ESCOLA NAVAL

talant de bien faire

Bruno Filipe Pinto Ramos

Sistema *Low Cost* para Análise de Vibrações a Bordo de Navios de Guerra

Projeto e Desenvolvimento de uma Aplicação para *Smartphone* com Sistema *Android*

Dissertação para obtenção do grau de Mestre em Ciências Militares Navais, na especialidade de Engenheiros Navais – Ramo de Mecânica

Alfeite

2019

Bruno Filipe Pinto Ramos

Sistema Low Cost para Análise de Vibrações a Bordo de Navios de Guerra

Projeto e Desenvolvimento de uma Aplicação para Smartphone com Sistema Android

Dissertação para a obtenção do grau de Mestre em Ciências Militares Navais, na especialidade de Engenheiros Navais – Ramo de Mecânica

Orientação de: Professor Doutor Rui Pedro Chedas Sampaio

Co-orientação de: Professor Doutor Bruno Duarte Damas

O Aluno Mestrando

O Orientador

_____

Bruno Filipe Pinto Ramos
ASPOF EN-MEC

_____

Professor Doutor Rui Pedro Chedas Sampaio

Alfeite

2019

Epígrafe

“Success is not final, failure is not fatal: It is the courage to continue that counts.”

Winston S. Churchill

Agradecimentos

O desenvolvimento deste trabalho não dependeu apenas de mim, mas também de muitas outras pessoas que contribuíram direta ou indiretamente para a conclusão do mesmo, às quais gostaria de deixar um agradecimento.

Ao Professor Chedas Sampaio e ao Professor Bruno Damas, por toda a disponibilidade e conhecimento que me passaram ao longo desta jornada, para que conseguisse concluir os meus objetivos para este trabalho.

Ao Sr. Hélder Ferreira do Gabinete de Avaliação de Condição da Direção de Navios, que apesar de todo o trabalho inerente à sua função, sempre teve disponibilidade para me ajudar neste trabalho e me indicar o melhor caminho.

À minha namorada, Ana Catarina, por toda a paciência e suporte nos meus momentos menos bons, e pelas ideias que sempre me deu quando não sabia como ultrapassar um obstáculo.

Aos meus pais e à minha irmã, que ao longo destes últimos 5 anos de percurso académico, me deram alento para continuar a lutar.

Ao meu camarada Moreira Fernandes, por todas as horas na sala de reuniões do Departamento de Ciências e Tecnologias. Travámos esta luta lado a lado, cada um com as suas dificuldades, ajudando-nos um ao outro, para que no fim ambos tivéssemos sucesso.

Aos camaradas do Curso Jorge Álvares que me ajudaram verdadeiramente neste percurso. A esses o meu profundo agradecimento.

Quero deixar igualmente o meu sincero agradecimento a todas as pessoas que durante a minha vida contribuíram para a minha formação como pessoa, desde professores a amigos. O que sou hoje, a vocês o devo.

Resumo

O elevado nível de operacionalidade exigido às marinhas de guerra só pode ser garantido se um bom programa de controlo de condição for implementado. Na Marinha Portuguesa, a monitorização da condição é feita em duas etapas: a deteção executada pela guarnição do navio e o diagnóstico realizado por uma equipa especializada sediada numa unidade em terra. A deteção é feita mensalmente através de medição de vibrações de máquinas selecionadas, como diesel geradores, bombas de incêndio, compressores de ar e bombas de leme hidráulicas. O diagnóstico é feito a pedido ao Gabinete de Avaliação de Condição, isto é, é requerido quando o navio faz uma deteção. Atualmente, existem menos de meia dúzia de analisadores de vibrações para todos os navios da Marinha, o que é manifestamente insuficiente para a ambição e o sucesso do controlo de condição de equipamentos a bordo das unidades navais. Deste modo, as guarnições dos navios enfrentam dificuldades para fazer a monitorização periódica, já que esta só pode ser realizada quando o navio estiver atracado. O objetivo deste trabalho é fornecer uma solução para um equipamento de baixo custo que possa ser utilizada por todos os navios da Marinha Portuguesa para realizar o controlo regular da condição das suas máquinas. Além disso, a aplicação melhorará o sistema de deteção, uma vez que, além dos critérios da ISO 10816, já utilizados pela Marinha Portuguesa, serão adicionados os critérios da DNV-GL RU-SHIPS Pt.6 Ch.8, sendo esta uma regra de sociedade classificadora dedicada a navios. Esta solução é baseada na medição de vibração usando um acelerómetro de tecnologia MEMS (*Micro Electro Mechanical System*) conectado a um dispositivo *Android* através da porta USB e uma aplicação (comummente designada de *app*) de análise desenvolvida especificamente para esta plataforma que pode ser facilmente utilizada pela guarnição. Trata-se de uma melhoria significativa dos custos de aquisição em comparação com a instrumentação profissional, cujo preço normalmente alcança valores de milhares de euros para cada dispositivo.

Palavras-chave: Vibrações mecânicas, Controlo de condição, Analisador de vibrações.

Abstract

The high rate of operability required to the navies can only be guaranteed if a good condition monitoring schedule is implemented. In the Portuguese Navy the condition monitoring is done in two stages: the detection executed by the crew of the ship and the diagnosis performed by a specialized team on a land-based unit. The detection is done every month by vibration measurement, on seleted machinery like diesel driven generators, fire pumps, air compressors, and steering hydraulic pumps. The diagnosis is done by request by Condition Evaluation Office, *i.e.,* it's required when the ship's crew does a detection. Currently there are less then an half of a dozen of vibration measurement and analysis devices for all the ships of the Navy, which is manifestly insuficiente for the ambition and success of the condition monitoring in machines on naval ships. Therefore, ships' crews struggles to do periodic monitoring, since this one only can be done when the ship is docked. The objective of this work is to provide a solution for a low-cost equipment that can be used by every Portuguese Navy ships to perform regular condition monitoring on their machines. Besides that, the application will improve the system of detection, since, in addition to ISO 10816 criteria, already used by Portuguese Navy, DNV-GL RU-SHIPS Pt.6 Ch.8 criteria will be added, since this rule is applied to ships. This solution is based on vibration measurement using an technology MEMS (*Micro Electro Mechanical System*) accelerometer connected to an Android device via an USB port, and a specifically developed analysis application (commonly denominated by app) for that platform that can be easily used by the crew. This is a tremendous cost improvement when compared to professional instrumentation, whose price can typically reach values of thousands of euro for each device.

Keywords: Mechanical vibrations, Condition Monitoring, Vibration analyser.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Abreviaturas, siglas e acrónimos

DME (Divisão de Mecânica)

DN (Direção de Navios)

DNV – GL (*Det Norske Veritas – Germanischer Loyd*)

DPA (Divisão da Plataforma)

FFT (*Fast Fourier Transform*)

GAV (Gabinete de Avaliação de Condição)

ISO (*International Organization for Standardization*)

RMS (*Root Mean Square*)

RPM (Rotações por Minuto)

OTG (*On-The-Go*)

USB (*Universal Serial Bus*)

1 Introdução

As Marinhas de Guerra são organizações importantíssimas para os interesses de um país na zona marítima a ele atribuída, não só na exploração dos recursos nela presentes como também na vigilância deste espaço e salvaguarda da vida daqueles que nele navegam. Inerente a isto está o elevado nível de operacionalidade exigido às Marinhas de Guerra, traduzindo-se em navios em prontidão quase constante.

Deste modo, é importante que um sólido programa de manutenção preventiva seja instaurado, por forma a que esse nível de operacionalidade seja alcançado e para que a segurança da guarnição seja garantida bem como uma franca redução de custos, inerente ao facto de se evitar paragem dos equipamentos devido a avarias inesperadas.

O controlo de condição, no âmbito da manutenção preventiva, acaba por ter um papel importantíssimo nestes objetivos, uma vez que, apesar de todas as ações sistemáticas previstas pelo fabricante para a manutenção das máquinas, por vezes desenvolvem-se danos nas mesmas que apenas com a vigilância da condição das mesmas se consegue detetar. O controlo de condição pode ser realizado de várias maneiras, como, por exemplo, análise termográfica, análise física e química de fluidos lubrificantes e medição e análise de vibrações mecânicas, sendo este último método o que será abordado ao longo da presente dissertação.

Na Marinha Portuguesa, já se encontra implementado o controlo de condição tendo como coordenador o Gabinete de Avaliação de Condição (GAV), integrante da DN. Para além disto existe também um programa implementado na Marinha denominado VibControlo, que assenta num controlo mensal das vibrações de equipamentos previamente selecionados e considerados vitais das unidades navais, estando a execução deste programa sob a alçada do GAV.

Com o aumento da sensibilidade das guarnições dos navios para a importância do controlo de condição, surgiu a problemática de falta de equipamentos disponíveis para fazer

a deteção. Uma vez que a aquisição de novos equipamentos se torna dispendioso, surge o desafio de desenvolver um sistema que permita realizar o controlo de condição necessário e que ao mesmo tempo seja de baixo custo.

1.1 Objetivo da Dissertação

Este trabalho de investigação surgiu para fazer frente ao desafio mencionado anteriormente. O objetivo desta dissertação passou por desenvolver uma aplicação destinada a *smatphones* com sistema operativo *Android*, que fosse compatível com um acelerómetro de baixo custo, podendo assim qualquer navio ter autonomia para medir vibrações a bordo e realizar deteção e diagnóstico de dano nos equipamentos, caso este exista.

1.2 Estrutura da Dissertação

Esta dissertação encontra-se dividida em cinco capítulos, cada um deles dividido em secções.

No primeiro capítulo, o da Introdução, faz se um breve enquadramento do tema no contexto da Marinha Portuguesa e das suas necessidades, bem como uma explicação do objetivo deste trabalho.

No segundo capítulo, o do Estado da Arte, é feita uma abordagem teórica ao controlo de condição e como ele está implementado na Marinha Portuguesa, seguido de uma contextualização histórica da evolução dos equipamentos de medição e análise de vibrações, bem como uma análise dos sistemas que existem na atualidade, focando-se mais nos equipamentos de custo reduzido.

No terceiro capítulo, o do Controlo de Condição em Máquinas a Bordo de Navios de Guerra, faz-se uma abordagem aos conceitos teóricos e matemáticos por trás da medição e análise de vibrações e por fim uma abordagem às normas e regras de sociedade classificadoras que ditam valores limite aconselhados para a deteção.

No quarto capítulo, o do Sistema de Medição e Análise de Vibrações para Sistema Operativo *Android*, faz-se uma abordagem metódica acerca do desenvolvimento da aplicação, e uma apresentação dos testes a que aplicação foi sujeita bem como resultados que alcançou.

