UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
CURSO DE GRADUAÇÃO EM ENGENHARIA CIVIL

CÁSSIO GALVÃO DOS SANTOS VICENTE
JOÃO PEDRO FRANÇA E CARVALHO

SISTEMA DE BAIXO CUSTO PARA MONITORAMENTO DINÂMICO DE ESTRUTURAS

Recife
2024

CÁSSIO GALVÃO DOS SANTOS VICENTE
JOÃO PEDRO FRANÇA E CARVALHO

SISTEMA DE BAIXO CUSTO PARA MONITORAMENTO DINÂMICO DE ESTRUTURAS

Monografia apresentada à Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Engenharia Civil. Área de concentração: Estruturas.

Orientador (a): Prof. Dr. Paulo Marcelo Vieira Ribeiro

Recife
2024

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Vicente, Cássio Galvão dos Santos.
Sistema de Baixo Custo para Monitoramento Dinâmico de Estruturas / Cássio Galvão dos Santos Vicente, João Pedro França e Carvalho. - Recife, 2024.
145

Orientador(a): Paulo Marcelo Vieira Ribeiro
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Engenharia Civil - Bacharelado, 2024.

1. Monitoramento. 2. Avaliação Estrutural. 3. Vibrações. 4. Instrumentação. I. Carvalho, João Pedro França e. II. Ribeiro, Paulo Marcelo Vieira . (Orientação). III. Título.

620 CDD (22.ed.)

CÁSSIO GALVÃO DOS SANTOS VICENTE
JOÃO PEDRO FRANÇA E CARVALHO

SISTEMA DE BAIXO CUSTO PARA MONITORAMENTO DINÂMICO DE ESTRUTURAS

Monografia apresentada à Universidade Federal de Pernambuco, como requisito parcial para obtenção do grau de Bacharel em Engenharia Civil. Área de concentração: Estruturas.

Aprovado em: 17/04/2024

BANCA EXAMINADORA

_____
Prof. Dr. Paulo Marcelo Vieira Ribeiro (Orientador)
Universidade Federal de Pernambuco - UFPE

_____
Prof. Ms. Geraldo Leite Maia Junior (Examinador Interno)
Universidade Federal de Pernambuco - UFPE

_____
Dr. Carlos Vitor da Silva Sarmento (Examinador Externo)
Universidade Federal de Pernambuco - UFPE

RESUMO

As infraestruturas no Brasil sofrem com a falta de manutenção, o DNIT (Departamento Nacional de Infraestrutura de Transportes), afirma por meio de pesquisa realizada para o gerenciamento de obras de artes especiais que em torno de 76% das obras cadastradas estão em estado potencialmente crítico. Isso ocorre principalmente devido à grande parte das pontes rodoviárias brasileiras são da década 60 e há anos são negligenciadas em sua manutenção. Tais ativos são de caráter estratégicos para a sociedade, onde seu não funcionamento ou colapso pode causar perdas financeiras, sociais e econômicas. Portanto, faz-se necessário utilizar metodologias para avaliar a integridade estrutural de obras de arte especiais, a fim de prever planos de manutenção e realizar intervenções quando necessário. O Monitoramento de Integridade Estrutural surge como uma forma de avaliar a saúde estrutural de um elemento. No entanto, sua implementação atual implica em altos custos, tornando necessário o desenvolvimento de sistemas de baixo custo de implementação e operação para viabilizar essa metodologia. Assim, sob esse viés da avaliação de integridade estrutural, este trabalho busca desenvolver e implementar um sistema de baixo custo baseado em acelerômetros de sistemas micro eletromecânicos (MEMS). Dentro do fluxo de desenvolvimento, este trabalho buscou entender as limitações dos materiais utilizados, validando o funcionamento dos sensores na análise dinâmica a partir dos ensaios realizados em laboratório e foi desenvolvido um sistema de aquisição e tratamento de dados com a finalidade de ser implantado numa estrutura real. O equipamento proposto de medição utiliza dois acelerômetros MPU-6050 conectados a uma ESP32, um microcontrolador responsável por receber as medições e enviá-las para o servidor local, que, neste caso, utiliza-se de uma Raspberry Pi 4. Os dados da medição são recebidos via comunicação TCP/IP, onde são armazenados e processados para análise dinâmica da estrutura avaliada. Esta solução atendeu ao quesito da economicidade do sistema, viabilizando a construção de um sistema de monitoramento de integridade estrutural de baixo custo. Foram realizados ensaios em laboratório com modelos estruturais que apresentaram uma diferença relativa em relação ao modelo teórico de até 15%, enquanto as medições realizadas na ponte foram coerentes com outro equipamento avaliado, favorecendo a aplicação do sistema desenvolvido em estruturas reais.

Palavras-chave: *Monitoramento; Avaliação Estrutural; Vibrações; Instrumentação.*

ABSTRACT

Infrastructure in Brazil suffers from a lack of maintenance. The DNIT (Departamento Nacional de Infraestrutura de Transporte), through a survey carried out for the management of special bridges, states that around 76% of the bridges registered are in a potentially critical state. This is mainly because most of Brazil's highway bridges date back to the 1960s and have been neglected for years. These assets are of a strategic nature for society, where their non-functioning or collapse can cause financial, social and economic losses. It is therefore necessary to use methodologies to assess the structural health of special engineering structures in order to plan maintenance and carry out interventions when necessary. Structural Health Monitoring (SHM) has emerged as a way of assessing the structural health of an element. However, its current implementation entails high costs, making it necessary to develop low-cost implementation and operation systems to make this methodology viable. With this in mind, this work aims to develop and implement a low-cost system based on microelectromechanical systems (MEMS) accelerometers. Within the development flow, this work sought to understand the limitations of the materials used for the desired purpose, validate the functioning of the sensors in the dynamic analysis and develop a data acquisition and processing system as a prototype for future implementation in real structures. The proposed measurement equipment uses two MPU-6050 accelerometers connected to an ESP32, a microcontroller responsible for receiving the measurements and sending them to the local server, which in this case uses a Raspberry Pi 4. The measurement data is received via TCP/IP communication, where it is stored and processed for dynamic analysis of the evaluated structure. This solution met the system's cost-effectiveness criterion, enabling the construction of a low-cost structural health monitoring system. Laboratory tests were carried out with structural models that showed a relative difference of up to 15% compared to the theoretical model, while measurements taken on the bridge were consistent with another evaluated equipment, favoring the application of the developed system in real structures.

Keywords: *Monitoring; Structural Assessment; Vibrations; Instrumentation.*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Existe uma tendência dentro da construção civil de unir esforços na gestão dos empreendimentos ao longo de sua vida útil, desde a concepção do projeto até o encerramento das atividades, devido ao alto valor investido. Lopes (1998, p. 3) afirma que "embora interpretada como uma atividade secundária, a manutenção é essencial e inevitável, repercutindo de forma direta na vida útil da edificação e em seu desempenho global."

A falta de manutenção nos ativos da construção civil é especialmente evidente nas obras de infraestrutura, não apenas em ações práticas, mas também na falta de capacidade técnica para realizar análises mais aprofundadas. No Brasil, um exemplo evidente é a carência de conhecimento técnico-científico aprofundado na avaliação das condições de estabilidade estrutural de pontes rodoviárias existentes. Geralmente, essas avaliações são conduzidas por meio de inspeção visual somente quando as patologias se tornam aparentes, o que sempre requer a experiência e o conhecimento específico dos fiscais (Vitório, 2013).

O Departamento Nacional de Infraestrutura de Transportes, por meio do Instituto de Pesquisa em Rodovias demonstra no Sistema de Gerência de Obras de Artes Especiais (SGO), que em torno de 76% das obras cadastradas encontram-se em estado potencialmente problemático e crítico (Silva, 2021).

Dentre as possibilidades de avaliação dos elementos estruturais, visando ser mais eficiente na previsão da manutenção, tem-se a opção tradicional de vistorias e inspeções visuais, visando obter informações sobre patologias e problemas observáveis no ativo. Também é possível determinar a saúde da estrutura avaliada a partir do Monitoramento da Integridade Estrutural (MIE) que busca avaliar a estrutura verificando as condições de operação. A campanha de monitoramento tem como finalidade compreender e validar o projeto com base nos resultados aferidos (Andrade, 2012).

Essa abordagem engloba sistemas de aquisição de dados que coletam informações em tempo real sobre o comportamento estrutural, permitindo identificar deformações, deslocamentos, vibrações, entre outros parâmetros. Sistemas de monitoramento de integridade estrutural são pouco utilizados, isso se dá pela complexidade e alto custo empregados na instalação de diversos sensores e interpretação de uma grande quantidade de dados (Lynch, 2014). Com os avanços tecnológicos, como o desenvolvimento de sensores de baixo custo e utilização de equipamentos IoT (*Internet of Things)*, tornou-se possível criar sistemas de monitoramento de integridade estrutural mais acessíveis.

1.1. JUSTIFICAÇÃO E MOTIVAÇÃO

A preservação da integridade de obras civis é essencial para garantir seu funcionamento, evitar possíveis danos e prejuízos. No entanto, a ocorrência de falhas estruturais pode representar sérios riscos a população, podendo causar mortes, além de perdas financeiras e a necessidade de intervenções corretivas, inviabilizando a operação do ativo. Portanto, é importante implementar medidas eficazes de monitoramento para identificar precocemente qualquer sinal de deterioração ou comprometimento das estruturas.

Atualmente, os métodos de detecção de danos muitas vezes dependem de inspeções visuais ou procedimentos experimentais locais, apresentando limitações em termos de acessibilidade e abrangência. Nesse sentido, o Monitoramento da Integridade Estrutural (MIE) surge como uma abordagem promissora, permitindo uma análise contínua e remota do estado das estruturas.

Assim, este trabalho propõe o desenvolvimento de sistema para monitoramento dinâmico de estruturas, visando fornecer uma solução econômica, prática e eficaz para a análise de estruturas em situação de carregamento dinâmico. Espera-se que essa iniciativa contribua para reduzir os custos na implementação de sistemas de monitorização das estruturas civis, possibilitando acompanhamento e previsibilidade da sua manutenção.

1.2. OBJETIVOS GERAIS E ESPECÍFICOS

1.2.1. Objetivo Geral:

Desenvolver sistema de baixo custo para monitoramento dinâmico de estruturas.

1.2.2. Objetivos Específicos:

Desenvolver equipamento com acelerômetros de baixo custo que sejam acessíveis e com a precisão adequada para análise dinâmica.

Desenvolvimento de plataforma de coleta e tratamento de dados em tempo real.

Validação dos sensores a partir de modelos estruturais reduzidos, com a finalidade de obter seus parâmetros dinâmicos.

Avaliar a eficácia da solução proposta em termos de custo, facilidade de implementação e capacidade de identificação precisa de problemas em comparação com abordagens convencionais.

2. MONITORAMENTO DE INTEGRIDADE ESTRUTURAL

A degradação da estrutura ao longo de sua operação pode colocar em risco seu funcionamento, gerando insegurança para os usuários. Fiscalizações e monitoramentos da estrutura avaliada podem auxiliar na tomada de decisão do gestor, permitindo, após análise, liberar ou interromper seu uso, propor intervenções de reforço estrutural ou até mesmo condenar a estrutura atual (Andrade, 2012). Essas ações podem ser realizadas através da inspeção visual, uma metodologia mais usual e antiga para a detecção de falhas. No entanto, vale ressaltar que a qualidade de sua aferição depende diretamente da experiência, da perspectiva de quem a executa e do espaço temporal que se é avaliado (Rytter, 1993). Adicionalmente, pode-se avaliar o comportamento da estrutura a partir de instrumentação, que naturalmente retratam o cenário naquele instante de medição. O MIE emerge como solução complementar na análise de detecção de danos, utilizando-se de monitoramento prolongado, com grande quantidade de sensores, processamento e transmissão remota dos dados, possibilitando maior espaço amostral no diagnóstico de falhas estruturais (Borges, 2021).

Figura 1 - Exemplo de monitoramento de estrutura.

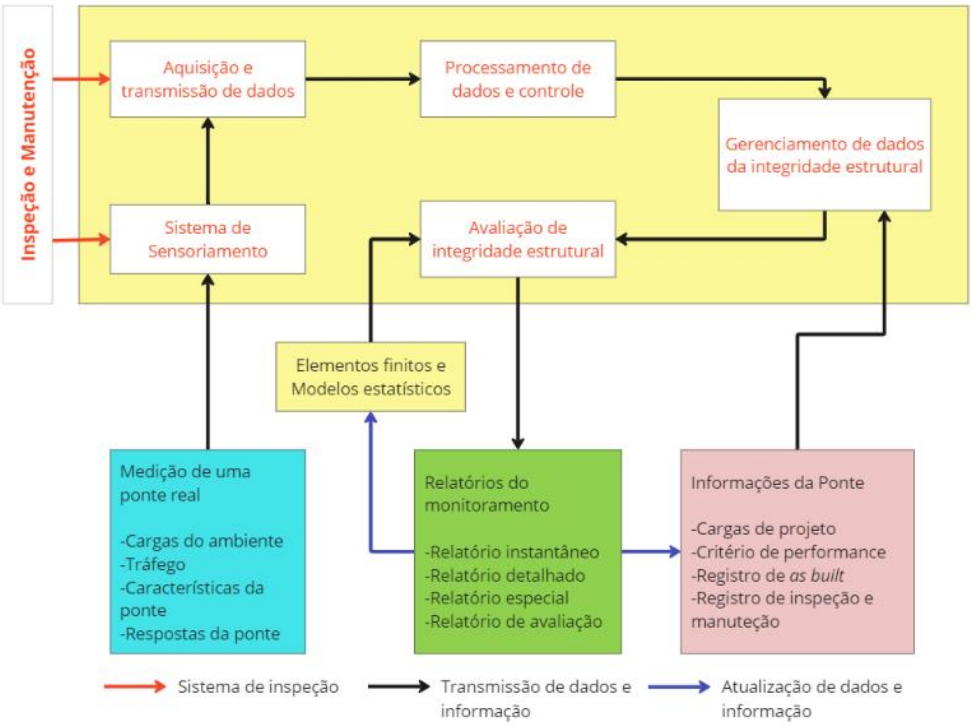
Fonte: Assis (2007).

A metodologia de monitoramento propõe avaliar a estrutura, de acordo com a fase de operação, sua resposta estrutural aos carregamentos, a fim de atestar seu estado atual e identificar possíveis falhas, conforma esquema de medição da ponte na Figura 1.

São diversas as aplicações, objetivos e variáveis a serem monitoradas pelo uso do MIE. Parâmetros como deformação, tensão, aceleração, temperatura, umidade, carregamento e deslocamento são opções relevantes, mas a escolha varia de acordo com as características da estrutura avaliada e as análises desejadas (Assis, 2007). As grandezas observadas devem abranger os parâmetros relevantes para a análise de falhas. Uma opção para atestar a saúde das estruturas seria a partir da resposta vibracional, que possibilita compreender a variação das suas propriedades dinâmicas, como massa e rigidez, em função dos carregamentos cíclicos ao longo do tempo (Kaminski, 1997). A resposta dinâmica pode ser medida por uma série de acelerômetros, analisada no domínio do tempo e da frequência, a fim de avaliar suas propriedades mecânicas, parâmetros como frequências naturais e modos de vibração direcionam a análise na detecção de falhas estruturais (Comisu, 2017).

A utilização de monitoramento contínuo da estrutura surge com a finalidade de validar os resultados dos modelos analíticos e numéricos (Andrade, 2012). A Figura 2 exemplifica a utilização de MIE em pontes.

Figura 2 - Exemplo de sistema de MIE



Fonte: Adaptado Wong (2007).

Pode-se apontar alguns fatores que justificam a implantação de um sistema de monitoramento estrutural, entre eles o avanço tecnológico, que leva as estruturas a incorporarem novos materiais, sendo necessário validar a estrutura projetada com o modelo analítico, calibrando as proposições de projeto. Outra oportunidade gerada é de aperfeiçoamento dos modelos teóricos e melhorias nas técnicas construtivas (Assis, 2007).

A identificação de danos em estruturas é uma justificativa plausível para a implementação de um sistema de monitoramento. Segundo Rytter (1997), um defeito em algum local da estrutura pode levar à alteração das características dinâmicas gerais. Vale salientar que apenas defeitos de determinada magnitude serão capazes de alterar de forma significativa suas propriedades. Para implantação do monitoramento é necessário dimensionar o sistema de obtenção e tratamento dos dados.

Para realizar as análises referente a medição das estruturas, é possível avaliar em função de sua operação ou excitação natural, a partir da Análise Operacional Modal (*OMA – Operacional Modal Analysis*). Sua principal característica é avaliar os parâmetros da estrutura em função da resposta vibracional (Barros, 2016).

As técnicas de OMA são de grande interesse para a avaliação de estruturas em operação, pois a metodologia não interfere em seu funcionamento e não há necessidade de equipamentos especiais para gerar uma excitação forçada. Nos casos de estruturas da Engenharia Civil, tais circunstâncias são bastante importantes, pois permitem que a avaliação das características dinâmicas das estruturas seja feita com custos significativamente reduzidos (Rodrigues, 2004).

Dentro da metodologia do OMA, existem duas vertentes de análise do sinal medido, seja pela avaliação no domínio da frequência ou do tempo. Os métodos no domínio da frequência são os mais antigos e tiveram desenvolvimento mais acelerado, principalmente com a popularização da Transformada Rápida de Fourier (FFT – Fast Fourier Transform) (Barros, 2016). Os métodos que utilizam da análise no domínio da frequência podem ser divididos em dois grupos, método de seleção de picos e método de decomposição do domínio da frequência. O método de seleção de picos (PP – Peack Picking) é baseado na construção da média normalizada de densidade espectral de potência, permitindo identificar os picos referentes a estimativa da frequência naturais da estrutura. A técnica da Decomposição no Domínio da Frequência foi desenvolvida com base na decomposição em valores singulares (SVD - Singular Value Decomposition). A matriz de densidade espectral é composta por funções autoespectrais, onde sua diagonal principal é composta pelos autovalores, representando o quadrado das

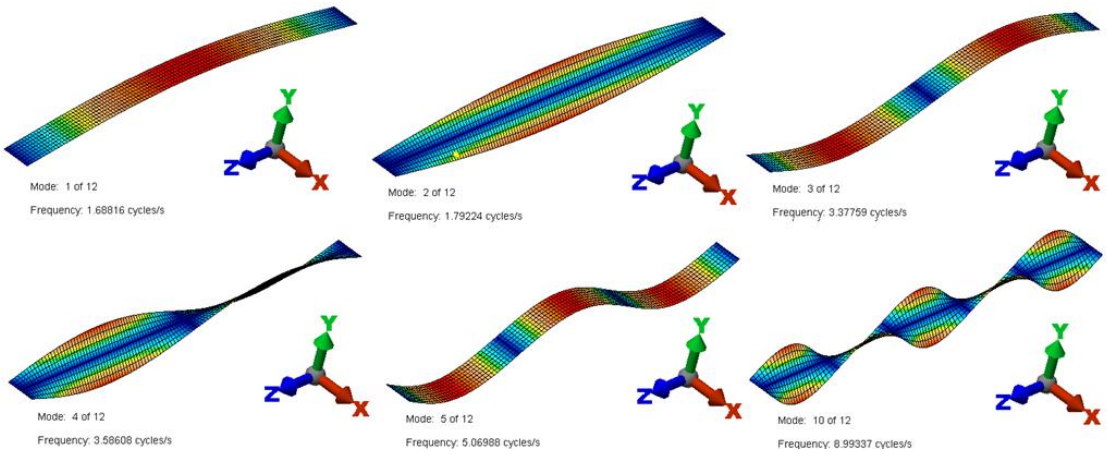
frequências naturais obtidas, e a matriz é preenchida pelos autovetores que representam a forma modal de cada frequência natural obtida (Barros, 2016).

Através dos métodos de análise é possível obter alguns parâmetros relevantes para análise de integridade estrutura. Segundo Rytter (1993, p. 65), “o uso de um Indicadores de Dano (Damage Indicators - DI) fornece principalmente uma indicação qualitativa da existência de danos.” Os DI’s em geral são características da resposta dinâmica da estrutura ao evento avaliado. Exemplos de tais indicadores incluem frequência natural e as formas modais.

As frequências naturais são funções de autovalores, geometria e material do elemento avaliado. São consideradas indicadores de dano clássico, pois seus valores dependem diretamente da rigidez e distribuição de massa da estrutura. Sua popularidade se dá pela facilidade e precisão na medição, onde sua variação pode apontar problemas diretamente relacionados com a vida útil da estrutura. Outro ponto importante desse parâmetro ocorre quando uma frequência natural que não diminui quando a outra o faz, sendo um indicador valioso de dano, pois indica que o dano está localizado em um ponto zero para a curvatura daquela forma modal (Rytter, 1993).

As formas modais (ϕ) são os padrões de vibração da estrutura em suas frequências naturais. Oferecem informações detalhadas sobre a distribuição de massa e rigidez ao longo da estrutura. Mudanças nas formas modais podem evidenciar áreas específicas onde ocorreram modificações na massa ou rigidez, auxiliando na identificação de danos locais, como mostra na Figura 3.

Figura 3 - Exemplo de forma modal



Fonte: John Evans (2013).

Esses métodos exploram as variações nas características dinâmicas da estrutura, representadas pelos parâmetros mencionados, para detectar potenciais falhas ou danos. Quando danos ocorrem na estrutura, eles impactam esses parâmetros, tornando-os sensíveis às mudanças nas propriedades mecânicas, como massa, rigidez e amortecimento. Portanto, esses indicativos são necessários para avaliar a integridade estrutural e para tomar medidas corretivas quando necessário.

3. SISTEMA DE MONITORAMENTO

Os sistemas de monitoramento, dentre a grande diversidade de opções, são constituídos por componentes integrados que são responsáveis pela medição, aquisição, envio, tratamento e interpretação dos resultados. Tais componentes apresentam subsistemas que completam o sistema de monitoramento, conforme o exemplo de sistema da Figura 4.

Figura 4 - Subsistemas de um sistema de monitoramento

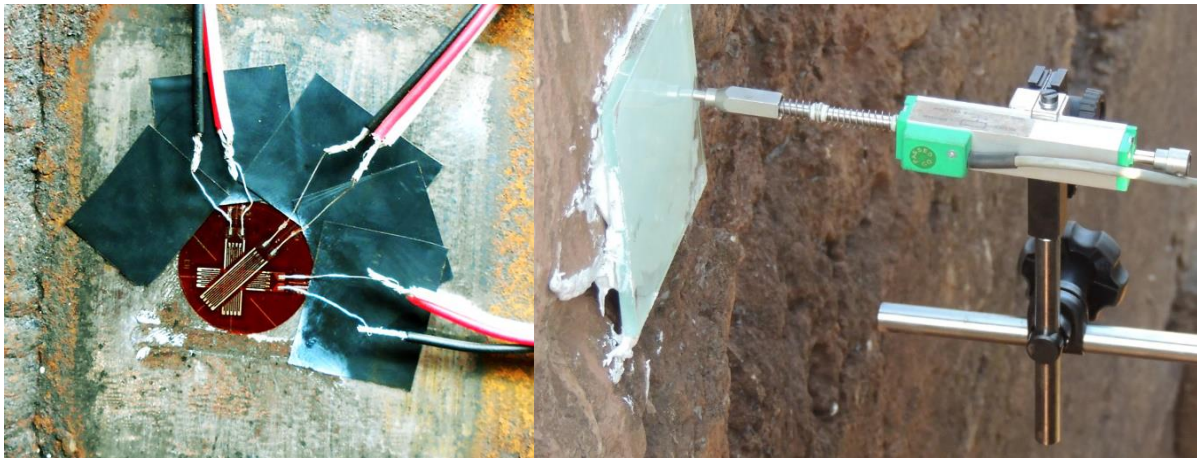
Fonte: Assis (2007).

3.1. REDES DE SENSORES

Os sensores são equipamentos que registram mudanças de comportamento ou propriedades quando estão expostos à influência de uma ação física ou química, podendo gerar um sinal que representa a dimensão dessa grandeza, seja de forma direta ou indireta (Regazzi, 2005).

No mercado, encontram-se vários tipos de sensores disponíveis para medições de diferentes grandezas, dentre elas temperatura, umidade, deformação, deslocamento, rotação e aceleração. Além dos sensores elétricos convencionais, os sensores de fibra óptica são amplamente utilizados no exterior para essas medições.

Figura 5 - Extensômetro e transdutor de deslocamentos



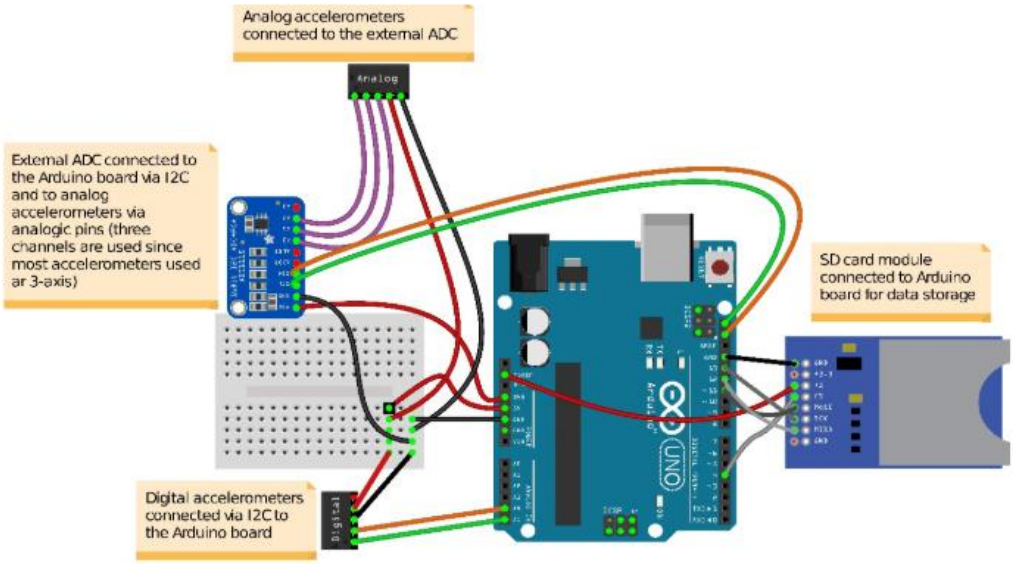
Fonte: Dynamis Techne (2015).

No decorrer dos anos, os sensores evoluíram ao passo do desenvolvimento dos circuitos integrados, principalmente nas décadas de 60 e 80. Como resultado dessa revolução, surgiram os dispositivos MEMS (Micro Electro Mechanical Systems – Sistemas Micro-Eletromecânicos), que funcionam a partir da integração de componentes mecânicos e elétricos em escala micro. Um exemplo desses dispositivos são os acelerômetros MEMS, que apresentam diversas versões, com a mais comum realizando as medições a partir da variação da capacitância medida em função de um deslocamento em x (Borges, 2021).

Devido ao alto custo na implantação dos sistemas clássicos de monitoramento, existem algumas alternativas desenvolvidas a partir da utilização de microcontroladores e sensores de baixo consumo energético como alternativa de sistema de obtenção de dados para o monitoramento de integridade estrutural.

Ribeiro (2019), em seu trabalho de estudo comparativo de acelerômetros, utiliza de sistema de medição conjunto sensor + Arduíno.

Figura 6 - Sistema de aquisição de dados com Arduino



Fonte: Ribeiro (2019).

Enquanto Borges (2021), propôs sistema de monitoramento de baixo custo utilizando conjunto ESP32 + MPU + Raspberry Pi. Portanto existe a oportunidade de criar sistemas de baixo custo para o monitoramento de integridade estrutural.

Figura 7 - Sistema de medição utilizando ESP32 + Raspberry Pi

Fonte: Borges (2021).

3.2. AQUISIÇÃO DE DADOS

O sistema de aquisição de dados tem por objetivo capturar as informações das grandezas medidas no objeto de estudo, a fim de obter seus valores numéricos e possibilitar manipulações posteriores. Dentro do sistema de monitoramento estrutural, outra parte responsável pelo subsistema de aquisição de dados é transformar os sinais elétricos gerados pelos sensores em valores absolutos das grandezas físicas, por muitas vezes esses valores são retornados de forma indireta. Os dados traduzidos em valores absolutos, podem ser armazenados e analisados posteriormente em um computador.

Segundo Assis (2007), tradicionalmente um sistema de aquisição de dados é constituído por três componentes:

- condicionador de sinais;
- conversor analógico-digital (conversor A/D);
- programa de aquisição de dados.

Os condicionadores de sinais são vários circuitos eletrônicos que adequam os sinais analógicos, normalmente sinais elétricos, para sinal digital. Os subcomponentes dos condicionadores são os filtros, amplificadores e isoladores (Lynx, 2006).

Os conversores por sua vez têm função de traduzir os dados analógicos para digitais, realizando as correlações necessárias para obter a grandeza. É importante enfatizar que os parâmetros de taxa de amostragem, resolução e faixa de entrada dos conversores A/D são essenciais no processo de conversão. A taxa de amostragem é um parâmetro importantíssimo na aquisição de dados, dependendo da sua proporção em função da frequência do sinal de entrada, pode-se perder informações sobre grandeza ou ocupar muito espaço de armazenamento. O teorema de Nyquist, conhecido como Teorema da Amostragem, define que a taxa de amostragem nas conversões analógicas/digital deve ser no mínimo duas vezes maior que a taxa do sinal a ser medido. Mas em muitas situações, um sinal pode ser melhor representado por uma taxa de amostragem de até dez vezes maior que a frequência do evento aferido (Campilho, 2000).

Com o avanço da microeletrônica, existem diversos tipos de microcontroladores que fazem essa função, como a plataforma Arduino (Arduino, 2018) e ESP32 (Espressif Systems, 2021).

3.3. COMUNICAÇÃO

Com as informações em sinal digital, elas podem ser disponibilizadas para o gestor da infraestrutura, assim, a partir desses dados podem ser racionalizadas ou validadas algumas ações acerca do objeto estudado. Portanto se faz necessário um sistema de comunicação entre a plataforma de aquisição de dados e o usuário. A partir disso deve ser planejado e dimensionado esse subsistema, que levará os dados aferidos até o servidor/computador. Segundo Assis (2007, p. 26), "um bom sistema de comunicação deve viabilizar o transporte dos dados com integridade, de modo que estes cheguem ao destino sem perdas qualitativas ou quantitativas".

Normalmente em monitoramento de integridade estrutural, a comunicação entre a central de aquisição de dados e o servidor é via cabo, pelos mais diversos protocolos de comunicação. Entretanto tal alternativa pode ser bem onerosa na implantação do sistema e inviável no processo de coleta de dados. Assim se faz necessário investigar novos protocolos de comunicação, baseado em sistemas IoT, podendo acessá-los de forma remota via internet.

Todas essas formas de comunicação são baseadas no envio a partir de redes públicas ou privadas, onde na outra ponta os dados são processados e armazenados pelo tempo necessário.

3.4. TRATAMENTO DOS DADOS

Utiliza-se de modelos matemáticos com a finalidade de entender o funcionamento da estrutura e validar seu comportamento. O subsistema de tratamento dos dados é de grande relevância dentro do processo de monitoramento, pois com os dados tratados é possível gerar interpretações que levam a análise da estrutura, embasando a tomada de decisão.

Os sistemas de tratamento devem ser amigáveis para o usuário, ou seja, de fácil entendimento e aprendizado para o analista utilizá-lo. Deve possuir interface gráfica, com ferramentas de visualização e manipulação dos dados, possibilitando os ajustes no processamento e a criação de relatórios.

Com os dados tratados, é possível obter informações acerca do objeto de estudo, dando uma visão preliminar do seu estado de funcionamento. Com ele é possível calibrar os modelos teóricos da estrutura estudada e avaliar a partir das informações do seu funcionamento se ela está trabalhando no regime projetado, caso não, os dados tratados constituem de justificativa para realização de intervenções na obra.

4. MATERIAIS E MÉTODOS

Para o desenvolvimento do sistema, a proposta passou por diversas fases até chegar no modelo final. Inicialmente, realizaram-se testes em laboratório, onde os recursos eram limitados. Esses primeiros testes foram conduzidos com sensores e recursos disponíveis no laboratório. O próximo passo foi o desenvolvimento do sistema de medição, seguido pela implantação e validação numa estrutura de testes.

4.1. MATERIAIS

O estudo desenvolvido utilizou-se dos seguintes materiais:

4.1.1. Low-g Accelerometer

O sensor de movimento Low-g Accelerometer fabricado pela Vernier Software & Technology, é uma ferramenta para fins educacionais, especialmente em laboratórios de física e ciências. O acelerômetro mede a aceleração linear em um único eixo.

Figura 8 - Low g Accelerometer

Fonte: Vernier Software & Technology, 2024.

De acordo com a Vernier Software & Technology (2024), o Acelerômetro Low-g utiliza-se de um circuito integrado (CI), originalmente projetado para o controle de airbags em automóveis, a partir da detecção de acelerações. Este circuito é composto por “dedos” (*fingers)*

extremamente finos, esculpidos em silício, que flexionam quando acelerados. Esses “dedos” são organizados e conectados como as placas de um capacitor, e à medida que se flexionam, a capacitância muda. A aceleração é calculada em função do sinal elétrico recebido durante ocorrência do evento. A saída do CI é então amplificada e filtrada por um circuito de amplificador operacional externo.

