. Na Figura 21 é possível observar a exibição da aba “Dados”, organizada em linhas e colunas que apresentam informações como coordenadas, data e hora.

Figura 21 – Aba de exibição de Dados do App

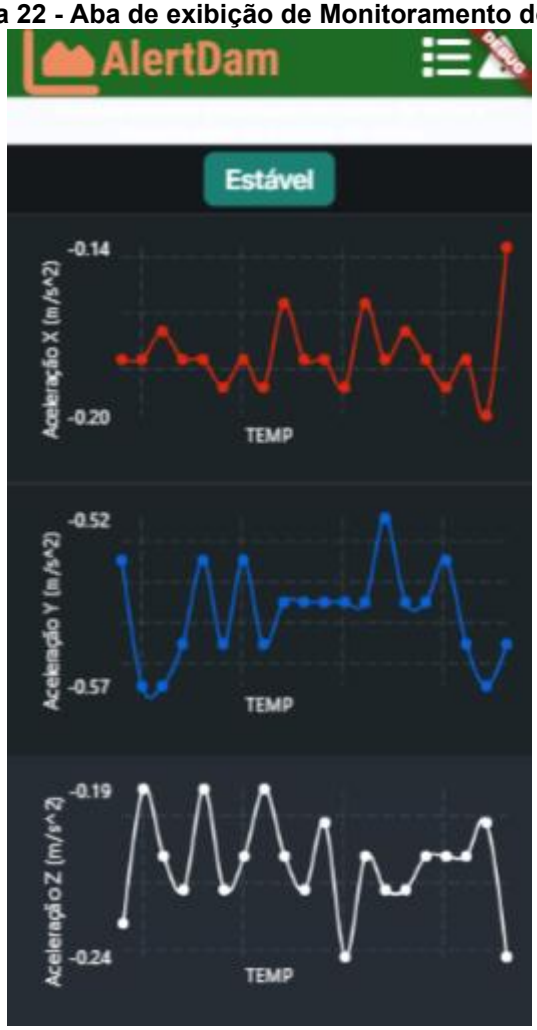
Fonte: Autoria própria (2025).

A aba “Monitoramento”, ilustrada na Figura 22, apresenta os dados de cada eixo separadamente em um gráfico de aceleração em função do tempo, proporcionando uma visualização clara e detalhada do comportamento de cada eixo. Esse formato facilita a identificação da direção predominante da aceleração, oferecendo uma análise mais funcional. Além disso, na parte superior da interface, é exibido o nível de alerta atual da barragem, permitindo um acompanhamento em tempo real das condições de estabilidade.

A aba “Alertas”, ilustrada na Figura 23, organiza as informações de monitoramento e alertas. Na parte superior, apresenta um gráfico cartesiano único que combina os três eixos, permitindo uma visualização integrada do comportamento dos dados. Na região central, a aba “Alertas” exibe o indicador de alertas, destacando o nível atual de criticidade. Já na parte inferior (Figura 23), são listados os detalhes de

cada alerta captado, incluindo as coordenadas, a data e a hora, garantindo um registro completo e acessível para análise.

Figura 22 - Aba de exibição de Monitoramento do App



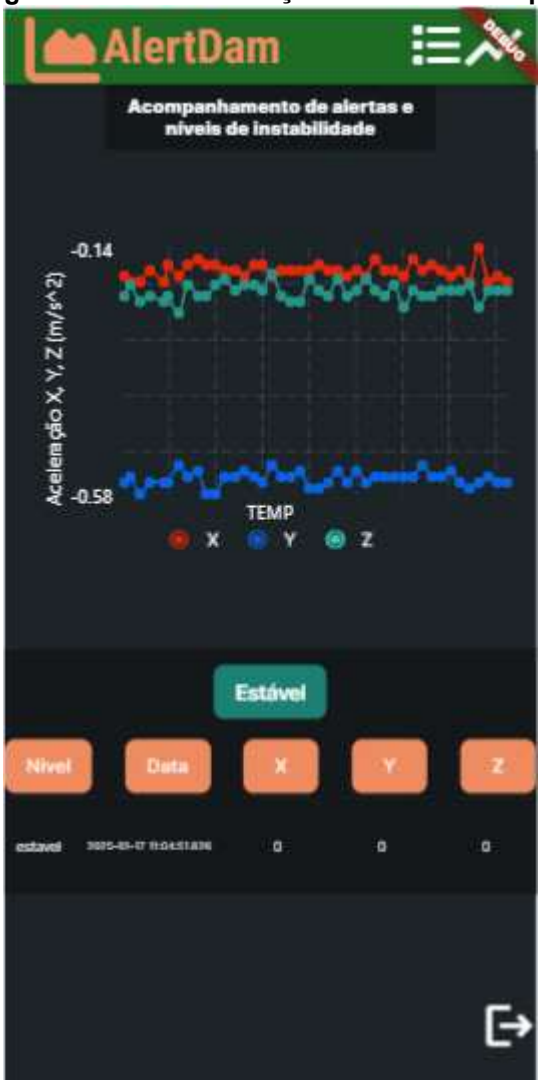
Fonte: Autoria própria (2025).