No último capítulo, o da Conclusão, faz-se um balanço final do trabalho desenvolvido, bem como dos objetivos alcançados, e por fim algumas sugestões para trabalhos futuros.

Por fim pode-se encontrar no final vários Apêndices, onde podem ser consultados tabelas e excertos de códigos de programação, destinados a uma melhor compreensão de alguns conteúdos abordados ao lodo da presente dissertação.

2 Estado da Arte

2.1 Controlo de Condição

Antes da Revolução Industrial, a produção de bens processados era assegurada pelo homem, recorrendo à manufatura, sendo que a produtividade e qualidade dependia maioritariamente da quantidade, qualidade e experiência da mão de obra. Com o desenvolvimento demográfico, e por forma a acompanhar as necessidades da população mundial, tornou-se crucial o desenvolvimento de métodos de produção de bens que permitissem um fornecimento maior dos mesmos ao mercado. Em resposta a esta necessidade, e com a revolução industrial, começam a surgir linhas de montagem acompanhadas de equipamentos eletromecânicos que vieram massificar o processo de produção. No início, as máquinas não eram sujeitas a manutenção preventiva, surgindo com frequência avarias que muitas vezes tinham consequências graves como perdas de vidas humanas e paragens de produção.

Surge assim o conceito de Manutenção Preventiva, com o objetivo de prevenir possíveis avarias, caindo em desuso a tão usual Manutenção Corretiva. É importante assim, e antes de avançarmos para a explicação do objetivo da manutenção preventiva, entender a diferença entre dois conceitos, o de dano e o de avaria, que apesar de parecerem semelhantes, apresentam significados bem diferentes. Quando nos referimos a dano referimo-nos a alterações verificadas desde o momento em que um bem começa a ser utilizado desde que veio de fábrica. Podemos mesmo afirmar que a partir do momento em que um bem é concebido na sua totalidade começam a desenvolver-se uma série de danos que vão-se manifestar nas suas caraterísticas de funcionamento.Quando nos referimos a avaria referimo-nos a um estado do bem em que este já não cumpre a função requerida (Instituto Português da Qualidade, 2007). Este conceito acaba por ser bastante subjetivo, uma vez que a função requerida de um bem varia de pessoa para pessoa, de organização para organização.

Segundo a Norma Portuguesa NP 13306:2007, a manutenção preventiva consiste em toda a "manutenção efetuada a intervalos de tempo pré-determinados ou de acordo com critérios prescritos com a finalidade de reduzir a probabilidade de avaria ou degradação do funcionamento de um bem". Essa mesma norma também define dois tipos de manutenção preventiva:

- Manutenção Preventiva Sistemática, ou manutenção preventiva efetuada a intervalos de tempo pré-estabelecidos ou segundo um número definido de unidades de utilização, mas sem controlo prévio do estado do bem (Instituto Português da Qualidade, 2007);

- Manutenção Preventiva Condicionada, ou manutenção preventiva baseada na vigilância do funcionamento do bem e/ou dos parâmetros significativos desse funcionamento, integrando as ações daí decorrentes (Instituto Português da Qualidade, 2007).

O Controlo de Condição é nada mais do que a vigilância do funcionamento do bem e/ou dos parâmetros significativos desse funcionamento, pelo que a implementação da manutenção preventiva condicionada passa sempre pela prática do Controlo de Condição.

Tal como as pessoas, que regularmente recorrem aos serviços de saúde a fim de realizar análises e exames médicos, as máquinas devem ser avaliadas em diferentes parâmetros, que, quando analisados, podem revelar sintomas de algum dano em rápida evolução.

O controlo de condição, ao tornar possível a deteção de um dano numa máquina pode evitar acidentes de trabalho que ponham em causa a vida humana, bem como desastres ambientais. Segundo Garrison, numa análise de 100 acidentes em centrais petroquímicas entre 1958 e 1987, a principal causa foi a falha mecânica dos equipamentos, seguido de falha humana na operação, representando 38 e 26 acidentes respetivamente (Rao B. K., 1998)

O controlo de condição tem como objetivo dar a conhecer o estado atual do equipamento, bem como a identificação do dano. Esta deve ser feita em duas fases e na ordem apresentada:

- Primeiro, através da deteção, que se trata de uma avaliação genérica, isto é, apenas nos diz se existe dano, caso se ultrapasse um certo valor limite, ou então pode-nos ajudar a prevenir a progressão de um dano ao realizar análise de tendência;

- Segundo, através do diagnóstico, em que se faz uma avaliação específica, ou seja, consegue-se saber a natureza do dano, a localização e o grau de gravidade do mesmo, bem como uma previsão de quando é que o dano pode causar uma avaria no bem.

O controlo de condição pode ser feito recorrendo a várias tecnologias:

- Termografia: na termografia faz-se um estudo da distribuição de temperatura de um equipamento, sendo possível detetar temperaturas excessivas num certo componente de um determinado equipamento, o que pode ser relacionado com algum dano em específico (Figura 1);

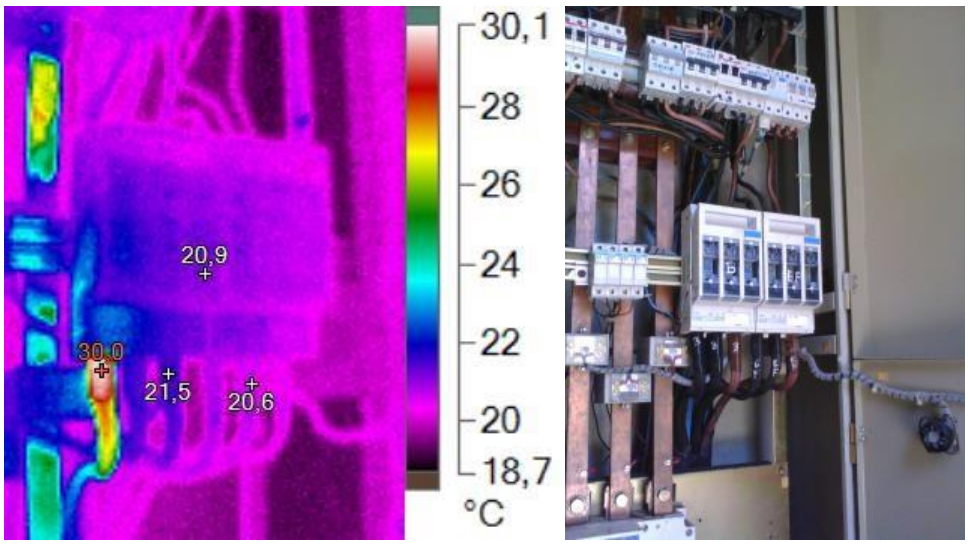


Figura 1 – Exemplo de aplicação de técnicas de termografia num quadro elétrico (Mundinstal - Técnicas de Instalações Elétricas, Lda, 2014)

- Análise Físico-Química de fluídos: nesta tecnologia, realiza-se a análise de fluidos como óleos de lubrificação ou combustíveis, sendo que, dependendo

das alterações verificadas nos parâmetros físico-químicos relativamente a valores padrão, pode-se aferir a existência de algum dano na máquina (ex.: presença de limalhas de ferro em óleos de lubrificação de um motor de combustão interna pode significar desgaste do veio de manivelas);

- Medição e análise de vibrações mecânicas: nesta técnica de controlo de condição é possível através da medição de vibrações detetar se existe algum dano numa máquina, ou numa fase mais precoce prever quando é que se está a desenvolver um dano. Através de uma análise mais específica é possível em alguns casos determinar a localização, amplitude e natureza do dano e quando é que este poderá levar a uma avaria.

Atualmente, num mundo cada vez mais industrializado e competitivo, em que um dia de paragem de produção devido à falha de equipamentos de produção pode vir a causar um prejuízo de milhares de euros (Figura 2), torna-se crucial que o controlo de condição seja implementado.

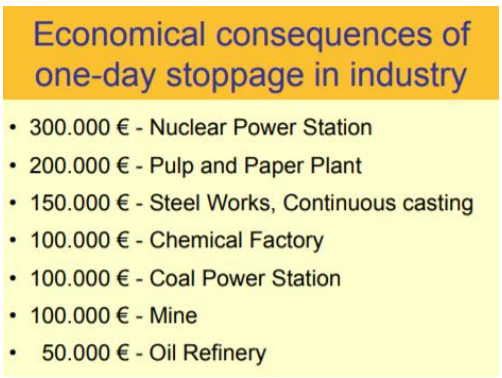


Figura 2 – Custos associados a um dia de paragem em diferentes tipos de indústria (VTT Technical Research Centre of Finland, 2006)

2.2 Controlo de Condição de Máquinas na Marinha Portuguesa

A Marinha Portuguesa trata-se de uma organização à qual se exige um elevado nível de operacionalidade, fruto das missões atribuídas, sendo elas a salvaguarda da vida humana

no mar, salvaguarda dos interesses nacionais no nosso território marítimo e representação de Portugal em missões e exercícios internacionais (Marinha Portuguesa, 2019). Para que essa operacionalidade seja garantida, é necessária a máxima disponibilidade dos meios navais e, consequentemente, de todos os sistemas que destes fazem parte.

Para que a disponibilidade de todos os sistemas de um navio seja possível, a Marinha apoia-se num Sistema de Gestão da Manutenção, que, baseando-se em métodos de manutenção preventiva, diminui a possibilidade de avaria dos equipamentos e, por inerência, o tempo de paragem forçada a que estes possam ser sujeitos.

Segundo a publicação ILA 5 (A) (Instruções para a Organização da Manutenção das Unidades Navais e outros Meios de Ação Naval), a Marinha entende por controlo de condição como a "função de gestão que consiste na análise e medição do comportamento real do sistema, atividade ou processo (…). A função inclui a escolha de ações corretivas a serem tomadas sempre que o comportamento real se afaste dos padrões pré-estabelecidos".

2.2.1 Gabinete de Controlo de Condição (GAV)

A Marinha dispõe já de meios para implementar o controlo de condição a bordo dos navios. O organismo responsável pela gestão e implementação desses meios é o GAV. O GAV encontra-se inserido na orgânica da DN, na alçada da Divisão de Mecânica (DME), que por sua vez se encontra sob alçada da Departamento da Plataforma (DPA) (Figura 3).