O acelerômetro possui as seguintes especificações técnicas:

Tabela 1 - Especificação técnica Low-g Accelerometer

Potência	30 mA @ 5 VDC
Faixas de Medição	±50 m/s² (±5g)
Precisão	±0,5 m/s² (±0,05g)
Resposta de Frequência	0–100 Hz
Resolução de 12 bits:	0,037 m/s²
Consumo normal de operação	Acelerômetro: 500 µA. Giroscópio: 3,6 mA
Calibrações Armazenadas para o Acelerômetro	Inclinação: 22,924 m/s²/V Interceptação: –51,751 m/s²

Fonte: Vernier Software & Technology (2024).

O Low-g Accelerometer é conectado ao LabQuest Mini (Figura 9), uma interface de coleta de dados de vários canais que pode ser utilizada para coletar dados de sensores Vernier em diversas plataformas.

Figura 9 - Low g Accelerometer

Fonte: Vernier Software & Technology, 2024.

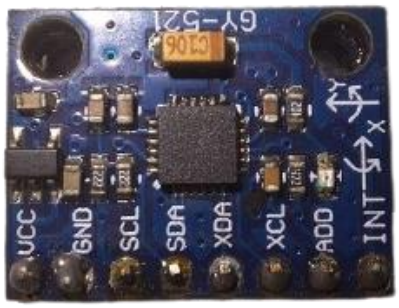
Conectando o Low-g Accelerometer ao LabQuest Mini, os dados coletados podem ser analisados utilizando o *Vernier Graphical Analysis*. Com o software, pode visualizar e interagir com os dados do acelerômetro. Este software permite que os usuários coletem, visualizem e

analisem dados em tempo real. Além disso, o *Graphical Analysis* oferece uma variedade de recursos avançados, incluindo diferentes tipos de gráficos, análise de dados detalhada e a capacidade de compartilhar dados instantaneamente entre dispositivos.

4.1.2. MPU-6050

O sensor de movimento MPU-6050 é fabricado pela TDK InvenSense Inc., possui 6 eixos, combinando um acelerômetro de 3 eixos e um giroscópio de 3 eixos. O acelerômetro mede a aceleração linear do dispositivo em relação aos eixos x, y e z, enquanto o giroscópio mede a velocidade angular de rotação em torno desses mesmos eixos (TDK Invensense, 2012).

Figura 10 - MPU6050



Fonte: Autores (2024).

Sensor baseado em MEMS, onde a aceleração é medida a partir da ação externa, em função do deslocamento de uma micro massa do sistema massa e mola dentro do chip. O deslocamento é calculado por sensores piezoelétricos, resultando na aceleração linear em cada eixo.

O MPU-6050 transfere os dados via protocolo I²C *(Inter-Integrated Circuit*), comunicação serial de dados bidirecional amplamente utilizado para troca de informações entre circuitos integrados, suportando uma frequência de dados de até 400kHz (TDK Invensense, 2012).

O MPU-6050 possui as especificações técnicas conforme Tabela 2:

Tabela 2 - Especificação técnica MPU-6050

Precisão	Acelerômetro: ±2% de erro Giroscópio: ±3% de erro
Faixas de Medição	Acelerômetro: até ±16g Giroscópio: ±250°/sec, ±500°/sec, ±1000°/sec, ±2000°/sec
A faixa de tensão de alimentação VDD	2.375V a 3.46V
Consumo normal de operação	Acelerômetro: 500 μA. Giroscópio: 3,6 mA
Consumo em modo de espera	5 μA
Largura de banda	500 Hz
Sensibilidade	16384 LSB/g
Sensibilidade do eixo cruzado	±2%
Densidade de ruído	400μg/√Hz
Resolução	16-bit

Fonte: TDK Invensense (2012).

4.1.3. ESP-WROOM-32

Linha de microcontroladores de alto desempenho e baixo consumo energético, possui um chip combo de Wi-Fi (*Wireless Fidelity*) e Bluetooth de 2,4 GHz. Contém microprocessadores Xtensa LX6 de 32 bits, suportando frequência de clock de até 240 MHz. Pode se comunicar com os periféricos através de GPIOs (*General-Purpose Input/Output*) programáveis, ADC (*Analog-to-Digital Converter*) de 12 bits, DAC (*Digital-to-Analog Converter*) de 8 bits, SPI (*Serial Peripheral Interface*), I²S (*Inter-IC Sound*), I²C, UART (*Universal Asynchronous Receiver-Transmitter*), Interface Ethernet MAC (*Media Access Control*) (Espressif Systems, 2023).

Figura 11 - Node MCU ESP32

Fonte: Autores (2024).

Características como conectividade com redes Wi-Fi, baixo consumo de energia, alta capacidade de processamento e compatibilidade com diversas interfaces de comunicação, torna o microcontrolador ESP32 de grande valor para confecção de sistemas IoT.

4.1.4. Adaptador Card Micro SD

O módulo permite a expansão da memória do microcontrolador, sendo possível escrever ou ler arquivos em cartão MicroSD. A comunicação do adaptador com o microcontrolador ocorre via SPI com nível de sinal em 3.3V.

Figura 12 - Adaptador Card Micro SD

Fonte: Autores (2024).

Outras características do módulo adaptador card micro SD:

- Tensão: 3.3V a 5V DC
- Dimensões: 42mm x 24mm x 12mm
- Pinos SPI: MOSI, SCK, MOSO e CS

- SD card Arduino/ leitor Micro SD card para Arduino;
- Suporte para cartão Micro SD e Micro SDHC (cartão de alta velocidade);
- Interface de comunicação SPI padrão;
- Permite salvar informações oferecidas pelos sensores;
- Possui trava que impede que o cartão saia do slot.

4.1.5. Raspberry Pi – Model 4B

Plataforma de desenvolvimento que tem funcionamento semelhante a um desktop miniaturizado. Foi criado no Reino Unido pela Fundação Raspberry Pi, uma organização sem fins lucrativos focada na promoção de ensino de ciência da computação básica para jovens em escolas e universidades da Europa.

Figura 13 - Raspberry Pi - Model 4B

Fonte: Autores (2024).

Possui um processador quad-core de 64 bits, suporta dois monitores de até 4K, decodificação de vídeo até 4Kp60, até 8GB de RAM, LAN sem fio de banda dupla, Bluetooth 5.0, Ethernet Gigabit e USB 3.0.

4.2. VALIDAÇÃO DO SENSOR

O processo de validação do equipamento de medição envolve uma série de testes rigorosos para garantir a confiabilidade de sua leitura, isso inclui verificar se o sensor captura e transmite dados de forma consistente. Dentro do processo abrange a comparação dos resultados do sensor com outros de mercado, a fim de atestar seu uso.

4.2.1. Modelo 1 - Chapa Metálica

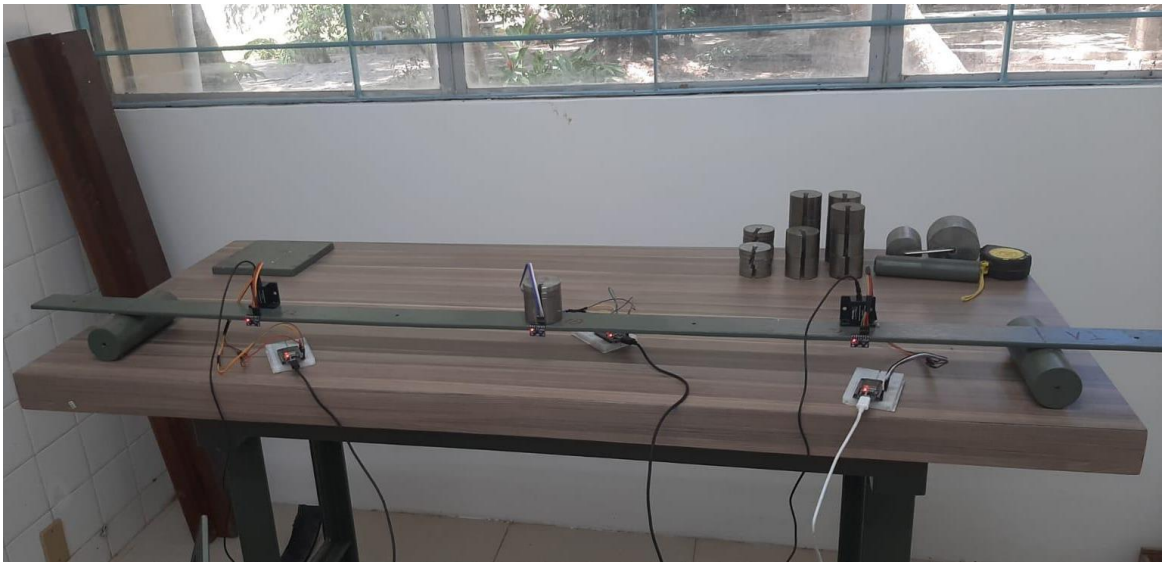
4.2.1.1. Objetivo

A primeira campanha de ensaios teve por objetivo principal validar as leituras realizadas pelos acelerômetros, em modelo de viga biapoiada, e entender o efeito variação de massa, através da adição de pesos, visando avaliar as variações nas frequências naturais da viga. Para fins comparativos, as frequências aferidas foram confrontadas com as calculadas em modelo de elementos finitos. Outro objetivo importante foi implementação de tratamento de dados para múltiplos sinais do evento avaliado.

4.2.1.2. Caracterização do ensaio

Foi utilizado um elemento bi apoiado, com dois trechos menores em balanço e um vão central. A chapa metálica possui uma seção retangular de 50x6mm com comprimento total de 1,51m, apoiada sobre roletes metálicos, que foram fixados na mesa de ensaio conforme Figura 14 - Aferição dos dados na viga 1. Foi considerado um módulo de elasticidade de 210 GPa para o aço.

Figura 14 - Aferição dos dados na viga 1



Fonte: Autores (2024).

4.2.1.3. Medição

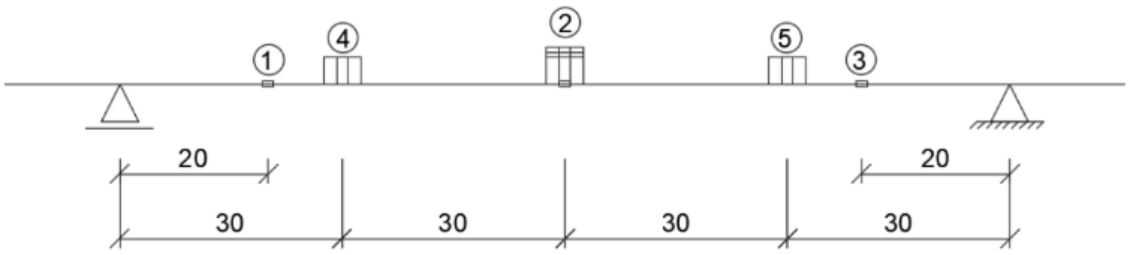
Os primeiros ensaios foram realizados medindo a aceleração da viga após excitação causada por deslocamento realizado no meio do vão, permitindo a viga entrar em vibração livre. Para este modelo estrutural, foram realizados cinco ensaios distintos, realizando a excitação para apenas com a viga e os outros quatro com adição de uma quantidade de massa em diferentes posições.

Tabela 3 - Resumo dos ensaios realizados

Ensaios	**Situações Avaliadas**
1	Peso próprio da viga
2	7N na posição 2
3	5N nas posições 4 e 5 e 7N na posição 2
4	10N nas posições 4 e 5 e 7N na posição 2
5	10N nas posições 4 e 5 e 17N na posição 2

Fonte: Autores (2024).

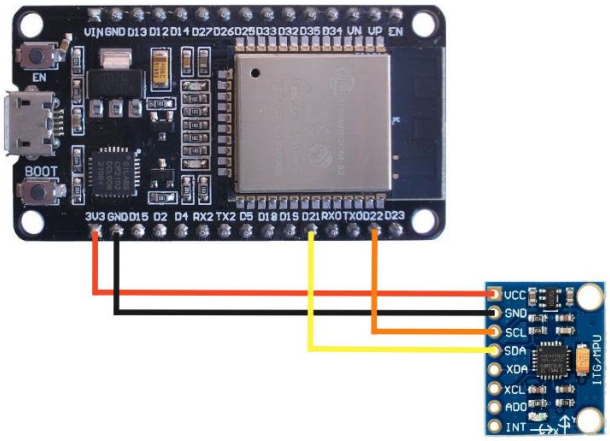
Figura 15 – Posicionamento dos sensores e massas, viga 1



Fonte: Autores (2024)

Utilizou-se da ESP32 para receber e processar os dados MPU-6050. Para comparar a resposta em frequência do MPU-6050, utilizou-se o Low-g Accelerometer.

Figura 16 - Esquema de ligação do sensor



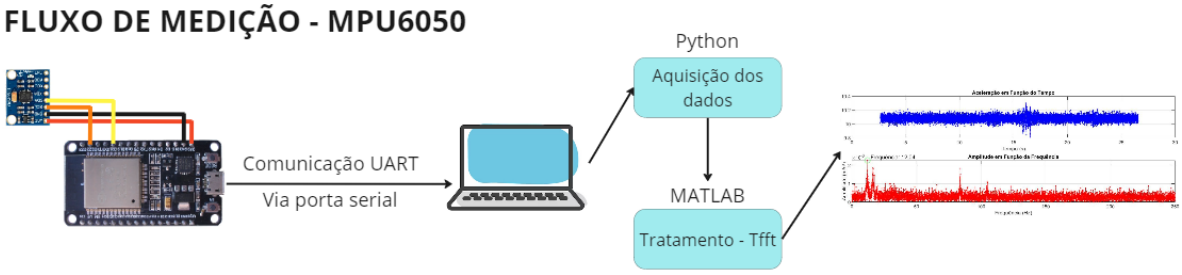
Fonte: Autores (2024).

O MPU-6050 e o Low-g foram posicionados a 20cm do apoio (posição 1 e 3) e outro MPU-6050 no centro do vão (posição 2), a fim que as medições possam obter pelo menos os primeiros harmônicos da viga. Para cada ensaio o experimento foi realizado dez vezes.

O fluxo de medição do ensaio varia conforme acelerômetro utilizado. Para o conjunto ESP32 e MPU-6050, os dados foram transmitidos da ESP32 para o desktop via conexão USB, enviados através da porta serial, processados usando um código em Python para a aquisição dos dados, salvos em um arquivo .txt e, em seguida, analisados utilizando a função Tfft e FDD no MATLAB (MATLAB, 1970). Não foi necessário metodologia para alinhamento dos dados, pois os conjuntos de medição foram iniciados no mesmo tempo, utilizando-se de um hub USB. O fluxo de Low-g Accelerometer difere anterior na aquisição dos dados, já que o sensor

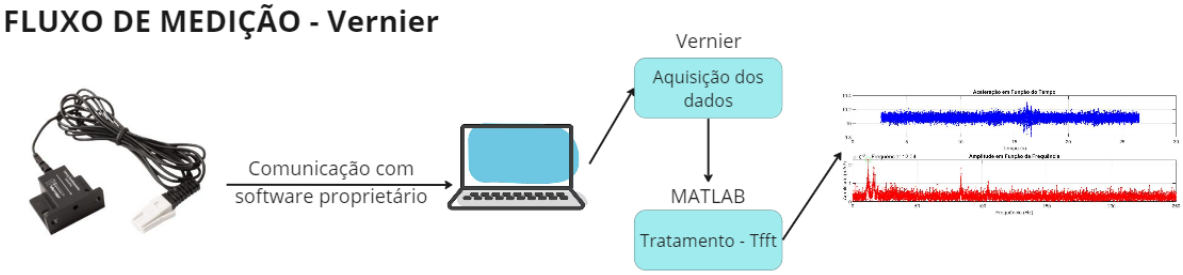
produzido pela empresa Vernier possui software proprietário para essa operação. Após medição, os dados são exportados do software para um arquivo .csv e interpretados na função Tfft e FDD. Ambos os processos são descritos nos fluxogramas na Figura 17 e Figura 18.

Figura 17 - Fluxo de medição, MPU-6050



Fonte: Autores (2024).

Figura 18 - Fluxo de medição, Vernier



Fonte: Autores (2024).

4.2.1.4. Captura dos dados:

O conjunto ESP32 e MPU-6050 foram programados na plataforma do IDE Arduino em C++, onde o acelerômetro comunica-se com microcontrolador via protocolo I^2C. Enquanto o acelerômetro produzido pelo Vernier possui sistema de aquisição de dados com software proprietário.

Tabela 4 - Configuração dos acelerômetros na ponte

Configurações	**Low-g Accelerometer**	**MPU-6050**
Frequência de amostragem	200 Hz	500 Hz
Faixa de medição	±5g	±2g
Tempo de medição	30 s	60 s
Eixos de medição	Unidirecional	Aceleração em x

Fonte: Autores (2024).

Figura 19 - Inicialização para medição

```
void setup() {

  Wire.begin();
  // Configuração do acelerometro
  Wire.beginTransmission(MPU_addr);
  Wire.write(0x6B);  // PWR_MGMT_1
  Wire.write(0);     // Iniciando acelerometro
  Wire.endTransmission(true);

  Serial.begin(115200);
}
```

Fonte: Autores (2024).

No setup do Arduino IDE, o acelerômetro foi configurado e inicializado. Já no loop, existe uma estrutura de condição para controlar o intervalo entre leituras, garantindo que a frequência de amostragem seja constante, garantindo que a frequência de amostragem sempre fosse mais que duas vezes maior que a frequência natural da estrutura. Dentro dessa estrutura condicional, os bytes da aceleração x são lidos, convertidos para m/s² de acordo com o fator de sensibilidade e, em seguida, são impressos no monitor serial.

Figura 20 – Estrutura de repetição de execução da medição

```
void loop() {
  unsigned long currentTime = micros();

  if (currentTime - previousTime >= delta) {      //Controle do tempo de leitura
    previousTime = currentTime;

    // Leitura do MPU-6050
    Wire.beginTransmission(MPU_addr);
    Wire.write(0x3B);  // Iniciando leitura dos dados a partir do registro da Aceleração em X
    Wire.endTransmission(false);
    Wire.requestFrom(MPU_addr, 2, true); // 2 registros que representam a leitura da aceleração em x
    AcX = Wire.read() << 8 | Wire.read();

  // Conversão da aceleração em m/s² de acordo com o fator de sensibilidade
    float acceleration_mps_X = (AcX / 16384.0) * 9.8;

    Tempo = currentTime / 1000;

    // Print data for the first accelerometer
    Serial.print(Tempo);
    Serial.print(" ");
    Serial.println(acceleration_mps2_X);
  }
}
```

Fonte: Autores (2024).

4.2.1.5. Aquisição dos dados

Os dados recebidos do MPU-6050 são enviados, via comunicação serial, para o desktop e posteriormente são interpretados. As rotinas foram elaboradas no ambiente de desenvolvimento *Virtual Studio Code* utilizando a linguagem Python, para leitura do sensor, retornando um arquivo de texto com o resultado da medição, com tempo e aceleração em colunas distintas. Para o Vernier, utilizou-se o software proprietário *Vernier Graphical Analysis* para coletar e organizar os dados, que em seguida foram exportados para o Excel, na mesma estrutura de dados do acelerômetro MPU-6050.

4.2.1.6. Tratamento de dados

A partir da identificação do sinal na leitura da aceleração no domínio do tempo, os dados de ambos os sensores foram analisados na função Tfft e FDD, a fim de verificar a frequência natural dominante aferida. Ambas as funções foram desenvolvidas no MATLAB.

A função Tfft calcula a Transformada Rápida de Fourier (FFT – *Fast Fourier Transform)* dos sinais de aceleração em relação ao tempo, fornecendo informações sobre as frequências presentes nos sinais. A FFT é uma técnica amplamente utilizada para transformar sinais do domínio do tempo para o domínio da frequência de forma eficiente. Para um sinal de aceleração a(t) em função do tempo t, a FFT nos fornece a representação de a(t) no domínio da frequência f, permitindo a análise das diferentes componentes de frequência presentes no sinal.

A fórmula da FFT é dada por:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-i2\pi kn/N} \quad (1)$$

Onde:

X(k) é a transformada de Fourier discreta no domínio da frequência.

x(n) é o sinal de entrada no domínio do tempo.

N é o número total de pontos no sinal.

k é o índice da frequência.

Na função Tfft, o processo de cálculo da FFT envolve:

Cálculo do comprimento do sinal de aceleração L, em número de amostras.

Cálculo do intervalo de amostragem Ts como a média das diferenças de tempo entre as amostras de tempo.

Cálculo da frequência de amostragem Fs como o inverso do intervalo de amostragem (Fs=1/Ts).

Para cada sinal de aceleração fornecido:

Remoção da média do sinal de aceleração para evitar componentes de frequência DC.

Cálculo da FFT normalizada do sinal de aceleração.

Determinação do vetor de frequência Fv com base na frequência de amostragem e no comprimento do sinal.

Identificação da frequência dominante e sua amplitude associada.

Armazenamento dos resultados para posterior análise.

A função retorna os vetores de frequência, as amplitudes, as frequências dominantes e as amplitudes máximas associadas para cada sinal de aceleração fornecido.

A Função Graf cria uma figura para cada conjunto de dados de aceleração fornecidos. Cada figura contém dois gráficos, avaliando o sinal no domínio do tempo e da frequência. Após a criação dos gráficos, eles são exportados como imagens JPEG para o caminho especificado.

O FDD é uma das técnicas para análise de respostas vibracionais no domínio do tempo. Desenvolvida por Brincker (2000) com objetivo de identificar as propriedades modais da estrutura instrumentada, a partir da análise operacional modal. O método se inicia calculando a matriz de densidade espectral de potência (PSD – *Power Spectral Density*) em função das acelerações coletadas da estrutura. Assim é formada a matriz de sinais de saída [G(w)], onde os valores da diagonal principal são constituídos da PSD e os valores da densidade espectral cruzada (CPSD – *Cross Power Spectral Density*) nas demais posições.

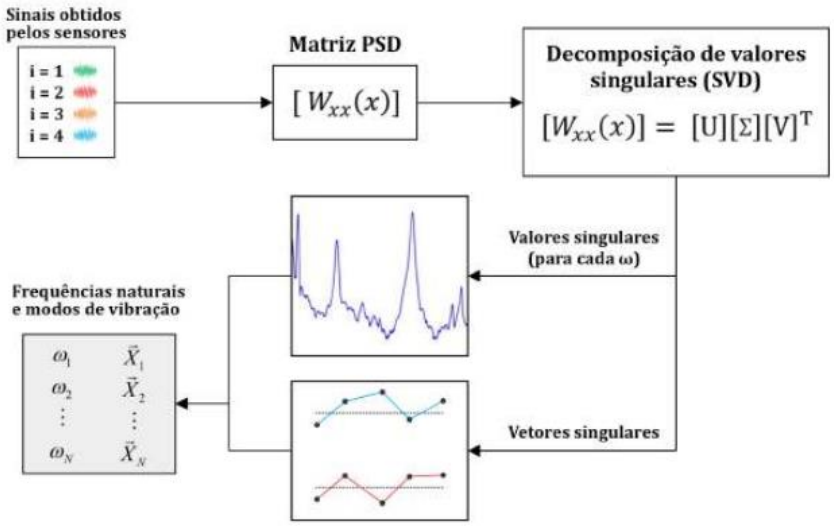
$$[G(w)] = \begin{bmatrix} G_{x1,x1}(w) & \cdots & G_{x1,xN}(w) \\ \vdots & \vdots & \vdots \\ G_{xN,x1}(w) & G_{xN,x2}(w) & G_{xN,xN}(w) \end{bmatrix} \quad (2)$$

Onde, *N* é o número de sinais de saída processadas pela FDD. A matriz PSD possui 3 dimensões, *N* x *N* x *NF*, onde *NF* é o número de amostras de sinal transformadas no domínio da frequência. Em seguida é realizada a decomposição em valores singulares, onde a decomposição é calculada para cada frequência (ω):

$$G(w) = [U][\Sigma][V]^T \quad (3)$$

Onde $[U]$é uma matriz unitária N x N, $[\Sigma]$ é uma matriz N x N diagonal com os valores singulares e $[V]^T$ é uma matriz N x N com vetores singulares.

Figura 21 - Ilustração FDD



Fonte: Silva (2022).

Foi utilizado uma função do OoMA Toolbox (Otto, 2024) para calcular o FDD, retornando os valores singulares, os vetores de frequência correspondente aos valores singulares e a matriz modal, para determinar a forma modal.

OoMA Toolbox é uma ferramenta dedicada à Análise Modal Operacional (OMA), focada especialmente na Identificação de Subespaço Estocástico (SSI), desenvolvida por Andrew Otto. O OoMA permite a análise da resposta vibratória de uma estrutura em situação de excitação ambiente para estimar suas características dinâmicas, como frequências naturais, formas modais e razões de amortecimento

Função GrafFDD gera o gráfico de densidade espectral em decibéis, onde os valores singulares são representados em função das frequências. Os picos identificados anteriormente são destacados no gráfico.

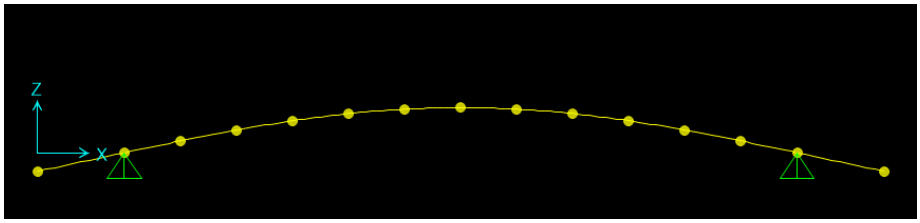
Os gráficos de aceleração em função do tempo foram examinados para verificar a presença de comportamento de amortecimento nos sinais. Durante a análise da Transformada Rápida de Fourier (FFT) e da Decomposição no Domínio da Frequência para comparação com

o modelo teórico, foi decidido optar pelo FDD devido à sua capacidade de processar múltiplos canais simultaneamente, utilizando uma matriz de correlação e sendo eficaz na identificação de frequências naturais. O modo de vibração foi empregado para comparar a forma das frequências avaliadas experimentalmente com as previstas utilizando os resultados do SAP2000 (SAP2000, 1975).

A partir do resultado dos ensaios foram realizadas a média, desvio padrão e covariância das frequências encontradas, além de avaliar uniformidade das leituras da aceleração em função do tempo e da resolução da informação de frequências no gráfico de aceleração no domínio da frequência.

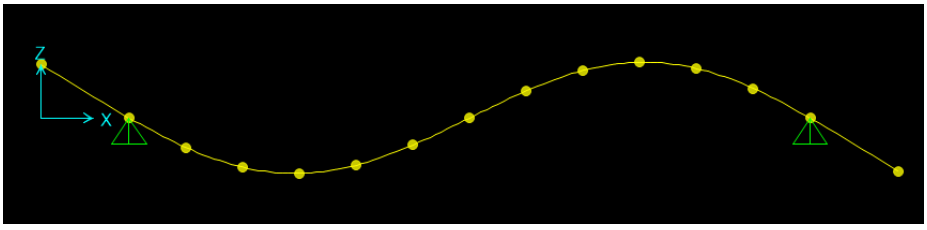
Para avaliar os ensaios, os resultados encontrados foram validados com modelo em elementos finitos. A estrutura do experimento foi modelada no SAP2000 com 14 elementos, entrando no cálculo a massa adicionada e o peso dos acelerômetros, a fim de obter os primeiros harmônicos do experimento e as formas modais.

Figura 22 - Forma modal do 1º modo de vibração



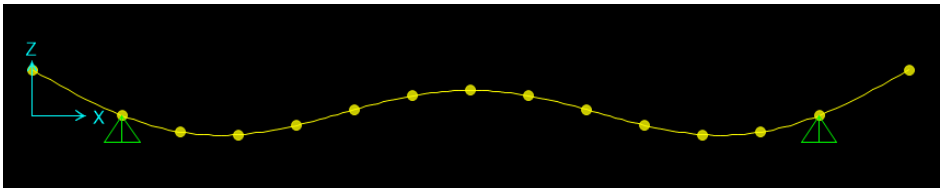
Fonte: Autores (2024).

Figura 23 - Forma modal do 2º modo de vibração



Fonte: Autores (2024).

Figura 24 - Forma modal do 3º modo de vibração



Fonte: Autores (2024).

Com as frequências aferidas e calculadas, utilizou-se da diferença relativa para fins de comparação. Além disso, foi realizada uma comparação entre as frequências obtidas pelo MPU-6050 e as medidas pelo low-g Vernier, utilizando a mesma métrica da diferença relativa.

$$e = \frac{|FA - FT|}{FT} * 100 \quad (4)$$

Onde,
e – Diferença relativa (%);
FA - Frequência Aferida (Hz);
FT - Frequência Teórica (Hz).

4.2.2. Modelo 2 – Viga I

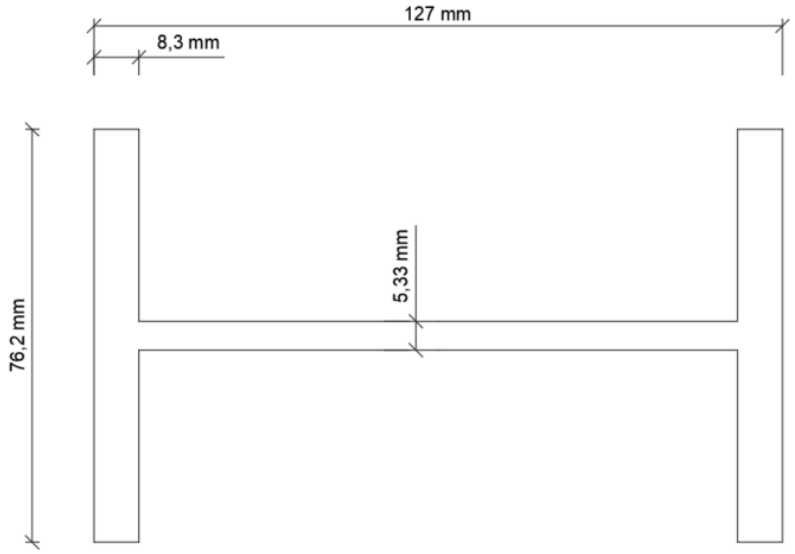
4.2.2.1. Objetivo

O objetivo do ensaio foi investigar a resposta vibracional de uma viga mais rígida, além de desenvolver uma rotina para o alinhamento dos dados. Em resumo, buscamos orientar dados defasados do mesmo evento medidos por sensores diferentes, resultando em todos os dados alinhados a partir do início do evento aferido.

4.2.2.2. Caracterização

O ensaio utilizou-se de um modelo estrutural bi apoiado, no qual foi suportado por dois blocos de concreto. A viga metálica em questão apresenta uma seção transversal em forma de I, com medidas de mesa bf de 76,2mm, espessura da mesa tf de 8,3mm, espessura da alma tw de 5,33mm e altura h de 127mm.

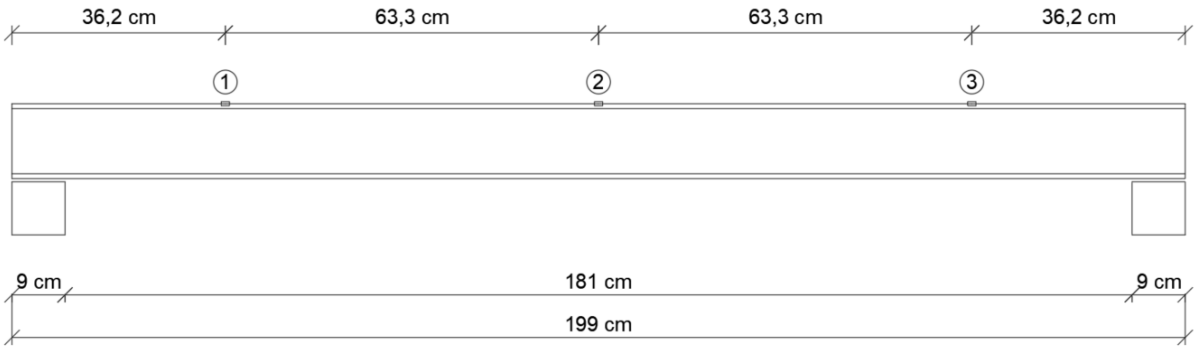
Figura 25 - Seção viga I



Fonte: Autores (2024).

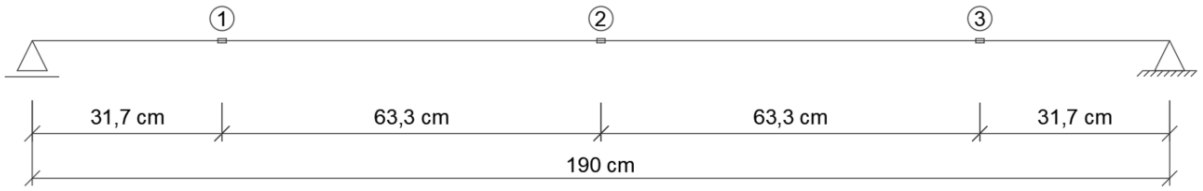
A viga foi posicionada no sentido de menor inércia, possuindo um comprimento total de 1,99m. Cada bloco de concreto apoiava 9cm da viga, situando-se a 4,5cm das extremidades, resultando em um vão teórico de 1,90m.

Figura 26 - Ilustração da viga 2



Fonte: Autores (2024).

Figura 27 - Estrutura teórica da viga 2



Fonte: Autores (2024).