As abas “Monitoramento” e “Alertas” foram desenvolvidas para apresentar as leituras do sensor em diferentes condições de alerta: Baixo, Médio e Alto. Os valores associados a cada nível de alerta são definidos pelos engenheiros responsáveis no código principal. Na simulação em questão, os parâmetros foram configurados com os seguintes valores: Baixo = 1, Médio = 2 e Alto = 3.

Na Figura 24 e Figura 25, as abas “Monitoramento” e “Alertas” ilustram os níveis de alerta por meio das variações na amplitude das acelerações, complementadas por indicações visuais. O nível Baixo exibe oscilações sutis, sinalizando a necessidade de atenção; o nível Médio destaca aumentos moderados,

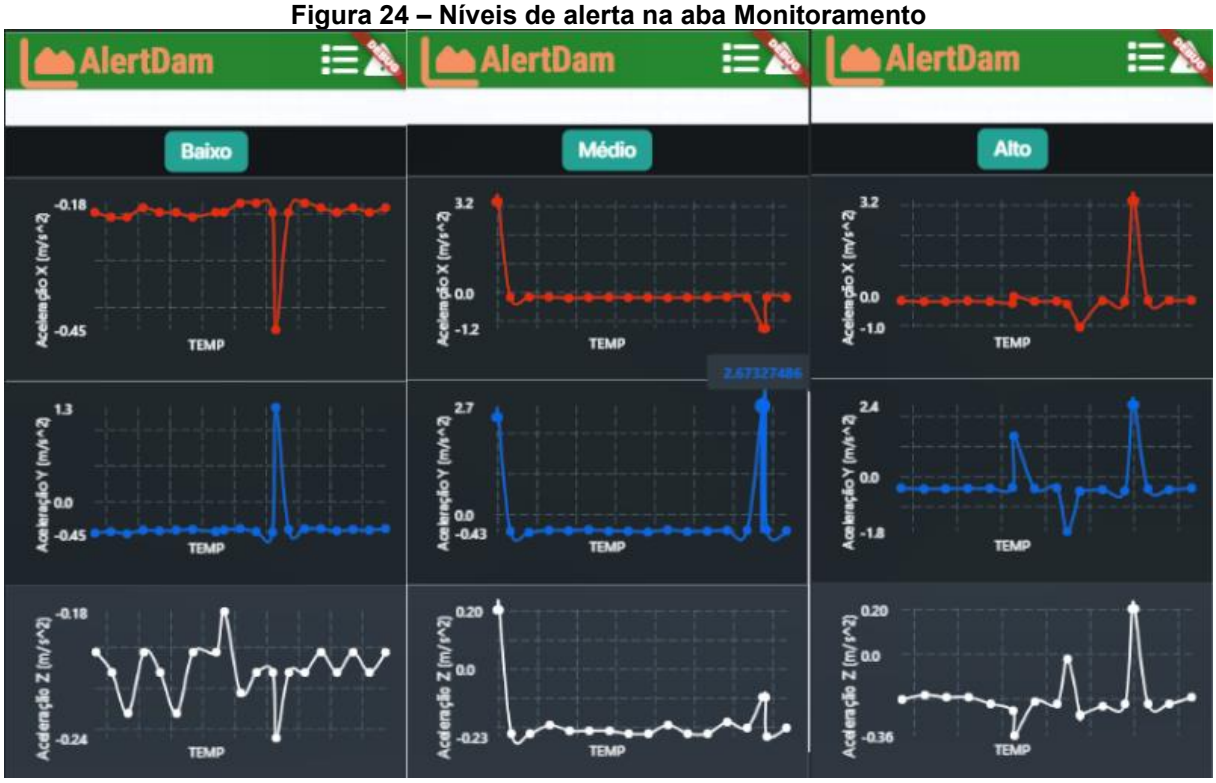
sinalizando maior movimentação; e o nível Alto apresenta picos acentuados, indicando uma situação crítica.

Figura 23 - Aba de exibição de Alertas do App



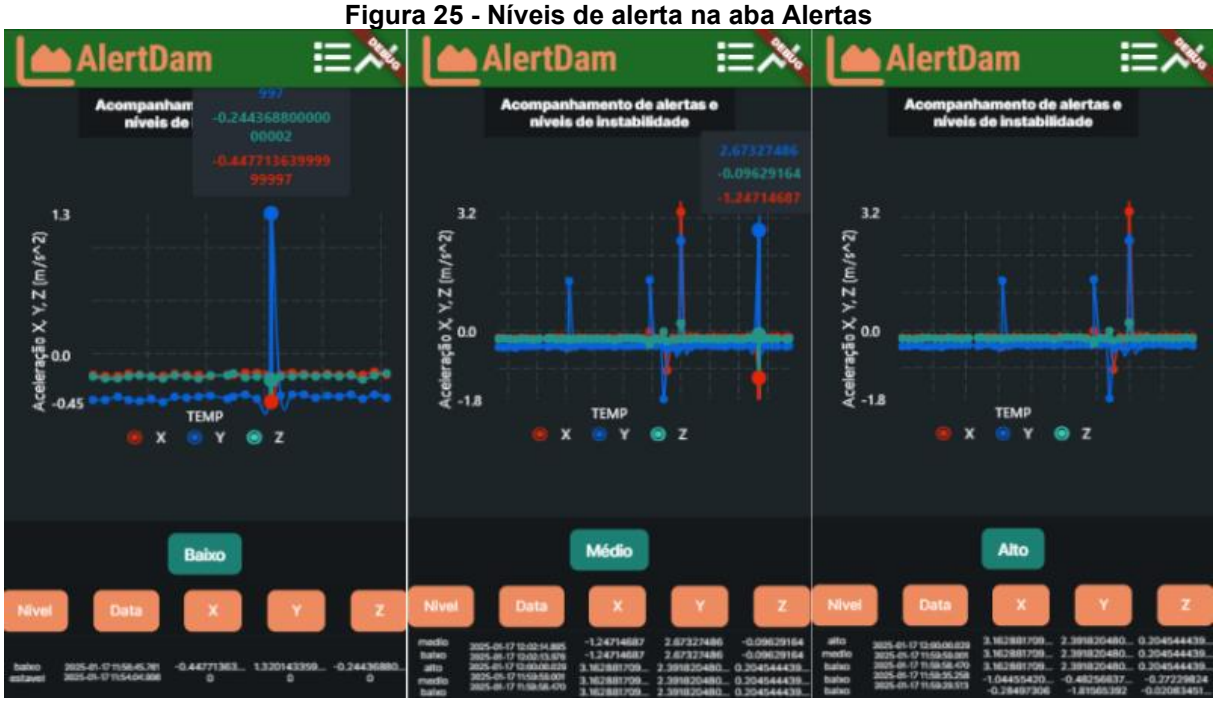
Fonte: Autoria própria (2025).