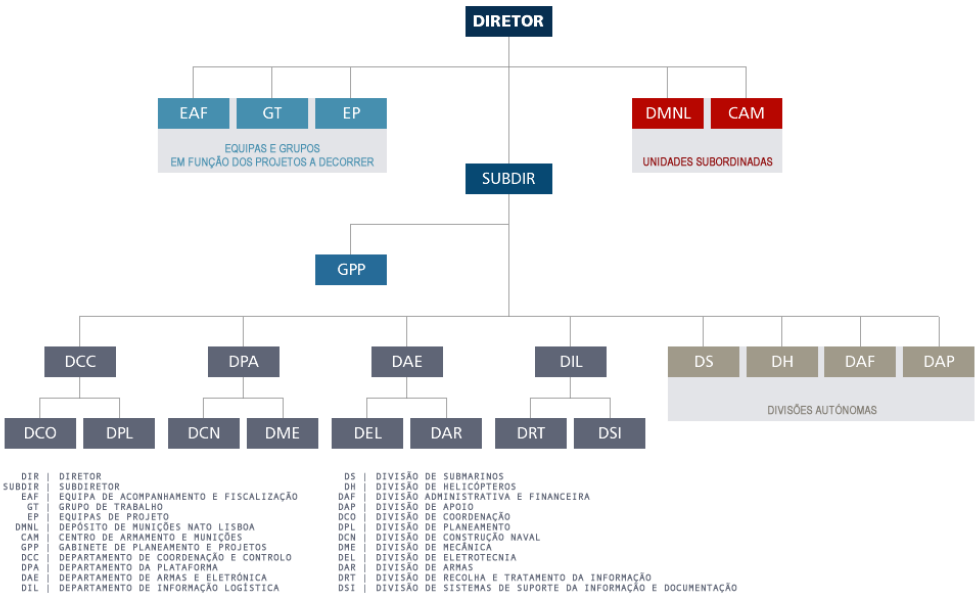


Figura 3 – Organograma da Direção de Navios.

A missão do GAV consiste na avaliação da condição das máquinas e dos combustíveis, lubrificantes e restantes fluidos que a elas dizem respeito, e ainda na avaliação qualitativa das vibrações estruturais, iluminação e ruído (Armada, Almirante Chefe de Estado-Maior da, 2016).

Em 2016 surge o projeto VibControlo@marinha.pt, cuja missão, retirada diretamente do Documento Iniciador de Projeto, consiste na implementação de "um plano de medição e análise de vibrações com a finalidade de reduzir as avarias nos navios, melhorando os indicadores de fiabilidade dos equipamentos e desenvolver o ensino da Manutenção na Escola Naval com investigação no âmbito do controlo de condição por medição de vibrações".

Este projeto vem alterar a metodologia do controlo de condição, por medição de vibrações, até então implementada na Marinha. Resumidamente, propõe que:

- a adesão dos navios ao projeto seja voluntária considerando a motivação fator primordial no sucesso do projeto;
- o GAV defina, com apoio de cada classe de navios, os equipamentos a monitorizar e os critérios de alarme;

- todas as medições, com a finalidade de deteção, passem a ser executadas exclusivamente por cada navio aderente, deixando de ser executadas pelo GAV;
- todas as medições e respetivas análises, com a finalidade de diagnóstico, passem a ser executadas exclusivamente pelo Comando Administrativo e/ou pelo GAV (Figura 4), deixando de ser partilhadas com o Arsenal do Alfeite.

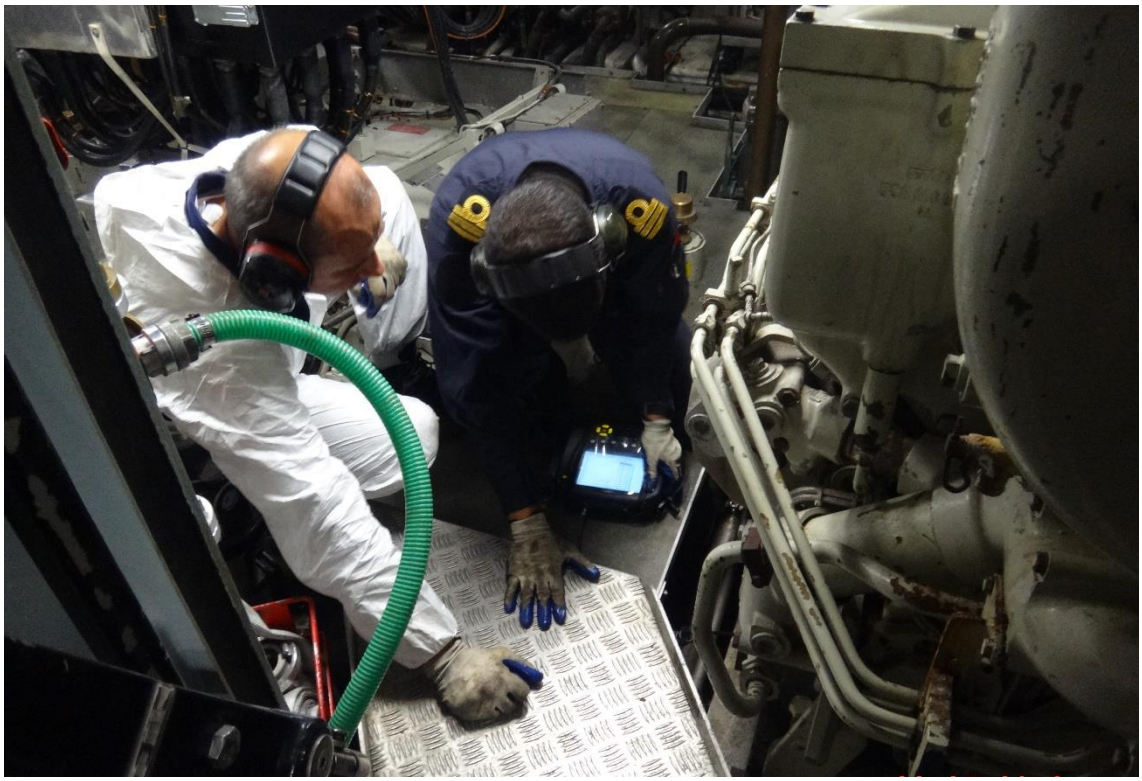


Figura 4 – Equipa do GAV a realizar medição de vibrações a bordo de um navio da Marinha Portuguesa.

Concluindo, de modo a que a Marinha Portuguesa tenha máxima disponibilidade para cumprimento das suas missões, e para defesa e implementação dos interesses da nação, é vital que os equipamentos que integram todas as unidades navais apresentem um elevado nível de fiabilidade e reduzidos períodos de paragem para manutenção corretiva, pelo que a organização deve ter por base um rígido programa de controlo de condição dos equipamentos.

2.3 Sistemas Portáteis de Medição de Vibrações

A medição e análise de vibrações mecânicas em equipamentos, possibilita a deteção e o diagnóstico de dano nas máquinas em funcionamento. Para isso, foram desenvolvidos ao longo das últimas décadas sistemas portáteis de medição e análise de vibrações que pudessem satisfazer as necessidades das empresas no que diz respeito a esta técnica de controlo de condição.

Os primeiros sistemas de medição e análise de vibrações mecânicas eram fixos e localizados em laboratórios (Archambault, 2003). O primeiro dispositivo portátil concebido para o efeito surgiu ainda na década de 40, desenvolvido por Art Crawford (Figura 5). Este dispositivo permitia ao utilizador realizar análise de fase da vibração. Em meados da década de 70 surgiram os primeiros analisadores em tempo real que aplicavam ao sinal no tempo a transformada de Fourier, estando limitados no tempo que demorariam a realizar uma análise de apenas uma amostra no tempo (Figura 6).

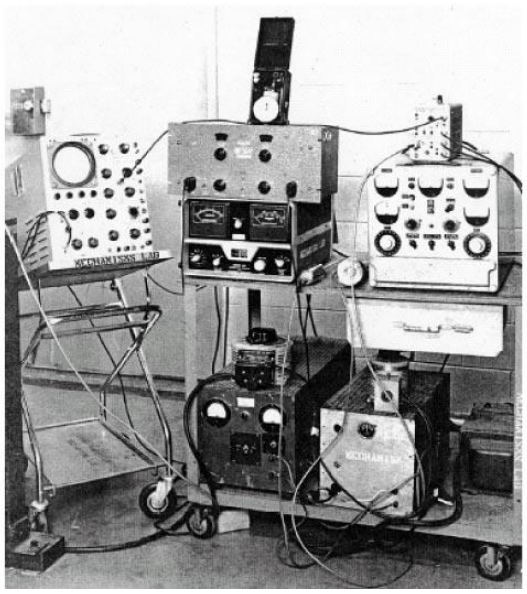


Figura 5 – Sistema fixo de medição e análise de vibrações da década de 50 (Archambault, 2003)

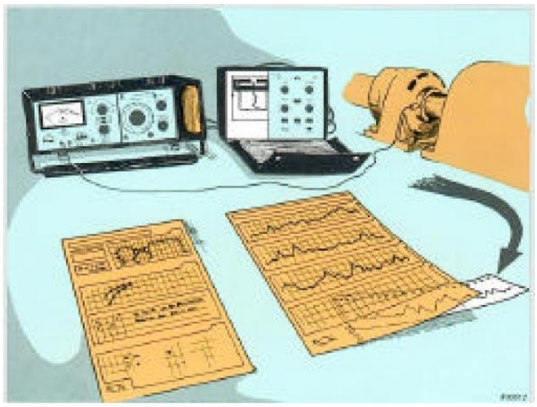


Figura 6 – Representação esquemática de um medidor de vibração portátil com capacidade de gravação de sinal e aplicação de filtros (Archambault, 2003)

A Transformada Rápida de Fourier (Fast Fourier Transform, FFT) veio revolucionar o processamento de sinal, tendo o seu algoritmo, atualmente utilizado, sido publicado em 1965 por Cooley e Tukey (Cooley, Lewis, & Welch, 1967), apesar de outros algoritmos de FFT terem sido desenvolvidos em décadas anteriores. No início da década de 80 surgiram os primeiros sistemas portáteis que gravavam a medição para, mais tarde, no laboratório, se aplicar o algoritmo FFT (Archambault, 2003).

Foi ainda na década de 80 que se verificou o proliferar do controlo de condição pelas empresas, que aliado ao aparecimento dos primeiros computadores pessoais, veio causar um aumento na venda de instrumentação para medição e análise de vibrações mecânicas. Os primeiros modelos de instrumentação eram capazes de calcular níveis globais, tendo mais tarde surgido equipamentos capazes de calcular FFT's com 400 linhas de espetro (Archambault, 2003).

Surgiram assim empresas especializadas no desenvolvimento de instrumentação para a implementação do controlo de condição, como é o caso da SKF Condition Monitoring Division, Vitec, DLI, entre outras (Archambault, 2003).

Por volta dos anos 80, houve uma grande evolução na medição de vibrações em rolamentos e engrenagens através da técnica do envelope, o que levou a que, na década de

90, as empresas que desenvolviam sistemas de medição de vibrações apostassem mais em equipamentos com capacidade para medir vibrações numa maior largura de banda e com uma resolução em frequência maior, por forma a permitir uma análise mais completa de todos os componentes de máquinas (Archambault, 2003).