4.2.2.3. Medição

A excitação na viga foi realizada com impacto de martelo de borracha no centro da viga. Para receber a resposta vibracional foram colocados dois sensores MPU-6050 a uma distância de 36,2cm das extremidades da viga (posições 1 e 3), enquanto um terceiro MPU foi posicionado no centro (posição 2). Cada sensor foi conectado a uma placa ESP32, conforme os ensaios anteriores. A diferença importante foi na alteração da faixa medição da aceleração, pois pelo impacto algumas acelerações atingiram os limites de medição anterior, assim foi modificado para ±4g. A captura de dados e aquisição foi conforme os ensaios realizados na viga 1.

Figura 28 - Instalação dos sensores na viga 2.

Fonte: Autores (2024).

4.2.2.4. Tratamento de dados

No tratamento de dados, o fluxo da análise foi similar aos anteriores, porém, devido aos acelerômetros serem ativados em momentos diferentes, foi necessário alinhar os dados. Para isso, foi utilizada a função “Xcorrelacao”. A Função calcula a correlação cruzada entre os sinais de aceleração registrados por diferentes sensores, permitindo comparar a semelhança entre esses sinais ao longo do tempo. A correlação cruzada entre duas sequências discretas de tempo, x e y, com um deslocamento (*lag*) de m, é definida matematicamente como:

$$R_{xy} = \sum_{n=0}^{N-1} x[n]y^*[n-m] \quad (5)$$

Onde:

Rxy [m] é a correlação cruzada entre x e y com um deslocamento (lag) de m.

x[n] é a amostra n da sequência x.

y*[n−m] é a amostra n da sequência y, deslocada por m unidades para a direita e com conjugação complexa.

O somatório é realizado sobre todos os valores possíveis de n, de 0 a N−1, onde N é o comprimento dos sinais.

A função realiza a leitura dos dados de aceleração e tempo de cada sensor. Em seguida, é utilizada a função *signal.correlate*, função do MATLAB, para calcular a correlação cruzada entre os sinais de aceleração de todos os pares de sensores. Este processo envolve determinar os lags que maximizam a correlação entre os sinais, indicando o deslocamento temporal necessário para alinhar os sinais de forma ideal.

Os *lags* calculados são então utilizados para ajustar os dados de aceleração, garantindo que estejam corretamente sincronizados. Isso é realizado cortando os sinais de aceleração para terem o mesmo comprimento, o menor encontrado entre os sinais. Além disso, o tempo associado aos dados de aceleração é ajustado para corresponder ao menor comprimento dos sinais, assegurando que os dados de tempo e aceleração permaneçam sincronizados.

Assim como nos ensaios realizados na viga 1, foram calculados os valores estatísticos e utilizou-se do modelo do SAP2000 para validar as frequências encontradas.

4.3. PROVA DE CONCEITO

A utilização de sensores de baixo custo para monitoramento dinâmico de estruturas requer uma análise muito cuidadosa de suas especificações técnica e do objeto de estudo que se pretende avaliar. Além disso, estruturas sujeitas a carregamentos predominantemente estáticos não são adequadas para análise dinâmica. Pontes urbanas e rodoviárias são bons exemplos de estruturas para o monitoramento a partir de vibrações. Tais estruturas sofrem ao longo do tempo com carregamentos cíclicos, podendo levar a danos por fadiga que consequentemente alteram seus parâmetros dinâmicos. Apesar de serem ótimas oportunidades de monitoramento, normalmente são estruturas muito rígidas, portanto deformam e, consequentemente, vibram em menor intensidade. Outro problema dos sensores da linha MPU são com ruídos, Borges (2021) em seu estudo verifica que eles estão bem acima do valor médio para utilização em sistemas de monitoramento dinâmico e diferente do especificado na documentação técnica. Assim se faz necessário avaliar se o sensor MPU-6050 é uma solução válida para análise dinâmica em estruturas reais.

4.3.1. Objetivo

A prova de conceito, por definição, consiste em verificar se determinada aplicação em estágio inicial atende ao conceito pré-estabelecido (Pinheiro, 2010). Portanto, foi realizado prova de conceito através da medição da aceleração em uma ponte, para verificar se o sinal aferido supera o nível do ruído e se é possível utilizá-lo para análise de frequências naturais.

4.3.2. Caracterização da ponte

A estrutura avaliada na prova de conceito fica localizada na avenida Reitor Joaquim Amazonas, sentido CAC (Centro de Artes e Comunicação), na Universidade Federal de Pernambuco - campus Recife. A estrutura é de concreto armado, possuindo três longarinas e tabuleiro com trecho em balanço na região do passeio. Possui tráfego moderado, composto em maioria de carros de passeio e veículos de transporte coletivo.

Figura 29 - Ponte avaliada na prova de conceito

Fonte: Google Maps (2021).

Figura 30 - Foto em perfil da ponte

Fonte: Autores (2024).

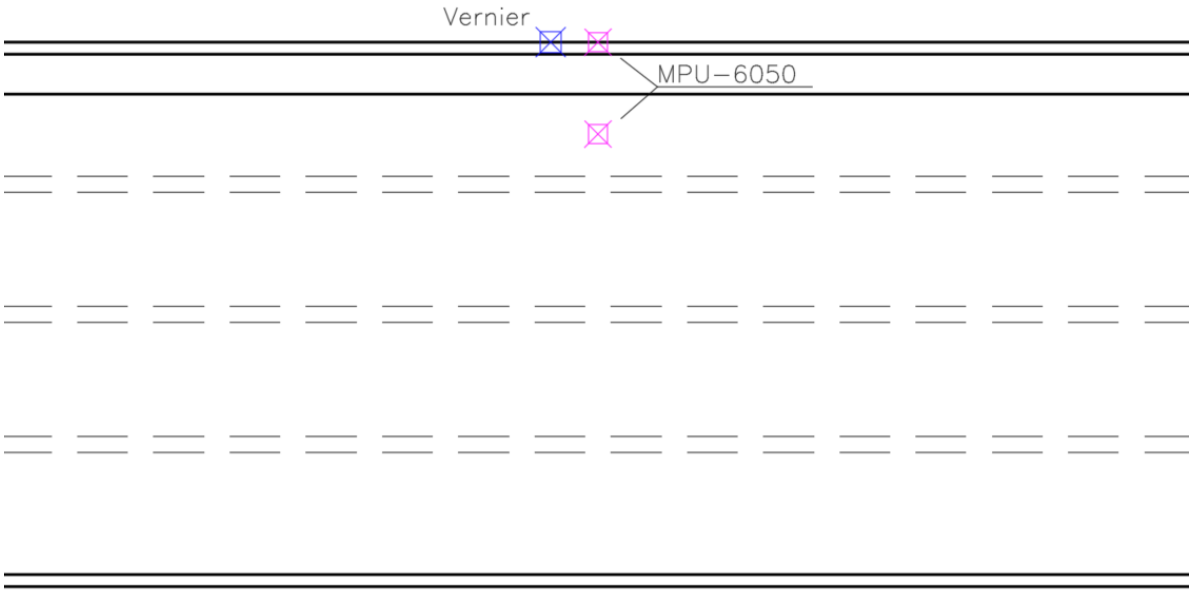
Como essa estrutura está sujeita a carregamentos cíclicos e de intensidade média (transporte coletivo), ela foi escolhida para o ensaio. Além disso, sua localização dentro do campus universitário torna mais fácil realizar as medições necessárias.

4.3.3. Medição

Utilizou-se do conjunto ESP32+MPU-6050 da mesma forma que aplicado nos ensaios anteriores. Para comparar a resposta em frequência do MPU-6050, utilizou-se o Low-g Accelerometer. Os ensaios foram realizados na metade da ponte para obter a maior amplitude de vibração da primeira frequência natural, e a medição foi realizada próximo ao passeio. A

aquisição dos dados iniciou-se pouco antes da passagem de veículos sobre a estrutura. A Figura 31 mostra a disposição dos acelerômetros durante execução do ensaio.

Figura 31 - Planta esquemática dos pontos de medição



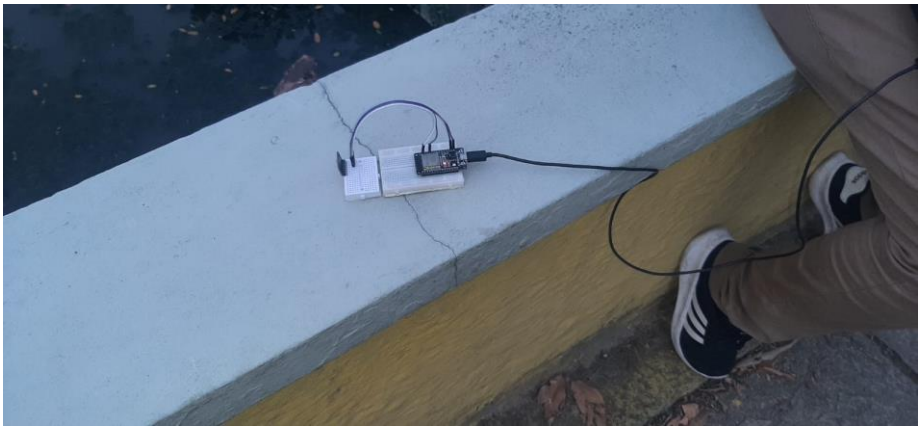
Fonte: Autores (2024).

As Figura 32, Figura 33 e Figura 34 exibem os sensores utilizados e os respectivos pontos de medição.

Figura 32 - Medição com acelerômetro Vernier

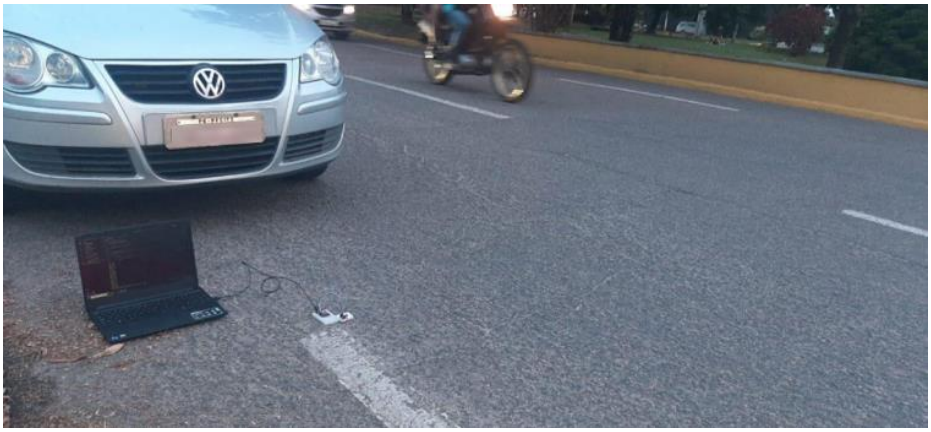
Fonte: Ribeiro (2023).

Figura 33 - Primeiro ponto de medição com MPU-6050



Fonte: Autores (2024).

Figura 34 - Segundo ponto de medição com MPU-6050



Fonte: Autores (2024).

Os ensaios foram realizados com a mesma metodologia da validação dos sensores, com exceção do tratamento de dados, onde foi utilizado apenas o FFT no tratamento dos dados, pois não existiam dois ou mais acelerômetros realizando a medição do mesmo evento em locais diferentes.

4.4. PROTOTIPAGEM

O desenvolvimento de sistemas de monitoramento exige alguns cuidados, começando pelos componentes escolhidos para compor o equipamento de medição, dimensionamento da alimentação de energia, verificação dos acessos as redes privadas ou públicas nos locais de monitoramento, validação do custo de aquisição e implantação, além de fatores relacionados à

instalação e manutenção do sistema. Outro ponto desafiador é definir as condições de contorno para que o monitoramento funcione de forma plena e eficiente. Em resumo, a solução adotada sempre vai depender do objeto de estudo, tornando-se um produto com nível considerável de customização.

4.4.1. Objetivo

O processo de prototipagem compreende da concepção até a produção de um produto mínimo viável (MVP – Minimum Viable Product). Para esse trabalho foi desenvolvido, sistema de aquisição, tratamento e monitoramento dos dados. Assim, o objetivo dessa etapa foi desenvolver o equipamento de medição, tanto em rotinas como sua estrutura física, construir o sistema de aquisição dos dados e tratamento através de servidor local para dados em quase tempo real.

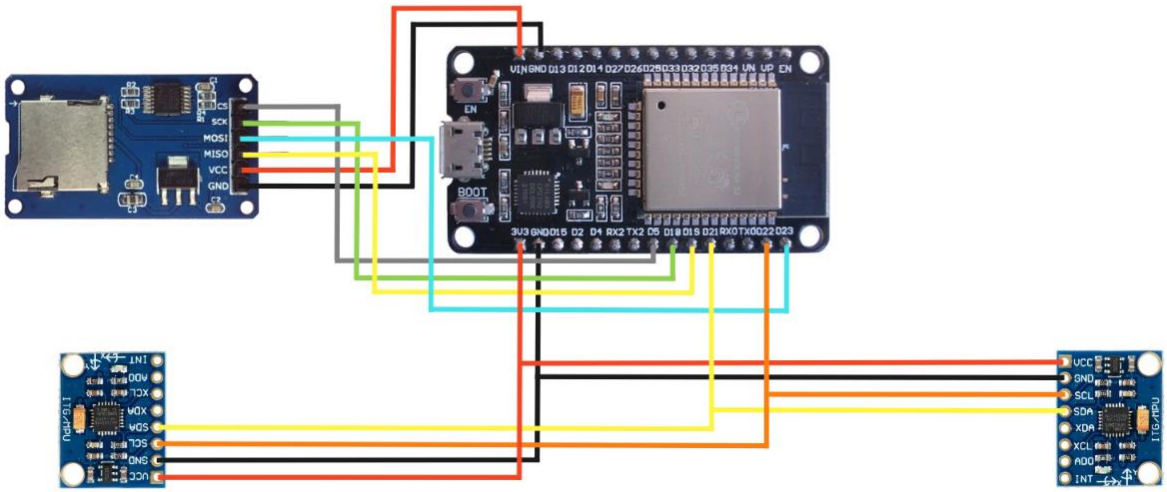
4.4.2. Sistema desenvolvido

O sistema é composto por sensores que realizam a medição dos dados, onde eles são armazenados em memória interna e logo em seguida enviados para servidor local, hospedado em Raspberry pi, via Wi-Fi, onde os dados são armazenados e tratados.

4.4.2.1. Aquisição de dados

Para aquisição de dados foram utilizados dois sensores MPU-6050, conectados via comunicação I^2C na ESP32, que junto a ela existe um adaptador de cartão micro SD, utilizando-se da comunicação SPI, para expansão de memória.

Figura 35 - Esquema de ligação final do sensor



Fonte: Autores (2024).

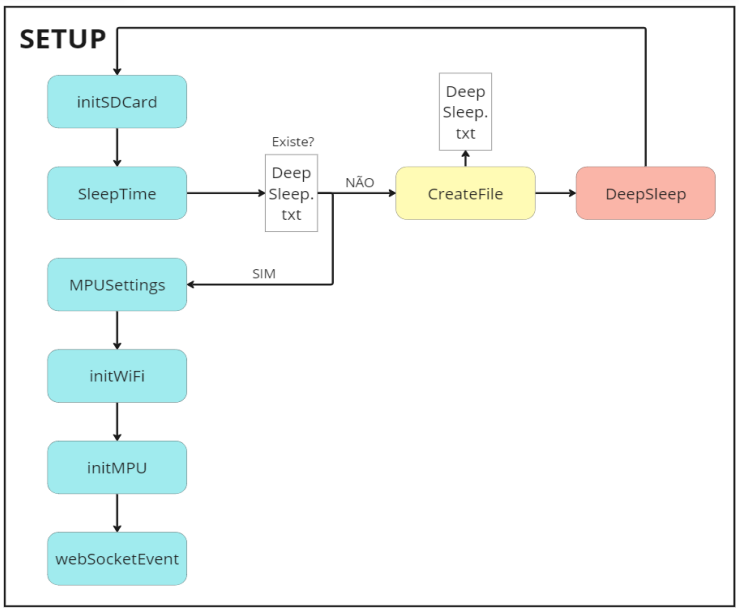
A rotina desenvolvida para o equipamento de medição foi construída no Arduino IDE, em C++, contemplando a realização das medidas com frequência de amostragem e intervalo de tempo determinados, conexão com servidor para configuração dos sensores, definição de intervalo de dormência e armazenamento interno dos ensaios. Trata-se de um código mais complexo, pois busca contemplar as situações de medição em campo.

Foram consideradas algumas premissas para o desenvolvimento da rotina, tais considerações são:

- O equipamento pode ser configurado de acordo com a necessidade de monitoramento, assim parâmetros como horários de medição, intervalo de tempo da amostra, taxa de amostragem e faixa de medição podem ser alterados após a implantação.
- Todas as medidas realizadas são salvas no armazenamento micro sd antes de enviar para o servidor local, pois esse processo garante integridade da medição, evitando perda de informação na comunicação Wi-Fi.
- Entre os intervalos de medições, o equipamento entrará em modo dormência para otimizar o consumo de energia.
- Os dados recebidos no servidor local não são exatamente em tempo real, pois existe o tempo que a leitura vai ser salva no armazenamento interno do equipamento e o tempo de envio dos dados, portanto é considerado em quase tempo real.

Dada todas essas considerações, foram confeccionadas uma série de funções para satisfazerem as condições de monitoramento da estrutura. Na rotina, o void setup configura a medição e verifica o estado de funcionamento do sensor.

Figura 36 - Esquema da inicialização do equipamento



Fonte: Autores (2024).

As seguintes funções foram utilizadas:

- void initSDCard: Inicializa o cartão SD, caso não seja reconhecido, a informação do erro é exibida na serial. Posteriormente verifica o tipo de cartão e informa a capacidade disponível.

- void SleepTime: Tem como objetivo controlar o modo de dormência do sensor para reduzir o consumo de energia durante o seu uso. Primeiramente, ela verifica se o arquivo “deepsleep.txt” existe. Se esse arquivo não existir, a função cria o arquivo e entra no modo de dormência pelo tempo especificado no arquivo “sleeptime.txt”. Após o término desse período de dormência, quando a função é executada novamente e o arquivo “deepsleep.txt” é encontrado, ela atualiza a variável iniciar para o valor 1 e segue para próxima função.

- void MPUSettings(String): Recebe uma string que informa o range de medição desejado para o MPU. Em seguida, ela seleciona as variáveis de configuração do endereço que altera o range e o fator de sensibilidade relativo a ele.

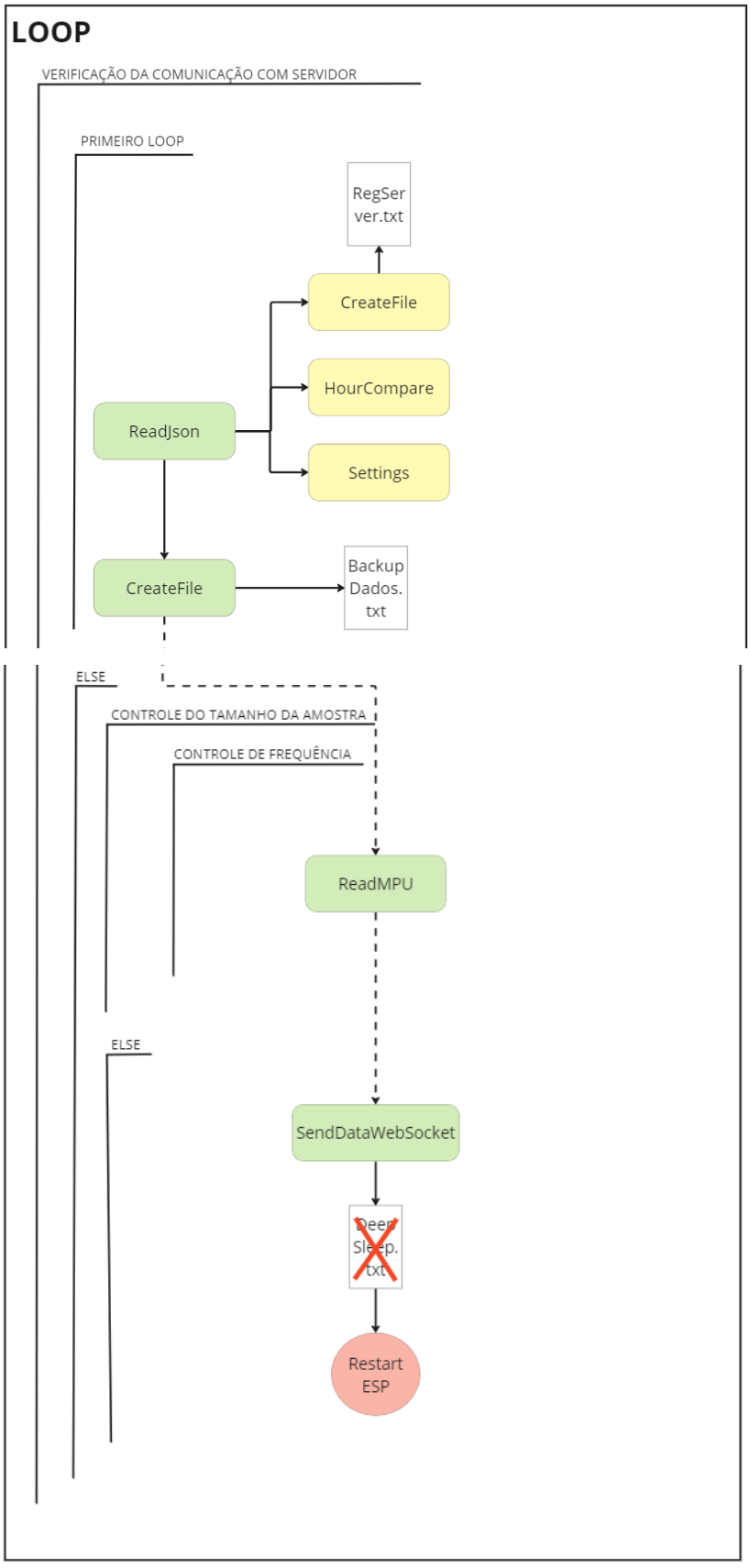
- void initWiFi: Configura a conexão com a rede WiFi de acordo com os parâmetros de nome e senha da rede. Retorna como informação os status da conexão e seu IP.

- void initMPU(int, int): Aplica as configurações selecionadas a cada sensor utilizado no sistema de medição. O primeiro valor de entrada é o endereço do sensor, enquanto o segundo é o valor hexadecimal que representa a configuração do range.

- void webSocketEvent: Funciona como um manipulador de eventos para a comunicação WebSocket. Ela recebe diferentes tipos de eventos, como desconexão, conexão, recebimento de texto ou binário, e erros. Dependendo do tipo de evento recebido, a função imprime mensagens no monitor serial, atualiza o status de conexão, e aloca memória para armazenar os dados recebidos, se for o caso. Em caso de texto recebido, a função copia os dados recebidos para a variável "DataHora", que é utilizada para armazenar a mensagem recebida do servidor, com o recebimento de uma mensagem do servidor é alterado o valor da variável "Conectado" para 1.

No void loop, a mensagem do servidor é recebida após a conexão, a qual é utilizada para configurar algumas das premissas e, posteriormente, realizar a medição. Passado o tempo de dormência especificado e com a conexão ativa, a primeira estrutura condicional é superada. Na Figura 37 é possível visualizar o fluxo de funcionamento do código e em seguida é descrita cada função utilizada.

Figura 37 - Esquema de estrutura de repetição para medição



Fonte: Autores (2024).

- void ReadJSON(SD, const char*, const char*) : Recebe a estrutura de armazenamento configurada, uma constante do tipo char* com a mensagem recebida do server e outra const char* com o nome do arquivo onde a mensagem será salva. Nessa função a char* da mensagem espera receber dados como Data e Hora da conexão com o servidor, a frequência de amostragem a ser utilizada, o tempo de amostragem e os horários que devem ser realizadas medições. Por fim chama as funções “Settings”, “CreateFile” e “HourCompare”.

- void Settings: Ajustar as novas configurações de medição nas variáveis globais, substituindo-as pelos valores recebidos na função “ReadJSON”.

- void CreateFile(SD, const char*, const char*): Cria um arquivo de texto dentro da estrutura de armazenamento configurada, uma constante tipo char* que informa o nome do arquivo e outra para mensagem que será o cabeçalho do arquivo. Nessa situação será criada o arquivo “RegServer.txt” que salva os dados temporais recebidos do server.

- void HourCompare(SD, const char*, const char*) : Tem como entrada a estrutura de armazenamento configurada, recebe uma constante char* que é o arquivo com os dados de data e hora do servidor e recebe outra constante char* com o arquivo que registra os intervalos de dormência do sensor. A função compara a hora recebida do servidor com as horas para realização da medição, a partir dessa comparação gera um intervalo de tempo em microssegundos onde o sensor entrará em modo dormência.

- void CreateFile(SD, const char*, const char*): A função é chamada novamente, agora para criar um arquivo “BackupDados.txt” com os dados da medição, onde no cabeçalho do arquivo estará o horário da medição e os valores de aceleração zerado. Esse arquivo será utilizado na estrutura de loop para escrever todos os dados aferidos.

- Finalizada as funções, existe um incremento na variável “i”. A próxima estrutura condicional garante que a quantidade de leituras está coerente com o tempo de leitura determinado e a seguinte garante que o tempo de amostragem, ou frequência de amostragem, continua constante.

- void ReadMPU: Realiza a leitura dos sensores nos registradores que estão os bytes referentes a aceleração no eixo x, realiza a conversão desses valores para m/s² de acordo com o fator de sensibilidade utilizado e escreve no arquivo de "BackupDados.txt".

Após finalizar todas as medições, a função SendDataWebSocket irá ler linha por linha do arquivo de medição e enviar para WebSocket server, em cada envio é verificado o status da conexão. O envio é configurado para ser o mais rápido possível, caso haja algum erro no envio, a verificação acusa desconexão e tenta reconectar. Reconectando, os dados são reenviados do início.

Enviando todos os dados, a partir da função "void DeleteFile", o arquivo "DeepSleep.txt" é apagado e a ESP32 é reiniciada para executar novamente o script.

4.4.2.2. Comunicação

A comunicação entre o equipamento de medição e o servidor local, se dá via Wi-Fi, pelo protocolo TCP/IP. O servidor local deve ser implantado em Raspberry pi, pois ela consegue gerar a rede Wi-Fi para conexão das placas e executar a rotina para criação do servidor. Essa solução diminui a quantidade de fios para instalação do sistema e consequentemente permite o gerenciamento dos dispositivos utilizados na medição, bem como sua configuração.

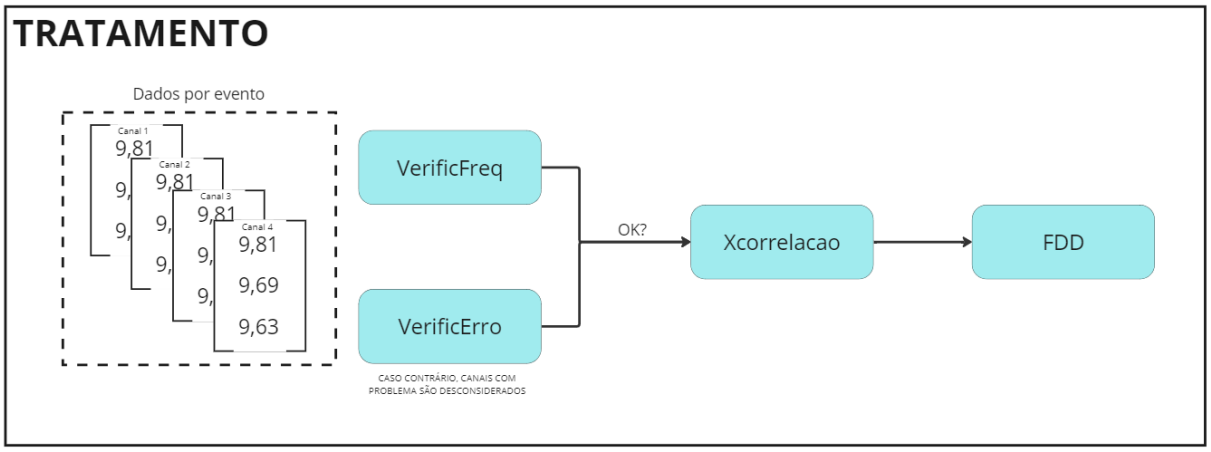
A rotina "WebSocketServer" desenvolvida na linguagem python é responsável por criar um servidor local *web socket,* onde os sensores podem conectar-se ao servidor e enviar os dados após leitura. A aplicação *web socket* é uma tecnologia de ponta que possibilita comunicação bidirecional entre o cliente e um servidor em tempo real. Quando um cliente se conecta, o servidor o adiciona à lista e envia uma mensagem no formato JSON com as configurações pertinentes para a captura dos dados. Em seguida, entra em loop para receber continuamente os dados enviados pelo cliente. Os dados recebidos são processados e armazenados em um dataframe, quando o recebimento dos dados é encerrado, eles são inseridos no banco de dados relacional SQLite (SQLite, 2024).

O SQLite é uma biblioteca em linguagem C que se destaca como um motor de banco de dados SQLite mais enxuto, porém de alto desempenho. Ele oferece uma solução completa e para gerenciamento de dados, com API (*Application Programing Interface* – Interface de Programação de Aplicação) aberta para manipulação em Python.

4.4.2.3. Tratamento de dados

Os ensaios são consultados no banco de dados relacional, retornando o conjunto de dados para cada evento avaliado. Com o retorno da medição, os dados são tratados, conforme o fluxo apresentado na Figura 38. As funções confeccionadas em Python e utilizadas no tratamento de dados estão descritas abaixo.

Figura 38 - Fluxo para tratamento dos dados



Fonte: Autores (2024).

Função VerificFreq: responsável por verificar se as diferenças entre os elementos de um vetor estão dentro de uma faixa de tolerância específica, o que pode ser útil para analisar a frequência de ocorrência de eventos em um conjunto de dados.

O processo de verificação é realizado da seguinte maneira:

É definida uma margem de erro, que representa a variação permitida em torno de um valor de referência.

Para cada par de elementos consecutivos no vetor, é calculada a diferença entre esses elementos.

Verifica-se se todas as diferenças calculadas estão dentro da faixa especificada, considerando a margem de erro.

Essa função é útil para analisar a regularidade ou periodicidade de eventos em séries temporais ou conjuntos de dados sequenciais, fornecendo uma maneira de avaliar se os eventos ocorrem com uma frequência consistente dentro de uma faixa de tolerância.

Função VerificErro: verifica a presença de erros em um conjunto de dados, especialmente erros representados por zeros consecutivos, o que pode ser comum em medições ou aquisições de dados.

O processo de verificação é realizado da seguinte maneira:

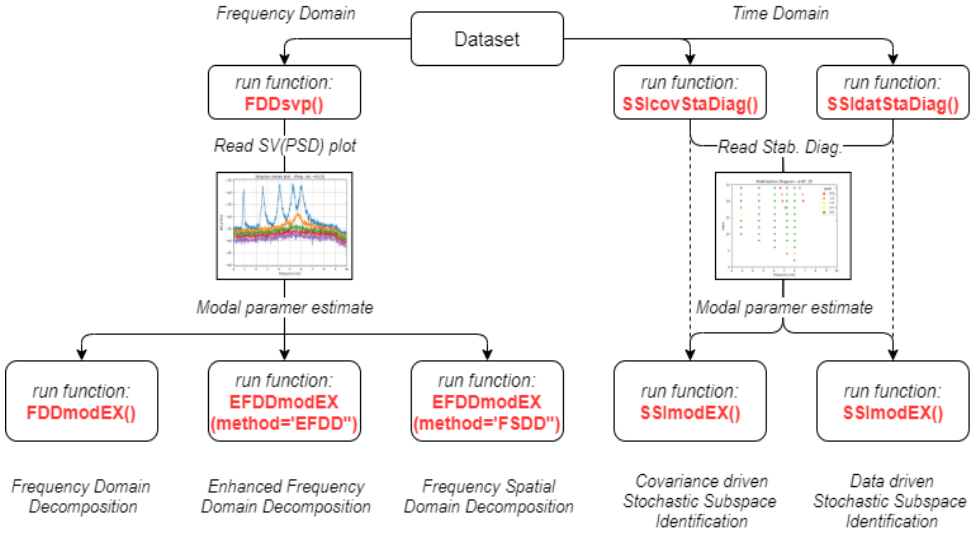
- É especificado o número de zeros consecutivos a serem verificados.
- Para cada sequência de zeros consecutivos encontrada no vetor (excluindo o primeiro elemento), verifica-se se o número de zeros consecutivos corresponde ao número especificado.
- Se for encontrada uma sequência de zeros consecutivos com o comprimento desejado, a função indica a ocorrência e o índice inicial dessa sequência.

Essa função é útil para identificar anomalias ou falhas em dados de medição, onde a presença de zeros consecutivos pode indicar problemas na aquisição ou no registro dos dados.

As funções Xcorrelação e FFD são as mesmas apresentadas nos ensaios de validação do sensor, onde o Xcorrelação foi transcrito para o Python. Já a função FFD foi utilizada da biblioteca Py-OMA.

A ferramenta PyOMA possui um framework em Python de código aberto para análise dinâmica de estruturas, em função dos seus dados de saída, assim é possível encontrar parâmetros modais como frequências naturais, formas modais e razões de amortecimento. Para OMA existem diversos algoritmos que conseguem interpretar a resposta dinâmica da estrutura. Entre eles, o Método de Identificação de Subespaço Estocástico (SSI) e a Decomposição de Domínio de Frequência (FDD) provaram ser algoritmos eficazes e confiáveis para identificação de parâmetros dinâmicos (Pasquale, 2022).