Figura 24 – Níveis de alerta na aba Monitoramento



Fonte: Autoria própria (2025).

Figura 25 - Níveis de alerta na aba Alertas



Fonte: Autoria própria (2025).

Os custos para a aquisição dos componentes do projeto estão apresentados no Quadro 1, que traz uma estimativa baseada nos valores praticados em março de 2023 para a IMU e o RF-Nano, e em janeiro de 2025 para a placa solar e o *power*

bank, totalizando R$ 530,66. Além disso, como o aplicativo foi desenvolvido utilizando a plataforma FlutterFlow, é necessário a aquisição do plano “Pro”, com um custo estimado de R$ 211,05 por mês. Este plano oferece funcionalidades essenciais, como a possibilidade de gerar o arquivo APK para dispositivos Android e realizar a publicação do aplicativo tanto na Apple Store quanto na Play Store, facilitando sua distribuição e acessibilidade aos usuários finais.

Quadro 1 - Custo dos componentes

Componentes	Quantidade	Valor Total
NXP *Precision 9DoF Breakout*	1	R$ 288,00
RF-Nano	2	R$ 65,02
Power Bank	1	R$ 91,63
Placa Solar 5V	1	R$ 86,01

Desta forma, este trabalho obteve como resultados o desenvolvimento de um dispositivo, denominado Módulo Sensor, desenvolvido com o objetivo de realizar leituras de aceleração, em função do tempo, para uma estrutura de barragem. Junto com o dispositivo, foi criado um aplicativo de monitoramento remoto, com comunicação por radiofrequência, que possibilita ao usuário visualizar os dados em tempo real, além de fornecer indicações de alerta, com base em limites pré-estabelecidos de variação de aceleração.

5 CONCLUSÃO

Este trabalho teve como objetivo principal a implementação prática e funcional de um sistema de sensoriamento remoto destinado ao monitoramento do risco de ruptura de barragens. A pesquisa e o desenvolvimento desta tecnologia destacam sua importância estratégica para o setor de mineração, contribuindo para a segurança estrutural e a mitigação de riscos ambientais e operacionais.

A implementação do sistema englobou etapas fundamentais, como a coleta e transmissão de dados em tempo real, com medições de vibração realizadas por meio de uma IMU. Esses dados foram transmitidos de maneira eficiente por meio de um *transceiver* RF para um dispositivo receptor, armazenados em um ambiente de nuvem e exibidos em um aplicativo. O aplicativo por sua vez, permitiu uma visualização clara e detalhada do comportamento do solo, traduzindo as leituras do sensor em informações acessíveis e úteis para análise e tomada de decisão. Além disso, o sistema incorporou um mecanismo de alertas configurados de acordo com limites pré-estabelecidos, facilitando a identificação de condições críticas e permitindo ações preventivas rápidas.

O sistema desenvolvido apresenta grande potencial de aplicabilidade prática, podendo ser adaptado para diferentes tipos de barragens e até mesmo para outros setores industriais que demandem monitoramento de vibrações estruturais. A flexibilidade de seu *design* permite sua implementação em áreas como construção civil, monitoramento de pontes ou viadutos, e inspeções em áreas de risco sísmico. A prova de conceito realizada demonstra que, com ajustes específicos, o modelo de sensoriamento remoro pode ser ajustado às necessidades particulares de cada aplicação, garantindo maior precisão e confiabilidade.

Embora o sistema desenvolvido tenha alcançado seus objetivos iniciais, existem diversas possibilidades de aprimoramento e expansão. A substituição da IMU por sensores de maior precisão, capaz de detectar variações ainda mais sutis, é uma das melhorias futuras sugeridas. Além disso, a integração de tecnologias avançadas, como algoritmos de aprendizado de máquina, pode permitir a análise preditiva de dados, identificando padrões que antecedem eventos críticos. A ampliação do monitoramento para outros parâmetros, como pressão ou temperatura, também pode fornecer uma visão mais holística da estabilidade estrutural, tornando o sistema ainda mais robusto.