A partir do início do novo milénio, para além do aperfeiçoamento de características técnicas dos equipamentos, tem-se verificado uma maior preocupação no desenvolvimento de capacidade de análise e interpretação de dados mais automatizada e inteligente e principalmente dotar os sistemas com capacidade de processamento de dados e análise preliminar bastante superior. Para além disso começaram a ser incluídas nos dispositivos valores limite de referência, de acordo com normativos internacionais, como é o caso das normas ISO, assunto este que será abordado posteriormente.

2.4 Sistemas *Low Cost* de Medição de Vibrações

Atualmente o mercado apresenta um vasto leque de analisadores de vibrações, cujo preço varia segundo a sua complexidade e funcionalidades oferecidas. Atualmente a Marinha, no âmbito do projeto VibControlo, possui três medidores de vibrações, da marca SKF, mas que, no entanto, apresentam preços avultados que rondam os 10 e os 30 mil euros.

A fim de cumprir um dos objetivos deste projeto, que é o desenvolvimento de uma solução *low cost* para a medição e análise de vibrações, tornou-se necessário realizar um estudo de mercado para que se perceba se o trabalho realizado consegue fazer concorrência ao nível da relação qualidade preço, e deste modo conseguir equipar as unidades navais com uma boa solução ao nível de analisadores de vibrações.

2.4.1 SKF QuickCollect Sensor

Este equipamento consiste em um sensor portátil com capacidade para medição de vibrações e conexão via *Bluetooth* ao *smartphone* (Figura 7), recorrendo à aplicação desenvolvida pela mesma empresa, SKF *QuickCollect* (Figura 8).

Em termos de características técnicas, este equipamento permite a medição de envelope de aceleração, velocidade e temperatura, sendo que apresenta uma largura de banda

entre 10 Hz e 1kHz, medindo valores RMS e pico a pico. Este equipamento ronda os 2000 euros, o que se torna um preço pouco acessível, mas aceitável tendo em conta as funcionalidades que o equipamento possui.

Figura 7 - SKF *QuickCollect Sensor* (SKF, s.d.)

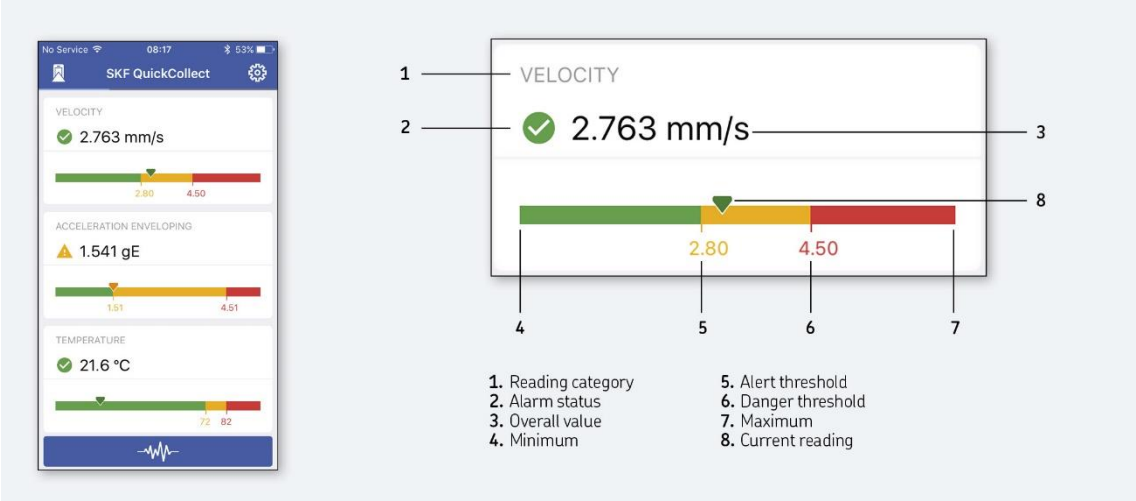


Figura 8 – Interface da aplicação desenvolvida pela SKF compatível com o SKF *QuickCollect Sensor* (SKF, s.d.)

2.4.2 SKF MCA (Machine Condition Advisor)

Tal como o SKF *QuickCollect Sensor*, o SKF MCA (*Machine Condition Advisor*) consiste em um equipamento portátil para medição e análise de vibrações (Figura 9). Este equipamento tem a capacidade também de medir velocidade e envelope de aceleração e, recorrendo a um pequeno visor integrado, alertar o operador para situação de alarme, ou

perigo, baseando-se na norma ISO 10816. Este dispositivo é equipado com um acelerómetro com capacidade de medir vibrações na largura de banda de 10 Hz a 1 kHz e o seu preço ronda os 2000 euros.

Figura 9 - SKF MCA (*Machine Condition Advisor*) e exemplo de utilização (SKF, s.d.)

2.4.3 SKF Machine Condition Indicator CMSS 200

O *SKF Machine Condition Indicator CMSS 200* trata-se de um equipamento mais simples e de preço mais acessível. Trata-se de um simples acelerómetro piezoelétrico, sem fios. Este equipamento, tal como os anteriores mede a aceleração em envelope de aceleração e em velocidade, na gama de 10 Hz a 1 kHz, e também a temperatura da superfície da máquina. O dispositivo vem equipado com 3 leds, que, consoante a cor acesa, alertam o utilizador de uma situação de perigo para que análises mais pormenorizadas sejam realizadas posteriormente (Figura 10).

Este equipamento pode ser encomendado em packs de 2, 10 ou 50, sendo que o preço médio por unidade ronda os 200 euros.

Figura 10 - SKF *Machine Condition Indicator* CMSS 200 (SKF, s.d.)

2.4.4 Vib Cloud e VibeCheck

A *Vib Cloud* e *VibeCheck* tratam-se de duas aplicações desenvolvidas para *smartphone*, tanto para sistema *Android* como para IOS (sistema operativo desenvolvido para *smartphones* da empresa *Apple*), criadas pela empresa Itnnovate, com a capacidade de medir e analisar vibrações com recurso a um acelerómetro piezoelétrico que se conecta via USB ao telemóvel.

A *VibeCheck* é uma aplicação gratuita sem nenhuma subscrição (Figura 11), sendo que apenas exige ao utilizador a aquisição do acelerómetro anteriormente referido, um Digiducer 333D01 USB *Digital Accelerometer* (Figura 12), e de um adaptador OTG (*On-The-Go*) que permita ao telemóvel receber a informação transmitida pelo sensor. Este tem um valor aproximado de 2000 euros.

Figura 11 – Logotipo da aplicação *VibeCheck* (Itnoovate, 2019)

Figura 12 – Acelerómetro piezoelétrico *Digiducer 333D01 USB Digital Accelerometer* (Digiducer, 2019)

Em termos de funcionalidades, para além da medição de vibrações, a aplicação avisa o utilizador no momento da medição caso exista algum valor fora do normal, sendo que essa avaliação será feita segundo a norma ISO 10816. O espetro de frequências é uma funcionalidade que o utilizador pode usufruir mediante o pagamento de 350 euros.

Relativamente à aplicação *Vib Cloud* (Figura 13), o seu download é igualmente gratuito, mas o seu usufruto está dependente da criação de uma conta sujeita a um pagamento de uma subscrição anual no valor de 5000 euros com a possibilidade de realizar *login* em 5 dispositivos diferentes, sendo que, por cada conta a mais requerida, acresce 900 euros à subscrição anual. Em termos de funcionalidades, esta aplicação aproxima-se mais de um equipamento profissional, como os que a Marinha possui, uma vez que para além da medição e análise de vibrações (Figura 14), possibilita a programação de rotas (lista ordenada de equipamentos a medir, tornando mais fácil e expedito para o utilizador a medição de vibrações para uma maior quantidade de equipamentos), armazenamento de dados de medições em base de dados online e análise das vibrações medidas por parte de profissionais da área, o que permitirá um estudo mais aprofundado do problema caso este exista (Itnoovate, 2019).

Tal como a aplicação *VibeCheck*, esta aplicação requer um acelerómetro que se ligue ao telemóvel via USB e respetivo adaptador.

Figura 13 - Logotipo da aplicação *Vib Cloud*

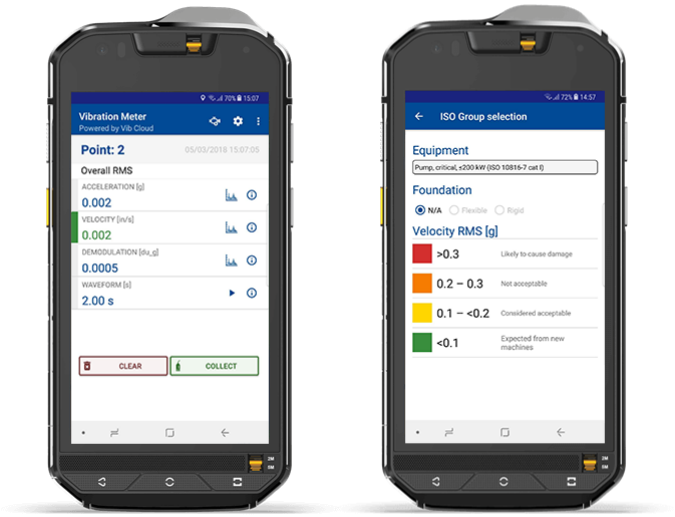


Figura 14 – Display da aplicação *Vib Cloud* (Itnoovate, 2019)

3 Controlo de condição de máquinas, por medição de vibrações, a bordo de navios de guerra

Neste capítulo será feita uma abordagem aos fundamentos teóricos da medição e análise de vibrações e por fim um estudo acerca de normas ISO e regras de Sociedades Classificadoras referentes a monitorização de vibrações mecânicas em navios.

3.1 Medição e análise de vibrações mecânicas

Para o desenvolvimento da aplicação, foi necessário um estudo prévio de conceitos teóricos relacionados com a medição e análise de vibrações mecânicas, que serão abordados nesta secção.

3.1.1 Vibração mecânica

A vibração mecânica consiste na resposta oscilante que um sistema mecânico apresenta quando afastado do equilíbrio estático por uma ação externa, força ou momento, sendo essa ação uma perturbação pontual ou contínua no tempo. A vibração pode também descrever-se como permutações alternadas da energia potencial e cinética do sistema cujo valor diminuirá com o atrito (Sampaio, 2017).