Figura 39 - Framework para Análise Operacional Modal



Fonte: Pasquale (2022).

4.4.3. Prototipagem do equipamento de medição

O setup final do equipamento foi projetado em função do custo, limitações do sistema, resiliência contra agentes externos, possíveis dificuldades na instalação do sistema e manutenções necessárias ao longo da operação do sistema. Algumas aferições e premissas foram discutidas na idealização do projeto, entre elas:

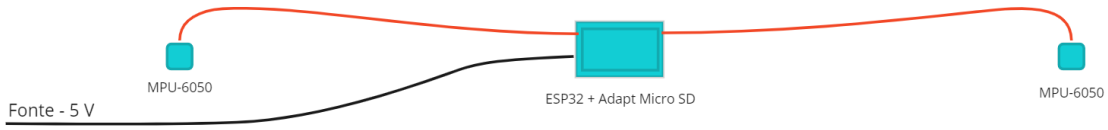
Quantidade de acelerômetros: A comunicação com a ESP32, via barramento I^2C, permite apenas a instalação de dois sensores MPU-6050 com endereços distintos. Uma solução para resolver esse problema seria a utilização de um multiplexador para criar vários endereços e consequentemente aumentar a quantidade de sensores por microcontrolador. Entretanto, foram realizados testes com mais de dois acelerômetros e um multiplexador, e o desempenho em termos de frequência foi abaixo do esperado, sendo possível realizar medições numa taxa de amostragem máxima de 100 Hz. Além da taxa de amostragem, a quantidade de fios do sistema dificultaria a instalação em estruturas reais.

Alimentação do sistema: Incialmente o objetivo seria alimentar o sistema de forma individualizada, onde cada equipamento teria uma fonte própria de energia. Porém, além do custo elevado, inviabilizaria possível utilização de rede de energia local e necessitaria de manutenção mensal na troca das baterias. Portanto, a alimentação do sistema poderá ser feita de forma generalizada, onde a fonte alimentadora irá abastecer cada sensor. Essa fonte pode ser

instalação própria da estrutura ou conjunto de baterias em local acessível para possíveis manutenções.

Assim a estrutura foi esboçada em três partes, uma para o microcontrolador e o armazenamento interno, outras duas para proteger os acelerômetros

Figura 40 - Esboço do equipamento de medição

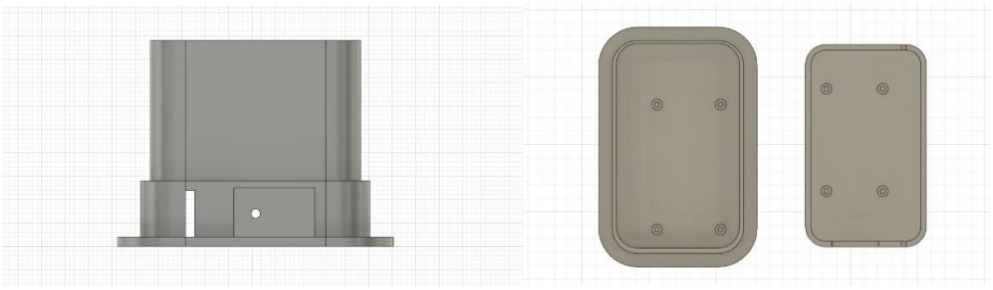


Fonte: Autores (2024).

4.4.3.1. Proteção da ESP32 + Adaptador Micro SD

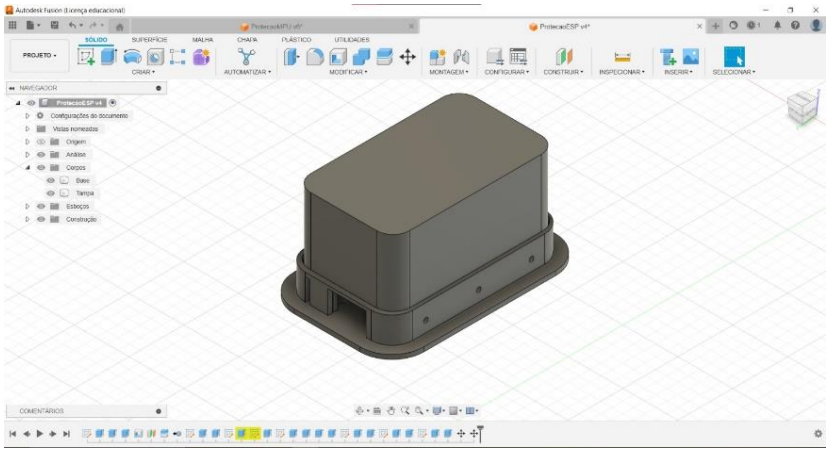
A estrutura projetada tem por objetivo possuir o menor tamanho possível para organização das placas e disposição dos fios. A Peça foi modelada no Autodesk Fusion, software de modelagem 3D e design paramétrico. Oferece uma plataforma integrada para modelagem, simulação, renderização e fabricação, permitindo o usuário visualizar e testar seus projetos antes da produção.

Figura 41 - Vista frontal e superior da proteção da ESP32



Fonte: Autores (2024).

Figura 42 - Isométrico da peça de proteção da ESP32



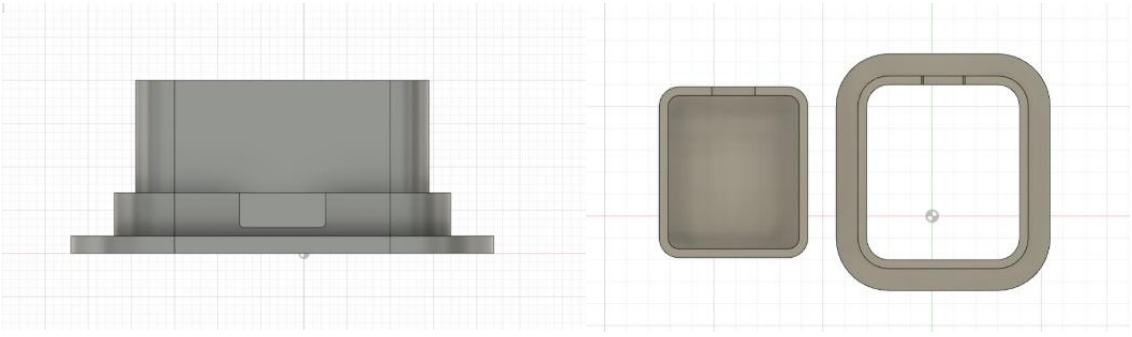
Fonte: Autores (2024).

A peça projetada possui um conjunto de aberturas para passagem dos fios que ligam aos sensores e a entrada de energia. A parte superior é conectada a inferior a partir de parafusos de 2x5mm, a região de encaixe das duas partes foi protegida com silicone. As placas eletrônicas também são fixadas em pinos elevados por parafusos do mesmo tipo utilizado encaixe das peças. A base inferior possui tamanho sobressalente em relação a peça para aumentar a área de contato onde será fixado na estrutura com adesivo químico.

4.4.3.2. Proteção da MPU-6050

A estrutura foi esboçada para possuir o menor tamanho possível, para comportar apenas o sensor e proporcionar a passagem dos fios de comunicação com microcontrolador.

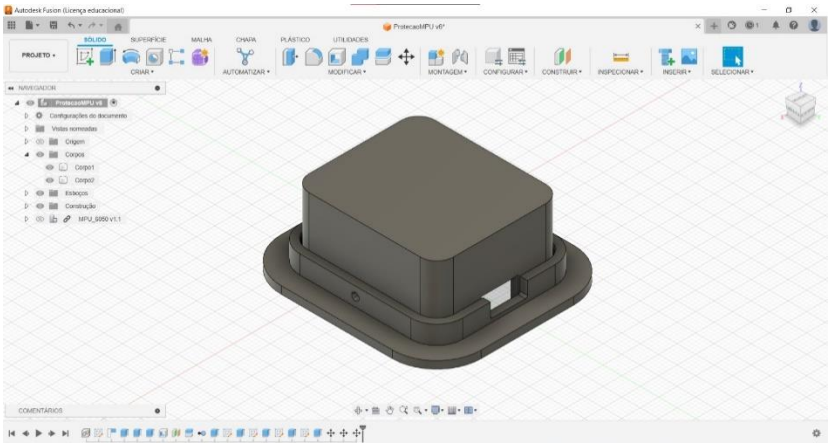
Figura 43 - Vista frontal e superior da proteção do MPU-6050



Fonte: Autores (2024).

A peça projetada possui uma abertura para passagem dos fios que ligam ao microcontrolador. A parte superior é conectada a inferior a partir de parafusos de 2x3mm, a região de encaixe das duas partes foi protegida com silicone. Diferente da proteção da ESP32, essa caixa foi projetada para proteger o sensor, que está fixado diretamente na estrutura, por isso todas as juntas da peça foram cuidadosamente impermeabilizadas com silicone. A região da interface proteção e estrutura também deve ser impermeabilizada com silicone, após a colagem com adesivo químico. Mais informações relativas a montagem do equipamento estão disponíveis no Apêndice A.

Figura 44 - Isométrico da peça de proteção do MPU-6050



Fonte: Autores (2024).

4.4.3.3. Custo do equipamento

O custo associado ao desenvolvimento do sensor está relacionado essencialmente a aquisição dos componentes eletrônico e estimativa do custo de impressão das proteções. Na Tabela 5 está descrito os custos por equipamento para desenvolvimento desse protótipo.

Tabela 5 - Custo por equipamento

Descrição	Quantidade	Custo unitário (R$)	Custo total (R$)
ESP32 - WiFi + Bluetooth	1	44,90	44,90
Módulo Cartão MicroSD	1	7,90	7,90
Acelerômetro e Giroscópio MPU-6050	2	17,90	35,80
Jumpers Fêmea-Fêmea	26	0,20	5,20
Impressão Proteção MPU-6050	2	8,00	16,00
Impressão Proteção ESP32	1	14,00	14,00
TOTAL	-	-	123,8

5. RESULTADOS

5.1. VALIDAÇÃO DO SENSOR

5.1.1. Modelo 1 - Chapa Metálica

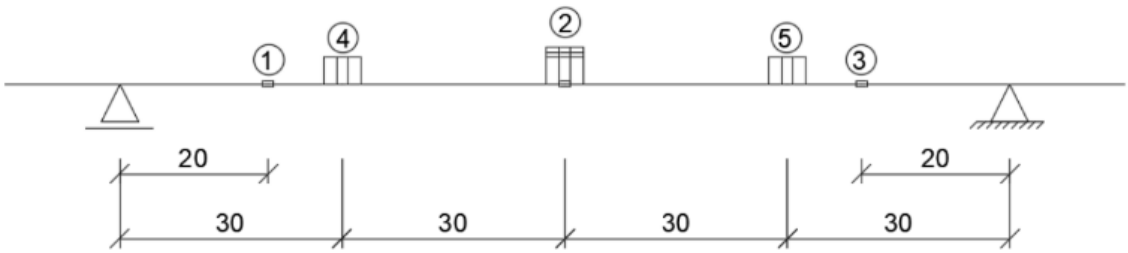
Neste modelo, foram realizados cinco ensaios, cujos resultados serão apresentados a seguir conforme a Figura 45 e a Tabela 6.

Tabela 6 - Resumo dos ensaios realizados

Ensaios	Situações Avaliadas
1	Peso próprio da viga
2	7N na posição 2
3	5N nas posições 4 e 5 e 7N na posição 2
4	10N nas posições 4 e 5 e 7N na posição 2
5	10N nas posições 4 e 5 e 17N na posição 2

Fonte: Autores (2024).

Figura 45 – Posicionamento dos sensores e massas, viga 1

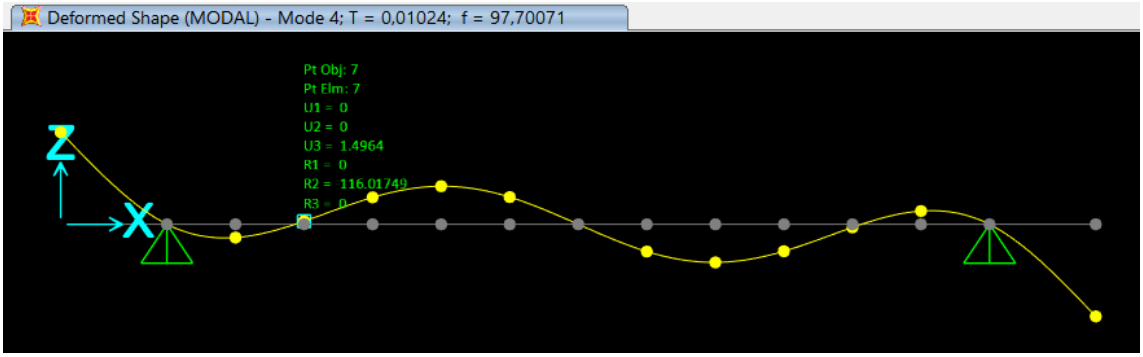


Fonte: Autores (2024)

Durante os ensaios realizados, destacou-se a presença de diferentes números de picos em cada experimento. No Ensaio 1, identificaram-se seis picos, associados aos modos 1, 2, 3, 5, 6 e 7 da estrutura. Nos Ensaios 2 e 3, foram registrados quatro picos, correspondendo aos modos 1, 2, 3 e 5. Já nos Ensaios 4 e 5, observaram-se cinco picos, envolvendo os modos 1, 2, 3, 5 e 7.

É relevante notar que a inexistência do quarto modo de vibração nos ensaios pode ser explicada utilizando o SAP2000. Nesse modelo o nó do modo de vibração tem deslocamento mínimo Figura 46 na posição do acelerômetro para o quarto modo, resultando na não detecção desse modo específico nos experimentos.

Figura 46 - Forma modal do quarto harmônico

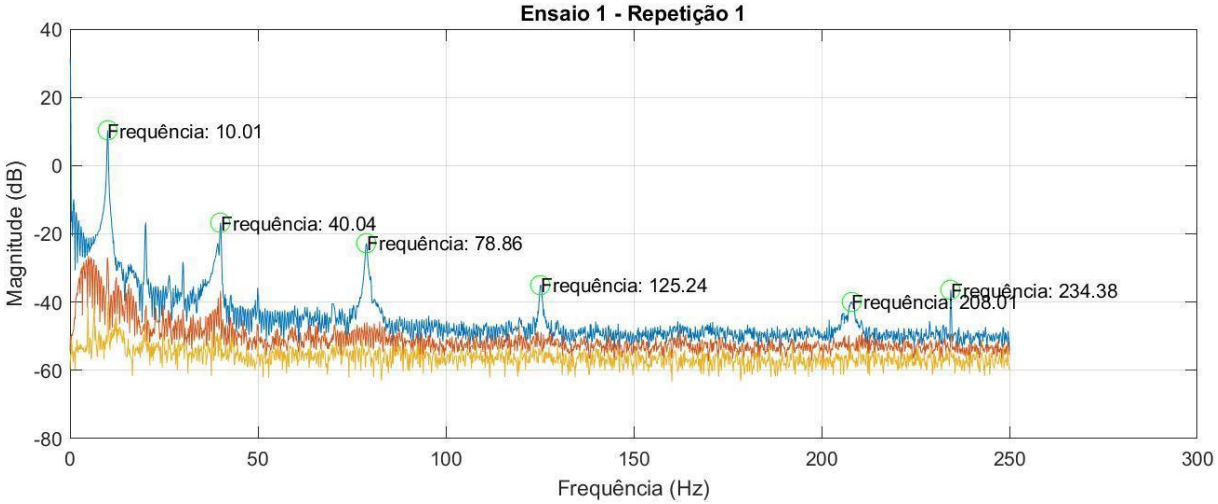


Fonte: Autores (2024).

5.1.1.1. Ensaio 1

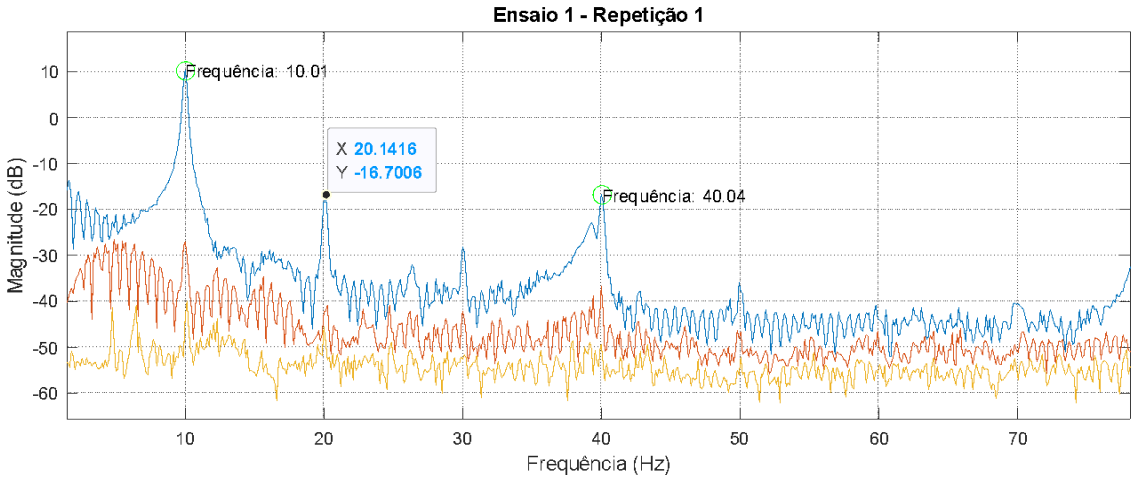
No Ensaio 1, obtivemos um total de 6 picos, sendo o ensaio que apresentou o maior número de picos entre todos os ensaios realizados. Destacam-se alguns pontos notáveis, incluindo a presença de um pico em 20 Hz em todas as repetições, ausente no modelo calculado, mostrado na Figura 47. Na nona repetição, não houve a aparição do 5º modo de vibração. A magnitude dos modos 6 e 7 variou consideravelmente entre as repetições, algumas não tão evidentes quanto a da repetição 1. As três curvas do gráfico surgem do resultado do FDD, que são os valores singulares (Sv).

Figura 47 - Resultado FDD ensaio 1, repetição 1



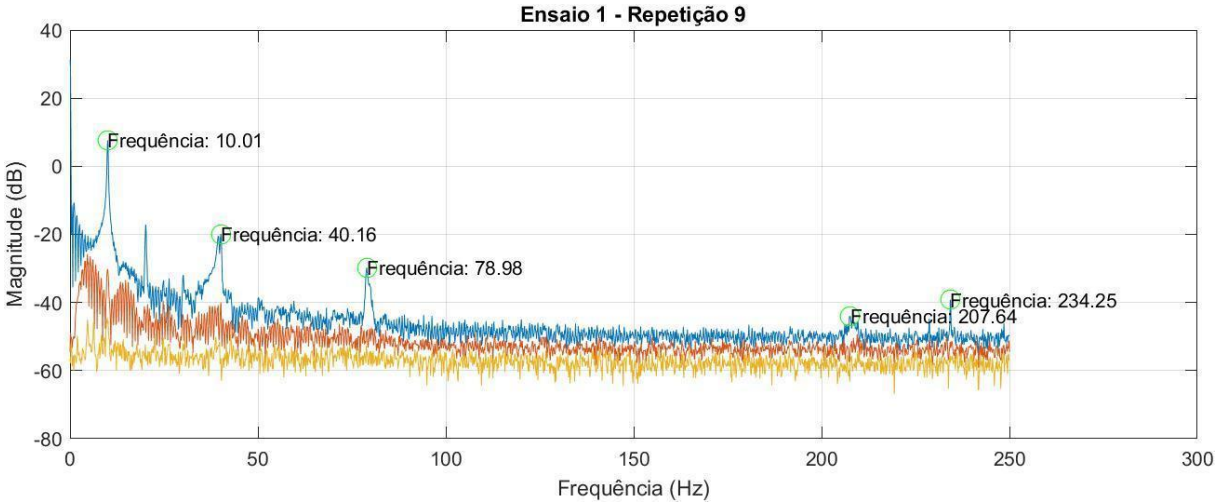
Fonte: Autores (2024).

Figura 48 - Detalhe para pico recorrente nas medições, ensaio 1 e repetição 1



Fonte: Autores (2024).

Figura 49 - Resultado FDD ensaio 1, repetição 9



Fonte: Autores (2024).

Tabela 7 - Frequências aferidas no ensaio 1

Repetição	1º Modo (Hz)	2º Modo (Hz)	3º Modo (Hz)	5º Modo (Hz)	6º Modo (Hz)	7º Modo (Hz)
1	10,010	40,039	78,857	125,244	208,008	234,375
2	10,010	40,039	78,857	125,610	210,205	234,375
3	10,010	39,795	78,857	125,366	209,229	234,619
4	10,010	39,429	78,857	119,629	209,106	234,497
5	10,010	39,307	78,857	125,244	209,106	234,619
6	10,010	39,429	78,857	125,488	206,665	234,497
7	10,010	39,429	78,979	125,488	208,740	234,497
8	10,010	39,429	78,857	125,244	208,740	234,497
9	10,010	40,161	78,979	0,000	207,642	234,253
10	10,010	39,551	78,979	125,244	207,642	234,375

Fonte: Autores (2024).

Tabela 8 - Estatísticas ensaio 1

Frequências	1º Modo	2º Modo	3º Modo	5º Modo	6º Modo	7º Modo
Média (Hz)	10,010	39,661	78,894	112,256	208,508	234,460
Desvio Padrão (Hz)	0,000	0,318	0,059	39,484	1,020	0,116
Covariância (%)	0,000	0,801	0,075	35,173	0,489	0,049

Fonte: Autores (2024).

Ao analisar as tabelas, é notado que o desvio padrão e covariância do primeiro modo foram praticamente nulos em quase todos os ensaios. O 5º modo, apesar de apresentar um desvio padrão elevado, foi considerado zero devido à ausência de pico na nona repetição. Desconsiderando essa repetição específica, a média das frequências foi de 124,720 Hz, com

desvio padrão de 1,917 e covariância de 1,537%, destacando-se como o ensaio com maior desvio e covariância.

Tabela 9 - Comparação entre os dados obtidos e calculados, ensaio 1

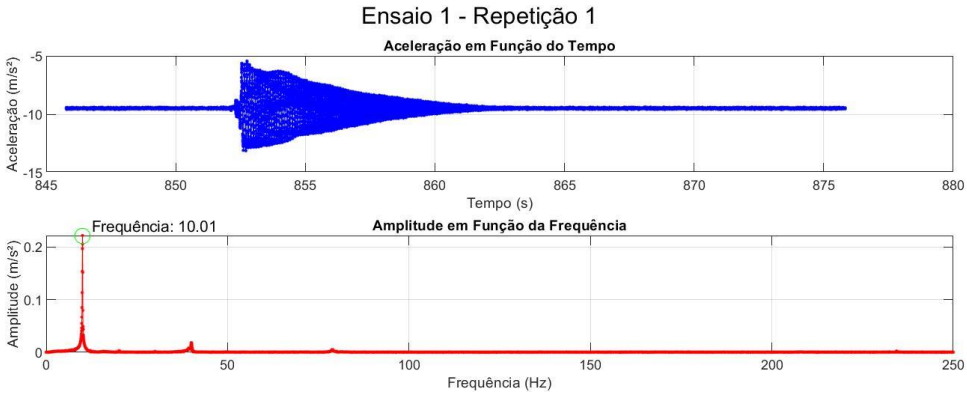
Frequências	1º Modo	2º Modo	3º Modo	5º Modo	6º Modo	7º Modo
Média (Hz)	10,010	39,661	78,894	112,256	208,508	234,460
SAP2000 (Hz)	9,872	36,562	69,946	130,569	190,462	278,865
Diferença relativa (%)	1,39	8,47	12,7	4,47	9,47	15,92

Fonte: Autores (2024).

Além disso, é observada uma diferença relativa superior a 10% no terceiro e sétimo modo. Mesmo com a presença consistente do sétimo pico em todas as repetições, o método do FDD tem uma limitação para essa frequência, que abrange apenas até a metade da frequência total, alcançando 250 Hz, enquanto a frequência pelo SAP2000 esperada seria 278 Hz.

O gráfico da aceleração no domínio do tempo exibe o comportamento característico de uma vibração amortecida. O valor do primeiro modo obtido experimentalmente, correspondente a 10,01 Hz, está próximo ao valor obtido pelo modelo no SAP2000.

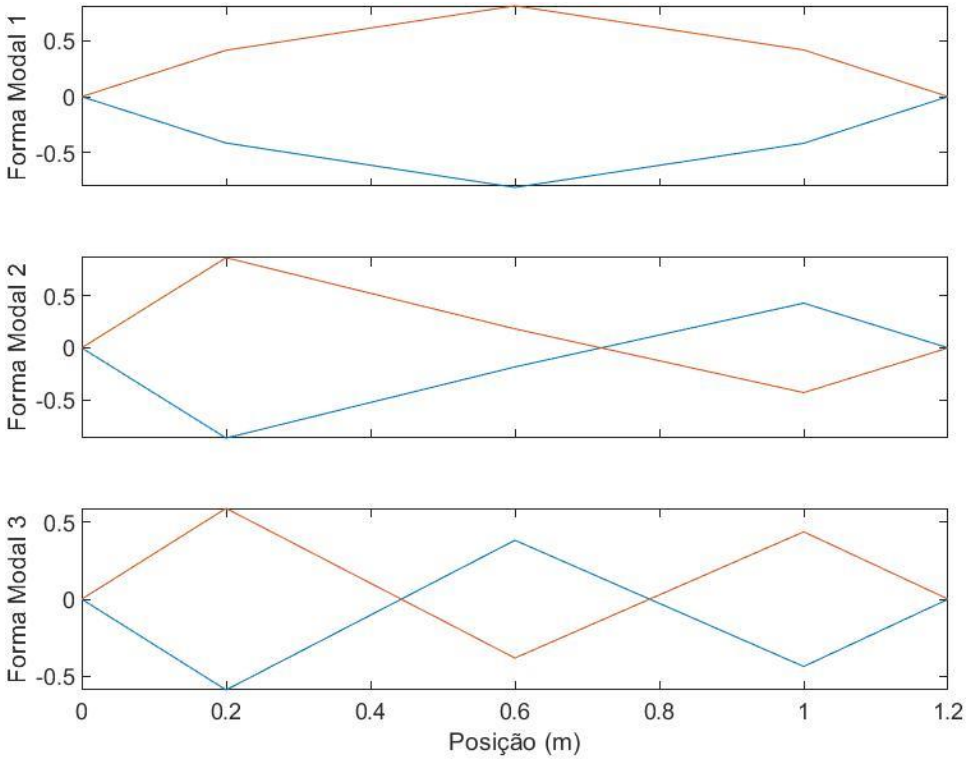
Figura 50 - Aceleração no domínio do tempo e da frequência, ensaio 1



Fonte: Autores (2024).

As formas modais obtidas experimentalmente estão em concordância com as formas modais obtida pelo SAP2000.

Figura 51 - Formas modais obtidas pelo FDD, ensaio 1

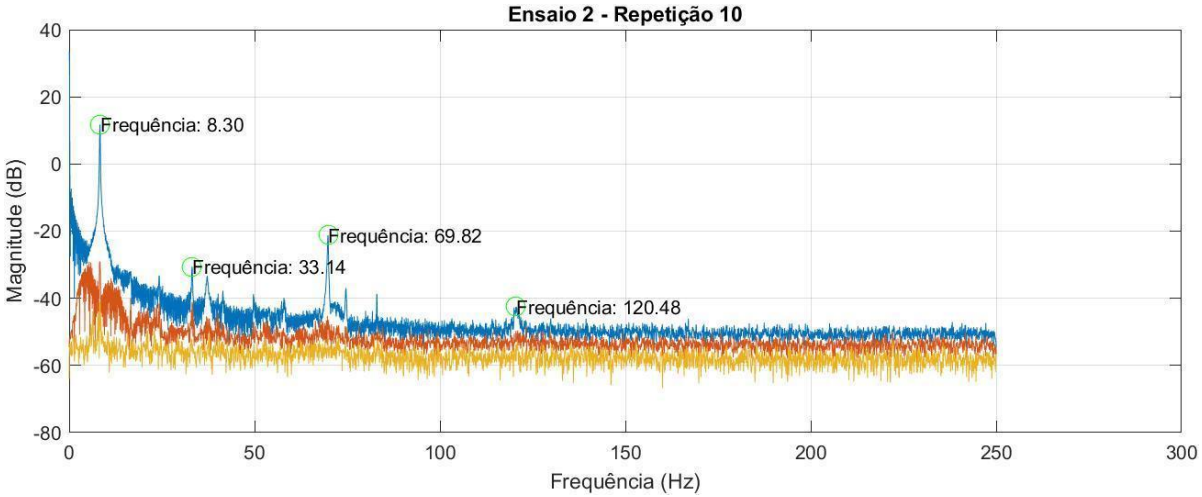


Fonte: Autores (2024).

5.1.1.2. Ensaio 2

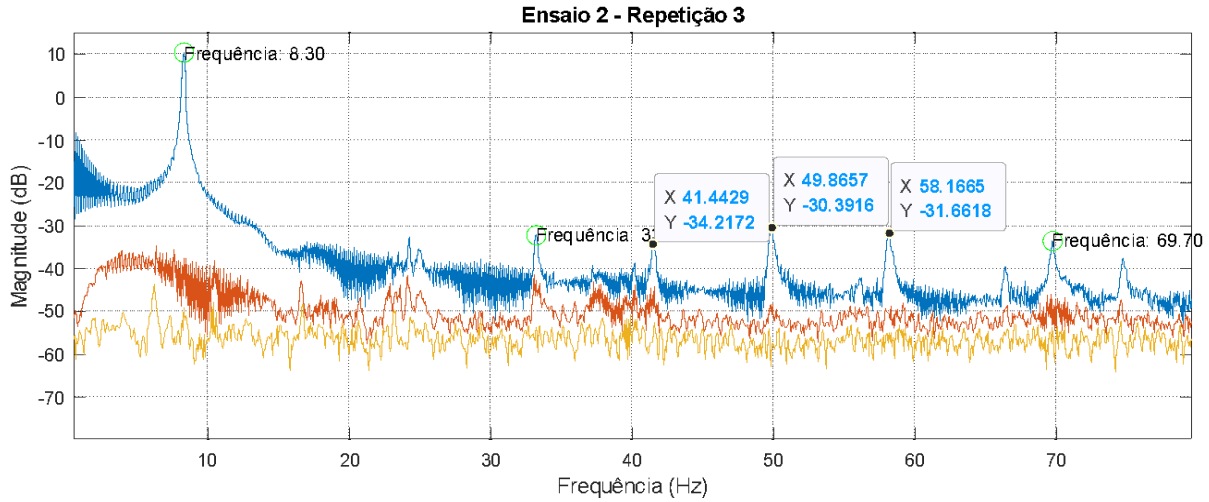
No decorrer do Ensaio 2, foram identificados quatro picos significativos. Na Figura 52, destacam-se três picos (41 Hz, 50 Hz e 58 Hz) que não foram previstos pelo modelo no SAP2000, persistindo em várias repetições. A magnitude do 5º modo apresentou variações em algumas repetições, sendo menos evidente do que na repetição 10, conforme ilustrado na Figura 53. O gráfico do FDD mostrou uma anomalia com vários picos, o que foi inesperado, considerando que todas as repetições foram realizadas simultaneamente, e somente essa repetição apresentou esse comportamento atípico, logo repetição 1 Figura 54 foi desconsiderada.

Figura 52 - Resultado FDD ensaio 2, repetição 10



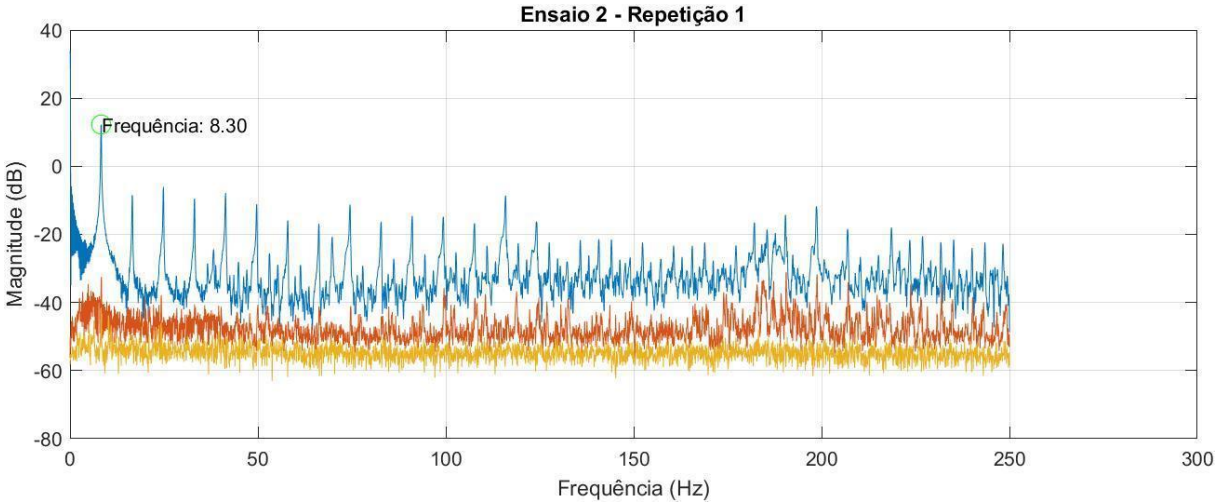
Fonte: Autores (2024).