O desenvolvimento deste sistema não apenas contribui para o avanço tecnológico no monitoramento de barragens, mas também possui um impacto social e ambiental significativo. Ao fornecer uma ferramenta eficaz para identificar possíveis falhas estruturais, o sistema promove a segurança das comunidades próximas a barragens, prevenindo desastres de grande escala. Além disso, ao minimizar os riscos de ruptura, ele auxilia na prevenção do meio ambiente, evitando contaminações e danos irreversíveis a ecossistemas.

Durante a implementação do sistema, foram enfrentados diversos desafios técnicos, como a calibração precisa da IMU e a garantia de transmissão confiável dos dados em tempo real. Superar essas dificuldades resultou em um amadurecimento técnico significativo, demonstrando que o projeto é viável mesmo diante de limitações tecnológicas e de recursos. O aprendizado adquirido durante essas etapas servirá de base para futuros desenvolvimentos e aprimoramentos.

Esse sistema representa um marco importante na busca por soluções acessíveis e eficientes para o monitoramento remoto de estruturas críticas. Sua implementação reforça o compromisso com a segurança, a sustentabilidade e a inovação tecnológica no setor de mineração, evidenciando que soluções tecnológicas simples, porém eficazes, podem fazer uma diferença significativa. O trabalho realizado demonstra que, com visão e dedicação, é possível criar tecnologias que protejam vidas, preservem o meio ambiente e contribuam para o desenvolvimento sustentável.

REFERÊNCIAS

ADAFRUIT INDUSTRIES. ***NXP Precision 9DoF Breakout***. 2017. Disponível em: https://learn.adafruit.com/nxp-precision-9dof-breakout. Acesso em: 22 abr. 2023.

ANATEL. Agência Nacional de Telecomunicações. **Regulação – Radiofrequência.** Disponível em: https://www.gov.br/anatel/pt-br/regulado/radiofrequencia. Acesso em: 03 abr. 2023.

Alexis, Kostas. ***Inertial Sensors***. Disponível em: http://www.kostasalexis.com/inertial-sensors.html&gt. Acesso em: 02 maio 2023.

BEZERRA, Juliana. Desastre de Mariana. **Toda Matéria** [s.d]. Disponível em: https://www.todamateria.com.br/desastre-de-mariana/. Acesso em: 04 abr. 2023.

BRASIL. Congresso Nacional. Projeto de Lei no 12.334 de 2010. **Estabelece a Política Nacional de Segurança de Barragens destinadas à acumulação de água para quaisquer usos, à disposição final ou temporária de rejeitos e à acumulação de resíduos industriais**. Diário Oficial [da] República Federativa do Brasil. Poder Executivo, Brasília, DF. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2007-2010/2010/lei/l12334.htm. Acesso em: 28 abr. 2023.

BRASIL. Congresso Nacional. Projeto de Lei no 14.066 de 2020. **Altera a Lei nº 12.334, de 20 de setembro de 2010, que estabelece a Política Nacional de Segurança de Barragens (PNSB)**. Diário Oficial [da] República Federativa do Brasil. Poder Executivo, Brasília, DF. Disponível em: http://www.planalto.gov.br/ccivil_03/_Ato2019-2022/2020/Lei/L14066.htm. Acesso em: 28 abr. 2023.

CARDOZO, Fernando Alves Cantini; PIMENTA, Matheus Montes; ZINGANO, André Cezar. **Métodos construtivos de barragens de rejeitos de mineração–uma revisão**. **Holos**, v. 8, p. 77-85, 2016. Disponível em: https://www2.ifrn.edu.br/ojs/index.php/HOLOS/article/view/5367/pdf. Acesso em: 03 abr. 2023.

CBDB, COMITÊ BRASILEIRO DE BARRAGENS. **A história das barragens no Brasil, Séculos XIX, XX e XXI: cinquenta anos do Comitê Brasileiro de Barragens**. 2011.

CHANG, Dah-Chung; SHIU, Tsung-Hau. Digital GFSK carrier synchronization. In: **APCCAS 2006-2006 IEEE Asia Pacific Conference on Circuits and Systems**. IEEE, 2006. p. 1523-1526.

GAD-EL-HAK, Mohamed. ***MEMS: design and fabrication***. 2nd ed. Boca Raton: CRC Press 2006.

HAYKIN, S. S.; MOHER, M. ***Communication systems***. Hoboken, Nj: Wiley, 2009.

HUSSAIN, Tassadaq et al. SDDRM: Software Defined Digital Radio Mondiale. In: **2019 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)**. IEEE, 2019. p. 1-5.

IBRAHIM, Dogan. ***Microcontroller based applied digital control***. John Wiley, 2006.

ICOLD. ***The History of the World Register of Dams***. *International Commission on Large Dams.* 2019. Disponível em: https://www.icold-cigb.org/GB/world_register/history.asp. Acesso em: 03 abr. 2023.

KUMARI, R. Shantha Selva; GAYATHRI, C. Interfacing of MEMS motion sensor with FPGA using I2C protocol. In: **2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)**. IEEE, 2017. p. 1-5.