Para compreender este conceito, peguemos no caso de vibração mais simples: a vibração harmónica, provocada por um impacto num sistema não amortecido de um grau de liberdade. Quando o sistema se encontra em repouso, a sua energia potencial e cinética é nula. Quando perturbado, o sistema absorverá a energia transferida e transformá-la-á em energia cinética. À luz do Princípio da Conservação da Energia, no "percurso" entre a posição de repouso e a posição de afastamento máximo do estado de equilíbrio, a energia cinética irá diminuir e a potencial aumentar. Quando o sistema atingir a amplitude máxima, a energia cinética será nula, uma vez que a velocidade é nula também. Contrariamente, a energia potencial será máxima, o que resulta numa reaproximação do seu estado

de equilíbrio. Assim a energia potencial começará a diminuir e a cinética voltará a aumentar, sendo que o sistema passará pela posição de equilíbrio com uma velocidade máxima e uma energia potencial nula, pelo que ocorrerá um afastamento da posição de equilíbrio novamente.

A vibração harmónica pode ser representada matematicamente pela seguinte função:

$$x(t) = X \cdot \cos(2\pi f t + \alpha) \qquad (1)$$

onde X é a amplitude máxima da função harmónica, f é a frequência da função, isto é, o número de ciclos que ocorrem por unidade de tempo, e α é a fase, ou seja, o desfasamento que a função tem relativamente a um determinado referencial.

A vibração pode classificar-se de:

- Vibração livre: esta vibração ocorre quando um determinado sistema, após uma perturbação isolada do seu estado de equilíbrio, é deixado a vibrar, sem influência de nenhuma força externa (Rao S. S., 2011). Quando um sistema vibra livremente, a(s) frequência(s) a que o mesmo vibra denominam-se de frequências naturais;
- Vibração forçada: quando um sistema é sujeito a uma excitação externa que se repete ao longo do tempo, diz se que este apresenta uma vibração forçada. Se a excitação que se vai repetindo ao longo do tempo o fizer a uma frequência igual a uma das frequências naturais do sistema, ocorrerá um fenómeno denominado de ressonância, em que o sistema começa a apresentar amplitudes de vibração cada vez maiores o que pode pôr em causa a integridade do sistema (Rao S. S., 2011).

Enquadrando estes conceitos em vibrações de máquinas, a vibração livre ocorre quando um equipamento que não se encontra em funcionamento e é alvo de um impacto. A vibração forçada em sistemas mecânicos ocorre quando os mesmos se encontram em funcionamento. No entanto, idealmente, qualquer sistema mecânico que acabasse de sair de fábrica não deveria apresentar qualquer tipo de vibração, pois não teria desequilíbrios nem imperfeições. No entanto para além do perfecionismo se tornar bastante dispendioso,

é impossível eliminar completamente a vibração de um sistema. Assim, considera-se que a vibração forçada de uma máquina em funcionamento é o resultado de todas as anomalias e imperfeições que a máquina tem assim que sai de fábrica e que vai desenvolvendo ao longo do seu ciclo de vida, anomalias essas que se traduzem em forças e momentos que se irão repetir ao longo do tempo.

Cada anomalia ou dano tem as suas frequências de excitação características, como é o caso de um desalinhamento, de um desequilíbrio ou de um rolamento degradado. Deste modo, quando estas frequências aparecem na vibração forçada de um sistema, ou quando a amplitude da vibração a essas frequências aumenta para valores inaceitáveis, devemos tomar medidas a fim de prevenir uma avaria.

3.1.2 Fast Fourier Transform (FFT)

Em 1822, Joseph Fourier, demonstrou, através da Transformada de Fourier (direta ou inversa), que uma função periódica pode ser representada como uma série de funções harmónicas (Figura 15). Deste modo, considerando que $x(t)$ é uma função periódica, pode-se concluir que:

$$x(t) = \sum_{k=1}^{N} X_k \cdot \cos(2\pi f_k t + \alpha_k) \qquad (2)$$

em que N é o número de harmónicas que compõe o sinal original, X_k a amplitude da harmónica de ordem k, f_k a frequência da harmónica de ordem k e α_k a fase da harmónica de ordem k.

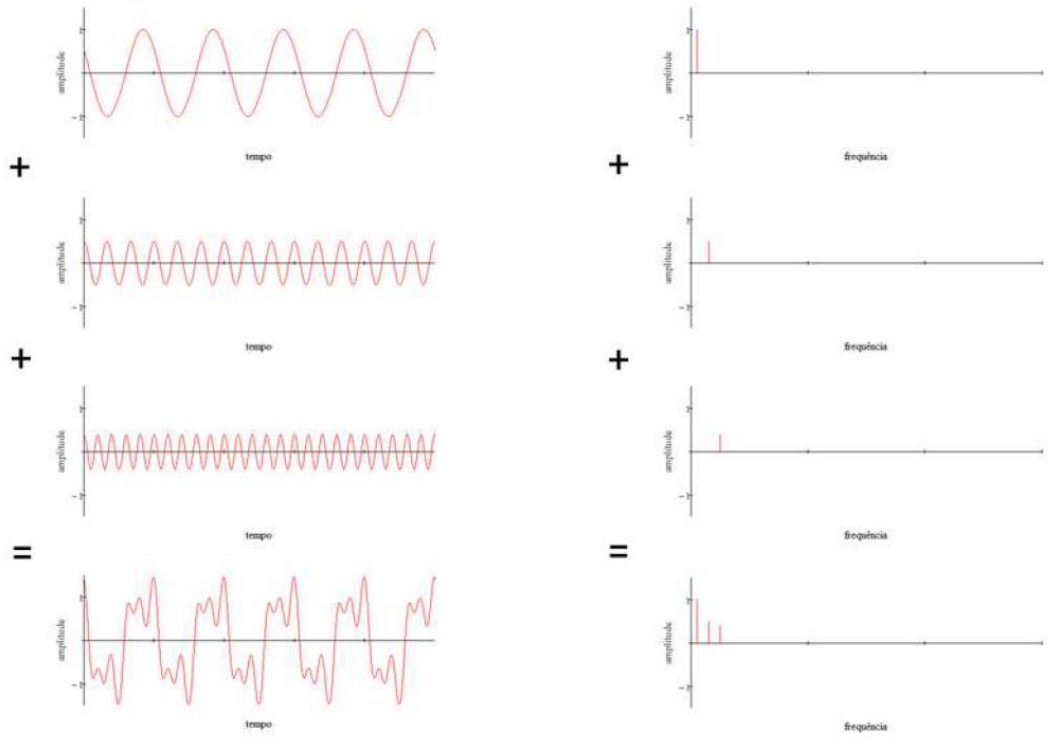


Figura 15 – Espetro de Frequência associado a uma função periódica composta por uma soma de harmónicas (Sampaio, 2017)

A transformada de Fourier têm várias versões, de acordo com o campo de aplicação das mesmas. Para o caso do sinal digital, a versão usada é a Transformada de Fourier Discreta, DFT.

No entanto, tendo em conta que um sinal digital é muitas vezes definido por um grande número de pontos, a DFT não se tornava muito útil devido à enorme quantidade de tempo que tinha de se despender no seu processamento. Deste modo o processamento de sinal digital encontrou o seu auge quando foi descoberto o algoritmo da FFT por Cooley e Tukey, na década de 60. De facto, para um sinal digital constituído por 2048 pontos, a DFT necessita de cerca de 640 vezes mais tempo para executar a transformada desse sinal, comparativamente com a FFT.

Quando se aplica a FFT a um sinal no tempo, obtém-se um espetro de frequências, que consiste num gráfico de barras, onde cada barra corresponde a uma frequência e amplitude de uma harmónica, como podemos verificar na figura 15. O espetro de frequência

torna-se assim importantíssimo, uma vez que nos permite identificar as harmónicas mais importantes (maior amplitude) e a sua origem (frequência) (Sampaio, 2017).

É assim importante perceber o que é que obtemos quando aplicamos o algoritmo da FFT a um sinal no tempo de uma determinada vibração.

Segundo a expressão da DFT,

$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n \cdot e^{-j\cdot\frac{\pi nk}{N}} \qquad (3)$$

em que X_k é a harmónica de ordem *k* do espetro de frequências para $k = 0,1,2,\dots,N$, N o número de pontos que compõem o sinal no tempo, x_n a amplitude de ordem *n* do sinal no tempo, para $n = 0,1,2,\dots,N$ e j a unidade imaginária de um número complexo.

A partir da expressão da DFT podemos concluir que o resultado será um vetor de números complexos, uma vez que, segundo a equação de Euler,

$$ze^{-j\theta} = z\cos\theta - jz\sin\theta \qquad (4)$$

pelo que podemos afirmar que

$$X_k = Re(X_k) + Im(X_k)j \qquad (5)$$

Assim, ao aplicarmos a DFT a um sinal no tempo representado num vetor de 1000 números reais, iremos obter como resultado um sinal em frequência representado por um vetor composto por 1000 números complexos. No entanto, se analisarmos o vetor obtido, podemos reparar que os números complexos que compõem a segunda metade são os conjugados dos que compõem a primeira metade, ou seja:

$$X_{N-k} = X_k^{\,*} \qquad (6)$$

Cada par de números complexos conjugados representa uma harmónica do sinal obtido, sendo que a amplitude e fase da mesma se obtém através das seguintes equações, respetivamente:

$$2|X_k| = 2\sqrt{(Re(X_k))^2 + (Im(X_k))^2} \qquad (7)$$

$$\alpha_k = \text{atan}\left(\frac{Im(X_k)}{Re(X_k)}\right) \qquad (8)$$

Se aplicarmos o algoritmo da FFT a um sinal digital, obteremos igualmente um vetor de números complexos, com a nuance de que o mesmo só pode ser aplicado a um sinal cujo o número de pontos seja uma potência de 2 (2^n, com $n \in \mathbb{N}^+$).

A frequência da harmónica é dada pelo índice *k* pela expressão *k/T* onde *T* é o tempo de aquisição.

3.1.3 Quantificação da Vibração

A quantificação da vibração refere-se à amplitude da vibração e à unidade de medida dessa amplitude. A quantificação torna-se importante quando queremos comparar a vibração medida com limites já estabelecidos, quer por normas, quer por regras de sociedades classificadoras, quer pelo fabricante do equipamento, ou quando queremos analisar o padrão da vibração por forma a retirar conclusões (diagnóstico) relativas ao estado do equipamento ou a algum tipo de dano no sistema (Sampaio, 2017).

A amplitude da vibração pode ser apresentada em unidades de deslocamento, velocidade ou aceleração. Normalmente as unidades utilizadas para medição de vibrações são o μm para medição em deslocamento, *mm/s* para medição em velocidade e m/s^2, ou *g* (aceleração da gravidade, 9.81 m/s^2), para medição em aceleração.

Para a passagem de unidades de aceleração em *g* para aceleração em m/s^2, multiplica-se o valor da aceleração (X_k) em *g* por 9.81.