Figura 53 - Resultado FDD ensaio 2, repetição 3



Fonte: Autores (2024).

Figura 54 - Resultado FDD ensaio 2, repetição 1



Fonte: Autores (2024).

Tabela 10 - Frequências aferidas no ensaio 2

Repetição	1º Modo (Hz)	2º Modo (Hz)	3º Modo (Hz)	5º Modo (Hz)
1	-	-	-	-
2	8,301	33,325	66,406	124,512
3	8,301	33,203	69,702	120,544
4	8,301	33,203	69,702	124,451
5	8,301	33,142	66,406	124,573
6	8,301	37,170	69,824	120,422
7	8,301	33,203	69,702	124,146
8	8,301	33,325	69,824	120,483
9	8,301	37,109	69,824	123,779
10	8,301	33,142	69,824	120,483

Fonte: Autores (2024).

Tabela 11 - Estatísticas ensaio 2

Frequências	1º Modo	2º Modo	3º Modo	5º Modo
Média (Hz)	8,301	34,092	69,024	122,599
Desvio Padrão (Hz)	0,000	1,730	1,485	2,021
Covariância (%)	0,000	5,073	2,152	1,649

Fonte: Autores (2024).

A repetição 1 foi excluída do cálculo estatístico para os modos diferentes de 1. A consistência nas frequências medidas evidenciou um comportamento modal uniforme, com o primeiro modo mantendo-se constante em 8,301 Hz em todas as repetições. A análise estatística revelou médias de frequências bem definidas, indicando consistência modal. O baixo desvio padrão sugere baixa dispersão das medições em relação à média, implicando em boa reprodutibilidade das condições experimentais.

Tabela 7 - Comparação entre os dados obtidos e calculados, ensaio 2

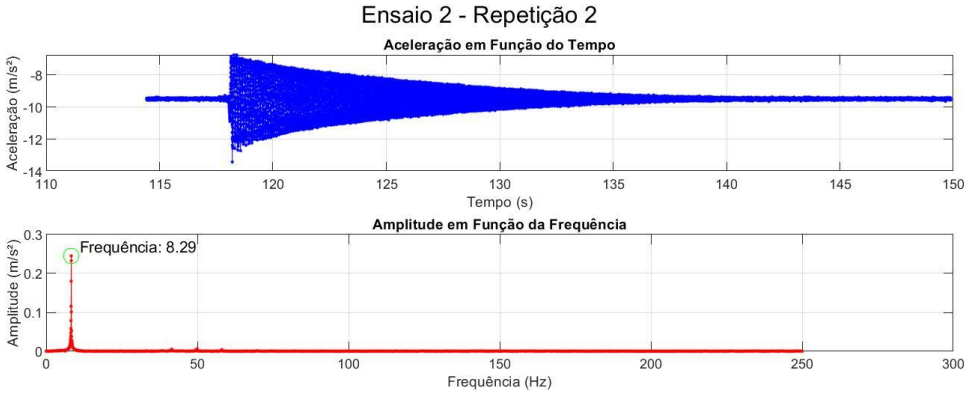
Frequências	1º Modo	2º Modo	3º Modo	5º Modo
Média (Hz)	8,301	34,092	69,024	122,599
SAP2000 (Hz)	8,016	36,562	62,854	120,444
Diferença relativa (%)	3,56	6,76	9,82	1,79

Fonte: Autores (2024).

Os resultados foram avaliados em relação ao modelo do SAP2000, revelando alta precisão para a maioria dos modos de vibração. O primeiro modo apresentou uma diferença relativa de 3,56%, indicando concordância significativa entre as frequências medidas e previstas. A covariância percentual demonstrou baixa variabilidade das medições em relação à média, corroborando a consistência observada nas repetições.

O gráfico da aceleração no domínio do tempo exibe o comportamento característico de uma vibração amortecida. O valor do primeiro modo obtido experimentalmente, correspondente a 8,29 Hz, está próximo ao valor obtido pelo modelo no SAP2000.

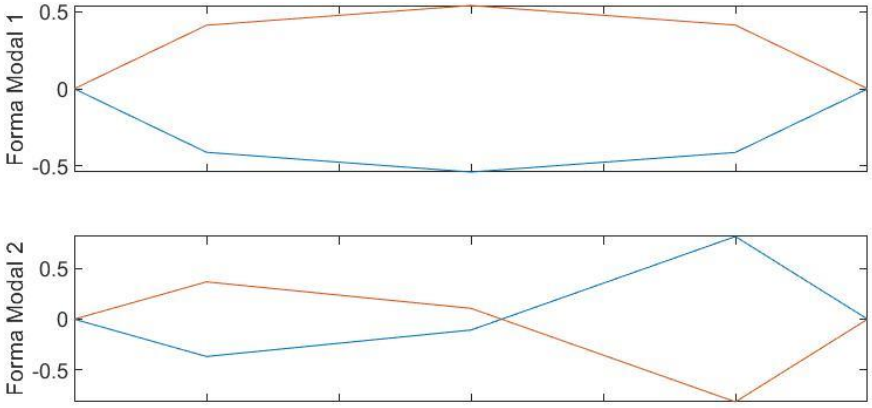
Figura 55 - Aceleração no domínio do tempo e da frequência, ensaio 2



Fonte: Autores (2024).

As primeiras formas modais (modos 1 e 2) obtidas experimentalmente estão em concordância com as formas modais obtida pelo SAP2000, porém a forma modal 3 não obteve o comportamento esperado para esse ensaio.

Figura 56 - Formas modais obtidas pelo FDD, ensaio 2

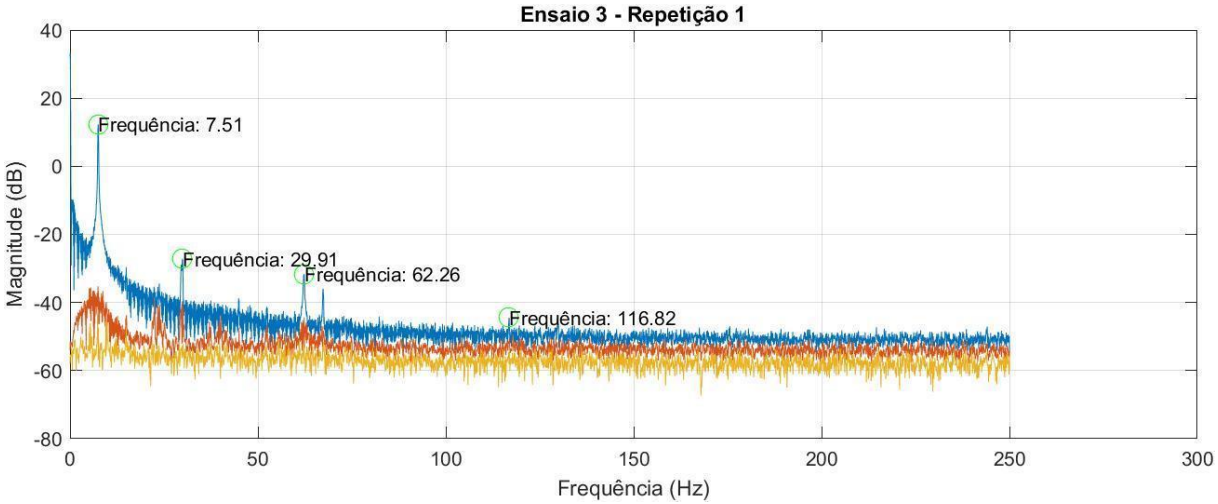


Fonte: Autores (2024).

5.1.1.3. Ensaio 3

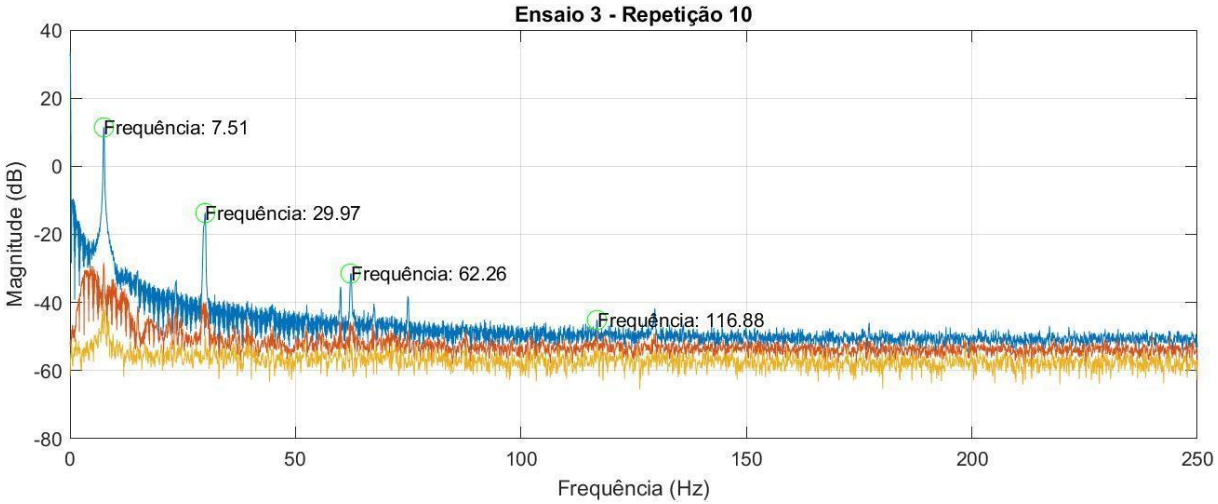
No Ensaio 3, foram identificados 4 picos significativos. Destaca-se que a magnitude do 5º modo foi relativamente pequena, e na repetição 10, como evidenciado na Figura 57, a identificação desse pico tornou-se difícil.

Figura 57 - Resultado FDD ensaio 3, repetição 1



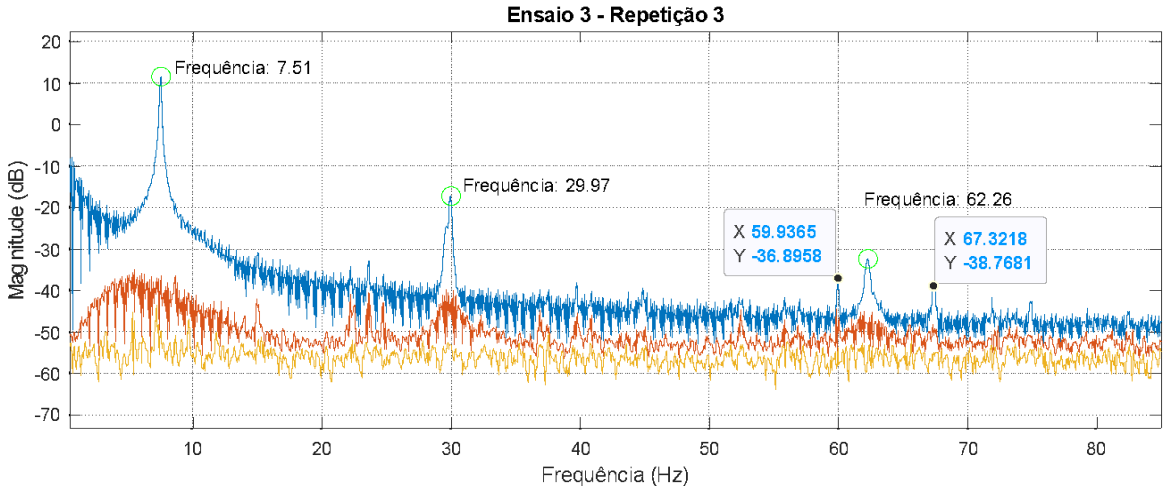
Fonte: Autores (2024).

Figura 58 - Resultado FDD ensaio 3, repetição 10



Fonte: Autores (2024).

Figura 59 - Resultado FDD ensaio 3, repetição 3



Fonte: Autores (2024).

Tabela 12 - Frequências aferidas no ensaio 3

Repetição	1º Modo (Hz)	2º Modo (Hz)	3º Modo (Hz)	5º Modo (Hz)
1	7,507	29,907	62,256	116,821
2	7,507	29,968	62,256	116,760
3	7,507	29,968	62,256	115,417
4	7,568	30,029	62,256	116,821
5	7,507	29,968	62,256	116,760
6	7,507	29,968	62,256	116,821
7	7,446	29,907	62,256	116,821
8	7,507	29,907	59,937	116,211
9	7,507	30,029	62,317	116,821
10	7,507	29,968	62,256	116,882

Fonte: Autores (2024).

Tabela 13 - Estatísticas ensaio 3

Frequências	1º Modo	2º Modo	3º Modo	5º Modo
Média (Hz)	7,507	29,962	62,030	116,614
Desvio Padrão (Hz)	0,029	0,045	0,736	0,462
Covariância (%)	0,383	0,150	1,186	0,396

Fonte: Autores (2024).

As repetições do Ensaio 3 revelaram consistência nas frequências medidas para cada modo de vibração. Embora haja variação nos valores, a média parece ser uma representação geral precisa. A análise estatística evidencia estabilidade nas medições, com baixa dispersão em relação à média, indicando boa reprodutibilidade das condições experimentais.

Tabela 14 - Comparação entre os dados obtidos e calculados, ensaio 3

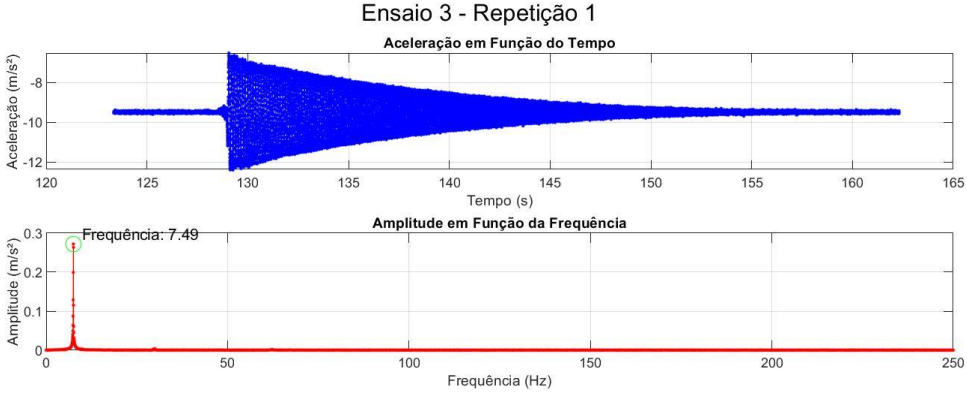
Frequências	1º Modo	2º Modo	3º Modo	5º Modo
Média (Hz)	7,507	29,962	62,030	116,614
SAP2000 (Hz)	7,193	28,719	57,694	105,092
Diferença relativa (%)	4,37	4,33	7,52	10,96

Fonte: Autores (2024).

Os resultados foram comparados ao modelo do SAP2000, demonstrando uma alta proximidade para a maioria dos modos de vibração. A diferença relativa aumentou em modos mais altos. A covariância percentual sugere uma variabilidade relativamente baixa das medições em relação à média, reforçando a consistência observada nas repetições.

O gráfico da aceleração no domínio do tempo exibe o comportamento característico de uma vibração amortecida. O valor do primeiro modo obtido experimentalmente, correspondente a 7,49, está próximo ao valor obtido pelo modelo no SAP2000.

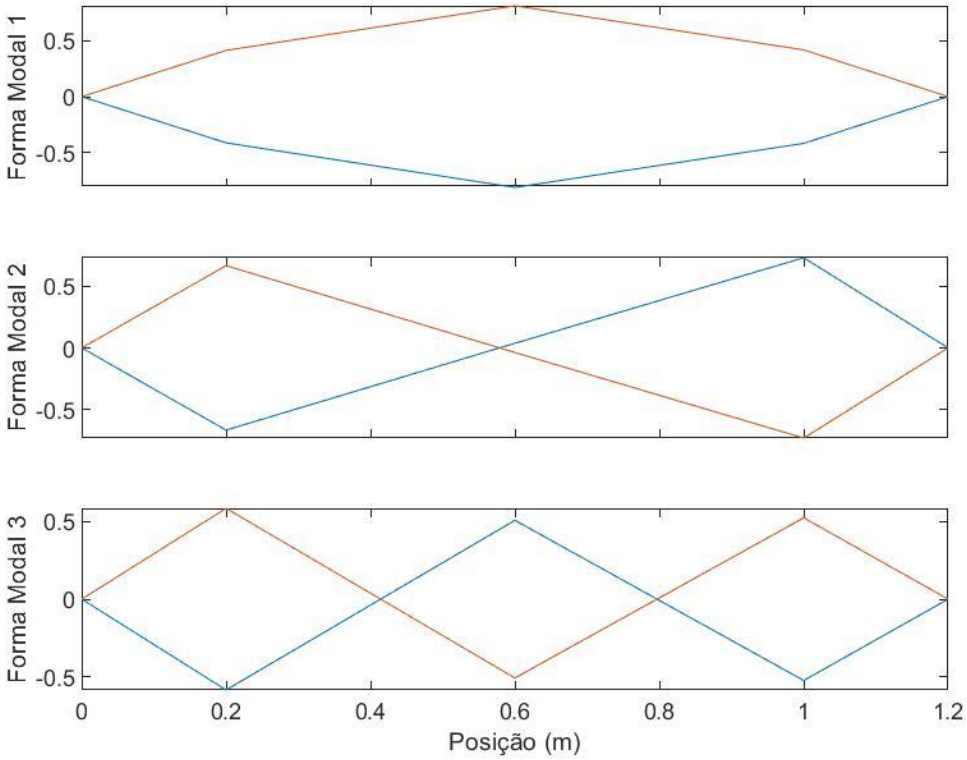
Figura 60 - Aceleração no domínio do tempo e da frequência, ensaio 3



Fonte: Autores (2024).

As formas modais obtidas experimentalmente estão em concordância com as formas modais obtida pelo SAP2000.

Figura 61 - Formas modais obtidas pelo FDD, ensaio 3

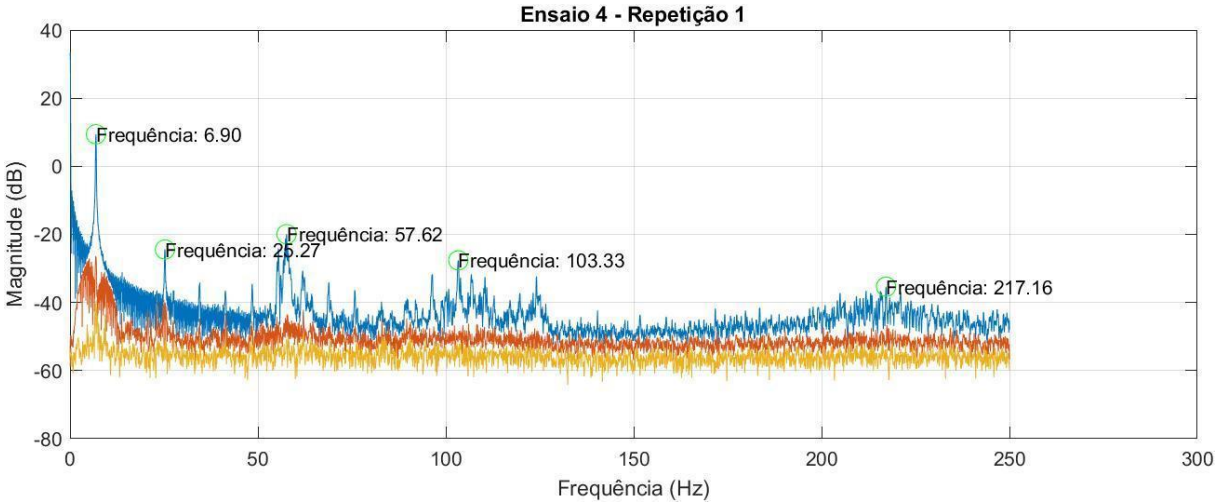


Fonte: Autores (2024).

5.1.1.4. Ensaio 4

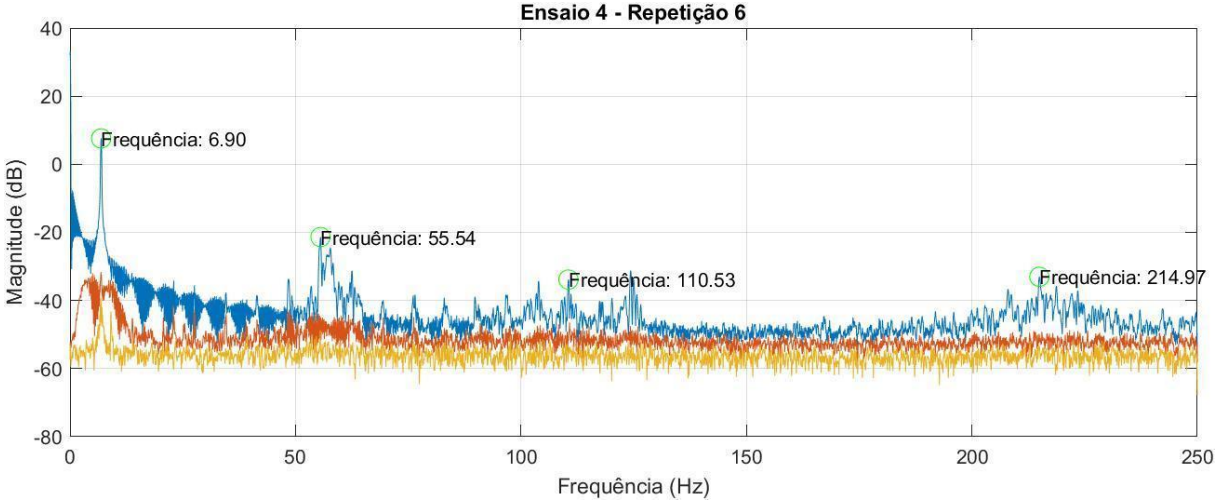
No Ensaio 4, foram observados cinco picos ao longo das repetições. A identificação do 5º modo mostrou-se difícil, pois estava frequentemente cercada por dois ou mais picos com magnitudes semelhantes. Além disso, a magnitude do 7º modo foi bastante reduzida, o que poderia dificultar sua detecção, especialmente em uma abordagem automatizada. Em metade das repetições, a magnitude do segundo modo foi pequena, enquanto na outra parte, ela foi clara. A repetição 6, conforme ilustrado na Figura 62, não apresentou pico.

Figura 62 - Resultado FDD ensaio 4, repetição 1



Fonte: Autores (2024).

Figura 63 - Resultado FDD ensaio 4, repetição 6



Fonte: Autores (2024).

Tabela 15 - Frequências aferidas no ensaio 4

Repetição	1º Modo (Hz)	2º Modo (Hz)	3º Modo (Hz)	5º Modo (Hz)	7º Modo (Hz)
1	6,897	25,269	57,617	103,333	217,163
2	6,897	25,208	57,251	103,394	213,135
3	6,897	25,269	57,495	106,628	223,572
4	6,897	34,485	55,481	107,239	217,773
5	6,897	25,208	57,617	103,882	223,450
6	6,897	-	55,542	110,535	214,966
7	6,836	25,269	57,129	109,741	217,407
8	-	-	-	-	-
9	6,897	25,330	55,481	110,474	222,656
10	6,897	34,546	57,434	110,229	220,215

Fonte: Autores (2024).

Tabela 16 - Estatísticas ensaio 4

Frequências	1º Modo	2º Modo	3º Modo	5º Modo	7º Modo
Média (Hz)	6,890	27,573	56,783	107,273	218,926
Desvio Padrão (Hz)	0,020	4,285	0,974	3,127	3,768
Covariância (%)	0,295	15,542	1,716	2,915	1,721

Fonte: Autores (2024).

No ensaio 8, os dados incompletos foram desconsiderados, assim como na sexta repetição, onde não foi identificado o pico no cálculo das estatísticas para o segundo modo. As repetições exibiram variações nas frequências medidas, especialmente nos 2º e 5º modos, indicando uma certa variabilidade. As análises de média, desvio padrão e covariância sugerem que, em geral, as medições são relativamente estáveis, com uma maior variabilidade no 2º modo.

Tabela 17 - Comparação entre os dados obtidos e calculados, ensaio 4

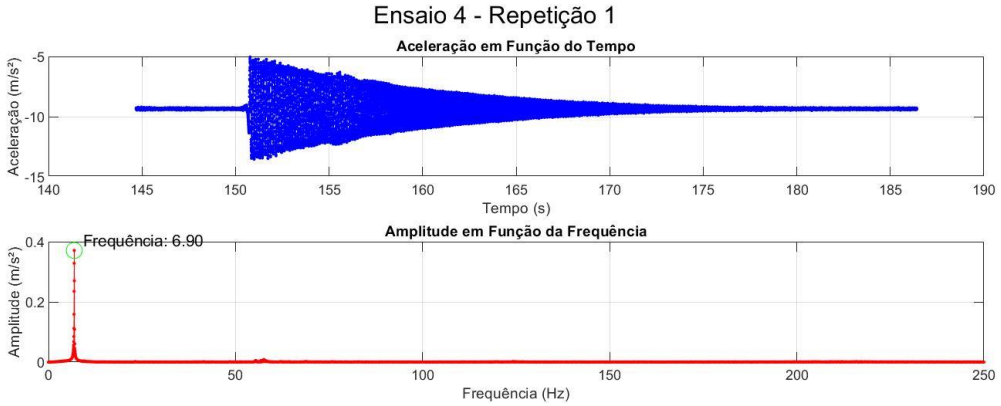
Frequências	1º Modo	2º Modo	3º Modo	5º Modo	7º Modo
Média (Hz)	6,890	27,573	56,783	107,273	218,926
SAP2000 (Hz)	7,193	28,719	57,694	105,092	242,090
Diferença relativa (%)	4,75	13,20	4,32	7,54	9,57

Fonte: Autores (2024).

A diferença relativa do modelo experimental em relação ao SAP2000 é razoavelmente alta, indicando uma concordância substancial entre as frequências medidas e as previstas pelo modelo. Contudo, observa-se uma menor precisão nos 2º e 7º modos.

O gráfico da aceleração no domínio do tempo exibe o comportamento característico de uma vibração amortecida. O valor do primeiro modo obtido experimentalmente, correspondente a 6,09 Hz, está próximo ao valor obtido pelo modelo no SAP2000.

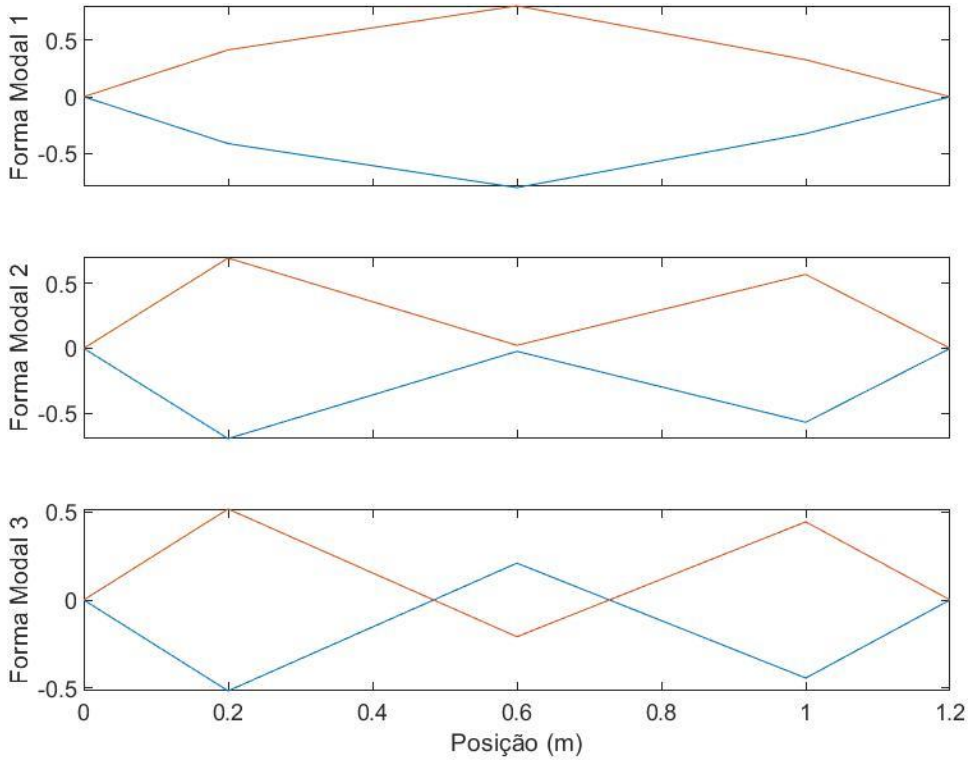
Figura 64 - Aceleração no domínio do tempo e da frequência, ensaio 4



Fonte: Autores (2024).

As formas modais obtidas experimentalmente estão em concordância com as formas modais obtida pelo SAP2000.

Figura 65 - Formas modais obtidas pelo FDD, ensaio 4

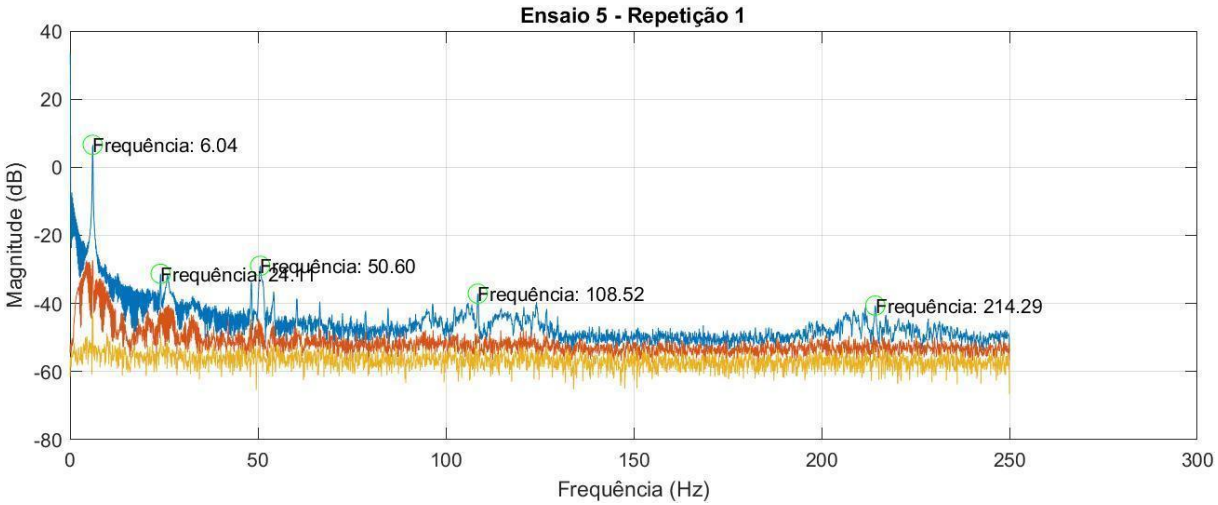


Fonte: Autores (2024).

5.1.1.5. Ensaio 5

O Ensaio revelou a presença de cinco picos significativos ao longo das repetições. Excluindo o primeiro e segundo modo, os picos teóricos frequentemente vieram acompanhados de picos com magnitudes semelhantes, o que dificultou a análise para determinar a frequência específica de cada modo de vibração.

Figura 66 - Resultado FDD ensaio 5, repetição 1



Fonte: Autores (2024).

Tabela 18 - Frequências aferidas no ensaio 5

Repetição	1º Modo (Hz)	2º Modo (Hz)	3º Modo (Hz)	5º Modo (Hz)	7º Modo (Hz)
1	6,042	24,109	50,598	108,521	214,294
2	6,042	24,048	51,086	120,178	213,013
3	6,042	24,048	50,781	120,178	214,722
4	6,042	24,048	51,147	108,093	216,614
5	6,042	24,109	50,964	121,521	214,600
6	6,042	24,109	50,842	108,765	210,754
7	6,042	24,109	50,415	108,887	214,294
8	6,042	25,818	50,720	108,704	210,815
9	6,042	24,109	50,903	108,704	210,754
10	6,042	24,170	50,537	108,948	212,402

Fonte: Autores (2024).

Tabela 19 - Estatísticas ensaio 5

Frequências	1º Modo	2º Modo	3º Modo	5º Modo	7º Modo
Média (Hz)	6,042	24,268	50,800	112,250	213,226
Desvio Padrão (Hz)	0,000	0,546	0,237	5,796	2,018
Covariância (%)	0,000	2,250	0,467	5,164	0,946

Fonte: Autores (2024).

As repetições apresentaram a mesma frequência para o 1º modo, indicando uma estabilidade considerável nesse modo. Variações nas frequências dos modos superiores sugerem uma certa variabilidade nas medições, sendo mais notável nos modos acima do 2º. O desvio padrão notavelmente baixo para o 1º modo sugere uma consistência robusta.