LIU, Cang et al. A flexible hardware architecture for slave device of i2c bus. In: **2019 International Conference on Electronic Engineering and Informatics (EEI)**. IEEE, 2019. p. 309-313.

MALUF, Nadim; WILLIAMS, K. ***An introduction to microelectromechanical systems engineering****,* 2000 *Artech House*. **Inc. Boston, MA**, 2000.
MICROCHIP TECHNOLOGY INC. **ATmega328P**. c2023. Disponível em: https://www.microchip.com/en-us/product/ATmega328P. Acesso em: 02 maio 2023.

MICROCHIP. Atmega328P. Disponível em: https://www.microchip.com/en-us/product/atmega328p. Acesso em: 03 maio 2023.

PAHLEVI, Rizka Reza et al. Fast uart and spi protocol for scalable iot platform. In: **2018 6th International Conference on Information and Communication Technology (ICoICT)**. IEEE, 2018. p. 239-244.

SANTOS, Vanessa Sardinha dos. **Tragédia de Brumadinho**. Escola Kids, [s.d.]. Disponível em: https://escolakids.uol.com.br/ciencias/tragedia-brumadinho.htm. Acesso em: 04 abr. 2023.

SHAEFFER, Derek K. MEMS inertial sensors: A tutorial overview. **IEEE Communications Magazine**, v. 51, n. 4, p. 100-109, 2013.

SILVEIRA, João Francisco Alves. **Instrumentação e segurança de barragens de terra e enrocamento**. Oficina de Textos, 2006.

TRIVEDI, Dvijen et al. SPI to I2C protocol conversion using verilog. In: **2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)**. IEEE, 2018. p. 1-4.

GOOGLE. Firebase Realtime Database. Disponível em: https://firebase.google.com/docs/database?hl=pt-br#cloud-firestore. Acesso em: 02 jan. 2025.

GOOGLE. Firebase Firestore Quotas. Disponível em:< https://firebase.google.com/docs/firestore/quotas?hl=pt-br>. Acesso em: 02 jan. 2025.

M. A. N. A and O. O. Khalifa, "Mobile-Enabled Traffic Monitoring with Alert System for a Better Drive," 2024 9th International Conference on Mechatronics Engineering (ICOM), Kuala Lumpur, Malaysia, 2024, pp. 121-126, doi: 10.1109/ICOM61675.2024.10652502.

FADAEE, Mohsen. Building the foundation for an ai-integrated career coaching application with flutterflow. 2024.

XIAO, Licheng et al. A Comprehensive Mobile Application to Assist the Beginner Snowboarder in Discovering Resources, Aid, Equipment, and Community Support. In: CS & IT Conference Proceedings. CS & IT Conference Proceedings, 2023.

DA SILVA, Lucas Emídio Roque. Calibração dos acelerômetros de uma unidade de medição inercial utilizando um manipulador robótico. Trabalho de Conclusão de Curso em Engenharia de Controle e Automação Universidade Federal de Minas Gerais, 2010.

SAILBOAT INSTRUMENTS. Magneto 1.2: improved magnetometer calibration- part 1. Disponível em: https://sailboatinstruments.blogspot.com/2011/09/improved-magnetometer-calibration-part.html. Acesso em: 20 dez. 2024.

MERCADO LIVRE. Power bank carregador portátil 20000mAh USB backup power PN-959 sem fio. Disponível em: https://www.mercadolivre.com.br/power-bank-carregador-portatil-20000mah-usb-backup-power-pn-959-sem-fio/p/MLB21342303?pdp_filters=item_id%3AMLB3955108083&from=gshop&matt_tool=31485049&matt_word=&matt_source=google&matt_campaign_id=22090354244&matt_ad_group_id=173090558156&matt_match_type=&matt_network=g&matt_device=c&matt_creative=727882729161&matt_keyword=&matt_ad_position=&matt_ad_type=pla&matt_merchant_id=735125422&matt_product_id=MLB21342303-product&matt_product_partition_id=2387641353766&matt_target_id=pla-2387641353766&cq_src=google_ads&cq_cmp=22090354244&cq_net=g&cq_plt=gp&cq_med=pla&gad_source=1&gclid=Cj0KCQiAhbi8BhDIARIsAJLOludmt1cA_DEQS5aKOyQq6eAtbjg-QWulj1mVUSJSstnD5s13ltldTwYaArGAEALw_wcB. Acesso em: 7 jan. 2025

MERCADO LIVRE. Painel solar USB de alta potência 5V células de acampamento. Disponível em: https://www.mercadolivre.com.br/painel-solar-usb-de-alta-potncia-5v-celulas-de-acampamento/p/MLB2000654965?pdp_filters=item_id%3AMLB3740564721&from=gshop&matt_tool=34755848&matt_word=&matt_source=google&matt_campaign_id=22090354298&matt_ad_group_id=173090578036&matt_match_type=&matt_network=g&matt_device=c&matt_creative=727882731279&matt_keyword=&matt_ad_position

=&matt_ad_type=pla&matt_merchant_id=735128761&matt_product_id=MLB2000654965-product&matt_product_partition_id=2419082984284&matt_target_id=pla-2419082984284&cq_src=google_ads&cq_cmp=22090354298&cq_net=g&cq_plt=gp&cq_med=pla&gad_source=1&gclid=Cj0KCQiAhbi8BhDIARIsAJLOlucmliynC_KlfkMykS9aMTW8w1CnbpZVBn11_2Z_dwpvQx-1d6szGFsaAgEZEALw_wcB. Acesso em: 7 jan. 2025.