Para a passagem de unidades de aceleração para unidades de velocidade, aplica-se a seguinte fórmula:

$$\dot{X}_k = \frac{\ddot{X}_k}{2 \cdot \pi \cdot f_k} * j * 1000 \qquad (9)$$

em que $\dot{X}_k$ é a velocidade de ordem *k* em *mm/s* e $\ddot{X}_k$ a aceleração de ordem *k* em m/s^2 e f_k a frequência de ordem *k* em Hz, com $k = 0,1,2,\ldots,K$, sendo *K* o número de pontos do vetor do sinal em frequência, e *j* a unidade imaginária de um número complexo.

Para a passagem de unidades de aceleração para unidades de deslocamento, aplica-se a seguinte fórmula:

$$X_k = -\frac{\ddot{X}_k}{(2\cdot\pi\cdot f_k)^2} * 10^6 \qquad (10)$$

em que X_k é o deslocamento de ordem *k* em μm e $\ddot{X}_k$ a aceleração de ordem *k* em m/s^2 e f_k a frequência de ordem *k* em Hz, com $k = 0,1,2,\dots,K$, sendo *K* o número de pontos do vetor do sinal em frequência.

Para o cálculo da frequência de ordem *k*, f_k, aplica-se a seguinte fórmula:

$$f_k = \frac{f_a}{2*K} * k \qquad (11)$$

em que f_k é a frequência de ordem *k* em Hz, com $k = 0,1,2,\dots,K$, sendo *K* o número de pontos do vetor do sinal em frequência, e f_a é a frequência de amostragem em Hz.

A frequência de amostragem calcula-se através da seguinte fórmula:

$$f_a = \frac{1}{\Delta t} \qquad (12)$$

em que Δt é a resolução no tempo (s/amostra).

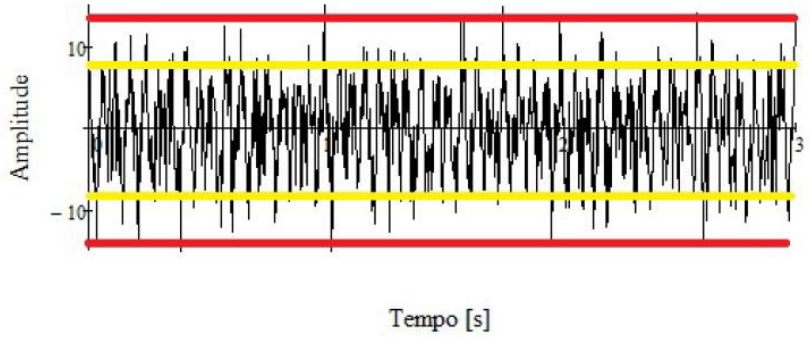
Para além das unidades de medição, a quantificação pode ser um valor instantâneo ou uma média de valores. Em termos de análise de vibração de máquinas costuma-se utilizar dois parâmetros: o pico, referente à amplitude máxima registada no período de aquisição do sinal, ou o valor RMS (Root Mean Square), referente à raiz quadrada das médias quadráticas das amplitudes registadas ao longo da aquisição. O valor RMS pode ser calculado a partir de um sinal no tempo, utilizando a equação (13) (em que *N* é o número de pontos do sinal adquirido e x_n refere-se à amplitude registada na amostra de ordem *n*), ou a partir do espetro de frequências, utilizado a equação (14) (em que *K* é o número de linhas do espetro de frequências e X_k refere-se à amplitude registada na frequência de ordem *k*).

$$X_{RMS} = \sqrt{\frac{\sum_{n=0}^{N-1} {x_n}^2}{N}} \qquad (13)$$

$$X_{RMS} = \sqrt{\frac{\sum_{k=0}^{K-1} X_k{}^2}{2}} \qquad (14)$$

3.1.4 Valores Limite

Uma vez que a quantificação da vibração é feita surge a necessidade de avaliar os dados obtidos. No entanto, só pelos valores obtidos, é difícil retirar conclusões acerca da condição do equipamento. Para isso, e por forma a realizar a deteção do dano, existem valores limite pré-definidos que servirão de referência para retirar conclusões acerca da vibração medida (Figura 16 e Figura 17). É comum existir dois valores limite de alarme: valor de alerta, valor esse que quando ultrapassado indica ao operador que a máquina começa a apresentar indícios de dano e, por isso, deve iniciar-se o diagnóstico para determinar a causa desse aumento de vibração, ou valor de perigo, que indica que a máquina apresenta indícios de que o dano está já numa fase avançada sendo aconselhado a paragem imediata do equipamento para evitar a avaria catastrófica.



,

Figura 16 – Exemplo de valores limite aplicados a um sinal no tempo (Sampaio, 2017)

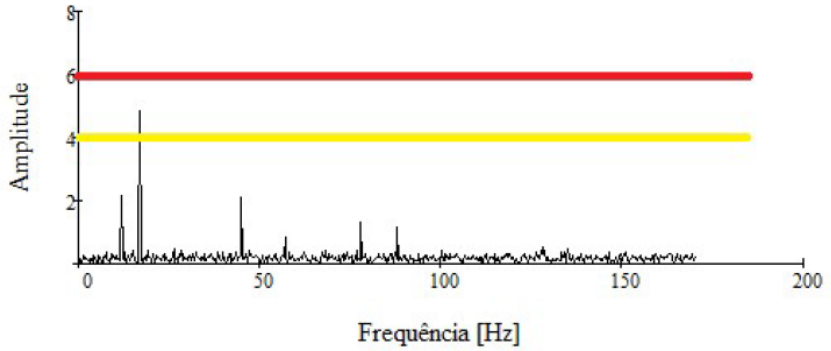


Figura 17 – Exemplo de valores limite aplicados a um sinal em frequência (Sampaio, 2017)

Estes valores de referência, idealmente, deveriam ser fornecidos pelo fabricante. No entanto, nem sempre isso acontece, pelo que normalmente são tomados como referência valores limite definidos por normas e regras de sociedades classificadoras.

Muitas vezes, a deteção de um dano não tem de passar pelo alarme de alerta ou perigo. A presença de dano pode ser detetada através de uma análise de tendência dos valores medidos ao longo do tempo, normalmente recorrendo a uma regressão linear, podendo mesmo serem tomadas ações antes de qualquer valor limite ser atingido (Figura 18).

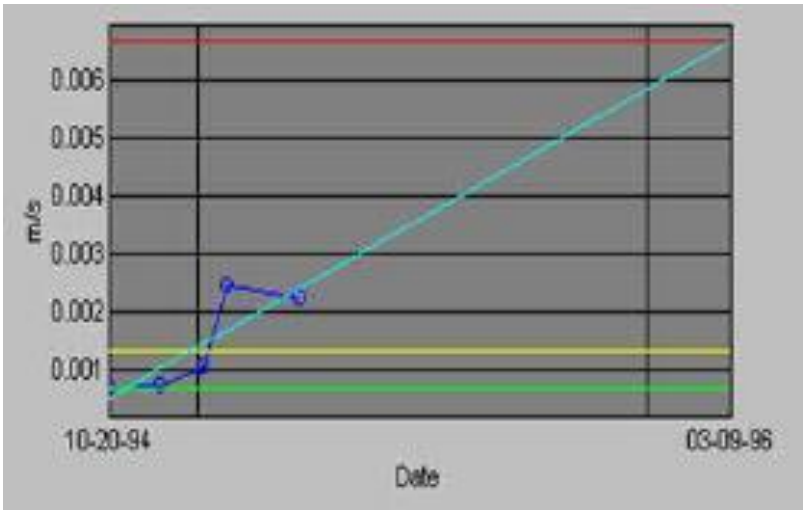


Figura 18 – Análise de tendência de vibração num equipamento (Sampaio, 2017).

3.2 Normas ISO e regras de Sociedades Classificadoras

Numa sociedade global em que para cada área de exploração de recursos, produção de bens e prestação de serviços conta com cada vez mais indústrias e empresas, surgiu a necessidade de padronizar procedimentos, condições de trabalho e até características de produtos ou serviços desenvolvidos e comercializados. Surgem assim normativos e regras de sociedades classificadoras com o intuito de uniformizar certos aspetos e procedimentos.

3.2.1 International Organization for Standardization (ISO)

A ISO é uma organização internacional não governamental, da qual fazem parte 164 organismos nacionais de normalização, sendo o Instituto Português da Qualidade membro integrante desta organização (International Organization for Standardization, s.d.). A ISO é assim responsável pela elaboração e publicação de Normas Internacionais.

As normas são publicações que dão especificações para produtos, serviços e sistemas, de forma a garantir a qualidade, segurança e rendimento (International Organization for Standardization, s.d.). Atualmente a ISO já publicou mais de 22 000 normas internacionais, abrangendo grande parte dos setores da indústria e serviços, como tecnologia, agricultura e saúde (International Organization for Standardization, s.d.)

As normas surgem assim como uma espécie de linguagem universal, em resposta a uma questão: "Qual é a melhor maneira de fazer isto?" (International Organization for Standardization, s.d.). Assim, podemos dizer que as normas são como regras básicas que as várias organizações devem cumprir para que o que quer que elas produzam, desde bens a serviços, seja seguro para o consumidor. Para além disso estas regras ditam as condições ideais para que o produtor possa trabalhar nas melhores condições possíveis e para que o rendimento da produção seja o melhor possível, nunca descurando a segurança do trabalhador e a qualidade do produto final (International Organization for Standardization, s.d.).

3.2.2 International Association of Classification Societies (IACS)

A IACS é uma organização associativa de sociedades classificadores que estabelece padrões e requisitos mínimos para a segurança marítima e sustentabilidade ambiental, garantindo ainda o seu cumprimento, sendo reconhecida como assessor técnico da International Maritime Organization (IMO) (International Association of Classification Societies, 2019).

Uma Sociedade Classificadora é uma organização que publica as suas regras de classificação e requisitos técnicos no que diz respeito ao projeto, construção e inspeção de navios, tendo ainda a capacidade de aplicar manter e melhorar essas regras autonomamente (International Association of Classification Societies, 2011). Verifica também, durante a construção, o cumprimento das regras de classificação bem como durante o resto do seu período de vida e publica regularmente um registo de todos os navios classificados (International Association of Classification Societies, 2011).

O objetivo de uma sociedade classificadora é assim fornecer serviços e assistência na classificação à indústria marítima e entidades regulamentares no âmbito da segurança no mar e prevenção da poluição, tendo por base conhecimento de técnicos especializados (International Association of Classification Societies, 2011).

O objetivo de classificação de um navio é verificar e garantir a integridade e resistência estrutural do casco, bem como a fiabilidade e correto funcionamento de sistemas de propulsão e governo, produção de energia e outros sistemas auxiliares e essenciais ao bom funcionamento do navio (International Association of Classification Societies, 2011). Este objetivo é alcançado através do desenvolvimento de regras das sociedades classificadoras e concordância entre a sociedade classificadora e entidades reguladoras nacionais/internacionais (International Association of Classification Societies, 2011).