Tabela 20 - Comparação entre os dados obtidos e calculados, ensaio 5

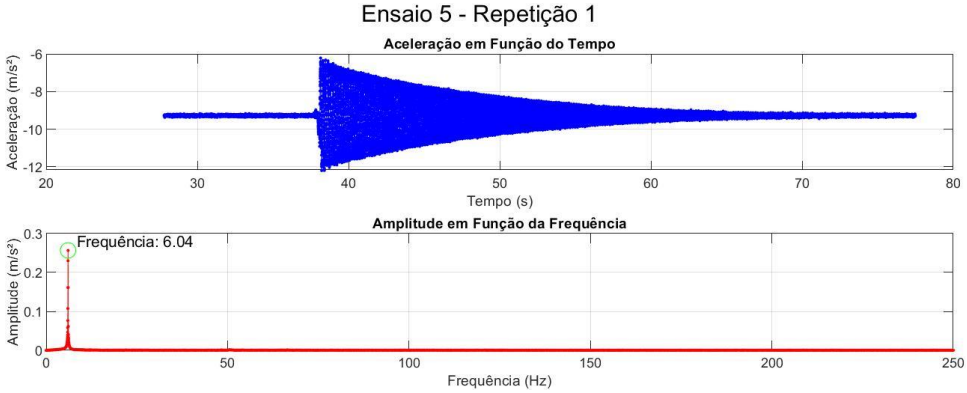
Frequências	1º Modo	2º Modo	3º Modo	5º Modo	7º Modo
Média (Hz)	6,042	24,268	50,800	112,250	213,226
SAP2000 (Hz)	5,709	24,356	47,794	97,739	236,316
Diferença relativa (%)	5,83	0,37	6,29	14,8	9,77

Fonte: Autores (2024).

A precisão em relação ao modelo do SAP2000 é geralmente alta, especialmente para o 2º modo. Modos superiores mostram uma precisão um pouco menor, mas ainda acima de 87%.

O gráfico da aceleração no domínio do tempo exibe o comportamento característico de uma vibração amortecida. O valor do primeiro modo obtido experimentalmente, correspondente a 6,04, está próximo ao valor obtido pelo modelo no SAP2000.

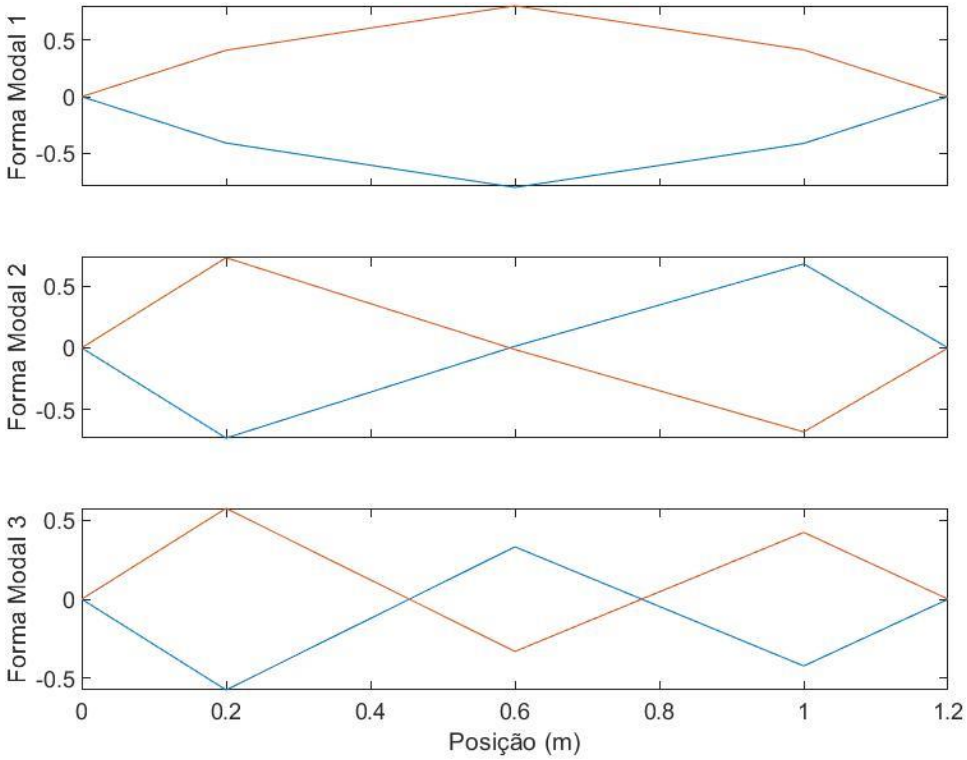
Figura 67 - Aceleração no domínio do tempo e da frequência, ensaio 5



Fonte: Autores (2024).

As formas modais obtidas experimentalmente estão em concordância com as formas modais obtida pelo SAP2000.

Figura 68 - Formas modais obtidas pelo FDD, ensaio 5



Fonte: Autores (2024).

5.1.1.6. Análise global do modelo

Os resultados obtidos nos cinco ensaios realizados na viga metálica revelam uma variedade de comportamentos modais sob diferentes condições de carregamento. Ao observar as médias das frequências modais Tabela 21, notamos padrões distintos em cada ensaio. O primeiro modo de vibração, geralmente mais estável, apresentou frequências consistentes, enquanto os modos superiores demonstraram maior variabilidade entre as repetições.

Tabela 21 - Resumo dos ensaios, chapa metálica

Ensaio	1º Modo (Hz)	2º Modo (Hz)	3º Modo (Hz)	5º Modo (Hz)	6º Modo (Hz)	7º Modo (Hz)
1	10,010	39,661	78,894	124,729	208,508	234,460
2	8,301	37,109	69,824	123,779	-	-
3	7,507	29,968	62,256	116,882	-	-
4	6,890	27,573	56,783	107,273	-	218,926
5	6,042	24,268	50,800	112,250	-	213,226

Fonte: Autores (2024).

Comparando as frequências obtidas experimentalmente com aquelas previstas pelo modelo do SAP2000. É notável a eficácia do método de FDD na identificação dos modos, embora alguns modos específicos possam apresentar menor precisão.

Tabela 22 - Resumo da modelagem no SAP2000

Ensaio	1º Modo (Hz)	2º Modo (Hz)	3º Modo (Hz)	5º Modo (Hz)	6º Modo (Hz)	7º Modo (Hz)
1	9,872	36,562	69,946	130,569	190,462	278,865
2	8,016	36,562	62,854	120,444	-	-
3	7,193	28,719	57,694	105,092	-	-
4	6,578	24,356	54,433	99,748	-	242,090
5	5,709	24,356	47,794	97,739	-	236,316

Fonte: Autores (2024).

A tabela abaixo mostra os resultados percentuais da diferença relativa entre os acelerômetros MPU e Vernier nos diferentes modos e ensaios. Ao analisar os resultados dos ensaios, fica evidente que os acelerômetros MPU e Vernier apresentam valores muito próximos.

Tabela 23 - Comparação entre MPU e Low g accelerometer

Equipamentos	Ensaio	1º Modo (Hz)	2º Modo (Hz)	3º Modo (Hz)	5º Modo (Hz)	6º Modo (Hz)
MPU6050	1	10.010	39.661	78.894	124.729	208.508
	2	8.301	34.092	69.024	122.599	-
	3	7.507	29.962	62.030	116.614	-
	4	6.890	27.573	56.783	107.273	-
	5	6.042	24.268	50.800	112.250	-
Low g accelerometer	1	10.010	39.697	78.992	123.947	209.699
	2	8.301	33.196	68.685	123.561	-
	3	7.483	29.944	62.262	116.937	-
	4	6.885	26.880	56.818	110.400	-
	5	6.042	24.097	52.435	117.706	-
Diferença relativa (%)	1	0,00	0,09	0,12	0,63	0,57
	2	0,00	2,63	0,49	0,78	-
	3	0,33	0,06	0,37	0,28	-
	4	0,08	2,51	0,06	2,92	-
	5	0,00	0,70	3,22	4,86	-

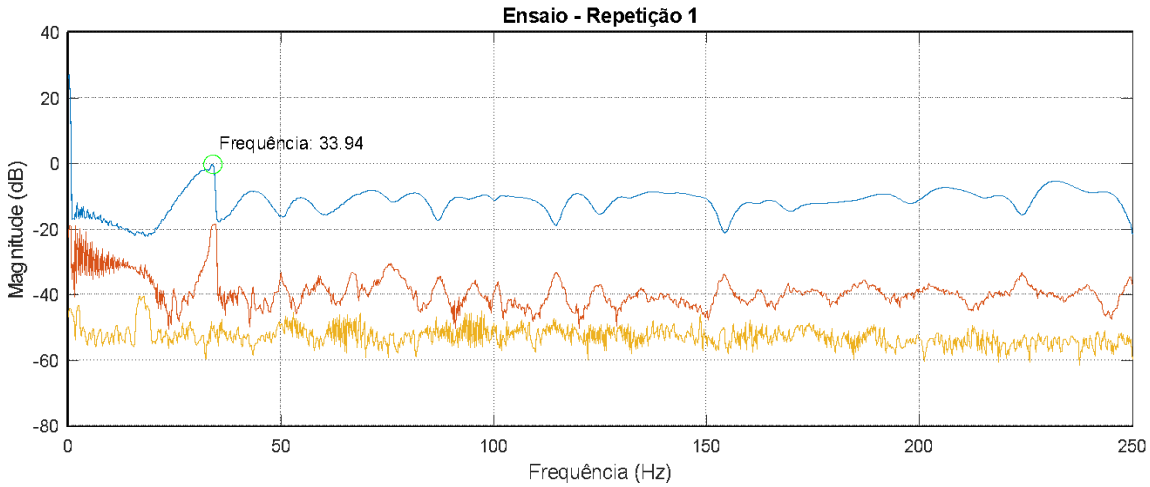
Fonte: Autores (2024).

Destacando o 1º Modo, a relação entre MPU e Low g accelerometer atinge 0,00%, indicando uma concordância completa. Mesmo nos casos de menor aproximação, como no 4º Modo durante o 5º ensaio, onde o percentual é de 4,86%, a concordância ainda é considerável.

5.1.2. Modelo 2 – Viga I

Durante o ensaio da viga I, apenas um pico foi observado ao longo das repetições. O segundo modo de vibração, conforme obtido pelo SAP2000 é de 162Hz, no entanto, não foram encontrados picos representativos durante as repetições, o que tornou a identificação deste modo de vibração inviável. Além disso, os demais modos possuem frequências superiores a 250Hz, o que torna a identificação por meio do método do FDD impossível.

Figura 69 - Resultado FDD viga I



Fonte: Autores (2024).

Tabela 24 - Resultados ensaio viga I

Ensaio	1º Modo (Hz)
1	33,936
2	33,203
3	33,203
4	33,936
5	33,936
6	33,447
7	33,813
8	33,936
9	33,936
10	34,180

Fonte: Autores (2024).

Ao analisar estatisticamente esses dados, a média das frequências do primeiro modo foi calculada como 33.752 Hz, com um desvio padrão de 0.342 Hz e uma covariância de 1.012%.

Tabela 25 - Estatísticas ensaio viga I

Frequências	1º Modo
Média	33,752 Hz
Desvio Padrão	0,342 Hz
Covariância (%)	1,012%

Fonte: Autores (2024).

A comparação com o modelo do SAP2000, que possui uma frequência de 40,801 Hz para o primeiro modo, revelou uma diferença relativa de 17,28%. Essa diferença pode ser atribuída à variação na forma como a perturbação foi aplicada durante o ensaio, utilizando uma batida com um martelo em vez de um deslocamento inicial, o que pode impactar as respostas dinâmicas da viga I.

É relevante mencionar que a viga I apresentava um leve empenamento. Para estabilização, foram utilizados calços, introduzindo uma imperfeição geométrica no sistema. As medidas das seções foram ajustadas para valores comerciais, conforme tabelas padrão, o que pode ter contribuído para as diferenças observadas na precisão, considerando a sensibilidade dessa estrutura a imperfeições geométricas e ajustes nas dimensões.

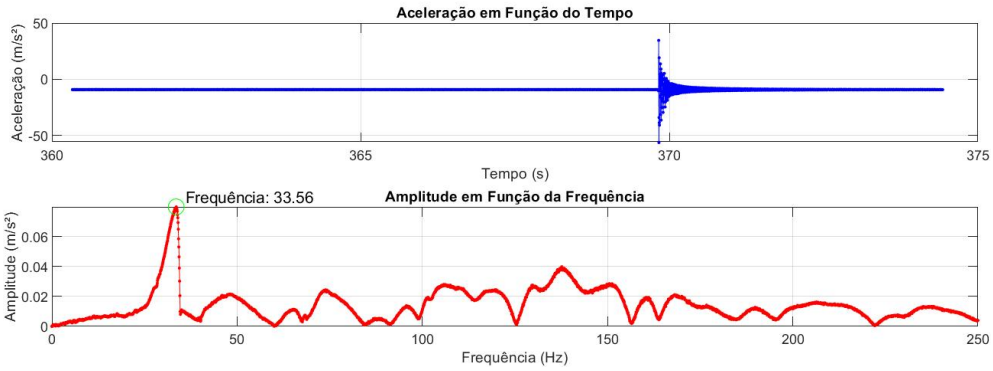
Tabela 26 - Comparação entre modelo teórico e experimental viga I

Frequências	1º Modo
Média	33,752 Hz
SAP2000 (Hz)	40,801 Hz
Diferença relativa (%)	17,28%

Fonte: Autores (2024).

O gráfico da aceleração no domínio do tempo exibe o comportamento característico de uma vibração amortecida. O valor do primeiro modo obtido experimentalmente, correspondente a 33,66 Hz, levando em consideração as condições do ensaio, o valor obtido pelo cálculo teórico está próximo.

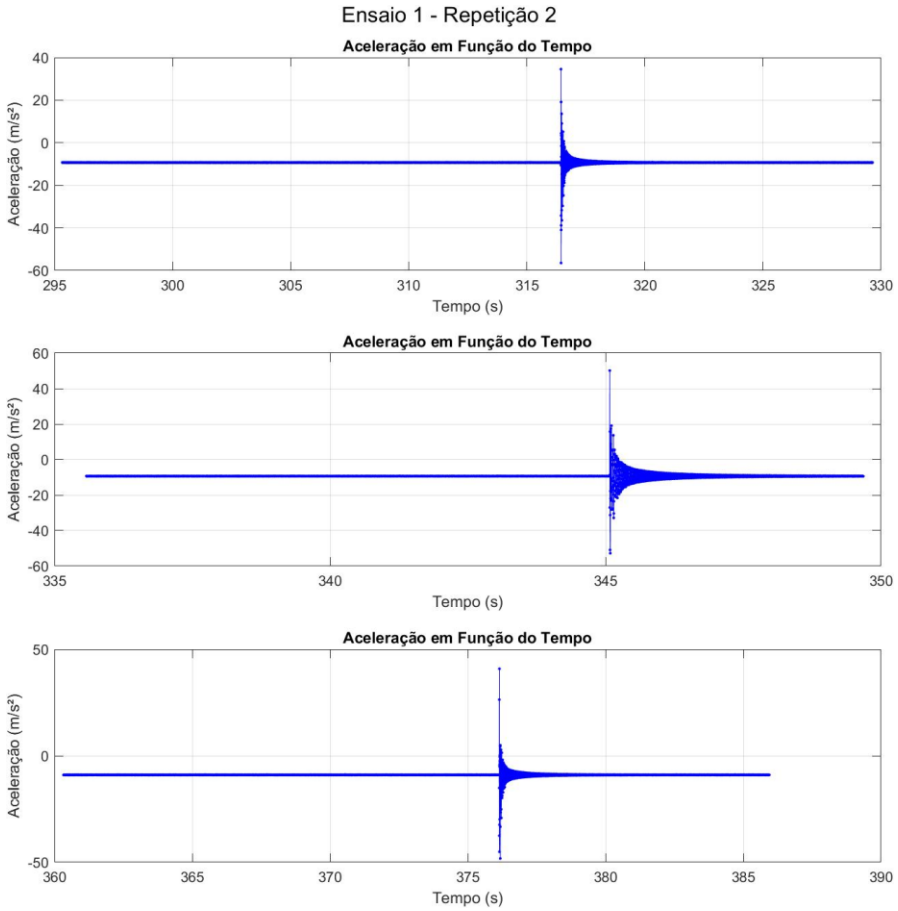
Figura 70 - Aceleração no domínio do tempo e da frequência



Fonte: Autores (2024).

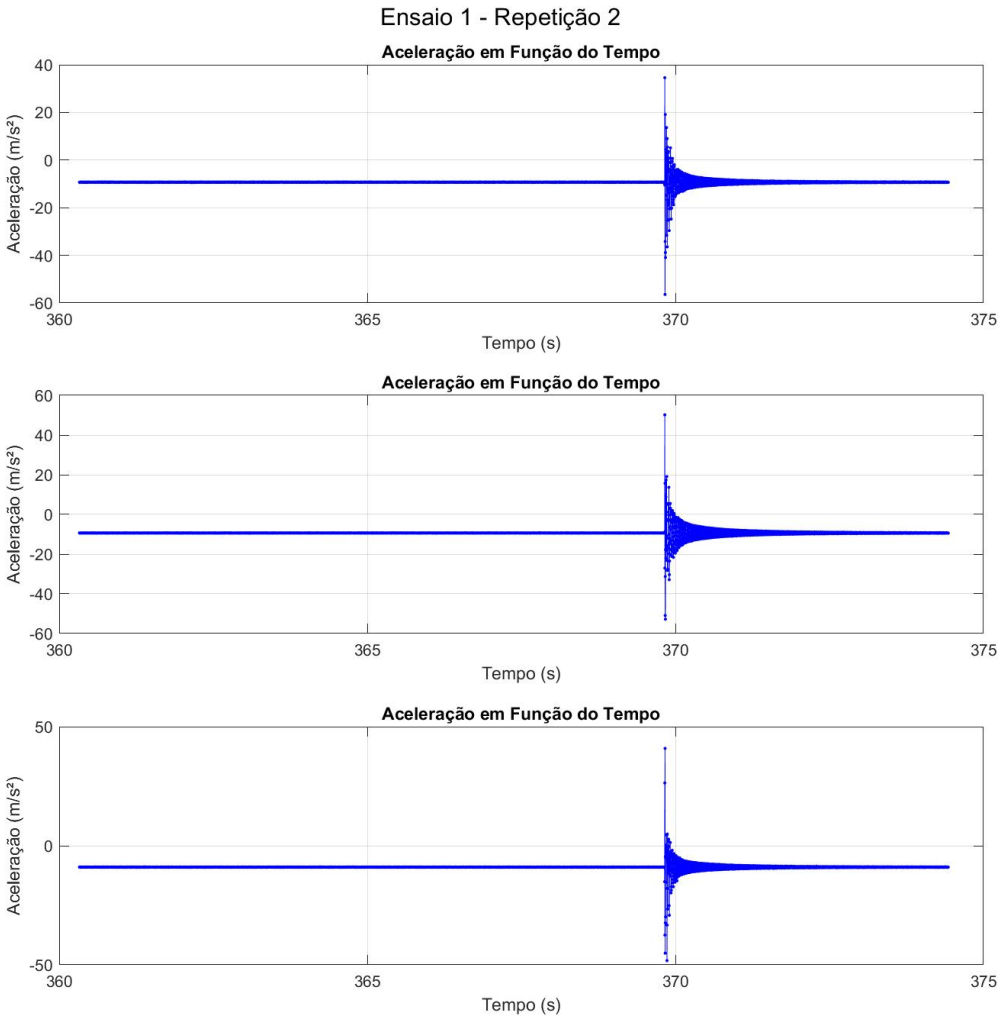
Como os acelerômetros foram ligados em tempos diferentes, foi utilizada a função 'Xcorrelação' para alinhar os dados. Na Figura 71, podemos observar a defasagem na aceleração entre os acelerômetro s, enquanto na Figura 72 os dados estão alinhados para análise posterior no domínio da frequência.

Figura 71 - Aceleração no Domínio do Tempo – Antes do Ajuste



Fonte: Autores (2024).

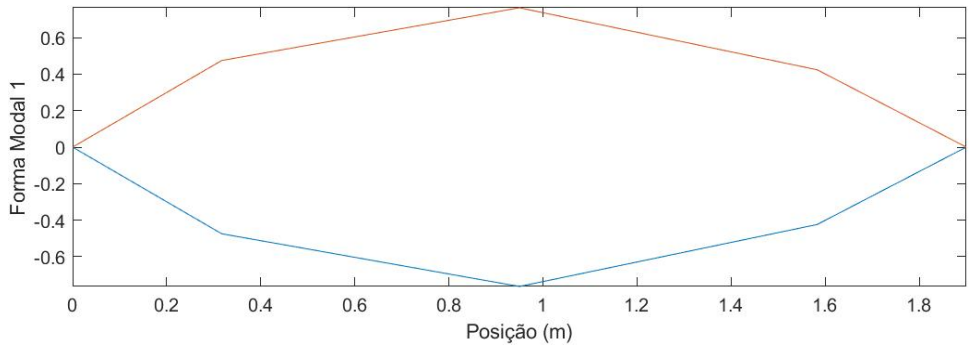
Figura 72 - Aceleração no Domínio do Tempo – Após do Ajuste



Fonte: Autores (2024).

A forma modal obtida experimentalmente está em concordância com a forma modal obtida pelo SAP2000.

Figura 73 - Forma modal obtida pelo FDD

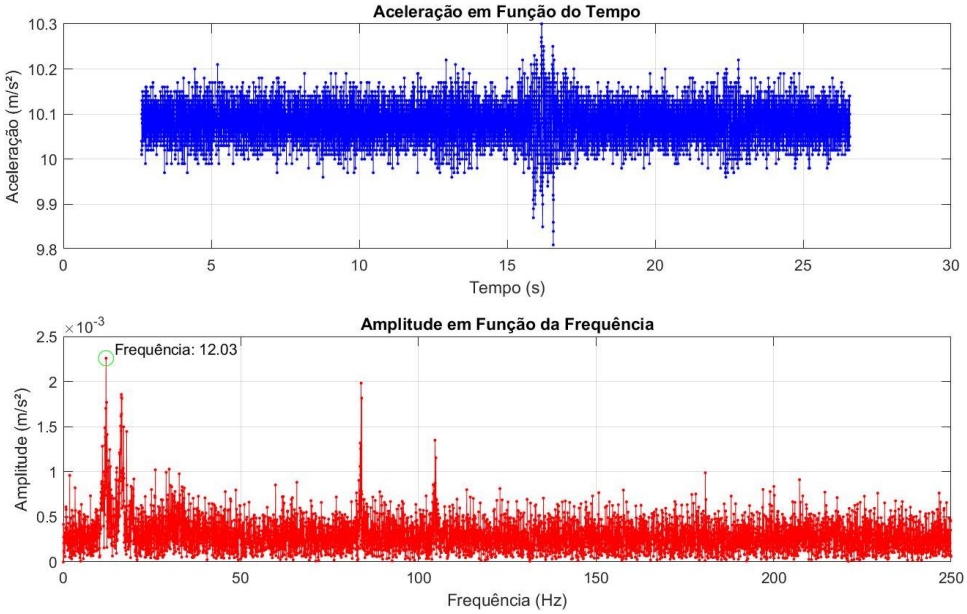


Fonte: Autores (2024).

5.2. PROVA DE CONCEITO

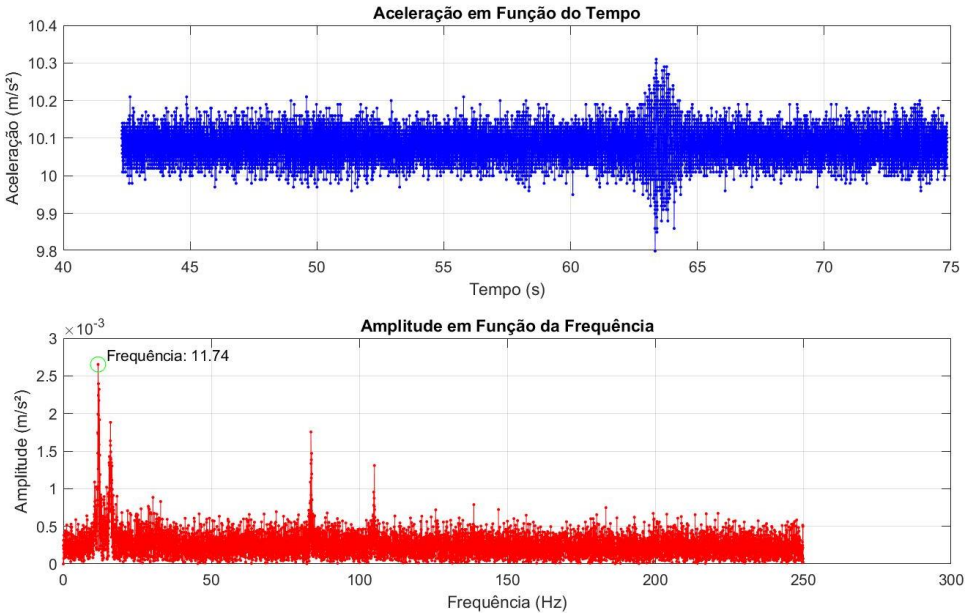
Durante as leituras realizadas com o MPU6050 na ponte, observamos uma distinção entre os valores de ruído e os dados capturados durante a passagem de veículos de grande porte, como caminhões e ônibus. Enquanto o ruído apresentava um nível relativamente alto, os valores de aceleração registrados durante a passagem desses veículos superaram o ruído de fundo. As imagens capturadas durante os ensaios destacam claramente esse fenômeno, revelando picos distintos nos momentos em que os veículos passavam sobre a ponte.

Figura 74 - Aceleração no domínio do tempo e da frequência, leitura 1 (MPU-6050)



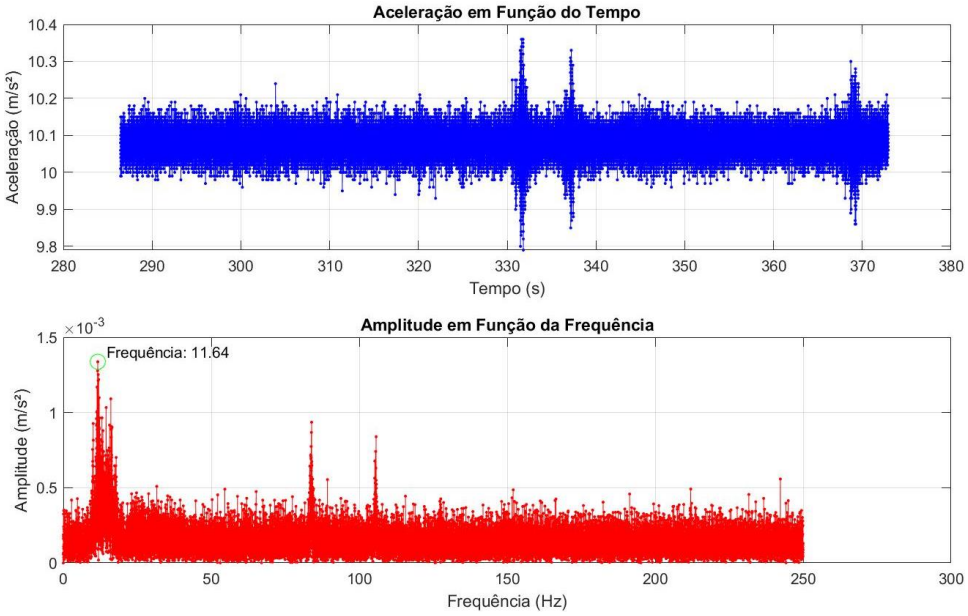
Fonte: Autores (2024).

Figura 75 - Aceleração no domínio do tempo e da frequência, leitura 2 (MPU-6050)



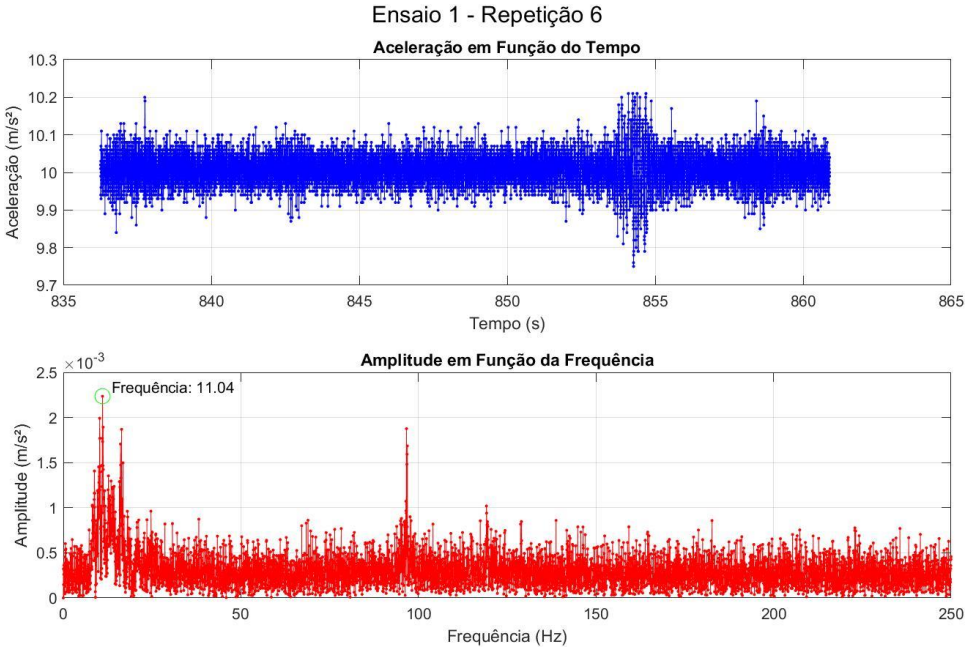
Fonte: Autores (2024).

Figura 76 - Aceleração no domínio do tempo e da frequência, leitura 3 (MPU-6050)



Fonte: Autores (2024).

Figura 77 - Aceleração no domínio do tempo e da frequência, leitura 4 (MPU-6050)



Fonte: Autores (2024).

Ao analisar o domínio da frequência, identificamos três frequências consistentes nos dados coletados, conforme mostrado na tabela a seguir:

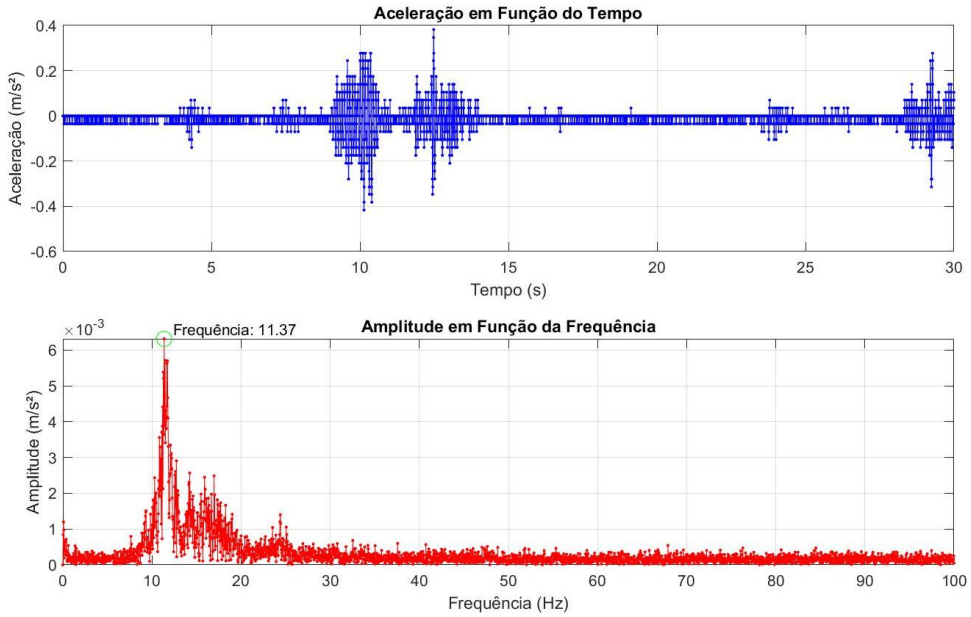
Tabela 27 - Resultados aferidos MPU-6050

Repetição	1ª Frequência (Hz)	2ª Frequência (Hz)	3ª Frequência (Hz)
1	12,03	83,86	104,66
2	11,74	83,67	105,13
3	11,64	83,90	105,70
4	11,04	96,63	119,56

Fonte: Autores (2024).

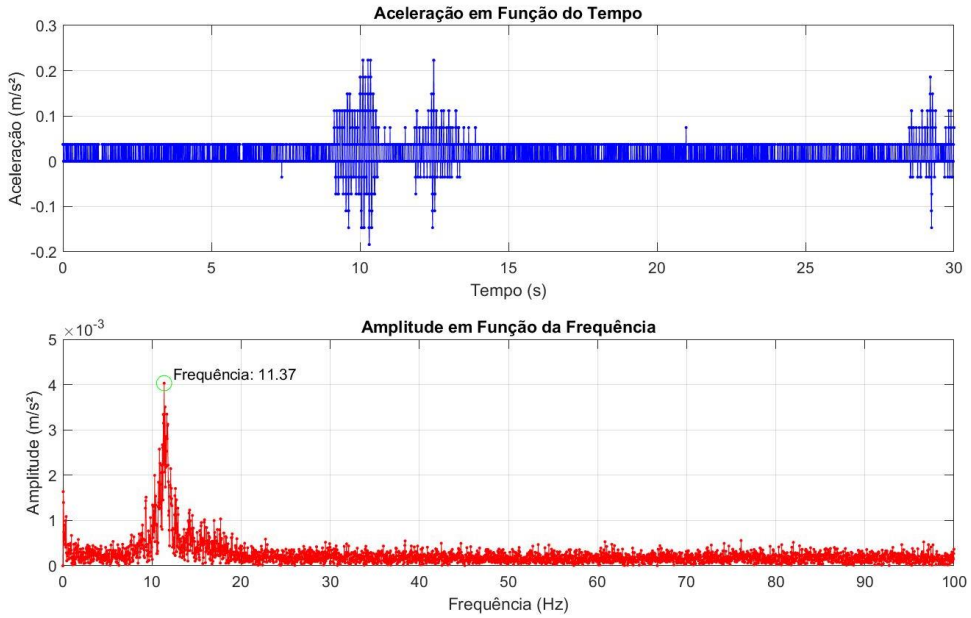
Nas leituras realizadas com o Low-g Accelerometer, observamos uma redução significativa no nível de ruído em comparação com as leituras do MPU6050. Isso é evidenciado nas figuras capturadas durante os ensaios.