NXP. **Precision 9-DOF Breakout Board**. Disponível em: https://cdn-learn.adafruit.com/downloads/pdf/nxp-precision-9dof-breakout.pdf. Acesso em: 24 fev. 2025.

APÊNDICE A: Código RF-Nano IMU.ino

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_FXOS8700.h>
#include <RF24.h>

// Configuração do acelerômetro
Adafruit_FXOS8700 accelmag = Adafruit_FXOS8700(0x8700A, 0x8700B);

// Configuração do módulo RF24
RF24 radio(10, 9); // Pinos CE e CSN do módulo RF24
const byte endereco[6] = "01001"; // Endereço RF

void setup() {
  Serial.begin(9600);

  // Inicializa o acelerômetro
  if (!accelmag.begin()) {
    Serial.println("Erro ao iniciar o FXOS8700!");
    while (1);
  }

  // Configuração do rádio
  if (!radio.begin()) {
    Serial.println("Erro ao iniciar o módulo RF!");
    while (1);
  }
  radio.openWritingPipe(endereco);
  radio.setPALevel(RF24_PA_HIGH);
  radio.stopListening(); // Configura como transmissor

  Serial.println("Sistema pronto!");
}

void loop() {
  sensors_event_t aevent, mevent;

  // Obtem os dados do acelerômetro
  accelmag.getEvent(&aevent, &mevent);

  // Prepara os dados para  e remove o valor da gravidade do eixo z

  float dados[3] = {aevent.acceleration.x, aevent.acceleration.y,
aevent.acceleration.z};

  // Envia os dados via RF
  bool sucesso = radio.write(&dados, sizeof(dados));

  if (sucesso) {
    // Envia somente os valores numéricos, separados por vírgulas
    Serial.print(dados[0]); Serial.print(","); // X
    Serial.print(dados[1]); Serial.print(","); // Y
    Serial.println(dados[2]);                  // Z
} else {
    Serial.println("Falha no envio dos dados!");
}
  delay(500); // Aguarda 500ms antes da próxima leitura}
```

APÊNDICE B: Código RF-NanoReceptor.ino

```
#include <RF24.h>

// Configuração do módulo RF24
RF24 radio(10, 9); // Pinos CE e CSN do módulo RF24
const byte endereco[6] = "01001"; // Endereço RF

void setup() {
  Serial.begin(9600);

  // Configuração do rádio
  if (!radio.begin()) {
    Serial.println("Erro ao iniciar o módulo RF!");
    while (1);
  }
  radio.openReadingPipe(0, endereco);
  radio.setPALevel(RF24_PA_HIGH);
  radio.startListening(); // Configura como receptor

  Serial.println("Receptor pronto!");
}

void loop() {
  if (radio.available()) {
    float dados[3]; // Array para armazenar os dados recebidos
    radio.read(&dados, sizeof(dados));

    // Remove a gravidade do eixo Z
    float gravidade = 9.81; // Valor da gravidade em m/s^2
    dados[2] -= gravidade;  // Subtrai a gravidade do valor do eixo Z

    // Envia os dados recebidos para o computador via serial
    Serial.print(dados[0]); Serial.print(","); // X
    Serial.print(dados[1]); Serial.print(","); // Y
    Serial.println(dados[2]);                  // Z (corrigido)
  }
}
```

APÊNDICE C: Código IMU-calibracao.ino

```
#include <Adafruit_FXOS8700.h>

// Inicialização do objeto do sensor
Adafruit_FXOS8700 accelmag = Adafruit_FXOS8700(0x8700A, 0x8700B);

// Função para exibir detalhes do sensor
void displaySensorDetails()
{
  sensor_t accel;
  accelmag.getSensor(&accel);
  delay(500);
}
// Variáveis globais
uint32_t delayPeriod = 20; // Delay para a medição [ms]

void setup()
{
  Serial.begin(115200);
  delay(100);

  // Inicialização do sensor
  if (!accelmag.begin())
  {
    Serial.println("[ERROR]: FXOS8700 não iniciou");
    while (1);
  }

  // Configuração do modo de sensor (somente acelerômetro)
  accelmag.setSensorMode(ACCEL_ONLY_MODE);

  // Exibe detalhes do sensor
  displaySensorDetails();
}

void loop()
{
  sensors_event_t accelEvent;

  if (!accelmag.getEvent(&accelEvent))
  {
    Serial.println("[ERROR]: Erro na leitura do sensor");
    return;
  }

  // Obtém um novo evento do sensor
  accelmag.getEvent(&accelEvent);
  // Fator de conversão de m/s^2 para G
  const float CONVERSION_FACTOR = 1.0 / 9.80665;

  // Envie as leituras dos eixos pela porta serial
  Serial.print(accelEvent.acceleration.x * CONVERSION_FACTOR, 6); // Leitura
do eixo X
  Serial.print(",");
  Serial.print(accelEvent.acceleration.y * CONVERSION_FACTOR, 6); // Leitura
do eixo Y
  Serial.print(",");
```