Como referido anteriormente, a IACS é constituída por sociedades classificadoras de todo o mundo, contando com a presença nomeadamente da American Bureau of Shiping (ABS), Bureau Veritas (BV), Lloyd's Register (LR), Royal Institute of Naval Architects (RINA), e a fusão de sociedades, Det Norske Veritas com Germanischer Lloyd (DNV-GL).

3.2.3 Normas ISO *vs* Regras de Sociedades Classificadoras

De acordo com o que foi referido até agora podemos dizer que a finalidade das normas internacionais e das regras das sociedades classificadoras é o de uniformizar procedimentos, para que um produto final seja seguro, fiável e rentável. A grande diferença acaba por residir no âmbito de trabalho de ambas as partes.

Enquanto a ISO elabora normas que abrangem uma grande variedade de áreas da indústria e serviços, não se especificando em nenhuma, as sociedades classificadoras focam-se na indústria naval. No entanto, tem-se verificado um alargar de horizontes no campo de aplicação das regras das sociedades classificadoras, como é o caso da DNV-GL que tem explorado outras áreas como indústria automóvel, segurança alimentar e cuidados de saúde (Det Norske Veritas - Germanischer Lloyd, 2019).

3.2.4 Norma ISO 10816

No âmbito do trabalho desenvolvido, foi necessário escolher normas e regras de sociedades classificadoras, de forma a selecionar valores limites a aplicar na funcionalidade de deteção da aplicação desenvolvida. Foram assim selecionadas a norma ISO 10816, uma vez que é a norma atualmente aplicada pelo GAV e é aplicada a máquinas industriais, e a regra DNV-GL RU-SHIPS Pt.6 Ch.8.

Nesta secção será explorado com mais detalhe a norma ISO 10816 como o seu campo de aplicação, requisitos de medição, etc.

No que diz respeito à medição de vibrações em máquinas a ISO desenvolveu duas normas: a ISO 7919 que se refere à medição de vibrações em máquinas rotativas, com base em medições da vibração dos veios com recurso a sensores de proximidade, o que não se adequaria aos objetivos da presente dissertação uma vez que estes devem estar permanentemente montado, visto que requerem calibração no local (International Organization for Standardization, 1996); e a ISO 10816 que se aplica a medição de vibrações mecânicas em partes não rotativas, o que implica o uso de sensores de contacto (International Organization for Standardization, 1995). Tendo em conta os objetivos da aplicação desenvolvida, a norma estudada foi a ISO 10816.

Esta norma é composta por nove partes (International Organization for Standardization, 2014):

- Parte 1 é referente às linhas gerais da norma;
- Parte 2, referente a turbinas a vapor e geradores instaladas em terra que excedam os 50 MW de potência;
- Parte 3, referente a máquinas industriais com potência nominal acima do 15 kW;
- Parte 4, referente a sistemas de turbinas a gás;
- Parte 5, referente a sistemas geradores de energia hidráulicos e sistemas de bombagem;
- Parte 6, referente a máquinas alternativas com potência acima do 10 kW;
- Parte 7, referente a bombas rotativas industriais;
- Parte 8, referente a sistemas de compressores alternativos;
- Parte 21, referente a eixos horizontais de turbinas eólicas com caixa redutora.

Tendo em conta a aplicabilidade da aplicação desenvolvida apenas serão abordadas as partes 1, 3, 4, 6, 7 e 8.

3.2.4.1 ISO 10816-1

Como referido anteriormente, esta parte da norma ISO 10816 diz respeito às suas linhas gerais. Esta parte estabelece assim condições gerais e procedimentos para a medição da vibração em partes das máquinas que não sejam rotativas e, quando aplicável, não alternativas (International Organization for Standardization, 1995).

Esta parte da norma define certos aspetos básicos da medição, como por exemplo a largura de banda de frequências que o equipamento de medição deve conseguir detetar (deve ser grande o suficiente para conseguir cobrir todas as frequências a que uma máquina costuma vibrar, normalmente localizado num intervalo de 10 a 1000 Hz), quantificação da vibração (unidades de medida para medição da vibração em aceleração, velocidade e deslocamento, sendo as mesmas que as abordadas na secção 3.1.3, Quantificação

da Vibração) e aspetos gerais da posição dos sensores quando se realiza uma medição (International Organization for Standardization, 1995).

No que diz respeito às condições de medição de vibrações, relativamente à operação da máquina, a norma indica que estas medições devem ser realizadas quando a máquina atingiu as condições normais de funcionamento. Relativamente às vibrações envolventes, a norma refere que, quando a vibração ultrapassa o limite recomendado, deve-se fazer uma medição de vibrações com a máquina parada a fim de medir as vibrações do meio envolvente, sendo que estas não devem ultrapassar um terço do valor limite recomendado para a máquina em análise (International Organization for Standardization, 1995).

Relativamente à instrumentação utilizada para a medição e análise de vibrações, a norma ISO 1016-1 refere que o equipamento deve estar preparado para realizar medições nas condições do meio envolvente, para que as mesmas não afetem a qualidade dos dados recolhidos. A norma refere igualmente que o sensor deve estar fixo à máquina corretamente e não deve influenciar o comportamento vibratório da máquina. A norma refere ainda que o equipamento deve ter capacidade para medir vibrações na gama de frequências em que a máquina pode vibrar, deve conseguir medir a vibração em RMS, e ser capaz de apresentar a vibração em aceleração, velocidade e deslocamento, quando o critério de avaliação de vibração estabelecido para a máquina em análise assim o exige (International Organization for Standardization, 1995).

A norma ISO 10816-1 considera dois critérios de avaliação: um critério que avalia a amplitude da vibração em termos absolutos e um critério que avalia a alteração da amplitude da vibração (International Organization for Standardization, 1995).

O primeiro critério mede a amplitude da vibração e enquadra-a em zonas/categorias, de acordo com os limites definidos pela norma para a máquina em análise. Deste modo este critério assume 4 zonas (International Organization for Standardization, 1995):

- Zona A: nesta zona enquadra-se a vibração característica de máquinas em início de vida;
- Zona B: a amplitude da vibração é maior do que na zona A, no entanto, considera-se aceitável para um longo período de funcionamento, sem restrições;

- Zona C: a amplitude da vibração é considerada indesejável para um longo período de funcionamento. A máquina pode ser operada, devendo fazê-lo num período de tempo limitado, até que surja a oportunidade de realizar uma manutenção;
- Zona D: uma máquina que apresente estes níveis, apresenta um elevado risco de avaria, aconselhando-se a sua paragem imediata e manutenção a fim de repor o correto funcionamento da mesma.

Apesar de a norma ISO 10816 definir valores limite para as zonas acima referidas, a ISO, salienta que estes valores não são mandatários, sendo que aconselha que estes valores sejam definidos pelo fabricante do equipamento. No entanto, caso tal não seja possível a ISO garante que os valores fornecidos cobrem grande parte das deficiências de operação que possam haver (International Organization for Standardization, 1995).

O segundo critério avalia alterações na amplitude da vibração medida, relativamente a um valor anteriormente medido e tido como referência. A norma assume que aumentos ou diminuições repentinas da amplitude da vibração devem ser investigadas, mesmo que, segundo o primeiro critério, a vibração não se enquadre na zona C ou D (International Organization for Standardization, 1995). Podemos assim dizer que o segundo critério da norma faz uma análise de tendência da amplitude da vibração.

Para a avaliação da vibração pelo segundo critério, é importante que as medições comparadas sejam referentes à mesma posição e direção de medição do sensor (International Organization for Standardization, 1995).

A ISO 10816 considera dois limites operacionais de vibração: o limite de Alerta (*ALARM*) e o de Perigo (TRIP), sendo que o primeiro indica que a vibração ou atingiu uma amplitude insatisfatória ou que ocorreu um aumento ou diminuição brusca da amplitude da vibração relativamente a um valor anteriormente ocorrido, aconselhando-se uma ação preventiva, não implicando a paragem imediata da máquina; o segundo indica que a vibração atingiu valores que podem significar uma avaria iminente, pelo que deve-se realizar a paragem imediata do equipamento (mas controlada), e tomar as ações necessárias

(International Organization for Standardization, 1995). Os valores normalmente praticados para estes limites devem ser discutidos com o fabricante, mas caso não seja possível aconselha-se que para o valor de Alerta se utilize um valor que tenha sido registado durante o período inicial de vida do equipamento ao qual se soma 25% do valor que separa a zona B da zona C (primeiro critério de avaliação da vibração); para o valor de Perigo a norma aconselha o valor de amplitude que representa a fronteira entre a zona C e a zona D (International Organization for Standardization, 1995).

Consideremos um exemplo: um motor elétrico, no seu primeiro mês de funcionamento, apresentou uma vibração de 5 *mm/s* RMS. Considerando que o valor que delimita a zona B da zona C é de 8 *mm/s*, o valor de Alerta será os 7 *mm/s* RMS. Considerando que valor separa a zona C da zona D é 10 *mm/s*, o valor de Perigo será os 10 mm/s.

Para efeitos de desenvolvimento da aplicação projetada, apenas será possível avaliar a vibração das máquinas segundo o primeiro critério, uma vez que, de forma a tornar a aplicação de simples utilização, não prevê a inserção do valor de vibração normal da máquina.

Esta parte da ISO 10816 indica, em anexo, valores gerais para grupos de máquinas específicos, dividindo os equipamentos em quatro classes (International Organization for Standardization, 1995):

- Classe I: componentes de máquinas e motores, essenciais ao normal funcionamento do sistema na sua integra (exemplo: motores elétricos até 15 kW, turbocompressores);
- Classe II: Máquinas de tamanho médio sem apoios especiais (exemplo: motores elétricos com potência entre 15 a 75 kW), motores de combustão interna rigidamente montados, máquina com potência até 300 kW, montadas em apoios especiais;
- Classe III: Motores principais de grandes dimensões e outras máquinas de massas rotativais montados em apoios rígidos;

- Classe IV: Motores principais de grandes dimensões e outras máquinas de massas rotativais montados em apoios flexíveis (exemplo: conjuntos de turbogeradores, turbinas a gás com potência superior a 10 MW).

No entanto, como foi referido, são apenas valores gerais, sendo que, em partes adicionais, fornece valores específicos para tipos de máquinas específicos.