Figura 78 - Aceleração no domínio do tempo e da frequência, leitura 1 (Low-g Accelerometer)



Fonte: Autores (2024).

Figura 79 - Aceleração no domínio do tempo e da frequência, leitura 2 (Low-g Accelerometer)



Fonte: Autores (2024).

Ao analisar o domínio da frequência dos dados do Low-g Accelerometer, encontramos uma única frequência comum, com o valor de 11,37 Hz.

Tabela 28 - Resultados aferidos Low-g Accelerometer

Repetição	1ª Frequência (Hz)
1	11,37
2	11,37

Fonte: Autores (2024).

Os sensores MPU6050 e Low-g Accelerometer foram eficazes para registrar as variações de aceleração durante a passagem de veículos pesados sobre a ponte. O MPU6050 detectou essas variações, embora seu sinal apresentasse um ruído considerável, aproximadamente 2/3 do tamanho da leitura durante os eventos de carga. Por outro lado, o Low-g mostrou um nível de ruído significativamente menor, cerca de 1/10 do tamanho da leitura durante os eventos de carga.

Enquanto o MPU6050 identificou múltiplas frequências durante a análise no domínio da frequência, o Vernier detectou apenas uma frequência comum em todas as análises. Isso sugere que a taxa de amostragem mais baixa do Low-g pode ter limitado sua capacidade de capturar outras frequências da ponte.

6. CONCLUSÃO

Este estudo teve por objetivo desenvolver sistema de baixo custo para o monitoramento dinâmico de estruturas, especialmente estruturas sujeitas a carregamentos cíclicos, a fim de que esse sistema possa posteriormente ser aplicado em estrutura real, caracterizando monitoramento de integridade estrutural. O sistema desenvolvido é composto por equipamento de aquisição dos dados, utilizando ESP32 e dois acelerômetros MPU-6050, interface de armazenamento e tratamento dos dados na Raspberry Pi. O trabalho buscou entender as limitações de cada material utilizado, validando seu uso para monitoramento e desenvolvendo protótipo para aplicação em estrutura real.

No processo de construção do sistema de monitoramento, diversas questões surgiram, assim conclui-se:

- O Acelerômetro MPU-6050 em teoria não contempla as necessidades para implantação em monitoramento de integridade estrutural, devido à alta taxa de ruído observada. Entretanto, na prova de conceito realizada numa estrutura real, notou-se que para carregamentos consideráveis é possível identificar o pulso vibracional e buscar valores em frequência da estrutura. Apesar das limitações relacionadas ao nível de ruído, o MPU-6050 obteve resultados excelentes na avaliação de parâmetros dinâmicos em modelos reduzidos, com diferença relativa com resultados teóricos aceitáveis. Conclui-se que dependendo da aplicação, ainda são necessários estudos adicionais, mas existe potencialidade de sua aplicação em estruturas reais.
- Quanto ao sistema de aquisição e tratamento dos dados, o propósito foi desenvolver o mais customizável possível, porém observou-se algumas limitações no processo. Na aquisição dos dados o maior desafio foi criar um equipamento com baixíssimo consumo energético, portanto o desenvolvimento de estrutura de dormência (*deepsleep*) foi primordial. A estrutura de conexão com o servidor ainda precisa ser aprimorada, explorando outras opções de comunicação entre o cliente o servidor. Do lado do servidor, em termos de desenvolvimento em código, se faz necessário criação de *log* das conexões e interações cliente/servidor. Em termos gerais o sistema funcionou de forma aceitável, mas existe oportunidade de melhoria.
- Em relação a estrutura física do equipamento de aquisição dos dados, atendeu parcialmente ao ensaio de validação do sistema, mas é necessário realizar testes em campo para identificar oportunidades de melhoria. Por se tratar de um protótipo, as

conexões entre placas não foram realizadas de forma definitiva e em alguns momentos causou transtornos nas medições. Uma oportunidade de melhoria seria na gestão dos cabos na ligação da ESP32 com os sensores acelerômetros, criando uma estrutura de fios mais robusta e resistente.

- Em termos de custo, o sistema final atende ao objetivo de baixo custo em relação aos sistemas tradicionais e o trabalho com dados em tempo real. O maior investimento do sistema está no servidor local utilizando a Raspberry Pi, porém por ser uma solução muito flexível, em situações que exista acesso a internet, é possível substituir o microcomputador por um servidor de baixo custo em nuvem. Em termos de implantação, o equipamento com dois acelerômetros exige apenas a fixação dos sensores na estrutura, onde seja possível alimentar com energia. A manutenção em geral será apenas na reposição dos equipamentos em casos de danos ou furtos. Na gestão de cabos acaba sendo bem econômico no geral, por possuir menor quantidade de cabos possíveis dentro das oportunidades de implementação do equipamento. Por fim em termos de segurança das medidas aferidas, pelo sistema salvar internamente as medições realizadas e enviar posteriormente os dados, garante que as informações nas duas pontas (cliente e servidor) sejam compatíveis.

Por fim, o sistema desenvolvido atende aos objetivos traçados, inovando na forma e nos equipamentos utilizados no monitoramento dinâmico de estruturas, utilizando equipamentos de baixo custo. Porém esse trabalho abre oportunidades para outros desenvolvimentos pensando em estruturas cada vez mais conectadas e informatizadas.

6.1. TRABALHOS FUTUROS

Para trabalhos futuros pretende-se validar o sistema aplicando numa estrutura real, uma ponte rodoviária, a fim de entender as oportunidades de melhorias e limitações. Planeja-se inserir dados em modelo BIM da estrutura avaliada, confeccionando um ativo digital. Por fim, têm-se o desejo de criar framework de análise de estruturas, a partir de vibração, realizando retro análise de modelos reais, implementando modelos computacionais robustos em elementos finitos e até realizando conexões via *API* com software de mercado.

6.2. SUGESTÕES DE TRABALHOS

A partir desse estudo foi possível observar uma gama muito grande de oportunidade de monitoramento de estruturas, assim existem alguns estudos que podem ser realizados, como:

1. Continuação de pesquisas por sensores de baixo custo que possuam menor densidade de ruído, possibilitando maior obtenção de dados.

2. Avaliação da variação de rigidez de peças em concreto armado em função da variação das frequências naturais.

3. Criação de algoritmos para identificação de regiões onde pode estar localizado possíveis danos de acordo com as frequências naturais obtidas.

4. Desenvolvimento de painel de monitoramento para avaliação dinâmica de estruturas.

REFERÊNCIAS

ANDRADE, R. G. M.; TRAUTWEIN, L. M.; BITTENCOURT, T. N. Comparativo e calibração de modelos numéricos a partir de dados de monitoramento de uma ponte rodoviária curva de concreto armado. **Revista Ibracon de Estruturas e Materiais**, pp. 121-138, 2013.

ARDUINO. **What is Arduino.** Disponível em: https://www.arduino.cc/en/Guide/Introduction. Acesso em: 02 abr. 2024

ASSIS, Wayne Santos de. **Sistemas computacionais de apoio à monitoração de estruturas de engenharia civil.** 2007. Tese (Doutorado em Engenharia Civil) - Departamento de Engenharia Civil, Escola Politécnica da Universidade de São Paulo, São Carlos, São Paulo, 2007.

BARROS, L. C. **Aplicação de métodos no domínio da frequência para identificação modal de estruturas**. 2016. Dissertação (Mestrado em Estruturas) – Programa de Pós-Graduação em Engenharia Civil, UFOP, Ouro Preto.

BORGES, J. L. S. **Desenvolvimento de um Dispositivo para Monitoramento Dinâmico de Estruturas**. 2021. Dissertação (Mestrado em Estruturas) – Programa de Pós-Graduação em Engenharia Civil, UFRGS, Porto Alegre.

CAMPILHO, **A. Instrumentação electrónica. Métodos e técnicas de medição**. Faculdade de Engenharia da Universidade do Porto, Porto, 2000.

COMISU, C. C.; TARANU, N.; BOACA, G.; SCUTARU, M. C. Structural health monitoring system of bridges. **X International Conference on Structural Dynamics**, 2017.

DESIGN AND MOTION. **SIMULATION MECHANICAL | NATURAL FREQUENCY MODES AND GRAVITY.** Disponível em: https://designandmotion.net/autodesk/simulation-mechanical-natural-frequency-modes-and-gravity/. Acesso em: 02 abr. 2024

DYNAMIS TECHNE. **Instrumentação com sensores de deslocamento** Disponível em: https://dynamistechne.com/nossos-servicos/instrumentacao-com-sensores-de-deslocamentos/ Acesso em: 02 abr. 2024

DYNAMIS TECHNE. **Ensaio de Extensometria (Instrumentação com extensômetros).** Disponível em: https://dynamistechne.com/nossos-servicos/instrumentacao-e-monitoracao-com-extensometros/. Acesso em: 02 abr. 2024

ESPRESSIF SYSTEMS. ESP32: **Technical Reference Manual, Shanghai.** Disponível em: http://www.hec.usace.army.mil/software/hec-hms/documentation/HEC-HMS_Technical Reference Manual_(CPD-74B).pdf. Acesso em: 02 abr. 2024.

HOLANDA, R. V.; DUARTE, M. A. V.; PENA, J. L. O.; OLIVEIRA, E. L. Um Estudo Sobre a Utilização de Análise Modal Operacional com Excitação Transiente. **VI Congresso Nacional De Engenharia Mecânica**, 2010.

KAMINSKI, J.; RIERA, J. D. Structural damage detection by means vibration tests. **INTERNATIONAL CONFERENCE ON STRUCTURAL MECHANICS IN REACTOR TECHNOLOGY.** 1997.

LINCH, J. P.; LOH, K. A Summary Review of Wireless Sensors and Sensor Networks for Structural Health Monitoring. **The Shock and Vibration Digest**, pp. 91-128, 2014.

LYNX TECNOLOGIA. **Conceitos básicos de aquisição de dados**. Disponível em: https://www.lynxtec.com.br/tutor_aqd1. Acesso em: 02 abr. 2024.

LOPES, Benedito de Arruda Ribeiro. **Sistema de Manutenção Predial para Grandes Estoques de Edifícios: Estudo para inclusão do componente "Estrutura de Concreto".** 1998. Dissertação (Mestrado em Engenharia Civil) - Departamento de Engenharia Civil, Universidade de Brasília, Brasília, Distrito Federal, 1998.

LUZURIAGA, J. E.; PEREZ, M.; BORONAT, P.; CANO, J. C.; CALAFATE, C.; MANZONI, P. A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks. **Consumer Communications and Networking Conference (CCNC)**, 2015.

MATLAB. R2024a. Natick: MathWorks, 1970. Computação numérica de alto desempenho e visualização. Disponível em: https://www.mathworks.com/products/matlab.html. Acesso em: 12 abr. 2024.

OTTO, A. **OoMA Toolbox**. Disponível em: https://www.mathworks.com/matlabcentral/fileexchange/68657-ooma-toolbox. Acesso em 12 abr. 2024.

PINHEIRO, J. M. S. **Prova de Conceito (PoC) no Projeto de Redes de Computadores** Disponível em: https://desmontacia.wordpress.com/2010/12/21/prova-de-conceito-poc-no-projeto-de-redes-de-computadores/
Acesso em: 04 abr. 2024

PASCA, D. P.; ALOISIO, A.; ROSSO, M. M.; SOTIROPOULOS, S. PyOMA and PyOMA_GUI: A Python module and software for Operational Modal Analysis. **Software** X, 2022.

REGAZZI, R. D.; PEREIRA, P. S.; SILVA, M. F. **Soluções práticas de instrumentação e automação – Utilizando a programação gráfica LabVIEW**, Rio de Janeiro, 2005, 456 p.

RODRIGUES, J. **Identificação modal estocástica: métodos de análise e aplicações em estruturas de engenharia civil**. 2004. Tese (Doutorado em Engenharia Civil) - Faculdade de Engenharia da Universidade do Porto - FEUP, 526 p, Porto, 2004.

RYTTER, A. **Vibrational Based Inspection of Civil Engineering Structures**. Aalborg Universitet, Aalborg, 1993
WONG, K. Design of a structural health monitoring system for long-span. **Structure and Infrastructure Engineering**, pp. 169–185, 2007

SAP2000. V14. Walnut Creek: CSI, 1975. Análise estrutural. Disponível em: https://multiplus.com/software/sap2000. Acesso em: 12 abr. 2024.

SILVA, A. C. G.; CAMPOS, G. R.; SANTOS, M. L. F. Análise de manifestações patológicas em obras de arte especiais - Estudo de caso e propostas de recuperação. **2º Simpósio Paranaense de Patologia das Construções (2º SPPC)**, Curitiba, pp. 159–170, 2017.

SQLITE. 3.45.2 . Virginia: SQLite Consortium, 2024. Banco de dados relacional. Disponível em: https://www.sqlite.org/. Acesso em: 12 abr. 2024.

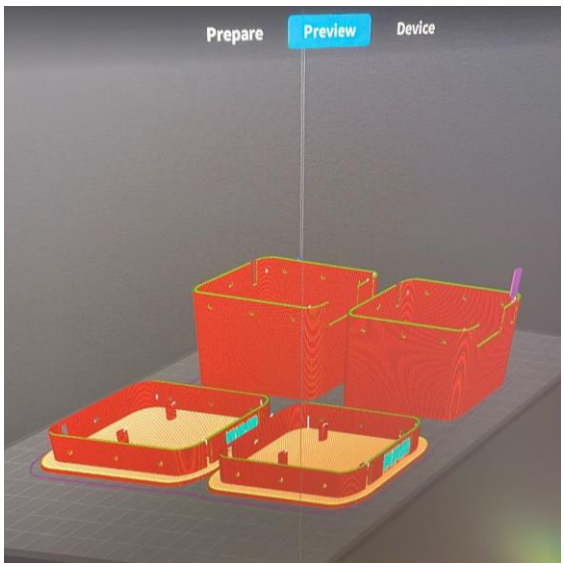
VITÓRIO, J. A. P.; BARROS, R. M. M. C. Análise dos danos estruturais e das condições de estabilidade de 100 pontes rodoviárias no brasil. **Segurança, Conservação e Reabilitação de Pontes**, Porto, p. 63, 2013.

VERNIER SOFTWARE & TECHNOLOGY. **Low-g Accelerometer: Order Code LGA-BTA**. 2024. Disponível em: https://www.vernier.com/manuals/lga-bta/. Acesso em: 30 mar. 2024.

APÊNDICE A – MONTAGEM DO EQUIPAMENTO

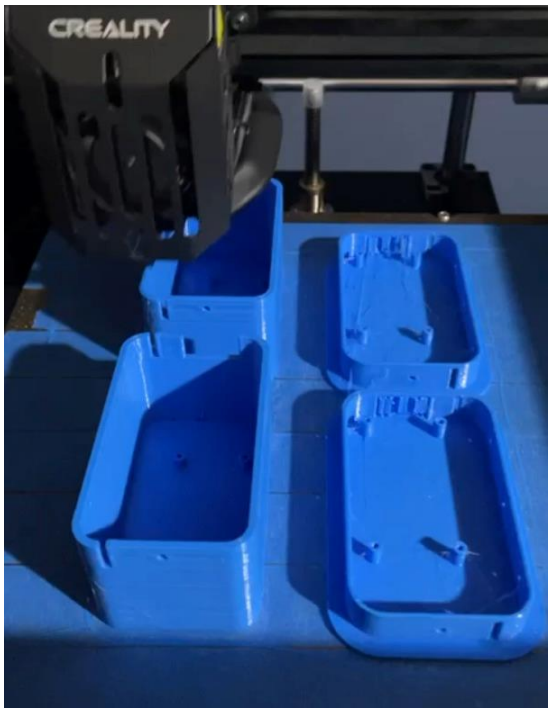
Dentro do fluxo de prototipagem, o desenvolvimento do equipamento de aquisição de dados foi o primeiro dos passos. As proteções foram impressas na Creality Ender - 5 S1 com PLA (Polyactic Acid – Ácido Poliático), material termoplástico comumente utilizado como filamento em impressões 3D.

Figura 80 - Preview do projeto para impressão



Fonte: Autores (2024).

Figura 81 - Impressão das proteções



Fonte: Autores (2024).

Em função do tipo de material utilizado na impressão é importante atentar-se para as retrações causadas pelo resfriamento. Para esse projeto, todos os encaixes foram dimensionados com uma folga de 0,5mm.

Figura 82 - Proteção ESP32 + Adaptador Micro SD

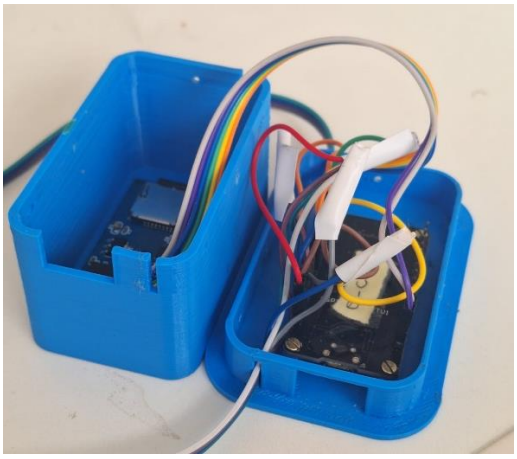
Fonte: Autores (2024).

Figura 83 - Proteção MPU-6050

Fonte: Autores (2024).

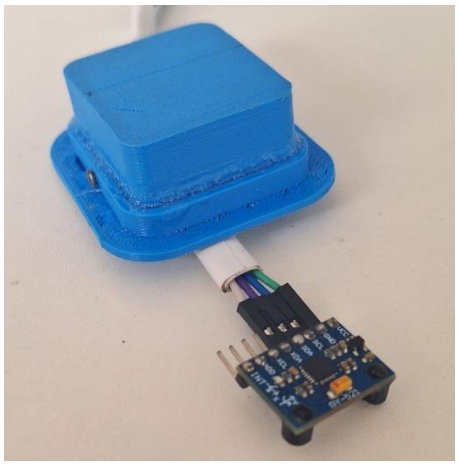
Outro ponto de atenção se dá as referências de projeto utilizadas para dimensionamento da estrutura de proteção, para esse projeto foram utilizados modelos de placas eletrônicas com dimensões compatíveis, mas qualquer erro de escala pode ocasionar em problemas na montagem do equipamento. A ligação entre os componentes foi realizada com *jumpers* do tipo fêmea-fêmea, evitando a utilização de solda, já que se trata de um modelo protótipo. Os pontos de emenda entre os fios foram isolados com tubos termo retráteis.

Figura 84 - Ligação entre os componentes da ESP32



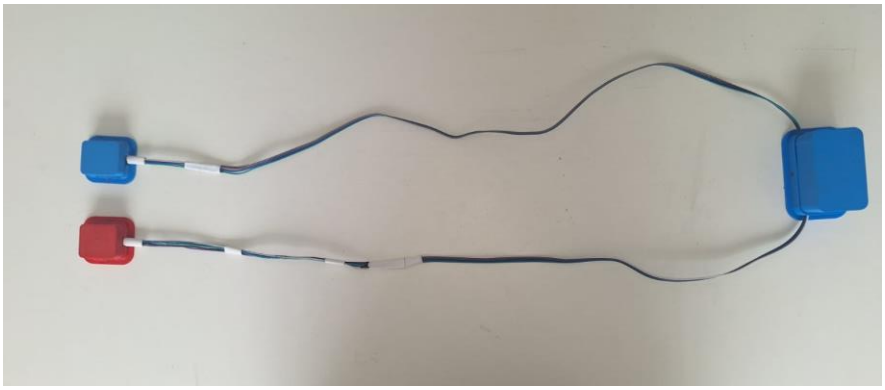
Fonte: Autores (2024).

Figura 85 - Ligação com MPU-6050



Fonte: Autores (2024).

Figura 86 - Estrutura final do equipamento



Fonte: Autores (2024).

A fonte de energia utilizada para operação do sistema deve abarcar o consumo do equipamento em dois estados, funcionamento pleno enviando dados e em *deepsleep* (sono profundo). A ESP32 é considerada um microprocessador de baixo consumo energético, mas em funções de conectividade, seja Wi-Fi ou bluetooth, seu consumo chega é níveis altos. Como o sistema dimensionado trabalha em dois regimes, ligado e desligado, o ideal para dimensionar a fonte é calcular consumo médio por hora, em função dos tempos de operação do sistema. Na Tabela 29 tem-se um exemplo de como dimensionar o consumo energético, mas vale salientar que o equipamento desenvolvido possui alta customização, logo parâmetros como tempo de funcionamento e consumo energético vão variar de acordo com a aplicação desejada.

Tabela 29 - Exemplo de consumo energético

MODO	**CONSUMO (mA)**	**TEMPO (min)**
Pleno Funcionamento	230	20
Deepsleep	8	40
Média de consumo	20,67	60

Fonte: Autores (2024).

APÊNDICE B – CÓDIGO ESP32

Funções para leitura e envio dos dados

```
//Bibliotecas para sistema de armazenamento
#include "FS.h"
#include "SD.h"
#include "SPI.h"
//Conexão com WIFI e comunicação IP/TCP
#include <WiFi.h>
//Biblioteca para consulta de data e hora
#include "time.h"
//Envio dos dados via requisição HTTP para WebSocket Server
#include <WebSocketsClient.h>
//Leitura dos sensores pelo protocolo I2C
#include <Wire.h>
//Leitura de JSON resposta do servidor
#include <ArduinoJson.h>
//Fator de conversão para microsegundos
#define uS_TO_S_FACTOR 1000

//Configuração da medição
float TaxaAmostragem = 200;                  //Taxa de amostragem da medição em Hz.
float TempoAmostra = 1;                      //Tempo da amostra em minutos
String Sensibilidade = "4g";
float InicioMedicao = 10.333;
float IntervaloMedicao = 10;
float FimMedicao = 23.333;
```

```
//Variáveis de apoio para configuração
float APOIODataHora;
float APOIOTaxaAmostragem;
float APOIOTempoAmostra;
float APOIOInicioMedicao;
float APOIOFimMedicao;
float APOIOIntervaloMedicao;
const char* APOIOSensibilidade;

int rep;

//Id Dispositivo
String  ID = "0001";

//Configuração de rede
const char* ssid     = "teste";           //Identificação da rede WIFI
const char* password = "12345678";              //Senha da Rede WIFI
const char *serverIP = "192.168.254.190";          //IP do servidor local
const int serverPort = 8080;                   //Porta do seu servidor
//IPAddress ip(10, 42, 0, 10);                  //IP ID para cada ESP32
//IPAddress gateway(10, 42, 0, 1);              //IP gateway, substituir pela da conexão
//IPAddress subnet(255, 255, 255, 0);             //IP subnet, substituir pela da conexão

//Variaveis para inicialização do algorítmo
float TamanhoAmostra;                       //Tamanho da amostra (tempo min*60s/min*Taxa
de amostragem amostras/s)
float delta;                              //Intervalo de aquisição dos dados em microsegundos
int iniciar = 0;                           //Varíavel que controla inicio da medição
int Conectado = 0;
int LimAcel;                               //Configuração do range acelerometro
float LSB;                                //Fator de sensibilidade para o range
```

```
float g = 9.8;                          //Aceleração da gravidade

const int MPU_addr_1 = 0x68;                //Endereço I2C do primeiro MPU-6050
const int MPU_addr_2 = 0x69;                //Endereço I2C do Segundo MPU-6050

int16_t AcX1;                           //Variáveis de medição do primeiro acelerômetro
int16_t AcX2;                           //Variáveis de medição do segundo acelerômetro

unsigned long StartTime;
unsigned long previousTime = 0;             //Variáveis para registro do tempo
unsigned long Tempo;                        //Variáveis para registro do tempo
float interval;                     //Tempo de standby
const char* DataHora = nullptr;                 //Data e Hora servidor
int i =0;                           //Variável contadora
int j =0;                           //Variável contadora
int l=0;
const char* BackupDados = "/LogMedicao.txt";        //Registro das medições realizadas
const char* RegServer = "/RegServer.txt";           //Registros de datas obtidas do server
const char* Intervalos = "/Intervalos.txt";         //Registros dos intervalos de delay
String LeituraDados;                    //String para gravar leitura no microSD
String linha;                           //String para leitura de linha de arquivo .txt
String header;                          //Cabeçalho
String DadosSensor;

float hora;

//Criação dos objetos
File file;                          //Arquivo para salvar dados

WebSocketsClient webSocket;                 //Objeto para conexão websockets
```

```
void Settings(){


  TaxaAmostragem = APOIOTaxaAmostragem;

  TempoAmostra = APOIOTempoAmostra;

  Sensibilidade = APOIOSensibilidade;

  InicioMedicao = APOIOInicioMedicao;

  FimMedicao = APOIOFimMedicao;

  IntervaloMedicao = APOIOIntervaloMedicao;


  TamanhoAmostra = TempoAmostra*60*TaxaAmostragem;  //Tamanho da amostra (tempo
min*60s/min*Taxa de amostragem amostras/s)

  delta = (1000/TaxaAmostragem)*1000;            //Intervalo de aquisição dos dados em
microsegundo


}
```

```
//Leitura de JSON resposta do servidor
void ReadJSON(fs::FS &fs, const char* message, const char *nomeArquivo){

 const size_t capacity = JSON_OBJECT_SIZE(5) + 150;

 DynamicJsonDocument doc(capacity);

 DeserializationError error = deserializeJson(doc, message);

 if (error) {
  Serial.print(F("Falha ao analisar o JSON: "));
  Serial.println(error.c_str());
  return;
 }

 APOIODataHora = doc["DataHora"];
 APOIOTaxaAmostragem = doc["Frequencia"];
 APOIOTempoAmostra =  doc["Tempo"];
 APOIOSensibilidade = doc["Sensibilidade"];
 APOIOInicioMedicao = doc["InicioMedicao"];
 APOIOFimMedicao = doc["FimMedicao"];
 APOIOIntervaloMedicao = doc["IntervaloMedicao"];

 Settings();

 static char bff[15];
 dtostrf(APOIODataHora, 4, 2, bff);
 const char *messageDataHora = bff;

 CreateFile(fs, nomeArquivo, messageDataHora);
 HourCompare(fs, nomeArquivo, Intervalos);}
```

```
//Seleciona parâmetros de configuração do acelerometro
void MPUSettings(String sensity){

 if(strcmp(sensity.c_str(), "2g") == 0){
  LimAcel = 0x00;
  LSB = 16384;
 } else if(strcmp(sensity.c_str(), "4g") == 0){
  LimAcel = 0x08;
  LSB = 8192;
 } else if(strcmp(sensity.c_str(), "8g") == 0){
  LimAcel = 0x10;
  LSB = 4096;
 } else if(strcmp(sensity.c_str(), "16g") == 0){
  LimAcel = 0x18;
  LSB = 2048;
 } else{
  Serial.println("Sensibilidade não identificada, será considerada 2g");
  LimAcel = 0x00;
  LSB = 16384;

 }
}
```

```
//envia os dados para o server
void sendDataWebSocket(const char *nomeArquivo) {
  uint8_t buffer[1024];
  size_t length;
  bool finalizado = false;

  File arquivo = SD.open(nomeArquivo, FILE_READ);

  if (arquivo){
    while (arquivo.available()){

      length = arquivo.readBytes((char *)buffer, sizeof(buffer));
      webSocket.sendBIN(buffer, length);

      if (length < sizeof(buffer)){
      finalizado = true;
      break;
      }

    }

    arquivo.close();
  } else{
  }
  if (finalizado) {
    uint8_t fim[1] = { 0xFF }; // Define um byte de fim de arquivo (pode ser qualquer valor que
você escolher)
    webSocket.sendBIN(fim, 1); // Envia o byte de fim de arquivo
  }
}
```

```
//Cria um arquivo novo
void CreateFile(fs::FS &fs, const char *nomeArquivo, const char * message){

 //Criando arquivo de backup
 File file = fs.open(nomeArquivo);
 if(!file) {
  String header = String(message) + "\n";
  writeFile(fs, nomeArquivo, header.c_str());
 }
 else {
  deleteFile(fs, nomeArquivo);
  String header = String(message) + "\n";
  writeFile(fs, nomeArquivo, header.c_str());
 }
 file.close();
}
```

```
//Compara o horario de ligar o acelerometro e o horario obtido no servidor, salva um intervalo de dormencia para o proxima leitura
void HourCompare(fs::FS &fs, const char *LeituraArquivo, const char *SaidaArquivo) {

 // Abre o arquivo para leitura
 File file = fs.open(LeituraArquivo);

 // Verifica se o arquivo foi aberto corretamente
 if (file) {
  // Lê o arquivo linha por linha até o final
  String linha = file.readStringUntil('\n'); // Lê uma linha
  sscanf(linha.c_str(), "%f", &hora);
  file.close();
 }

 int n = int((FimMedicao - InicioMedicao)/IntervaloMedicao);
 float d;
 float Ad =1000;

 for(int k=0; k<n; k++){

  d = ((k*IntervaloMedicao)+InicioMedicao)-hora;

  if(d>0){
   if(d<Ad){
    Ad = d;
    rep = k;
   }
  }
 }
```

```
  interval = Ad;

  if(interval==(1000)){
    interval = ((24-FimMedicao)+InicioMedicao);

  } else{
    interval = interval - (TempoAmostra/60);
    if(interval<0){
      interval=0;
    }
  }
  String ApoioInterval = String(interval);

  CreateFile(fs, SaidaArquivo, ApoioInterval.c_str());
}
```

```
//Executa os cases de acordo com status de conexão da rede
void webSocketEvent(WStype_t type, uint8_t * payload, size_t length) {
  switch(type) {
    case WStype_DISCONNECTED:
      Serial.printf("[WSc] Disconnected!\n");
      Conectado = 0;
      break;
    case WStype_CONNECTED:
      Serial.printf("[WSc] Connected to url: %s\n", payload);
      break;
    case WStype_TEXT:{
      Serial.printf("[WSc] get text: %s\n", payload);
      Conectado = 1;
      // Libera a memória se já houver uma mensagem anterior
      if (DataHora != nullptr) {
        delete[] DataHora;
        DataHora = nullptr;
      }
      // Aloca memória para a nova mensagem e copia o payload
      char* buffer = new char[length + 1];
      strncpy(buffer, reinterpret_cast<const char*>(payload), length);
      buffer[length] = '\0'; // Garante que a string seja terminada corretamente

      // Atribui o buffer à variável mensagemRecebida
      DataHora = buffer;
    }
    case WStype_BIN:
      Serial.printf("[WSc] get binary length: %u\n", length);
      break;
    case WStype_ERROR:
    case WStype_FRAGMENT_TEXT_START:
```

```
//Coloca o equipamento em modo standby
void SleepTime(fs::FS &fs, const char *nomeArquivo){

 File deep = fs.open("/deepsleep.txt");

 if (!deep){
  File teste = fs.open(nomeArquivo);
  // Verifica se o arquivo foi aberto corretamente
  if (teste) {
   String linha = teste.readStringUntil('\n'); // Lê uma linha
   float intervalo;
   sscanf(linha.c_str(), "%f", &intervalo);
   teste.close();
   intervalo = intervalo*60*60*1000;
   unsigned long apoiointervalo = intervalo;
   esp_sleep_enable_timer_wakeup(apoiointervalo * uS_TO_S_FACTOR);
   CreateFile(SD, "/deepsleep.txt", "OK");
   esp_deep_sleep_start();
  }
 }
 iniciar = 1;
}
```

```
//Executa os cases de acordo com status de conexão da rede
void webSocketEvent(WStype_t type, uint8_t * payload, size_t length) {
  switch(type) {
    case WStype_DISCONNECTED:
      Serial.printf("[WSc] Disconnected!\n");
      Conectado = 0;
      break;
    case WStype_CONNECTED:
      Serial.printf("[WSc] Connected to url: %s\n", payload);
      break;
    case WStype_TEXT:{
      Serial.printf("[WSc] get text: %s\n", payload);
      Conectado = 1;
      // Libera a memória se já houver uma mensagem anterior
      if (DataHora != nullptr) {
        delete[] DataHora;
        DataHora = nullptr;
      }
      // Aloca memória para a nova mensagem e copia o payload
      char* buffer = new char[length + 1];
      strncpy(buffer, reinterpret_cast<const char*>(payload), length);
      buffer[length] = '\0'; // Garante que a string seja terminada corretamente

      // Atribui o buffer à variável mensagemRecebida
      DataHora = buffer;
    }
    case WStype_BIN:
      Serial.printf("[WSc] get binary length: %u\n", length);
      break;
    case WStype_ERROR:
    case WStype_FRAGMENT_TEXT_START:
```

```
      case WStype_FRAGMENT_BIN_START:
      case WStype_FRAGMENT:
      case WStype_FRAGMENT_FIN:
        break;
   }
}


// Inicializa a conexão com rede WIFI
void initWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  //WiFi.config(ip, gateway, subnet);
  Serial.println(WiFi.localIP());
}
```

```
// Inicializa Micro SD Card
void initSDCard(){
  if (!SD.begin()) {
   Serial.println("Card Mount Failed");
   return;
  }
  uint8_t cardType = SD.cardType();

  if(cardType == CARD_NONE){
   Serial.println("No SD card attached");
   return;
  }
  Serial.print("SD Card Type: ");
  if(cardType == CARD_MMC){
   Serial.println("MMC");
  } else if(cardType == CARD_SD){
   Serial.println("SDSC");
  } else if(cardType == CARD_SDHC){
   Serial.println("SDHC");
  } else {
   Serial.println("UNKNOWN");
  }
  uint64_t cardSize = SD.cardSize() / (1024 * 1024);
  Serial.printf("SD Card Size: %lluMB\n", cardSize);
}
```

```
//Escreve no SD Card
void writeFile(fs::FS &fs, const char * path, const char * message) {
 Serial.printf("Writing file: %s\n", path);

 File file = fs.open(path, FILE_WRITE);
 if(!file) {
  Serial.println("Failed to open file for writing");
  return;
 }
 if(file.print(message)) {
  Serial.println("File written");
 } else {
  Serial.println("Write failed");
 }
 file.close();
}

//Deleta arquivo no SD Card
void deleteFile(fs::FS &fs, const char * path){
  if(fs.remove(path)){
    Serial.println("File deleted");
  } else {
    Serial.println("Delete failed");
  }
}
```

```
//Configura os acelerômetros na comunicação I2C
void initMPU(int addr, int rangeSetting){

  // Configuração do modo de energia
  Wire.beginTransmission(addr);
  Wire.write(0x6B);  // PWR_MGMT_1 Registro
  Wire.write(0);     // Retornando 0 para inicializar MPU
  Wire.endTransmission(true);

  // Configuração do range do acelerômetro para ±4g
  Wire.beginTransmission(addr);
  Wire.write(0x1C);  // ACCEL_CONFIG Registro
  Wire.write(rangeSetting);  // Configurando para ±4g
  Wire.endTransmission(true);
}
```