```
  Serial.println(accelEvent.acceleration.z * CONVERSION_FACTOR, 6); // Leitura
do eixo Z
  Serial.print(",");

  // Aguarda o período de delay
  delay(delayPeriod);
}
```

APÊNDICE D: Código IntegracaoPFAlertas.py

```
import serial
import time
import json
import numpy as np
from firebase_admin import credentials, firestore, initialize_app #Biblioteca
que permite interação com Firebase

# Configurações da porta serial
PORTA_SERIAL = 'COM4'  # Porta serial
BAUD_RATE = 9600
TIMEOUT = 1            # Timeout para leitura da porta serial (em segundos)

# Inicialização do Firebase
CAMINHO_CREDENCIAIS = 'C:/Users/caminho.json'
cred = credentials.Certificate(CAMINHO_CREDENCIAIS) # Autenticação de
credencial da chave
initialize_app(cred) # Incializa o Firebase no programa Python
db = firestore.client() # Cria um cliente para acessar o banco de dados
Firestore

# Nome da coleção no Firebase
NOME_COLECAO = "dados_acelerometro"
NOME_COLECAO_ALERTAS = "alertas_acelerometro"

# Parâmetros de calibração ajustados
A = np.array([[0.997670, -0.001004, 0.000187],  # Matriz de correção
              [-0.001004, 1.024630, -0.001222],
              [0.000187, -0.001222, 0.998932]])
b = np.array([-0.087137, -0.094403, -0.082786])  # Vetor de correção de bias

# Limites de alerta para vibração
LIMITES_ALERTA = {
    "baixo": 1,
    "medio": 2,
    "alto": 3
}

# Controle de envio de alertas
alertas_enviados = {"baixo": False, "medio": False, "alto": False}

# Controle do estado inicial "Estável"
estado_estavel_enviado = False  # Garantir que o estado estável seja enviado
apenas uma vez

# Configuração de controle de envio
ENVIO_INTERVALO = 10  # Intervalo em segundos
ultimo_envio = time.time()  # Inicializa o tempo do último envio

def enviar_estado_estavel():
    """
    Envia o estado inicial "Estável" ao Firebase.
    """
    global estado_estavel_enviado
    if not estado_estavel_enviado:
        estado_estavel = {
            "nivel": "estavel",
```

```
            "timestamp": firestore.SERVER_TIMESTAMP
        }
        try:
            db.collection(NOME_COLECAO_ALERTAS).add(estado_estavel)
            estado_estavel_enviado = True
            print("Estado inicial 'Estável' enviado ao Firebase:",
estado_estavel)
        except Exception as e:
            print(f"Erro ao enviar o estado 'Estável': {e}")

def aplicar_calibracao(dados):
    """
    Aplica a calibração nos dados do acelerômetro.
    """
    dados = np.array(dados)
    calib_data = (A @ dados) + b
    return calib_data

def verificar_alertas(dados_calibrados):
    """
    Verifica se os dados excedem os limites definidos e envia alertas ao
Firebase.
    """
    global alertas_enviados
    for nivel, limite in LIMITES_ALERTA.items():
        if any(abs(dado) > limite for dado in dados_calibrados):
            if not alertas_enviados[nivel]:
                alerta = {
                    "x": float(dados_calibrados[0]),
                    "y": float(dados_calibrados[1]),
                    "z": float(dados_calibrados[2]),
                    "nivel": nivel,
                    "timestamp": firestore.SERVER_TIMESTAMP
                }
                try:
                    db.collection(NOME_COLECAO_ALERTAS).add(alerta)
                    alertas_enviados[nivel] = True
                    print(f"Alerta de nível '{nivel}' enviado ao Firebase:",
alerta)
                   # Envia os mesmos dados para a coleção dados_acelerometro
                    dados = {
                        "x": float(dados_calibrados[0]),
                        "y": float(dados_calibrados[1]),
                        "z": float(dados_calibrados[2]),
                        "timestamp": firestore.SERVER_TIMESTAMP
                    }
                    db.collection(NOME_COLECAO).add(dados)
                    print("Dados do alerta também enviados ao Firebase na
coleção dados_acelerometro:", dados)
                except Exception as e:
                    print(f"Erro ao enviar alerta para o Firebase: {e}")
        else:
            alertas_enviados[nivel] = False

try:
    # Inicializa a conexão serial
```

```
    arduino = serial.Serial(PORTA_SERIAL, BAUD_RATE, timeout=TIMEOUT)
    print(f"Conectado à porta serial {PORTA_SERIAL}")

    # Aguarda o Arduino estar pronto
    time.sleep(2)

    # Envia o estado inicial "Estável" antes de iniciar o loop principal
    enviar_estado_estavel()

    while True:
        # Lê a linha de dados do Arduino
        linha = arduino.readline().decode('utf-8').strip()
        if linha:
            try:
                x, y, z = map(float, linha.split(','))
                dados_recebidos = [x, y, z]

                # Aplica a calibração
                dados_calibrados = aplicar_calibracao(dados_recebidos)
                print(f"Dados após calibração: {dados_calibrados}")