Inicialização do sensor

```
void setup() {


  Serial.begin(115200);
  initSDCard();
  SleepTime(SD, Intervalos);
  MPUSettings(Sensibilidade);
  initWiFi();
  Wire.begin();
  Serial.print(LimAcel)
  initMPU(MPU_addr_1, LimAcel);
  initMPU(MPU_addr_2, LimAcel);



  webSocket.begin(serverIP, serverPort, "/");
  webSocket.onEvent(webSocketEvent);
  webSocket.setReconnectInterval(5000);


}
```

Estrutura de repetição

```
void loop(){

 webSocket.loop();

 if(iniciar!=0 && Conectado!=0){

  if(i<1){

    ReadJSON(SD, DataHora, RegServer);

    StartTime = (hora*60*60*1000);

    String Apoio = String(StartTime) + " 0.00 0.00";

    CreateFile(SD, BackupDados, Apoio.c_str());

    file = SD.open(BackupDados, FILE_APPEND);
    //webSocket.sendTXT("Iniciando!!");
    DadosSensor = String(ID) + "_" + String(rep);
    webSocket.sendTXT(DadosSensor);
    i = i+1;

  }else{
    unsigned long currentTime = micros();
    if(i<TamanhoAmostra){
      if (currentTime - previousTime >= delta){
        previousTime = currentTime;
```

```
      Wire.beginTransmission(MPU_addr_1);
      Wire.write(0x3B);                  //Iniciando a leitura pelo registro de aceleração em X
      Wire.endTransmission(false);
      Wire.requestFrom(MPU_addr_1, 2, true);  //Requerimento dos primeiros 2 registros
      AcX1 = Wire.read() << 8 | Wire.read();  //Leitura dos dois primeiros bytes para
aceleração em X


      Wire.beginTransmission(MPU_addr_2);
      Wire.write(0x3B);                  //Iniciando a leitura pelo registro de aceleração em X
      Wire.endTransmission(false);
      Wire.requestFrom(MPU_addr_2, 2, true);  //Requerimento dos primeiros 2 registros
      AcX2 = Wire.read() << 8 | Wire.read();  //Leitura dos dois primeiros bytes para
aceleração em X
      //Conversão da leitura para m/s^2
      float acceleration_mps2_1X = (AcX1 / LSB) * g;
      float acceleration_mps2_2X = (AcX2 / LSB) * g;
      //Tempo em milissegundos
      Tempo = currentTime/1000;
      //Criação de string com leitura dos dados
      LeituraDados = String(Tempo) + " " + String(acceleration_mps2_1X) + " " +
String(acceleration_mps2_2X)+ "\r\n";
      //Esrevendo dados no arquivo locado no SD
      file.print(LeituraDados.c_str());


      i=i+1;
     }


    }else{


     //Fechando arquivo de leitura
     file.close();
```

```
      sendDataWebSocket(BackupDados);
      deleteFile(SD, "/deepsleep.txt");
      ESP.restart();

    }

  }

 }

}
```

APÊNDICE C – CÓDIGO RASPBERRY

Funções para WebSocket Server

```
import datetime
import websockets
import asyncio
import json
import os
from dotenv import load_dotenv

load_dotenv()

clients = []
dados = []

#Configuracao da medicao
FREQUENCIA_AMOSTRAGEM = 500
TEMPO_AMOSTRAGEM = 1
INTERVALO_MEDICAO = 10
INICIO_MEDICAO = "10:30"
FIM_MEDICAO = "23:30"
SENSIBILIDADE_MEDICAO = "2g"

#Configuracao server
IP_SERVER = os.environ.get("IP_SERVER")
PORT_SERVER = os.environ.get("PORT_SERVER")

#Pasta Destino
DIRECTORY = 'BancoDados'

TAM_AMOSTRA = FREQUENCIA_AMOSTRAGEM*TEMPO_AMOSTRAGEM*60
```

```
async def send_message(filename):
   async with websockets.connect(f"ws://{IP_SERVER}:{PORT_SERVER}") as websocket:
      message = f"{filename}"
      await websocket.send(message)
      print(f"Mensagem enviada: {message}")



async def hour_calculate(hora_string):
      # Dividir a string da hora nos dois pontos
      partes = hora_string.split(":")

      # Converter as partes para inteiros
      horas = int(partes[0])
      minutos = int(partes[1])

      # Calcular o total de horas
      total_horas = horas + minutos / 60

      return total_horas
```

```
async def ConnectionHandler(websocket, path):
    # Adiciona o cliente à lista quando ele se conecta
    clients.append(websocket)
    print(f"Novo dispositivo conectado: {websocket.remote_address}. Total de clientes: {len(clients)}")

    TInicioMedicao = await hour_calculate(INICIO_MEDICAO)
    TFimMedicao = await hour_calculate(FIM_MEDICAO)
    TIntervaloMedicao = INTERVALO_MEDICAO/60

    # Envia uma mensagem de boas-vindas ao cliente
    HoraAtual = datetime.datetime.now().strftime("%H:%M")
    THoraAtual = await hour_calculate(HoraAtual)

    MessageJson = {
      "DataHora": THoraAtual,
      "Frequencia": FREQUENCIA_AMOSTRAGEM,
      "Tempo": TEMPO_AMOSTRAGEM,
      "Sensibilidade": SENSIBILIDADE_MEDICAO,
      "InicioMedicao": TInicioMedicao,
      "FimMedicao": TFimMedicao,
      "IntervaloMedicao": TIntervaloMedicao
    }
    ClientMessage = json.dumps(MessageJson)
    await websocket.send(ClientMessage)

    if not os.path.exists(DIRECTORY):
                os.makedirs(DIRECTORY)

    device_name = await id_detect(websocket)
    currentTime = datetime.datetime.now().strftime("%Y-%m-%d-%H-%M-%S")
```

```
    filename = f"{device_name}_{currentTime}.txt"
    file_path = os.path.join(DIRECTORY, filename)

    confirmation_message = f"Nome do dispositivo recebido: {device_name}. Arquivo será salvo como {filename}"
    print(confirmation_message)

    try:
      with open(file_path, 'wb') as file:
        while True:
          data = await websocket.recv()
          if data == b'FIM':  # Verifica se recebeu o sinalizador de fim do arquivo
            print(f"Arquivo {filename} recebido e salvo com sucesso")
            await send_message(filename)
            await websocket.close()
            break
          if data:
            file.write(data)

    except websockets.exceptions.WebSocketException as e:
      # Remove o cliente da lista quando ele se desconecta
      clients.remove(websocket)
      print(f"Dispositivo {websocket.remote_address[0]} desconectou: {e}")

    except Exception as e:
      clients.remove(websocket)
      print(f"Erro inesperado: {e}")
```

```
async def id_detect(websocket):

    try:
        data = await websocket.recv()

        if isinstance(data, str):
            device_name = data.strip()  # Remove qualquer espaço extra

            return device_name
        else:
            print("Não é uma string")

            return None

    except websockets.exceptions.WebSocketException as e:
        # Remove o cliente da lista quando ele se desconecta
        clients.remove(websocket)
        print(f"Dispositivo {websocket.remote_address[0]} desconectou: {e}")

        return None
```

```
def server():

    loop = asyncio.new_event_loop()
    asyncio.set_event_loop(loop)

    start_server_coroutine = websockets.serve(ConnectionHandler, IP_SERVER,
PORT_SERVER, ping_timeout=None, close_timeout=None)

    print(f"Servidor ativo no IP {IP_SERVER} porta {PORT_SERVER}")

    loop.run_until_complete(start_server_coroutine)
    loop.run_forever()
```

Funções para análise dos dados

```
def last_archive(diretorio):
    # Inicializa uma variável para armazenar o nome e a data do arquivo mais recente
    arq_more_recent = None
    date_more_recent = None

    # Itera sobre todos os arquivos no diretório
    for arquivo in os.listdir(diretorio):
        if arquivo.startswith("0001_") and arquivo.endswith(".txt"):
            path_way = os.path.join(diretorio, arquivo)
            # Obtém a data de modificação do arquivo
            date_mod = os.path.getmtime(path_way)
            # Verifica se é o arquivo mais recente até agora
            if date_more_recent is None or date_mod > date_more_recent:
                date_more_recent = date_mod
                arq_more_recent = path_way

    return arq_more_recent
```

```
def verify_archive(caminho_arquivo):
    # Obtém o tamanho do arquivo em bytes
    tamanho = os.path.getsize(caminho_arquivo)

    # Verifica se o tamanho do arquivo é maior que 0
    if tamanho > 0:
        print("arquivo completo")
        return True
    else:
        print("arquivo incompleto")
        return False


def search_arq_relevant(DataBase, Archive, EN):
    list = []
    i = 1
    for i in range(EN+1):
        for arq in os.listdir(DataBase):
            name = os.path.basename(Archive)
            standard = r"_\d+_\d{4}-\d{2}-\d{2}"
            dateap = re.search(standard, name)
            date = dateap.group()
            prefix = f"000{i}"+date
            if arq.startswith(prefix):
                path_way = os.path.join(DataBase, arq)
                list.append(path_way)

    return list
```

```
def Xcorrelacao(caminho_completo, Nace,Nesp):
    # função xCorr para ajeitar a defagem o tempo e deixar o vetor de mesmo tamanho
    tempo = [None] * Nesp
    aceleracao = [None] * Nesp * Nace
    tamanho = [0] * Nesp
    tempoGlobal = [None] * Nesp
    # 2.1.3 Função para ler arquivo e obter tempo e aceleração
    for j1 in range(len(caminho_completo)):
        auxtempo, auxaceleracao = ArquivoLeitura(caminho_completo[j1])
        #olhar direito o tempo
        tempoGlobal[j1] = auxtempo[0] * 1000
        tempo[j1] = auxtempo[1:]
        aceleracao[j1*2] = auxaceleracao[0,1:]
        aceleracao[2*j1+1] = auxaceleracao[1,1:]

    print(len(aceleracao))

    # 2.1.4 Função xcorr
    lag = np.zeros((Nesp * Nace, Nesp * Nace))
    for j1 in range(Nesp * Nace):
        for j2 in range(Nesp * Nace):
            if j1 != j2:
                # Calcula a correlação cruzada
                print(aceleracao[j1].shape)
                print(aceleracao[j2].shape)
                xCorr = signal.correlate(aceleracao[j1], aceleracao[j2], mode='full')
                # Calcula a matriz de índices de deslocamento/lag
                lags = signal.correlation_lags(len(aceleracao[j1]), len(aceleracao[j2]), mode='full')
                IxCorr = np.argmax(xCorr)
                lag[j1, j2] = lags[IxCorr]
    lag = lag.astype(int)
```

```
# Ajustar os dados com base nas defasagens calculadas
linhas_negativas = np.where(np.all(lag <= 0, axis=1))[0]
I = linhas_negativas[0]
for j1 in range(Nesp * Nace):
   if j1 != I:
      if lag[j1][I] != 0:
         aceleracao[j1] = aceleracao[j1][abs(lag[j1][I]):]



# Ajustar para o menor comprimento
tamanho = [aceleracao[j1].shape[0] for j1 in range(Nesp * Nace)]
Menor = min(tamanho)
for j1 in range(Nesp * Nace):
   aceleracao[j1] = aceleracao[j1][:Menor]

# Ajustar o tempo para o menor comprimento
#tem a opção de somar aqui com o tempo da ESP de menor vetor
tempoGlobal = np.array(tempoGlobal)
I2=I/Nace
I2=I2.astype(int)
tempo = tempo[I2][:Menor]
tempo  = np.add(tempo, tempoGlobal[I])

# Converter listas para matriz
aceleracao = np.array(aceleracao)

return tempo, aceleracao
```

```
def Tfft(tempo, aceleracao):
    Fv_list = []
    ft_dd_list = []
    Fvm_list = []
    ft_ddm_list = []

    L = len(tempo)  # Signal Length (samples)
    Ts = np.mean(np.diff(tempo))  # Sampling Interval (second)
    Fs = 1 / Ts  # Sampling Frequency (Hz)

    for i in range(aceleracao.shape[0]):
        # Transformada
        dm = aceleracao[i] - np.mean(aceleracao[i])
        Fn = Fs / 2  # Nyquist Frequency (Hz)
        ft_d = np.fft.fft(dm) / L  # Normalized Fourier Transform
        Fv = np.linspace(0, 1, int(np.fix(L/2)) + 1) * Fn  # Frequency Vector
        Iv = np.arange(1, len(Fv)+1)  # Index Vector

        # Ponto mais alto
        ft_dd = np.abs(ft_d[Iv-1])
        ft_ddm = np.max(ft_dd)
        Fvm = Fv[np.argmax(ft_dd)]

        Fv_list.append(Fv)
        ft_dd_list.append(ft_dd)
        Fvm_list.append(Fvm)
        ft_ddm_list.append(ft_ddm)

    return np.array(Fv_list), np.array(ft_dd_list), np.array(Fvm_list),np.array(ft_ddm_list)
```

```
def Graf(tempo, aceleracao, fv, ft_dd, fvm, ft_ddm, j2, j1, caminho_destino):
    # Localização e tamanho
    for i in range(aceleracao.shape[0]):
        # Gráfico 1 - Aceleração em Função do Tempo
        plt.figure(figsize=(20, 10))
        plt.subplot(2, 1, 1)
        plt.plot(tempo, aceleracao[i, :], 'b.-',linewidth=0.5, markersize=2)
        plt.title('Aceleração em Função do Tempo')
        plt.xlabel('Tempo (s)')
        plt.ylabel('Aceleração (m/s²)')
        plt.grid(True)

        # Gráfico 2 - Amplitude em Função da Frequência
        plt.subplot(2, 1, 2)
        plt.plot(fv[i,:], ft_dd[i,:], 'r.-',linewidth=0.5, markersize=2)
        plt.plot(fvm[i], ft_ddm[i], 'go', markersize=10)
        plt.text(fvm[i], ft_ddm[i], f'  Frequência: {fvm[i]:.2f}', verticalalignment='bottom')
        plt.title('Amplitude em Função da Frequência')
        plt.xlabel('Frequência (Hz)')
        plt.ylabel('Amplitude (m/s²)')
        plt.grid(True)

        # Adiciona o título geral à figura
        titulo_geral = f'Ensaio {j2} - Repetição {j1}'
        plt.suptitle(titulo_geral)

        # Salva a figura
        plt.savefig(f'{caminho_destino}_AceFFt_{i}.jpg')
        # Fecha a figura
        plt.close()
```

```
# ---------------------------------------------- PyOMA -------------------------------------
def FDDsvp(data, fs, df=0.05, pov=0.5, window='hann'):
    '''
    This function perform the Frequency Domain Decomposition algorithm.

    The function return the plot of the singular values of the power spectral
    density. The cross power spectral density is estimated using
    scipy.signal.csd() function, which in turn is based on Welch's method.
    Furthermore it returns a dictionary that contains the results needed
    by the function FDDmodEX().
    ----------
    Parameters
    ----------
    data : array
        The time history records (Ndata x Nchannels).
    fs : float
        The sampling frequency.
    df : float
        Desired frequency resolution. Default to 0.01 (Hz).
    pov : float
        Percentage of overlap between segments. Default to 50%.
    window : str or tuple or array_like
        Desired window to use. Window is passed to scipy.signal's get_window
        function (see SciPy.org for more info). Default to "hann" which stands
        for a “Hanning” window.

    -------
    Returns
    -------
    fig1 : matplotlib figure
        Plot of the singular values of the power spectral matrix.
```

```
    Results : dictionary
        Dictionary of results to be passed to FDDmodEX()
    '''

    # ndat=data.shape[0] # Number of data points
    nch=data.shape[1] # Number of channels
    freq_max = fs/2 # Nyquist frequency
    nxseg = fs/df # number of point per segments
#    nseg = ndat // nxseg # number of segments
    noverlap = (nxseg//(1/pov)) # Number of overlapping points
    # Initialization
    PSD_matr = np.zeros((nch, nch, int((nxseg)/2+1)), dtype=complex)
    S_val = np.zeros((nch, nch, int((nxseg)/2+1)))
    S_vec = np.zeros((nch, nch, int((nxseg)/2+1)), dtype=complex)

    # Calculating Auto e Cross-Spectral Density
    for _i in range(0, nch):
        for _j in range(0, nch):
            _f, _Pxy = signal.csd(data[:, _i],data[:, _j], fs=fs, nperseg=nxseg, noverlap=noverlap,
window=window)

            PSD_matr[_i, _j, :] = _Pxy

    # Singular value decomposition
    for _i in range(np.shape(PSD_matr)[2]):
        U1, S1, _V1_t = np.linalg.svd(PSD_matr[:,:,_i])
        U1_1=np.transpose(U1)
        S1 = np.diag(S1)
        S_val[:,:,_i] = S1
        S_vec[:,:,_i] = U1_1
```

```
# Plot of the singular values in log scale
fig, ax = plt.subplots()
for _i in range(nch):
#    ax.semilogy(_f, S_val[_i, _i]) # scala log
   ax.plot(_f[:], 10*np.log10(S_val[_i, _i])) # decibel
ax.grid()
ax.set_xlim(left=0, right=freq_max)
ax.xaxis.set_major_locator(MultipleLocator(freq_max/10))
ax.xaxis.set_major_formatter(FormatStrFormatter('%g'))
ax.xaxis.set_minor_locator(MultipleLocator(freq_max/100))
ax.set_title("Singular values plot - (Freq. res. ={0})".format(df))
ax.set_xlabel('Frequency [Hz]')
ax.set_ylabel(r'dB $[g^2/Hz]$')
# ax.set_ylabel(r'dB $\left[\frac{\left(\frac{m}{s^2}\right)^2}{Hz}\right]$')
mplcursors.cursor()
plt.show()

Results={}
Results['Data'] = {'Data': data}
Results['Data']['Samp. Freq.'] = fs
Results['Data']['Freq. Resol.'] = df
Results['Singular Values'] = S_val
Results['Singular Vectors'] = S_vec
Results['PSD Matrix'] = PSD_matr

return fig, Results
```

```
def FDDmodEX(FreQ, Results, ndf=2):
    '''
    This function returns the modal parameters estimated according to the
    Frequency Domain Decomposition method.

    ----------
    Parameters
    ----------
    FreQ : array (or list)
        Array containing the frequencies, identified from the singular values
        plot, which we want to extract.
    Results : dictionary
        Dictionary of results obtained from FDDsvp().
    ndf : float
        Number of spectral lines in the proximity of FreQ[i] where the peak
        is searched.

    -------
    Returns
    -------
    fig1 : matplotlib figure
        Stabilisation diagram ...
    Results : dictionary
        Dictionary of results ...
    '''

#   data = Results['Data']['Data']
    fs = Results['Data']['Samp. Freq.']
    df = Results['Data']['Freq. Resol.']
    S_val = Results['Singular Values']
    S_vec = Results['Singular Vectors']
```

```
    deltaf=ndf*df
#   ndat=data.shape[0] #
#   nch=data.shape[1] #
    freq_max = fs/2 # Nyquist
#   nxseg = fs/df #

    f = np.linspace(0, int(freq_max), int(freq_max*(1/df)+1)) # spectral lines

    Freq = []
    index = []
    Fi = []

    for _x in FreQ:
#       idx = np.argmin(abs(f-_x))
        lim = (_x - deltaf, _x + deltaf) # frequency bandwidth where the peak is searched
        idxlim = (np.argmin(abs(f-lim[0])), np.argmin(abs(f-lim[1])))
        # ratios between the first and second singular value
        diffS1S2 = S_val[0,0,idxlim[0]:idxlim[1]]/S_val[1,1,idxlim[0]:idxlim[1]]
        maxDiffS1S2 = np.max(diffS1S2) # looking for the maximum difference
        idx1 = np.argmin(abs(diffS1S2 - maxDiffS1S2))
        idxfin = idxlim[0] + idx1
#
=============================================================================
=====
        # Modal properties
        fr_FDD = f[idxfin] # Frequency
        fi_FDD = S_vec[0,:,idxfin] # Mode shape
        idx3 = np.argmax(abs(fi_FDD))
        fi_FDDn = fi_FDD/fi_FDD[idx3] # normalised (unity displacement)
        fiFDDn = np.array(fi_FDDn)

        Freq.append(fr_FDD)
```

```
    Fi.append(fiFDDn)
    index.append(idxfin)

Freq = np.array(Freq)
Fi = np.array(Fi)
index = np.array(index)

Results={}
Results['Frequencies'] = Freq
Results['Mode Shapes'] = Fi.T
Results['Freq. index'] = index

return Results
```

```
def MaC(Fi1,Fi2):
    '''
    This function returns the Modal Assurance Criterion (MAC) for two mode
    shape vectors.

    If the input arrays are in the form (n,) (1D arrays) the output is a
    scalar, if the input are in the form (n,m) the output is a (m,m) matrix
    (MAC matrix).

    ----------
    Parameters
    ----------
    Fi1 : array (1D or 2D)
        First mode shape vector (or matrix).
    Fi2 : array (1D or 2D)
        Second mode shape vector (or matrix).

    -------
    Returns
    -------
    MAC : float or (2D array)
        Modal Assurance Criterion.
    '''

    MAC = np.abs(Fi1.conj().T @ Fi2)**2 / \
        ((Fi1.conj().T @ Fi1)*(Fi2.conj().T @ Fi2))

    return MAC
```

APÊNDICE D – CÓDIGOS MATLAB

Análises desenvolvidas com as rotinas do MATLAB

```
%% 1 Dados Iniciais
% Número de acelerômetros
Nace = 3;
% Quantos modos pretende pegar
Nmodos = 3;
% Quantas repetições do ensaio
Nrepet = 1;
% Número de ensaios
Nensaio = 1;
% Matriz com todos os possíveis dados
Dados = zeros(Nrepet, Nmodos, Nensaio);

%% 2 Loop para Plotar gráficos e pegar as Frequências com os modos
for j2 = 1:Nensaio
   for j1=1:Nrepet
      %% 2.1 Ler os Arquivos
      % 2.1.1 Criar nome do arquivo
      % Nome dos arquivos em geral
      nome_arquivo1 = sprintf('ensaio1_%d_15_01_2024', j1);
      nome_arquivo2 = sprintf('ensaio1_%d_P8_15_01_2024', j1);
      nome_arquivo3 = sprintf('ensaio1_%d_P10_15_01_2024', j1);
      nome_arquivo4 = sprintf('ensaio1_%d_P12_15_01_2024', j1);
      nome_arquivo5 = sprintf('ensaio%d_15_01_2024', j2);
      nome_arquivo = {nome_arquivo1; nome_arquivo2; nome_arquivo3;
nome_arquivo4};
      % Caminho
      caminho = 'C:\Cássio\UFPE\10 Periodo\TCC 2\Ensaios\15-01-2024 - MODELO
3\';
      % Nome da pasta do caminho
      nome_pasta = sprintf('ENSAIO %d', j2);
      % Função para os nomes
      [caminho_completo, caminho_destino, caminho_destino2] =
NomesArquivos2(nome_arquivo, caminho, nome_pasta);

      % Função xCorr para alinhar as acelerações
      [tempo, aceleracao] = Xcorrelacao(caminho_completo, Nace);
      % 3 Função para FFT
      [Fv, ft_dd, Fvm, ft_ddm] = Tfft(tempo, aceleracao);
      % 4 Função para Gerar Gráfico FFT e Aceleração vs Tempo
      Graf(tempo, aceleracao, Fv, ft_dd, Fvm, ft_ddm, j2, j1, caminho_destino);

      %% 5 OMA toolbox FFD
      % Frequência das amostras
      fs = (tempo(1001) - tempo(1000))^-1;
```

```
        % Função da decomposição de frequência (FDD) do OMA toolbox
        [SV, F, Phi, I, SV_num] = fdd(aceleracao, [], fs, []);

        %% 6 Plotar Frequência (Hz) vs Magnitude (dB)
        GrafFDD(j2, j1, F, SV, I)

        %% 7 Salvar as Frequências - em cada linha tem um modo achado
        Dados(j1, 1:size(I, 2), j2) = F(I)';

        if size(I, 2) < Nmodos
           Dados(j1, :) = ModosZero(Dados(j1, :), Nmodos - size(I, 2));
        end
        % 7.1 Cálculo das estatísticas
        Estat(:, :, j2) = [mean(Dados(:, :, j2));
                           std(Dados(:, :, j2));
                           std(Dados(:, :, j2))./mean(Dados(:, :, j2))*100];

        %% 8 Salvar as Figuras
        % 8.1 Salvar como .fig
        saveas(gcf, caminho_destino);
        % 8.2 Salvar como .jpeg
        exportgraphics(gcf, caminho_destino2);
        % Fechar a figura
        close(gcf);
    end
end

function
[caminho_completo,caminho_destino,caminho_destino2]=NomesArquivos(nome_arqui
vo,caminho,nome_pasta)
%%ENTRADA
    %nome_arquivo1 - O nome do arquivo que vai variando
ex:sprintf('ensaio%d_%d_13_12_2023', j2,j1)
    %caminho - Até a penultima pasta
    %nome_pasta - que vai ficar o arquivo
%%Saida
    %caminho_completo - Todas as pastas mais o arquivo texto que quer abrir
    %caminho_destino - Todas as pastas mais o arquivo que quer salvar sem a extensão
    %caminho_destino2 - Todas as pastas mais o arquivo que quer salvar com a extensão
jpg
% Repetição do ensaio
for i=2:size(nome_arquivo,1)
    auxnome_arquivo{i-1,1}=[nome_arquivo{i},'.txt'];
    % 2.1.2 Local da pasta
    % Caminho completo para o arquivo
    caminho_completo{i-1,1} = fullfile(caminho, nome_pasta, auxnome_arquivo{i-
1,1});
end
```

```
% 8.1 Salvar como .fig
% Onde vai salvar
caminho_destino = fullfile(caminho, nome_pasta, nome_arquivo{1});
% 8.2 Salva com .jpeg
% % Nome do arquivo da figura
nome_figura2 = [nome_arquivo{1},'.jpg'];
% Onde vai salvar
caminho_destino2 = fullfile(caminho, nome_pasta, nome_figura2);
end

function [tempo, aceleracao]=Xcorrelacao(caminho_completo,Nace)
%Função Xcorrelacao para alinhar aceleração
tempo = cell(1,Nace);
aceleracao = cell(1,Nace);
tamanho = zeros(1,Nace);
% 2.1.3 Função para ler arquivo e obter tempo e aceleração
for j1=1:size(caminho_completo,1)
   [auxtempo, auxaceleracao]=ArquivoLeitura2(caminho_completo{j1}, Nace);
   tempo(:,j1) = {auxtempo};
   aceleracao(:,j1) = {auxaceleracao};
end
% 2.1.4 Função xcorr
for j1=1:Nace
   for j2=1:Nace
      if j1~=j2
         [xCorr, lags] = xcorr(aceleracao{j1}, aceleracao{j2});
         [~, IxCorr]=max(xCorr);
         lag(j1,j2) = lags(IxCorr);
      end
   end
end
% Ajustar os dados com base nas defasagens calculadas
linhas_negativas = find(all(lag <= 0, 2));
I = linhas_negativas(1);
for j1 = 1:Nace
   if j1 ~= I
      if lag(j1, I) ~= 0
         aceleracao{j1} = aceleracao{j1}(1 + abs(lag(j1, I)):end);
      end
   end
end
% Ajustar para o menor comprimento
for j1 = 1:Nace
   tamanho(j1) = size(aceleracao{j1}, 1);
end
Menor = min(tamanho);

for j1 = 1:Nace
   aceleracao{j1} = aceleracao{j1}(1:Menor);
end
```

```
% Ajustar o tempo para o menor comprimento
tempo = auxtempo(1:Menor);

% Converter células para matriz
aceleracao = cell2mat(aceleracao);
end


function [Fv,ft_dd,Fvm,ft_ddm]=Tfft(tempo, aceleracao)
% Função para transformada rápida de Fourier
for i=1:size(aceleracao,2)
   %Transformada Este trecho de código é utilizado na função response_spectrum.m
   L = length(tempo); % Signal Length (samples)
   Ts = mean(diff(tempo)); % Sampling Interval (second)
   Fs = 1 / Ts; % Sampling Frequency (Hz)
   dm = aceleracao(:,i) - mean(aceleracao(:,i));
   Fn = Fs / 2; % Nyquist Frequency (Hz)
   ft_d = fft(dm) / L; % Normalized Fourier Transform
   Fv(:,i) = linspace(0, 1, fix(L/2) + 1) * Fn; % Frequency Vector
   Iv = 1:length(Fv(:,i)); % Index Vector

   %Ponto mais alto
   ft_dd(:,i)=abs(ft_d(Iv));
   ft_ddm(:,i)=max(ft_dd(:,i));
   Fvm(:,i)=Fv(find(ft_dd(:,i)==max(ft_dd(:,i))));
end
end

function [] = Graf(tempo, aceleracao, Fv, ft_dd, Fvm, ft_ddm, j2, j1, caminho_destino)
%% Plotar no domínio do tempo e no domínio da frequência
% Localização e tamanho
for i=1:size(aceleracao,2)
   f1=figure;
   f1.Position=[200 100 1000 350];

   % Gráfico 1 - Aceleração em Função do Tempo
   subplot(2, 1, 1);
   hLine1 = plot(tempo, aceleracao(:,i), 'b.-');
   title('Aceleração em Função do Tempo');
   xlabel('Tempo (s)');
   ylabel('Aceleração (m/s²)');
   grid on
```

```
% Gráfico 2 - Amplitude em Função da Frequência
   subplot(2, 1, 2);
   hLine2 = plot(Fv(:,i), ft_dd(:,i), 'r.-');
   hold on;
   plot(Fvm(:,i), ft_ddm(:,i), 'go', 'MarkerSize', 10); % Adiciona o ponto máximo
   text(Fvm(:,i), ft_ddm(:,i), sprintf('  Frequência: %.2f', Fvm(:,i)), 'VerticalAlignment',
'bottom'); % Adiciona o valor do ponto máximo
   % text(Fvm, ft_ddm, sprintf('  Max: %.2f', ft_ddm), 'VerticalAlignment', 'bottom',
'HorizontalAlignment', 'left', 'Color', 'blue', 'FontSize', 10, 'FontWeight', 'bold',
'BackgroundColor', [1 1 1]);
   hold off;
   title('Amplitude em Função da Frequência');
   xlabel('Frequência (Hz)');
   ylabel('Amplitude (m/s²)'); % Unidade dependerá das unidades originais dos seus
dados
   grid on
   % Adiciona o título geral à figura
   % Nome para a Legenda
   TituloGeral = sprintf('Ensaio %d - Repetição %d', j2,j1);%colocar no começo
   sgtitle(TituloGeral);

   exportgraphics(gcf,[caminho_destino,sprintf('_AceFFt_%d',i),'.jpg']);
   % Feche a figura
   close(gcf);
end
end
```

```
function [] = GrafFDD(j2, j1, F, SV, I)
% Plotar as formas modais no domínio da frequência
% Nome para a Legenda
Legenda = sprintf('Ensaio %d - Repetição %d', j2, j1); % colocar no começo

f1 = figure;
% Tamanho do plot
f1.Position = [200 100 1000 350];
plot(F, 10*log10(abs(SV)));
hold on;
% Plotar os pontos de máximo
% Pegar o SV do CMFI 1
k = SV(1, :);
k = 10*log10(abs(k(I)'));
% Pegar F
kk = F(I);
plot(F(I), k', 'go', 'MarkerSize', 10); % Adiciona o ponto máximo
for jj = 1:size(I, 2)
    texto = sprintf('Frequência: %.2f', kk(jj));
    text(kk(jj), k(jj), texto); % Adiciona o valor do ponto máximo
end
xlabel('Frequência (Hz)');
ylabel('Magnitude (dB)');
grid on;
title(Legenda);
hold off;
end
```