                # Verifica alertas
                verificar_alertas(dados_calibrados)

                # Verifica se é o momento de enviar os dados ao Firebase
                tempo_atual = time.time()
                if tempo_atual - ultimo_envio >= ENVIO_INTERVALO:
                    dados = {
                        "x": float(dados_calibrados[0]),
                        "y": float(dados_calibrados[1]),
                        "z": float(dados_calibrados[2]),
                        "timestamp": firestore.SERVER_TIMESTAMP
                    }
                    db.collection(NOME_COLECAO).add(dados)
                    print("Dados calibrados enviados ao Firebase:", dados)
                    ultimo_envio = tempo_atual
            except ValueError:
                print("Erro no formato dos dados recebidos. Certifique-se de
enviar valores no formato correto.")
except serial.SerialException as e:
    print(f"Erro ao acessar a porta serial: {e}")
except KeyboardInterrupt:
    print("Encerrando leitura...")
finally:
    if 'arduino' in locals():
        arduino.close()
```

ANEXO A: Código record-data.py

```
"""
RECORD ACCELEROMETER MEASUREMENTS FOR ACCELEROMETER CALIBRATION
VIA MAGNETO

Code By: Michael Wrona
Created: 17 Aug 2021

"""

import os
import math
import pandas
import serial


# global variables
MAX_MEAS = 200  # max number of readings in the session, so that we don't
create an infinite loop
AVG_MEAS = 25  # for each reading, take this many measurements and average
them
SER_PORT = 'COM4'  # serial port the device is connected to
SER_BAUD = 115200  # serial port baud rate
FILENAME = os.path.join(os.getcwd(), 'acceldata.txt')  # output file


class SerialPort:
    """Create and read data from a serial port.

    Attributes:
        read(**kwargs): Read and decode data string from serial port.
    """

    def __init__(self, port, baud=9600):
        """Create and read serial data.

        Args:
            port (str): Serial port name. Example: 'COM4'
            baud (int): Serial baud rate, default 9600.
        """
        if isinstance(port, str) == False:
            raise TypeError('port must be a string.')

        if isinstance(baud, int) == False:
            raise TypeError('Baud rate must be an integer.')

        self.port = port
        self.baud = baud

        # Initialize serial connection
        self.ser = serial.Serial(
            self.port, self.baud, timeout=None, xonxoff=False, rtscts=False,
dsrdtr=False)
        self.ser.flushInput()
        self.ser.flushOutput()

    def Read(self, clean_end=True) -> str:
```

```
        """
        Read and decode data string from serial port.

        Args:
            clean_end (bool): Strip '\\r' and '\\n' characters from string.
Common if used Serial.println() Arduino function. Default true

        Returns:
            (str): utf-8 decoded message.
        """
        self.ser.flushInput()
        bytesToRead = self.ser.readline()
        decodedMsg = bytesToRead.decode('utf-8')

        if clean_end == True:
            decodedMsg = decodedMsg.rstrip()  # Strip extra chars at the end

        return decodedMsg

    def Close(self) -> None:
        """Close serial connection."""
        self.ser.close()


def RecordDataPt(ser: SerialPort) -> tuple:
    """Record data from serial port and return averaged result."""
    # do a few readings and average the result
    ax = ay = az = 0.0

    for _ in range(AVG_MEAS):
        # read data
        try:
            data = ser.Read().split(',')
            ax_now = float(data[0])
            ay_now = float(data[1])
            az_now = float(data[2])
        except:
            ser.Close()
            raise SystemExit("[ERROR]: Error reading serial connection.")
        ax += ax_now
        ay += ay_now
        az += az_now

    return (ax / AVG_MEAS, ay / AVG_MEAS, az / AVG_MEAS)


def List2DelimFile(mylist: list, filename: str, delimiter: str = ',',
f_mode='a') -> None:
    """Convert list to Pandas dataframe, then save as a text file."""
    df = pandas.DataFrame(mylist)
    df.to_csv(
        filename,  # path and filename
        sep=delimiter,
        mode=f_mode,
        header=False,  # no col. labels
        index=False  # no row numbers
```

```
    )


def main():
    ser = SerialPort(SER_PORT, baud=SER_BAUD)
    data = []  # data list

    print('[INFO]: Place sensor level and stationary on desk.')
    input('[INPUT]: Press any key to continue...')

    # take measurements
    for _ in range(MAX_MEAS):
        user = input(
            '[INPUT]: Ready for measurement? Type \'m\' to measure or \'q\' to
save and quit: ').lower()
        if user == 'm':
            # record data to list
            ax, ay, az = RecordDataPt(ser)
            magn = math.sqrt(ax**2 + ay**2 + az**2)
            print('[INFO]: Avgd Readings: {:.4f}, {:.4f}, {:.4f} Magnitude:
{:.4f}'.format(
                ax, ay, az, magn))
            data.append([ax, ay, az])
        elif user == 'q':
            # save, then quit
            print('[INFO]: Saving data and exiting...')
            List2DelimFile(data, FILENAME, delimiter='\t')
            ser.Close()
            print('[INFO]: Done!')
            return
        else:
            print('[ERROR]: \'{}\' is an unknown input.
Terminating!'.format(user))
            List2DelimFile(data, FILENAME, delimiter='\t')
            ser.Close()
            return

    # save once max is reached
    print('[WARNING]: Reached max. number of datapoints, saving file...')
    List2DelimFile(data, FILENAME, delimiter='\t')
    ser.Close()
    print('[INFO]: Done!')


if __name__ == '__main__':
    main()
```