3.2.4.2 ISO 10816-3

Esta parte da ISO 10816 fornece os critérios de avaliação da vibração em máquinas com potência superior a 15 kW e que operem entre 120 RPM e 15 000 RPM. São assim abrangidas por esta parte (International Organization for Standardization, 2009):

- Turbinas a vapor com potência até 50 MW;
- Turbinas a vapor com potência superior a 50 MW e que não estejam incluídas na parte dois da norma;
- Compressores rotativos;
- Turbinas a gás com potência até 3 MW;
- Geradores;
- Motores elétricos;
- Ventiladores (acima de 300 kW);

Esta parte não abrange as seguintes máquinas (International Organization for Standardization, 2009):

- Turbinas a vapor que sejam abordadas na parte 2 da norma;
- Turbinas a gás com potência acima dos 3 MW (parte 4 da norma);
- Conjuntos de máquinas em estações de produção e energia hidráulica e em sistemas de bombagem;
- Máquinas acopladas a máquinas alternativas;
- Bombas rotativas bem como motores elétricos associados se estes se encontrarem rigidamente acoplados à bomba (parte 7 da norma);
- Compressores rotativos de deslocamento positivo (compressores de parafuso);
- Compressores alternativos;

- Bombas alternativas;
- Motobombas submersíveis;
- Turbinas eólicas.

Relativamente ao equipamento de medição das vibrações, esta parte da norma não faz nenhuma consideração adicional relativamente às da parte 1. Em termos de posição do sensor durante a medição apenas refere que este deve ser fixo em um sítio acessível e aconselha a medição em 3 direções ortogonais e que sejam efetuadas principalmente nas chumaceiras das máquinas (International Organization for Standardization, 2009).

No que diz respeito às condições de funcionamento, em nada se altera ao referido na parte 1 da norma, exceto o limite da vibração envolvente que é um quarto do valor limite recomendado para a vibração da máquina (International Organization for Standardization, 2009).

Esta parte da norma agrupa as máquinas por ela abrangidas, consoante o tipo de máquina, potência ou altura do veio, em dois grupos (International Organization for Standardization, 2009):

- Grupo 1: máquinas de dimensões acrescidas com potência acima dos 300 kW e motores elétricos com altura de veio acima dos 315 *mm*;
- Grupo 2: máquinas de dimensão médias com potência entre os 15 e os 300 kW e motores elétricos com altura de veio entre 160 e 315 *mm*.

Para além destes grupos, a máquina em análise é avaliada consoante o tipo de apoios em que considera apoio rígido sempre que a primeira frequência natural do sitema global é 25% (ou mais) superior à frequência de excitação principal, normalmente a frequência de rotação. Todos as máquinas que não cumpram com estes requisitos, consideram-se os seus apoios como flexíveis (International Organization for Standardization, 2009).

Relativamente aos critérios de avaliação da vibração, esta parte em nada difere da parte 1 da norma, bem como os limites operacionais para a vibração (Alerta e Perigo) (International Organization for Standardization, 2009).

Esta parte da ISO 10816 indica, em anexo, os valores a usar nas avaliações segundo o primeiro critério.

3.2.4.3 ISO 10816-4

Esta parte da norma ISO 10816 aplica-se a turbinas a gás utilizadas em aplicações elétricas ou mecânicas, com potência superior a 3 MW e uma velocidade de rotação entre os 3 000 e os 30 000 RPM, aplicando-se igualmente a equipamentos acoplados a uma turbina a gás, exceto (International Organization for Standardization, 2009):

- Bombas acopladas;
- Turbinas a vapor;
- Geradores;
- Compressores;
- Caixas redutoras.

Os requisitos para o equipamento de medição e pontos de medição aconselhados são os mesmos que os referidos na parte 1 da norma (International Organization for Standardization, 2009).

Relativamente a critérios de avaliação, esta parte da norma assume os dois critérios já anteriormente abordados bem como os limites operacionais para a vibração (Alerta e Perigo) (International Organization for Standardization, 2009).

Esta parte da ISO 10816 indica, em anexo, os valores a usar nas avaliações segundo o primeiro critério.

3.2.4.4 ISO 10816-6

Esta parte da norma aplica-se a máquinas alternativas com potência acima dos 100 kW, como por exemplo motores diesel marítimos, quer para propulsão quer para outros fins como produção de energia, e ainda compressores de gás (International Organization for Standardization, 1995). No entanto, como teremos oportunidade de ver mais à frente, a ISO publicou uma parte para esta norma (parte 8) específica para compressores alternativos. Deste modo considera-se esta parte da norma apenas aplicável a motores alternativos de combustão interna.

Na descrição dos requisitos para os equipamentos de medição, esta parte da norma apenas altera, relativamente à primeira parte, a frequência mínima requisitada que passa a ser 2 Hz, ao invés de 10 Hz como referido na parte 1 da norma (International Organization for Standardization, 1995).

No que diz respeito à posição dos sensores na medição da vibração, nada acrescenta relativamente às regras gerais mencionadas na parte 1 da norma. Em relação às condições de funcionamento da máquina em análise, a norma refere que as medições apenas devem ser realizadas assim que a máquina atingir os parâmetros normais de funcionamento (International Organization for Standardization, 1995).

Esta parte da norma, ao contrário do que verificámos nas partes anteriores apenas se refere a um critério dos que já abordámos: a avaliação segundo a amplitude da vibração, não falando no critério da avaliação segundo alterações na amplitude da vibração (International Organization for Standardization, 1995).

Para aplicar este critério a parte 6 da ISO 10816, divide em 7 grupos as máquinas abrangidas por esta parte, sendo que a divisão se baseia na potência e dimensão da máquina, sendo o grupo 1 referente a máquinas de baixa potência e dimensões reduzidas e o grupo 7 para máquinas de elevada potência e dimensões acrescidas. Tipicamente podemos considerar que motores marítimos abrangidos por esta parte da norma estejam inseridos no grupo 4, 5 ou 6 (International Organization for Standardization, 1995).

Relativamente aos valores para avaliação da vibração, estes encontram-se num dos anexos da parte 6 da norma, sendo que a vibração é avaliada segundo a aceleração, velocidade e deslocamento, tudo em valores RMS, na banda de frequência de 2 a 1000 Hz.

3.2.4.5 ISO 10816-7

Esta parte é referente a bombas rotativas com potência superior a 1 kW. Caso as bombas estejam rigidamente conectadas a motores elétricos, os critérios de avaliação referidos nesta parte aplicam-se igualmente aos motores elétricos. Esta parte da norma não se aplica a bombas de deslocamento positivo, quer alternativas quer rotativas, bombas movidas por motores alternativos, bombas em estações hidroelétricas ou em sistemas de bombagem

com potência superior a 1 MW (parte 5 da norma) e a bombas submersíveis (International Organization for Standardization, 2009).

Esta parte da norma não acrescenta nada relativamente à parte 1 no que diz respeito às características do equipamento de medição de vibração, à posição dos sensores para a medição de vibrações, aos critérios de avaliação das vibrações medidas e aos limites operacionais (Alerta e Perigo).

Para aplicar os critérios de avaliação, esta parte da norma divide as bombas por ela abrangidas em duas categorias, de acordo com o tipo de fluido de trabalho, sendo a categoria I para bombas que exigem elevados níveis de fiabilidade, uma vez que são utilizadas para bombear substâncias perigosas, e a categoria II para bombas cujo fluido de trabalho não representa nenhum perigo (International Organization for Standardization, 2009).

Para cada categoria, a parte 7 apresenta valores diferentes para avaliação da vibração segundo a sua amplitude, sendo que dentro das categorias os valores variam consoante a potência da bomba (se a potência é superior ou inferior a 200 kW) (International Organization for Standardization, 2009). Adicionalmente esta parte da norma, avalia as bombas cuja velocidade de rotação é inferior a 600 RPM, sendo que para além dos valores acima mencionados, avalia segundo valores de deslocamento (pico a pico), para a frequências específicas de 0.5, 1 e 2 vezes a frequência de rotação da bomba (International Organization for Standardization, 2009).

3.2.4.6 ISO 10816-8

Esta parte da ISO abrange os compressores alternativos, sendo que se preocupa mais com a fadiga que possa ocorrer, não só no compressor em si (armação e base), mas também com a canalização a ele conectada e outro equipamento auxiliar (International Organization for Standardization, 2014). Comparativamente à parte 6 da norma que também abrange compressores alternativos, esta parte é mais pormenorizada, uma vez que para cada parte da máquina, irá fornecer valores limite diferentes.

Assim, aplicamos esta parte da norma a compressores alternativos, independentemente da configuração do mesmo (em linha, em V, em W, etc.), a compressores de velocidade constante ou variável, compressores movidos por qualquer tipo de fonte motora (motor

elétrico, motor diesel, turbinas a gás ou vapor, etc.) e a compressores de cárter seco ou húmido (International Organization for Standardization, 2014).

Relativamente às características do equipamento de medição de vibrações, esta parte da norma aconselha um equipamento que consiga medir numa banda de frequências dos 2 aos 1 000 Hz, e que consiga medir a vibração em aceleração, velocidade e deslocamento, todos em valor RMS (International Organization for Standardization, 2014).

Comparativamente a outras partes da norma, a parte 8 é mais exaustiva no que diz respeito à localização dos sensores para a medição da vibração, assumindo os seguintes pontos de medição (International Organization for Standardization, 2014):

- Base do compressor;
- Armação do compressor;
- Apoios do compressor;
- Cilindros (topo e lateral);
- Canalização.

As medições em compressores devem ser efetuadas quando o compressor atingir os parâmetros normais de funcionamento (International Organization for Standardization, 2014).

Esta parte da norma apenas utiliza o critério I mencionado na parte 1, isto é, apenas avalia a amplitude da vibração medida, não avaliando as alterações em amplitude.

Em relação aos valores limite para a vibração do compressor, a parte 8 da norma dá valores para compressores verticais e para compressores horizontais, sendo que, se o compressor apresentar uma geometria em V ou em W, os valores limite a utilizar são os do compressor vertical. Os valores estabelecidos dividem-se em aceleração, velocidade e deslocamento RMS, sendo que para a aceleração mede-se numa gama de frequências de 200 a 1 000 Hz, para a velocidade numa gama de frequências de 10 a 200 Hz e para o deslocamento numa gama de 2 a 10 Hz. Os valores definidos variam consoante o ponto em que o sensor estiver a medir (International Organization for Standardization, 2014).

No que diz respeito à medição de vibrações nas canalizações principais a norma afirma que se o valor que delimita a zona C da zona D da amplitude da vibração for ultrapassado, tal não significa que um problema existe, uma vez que, normalmente, as situações de fadiga acontecem em canalizações de baixo calibre e equipamentos a elas ligados (pressostatos e termostatos). Assim, esta parte da norma apenas considera preocupante se o valor de 45 mm/s de velocidade RMS for ultrapassado, se se registar vibrações acrescidas em canalizações de menor calibre ou se os valores de deslocamento estiverem abaixo do valor que delimita a zona C da zona D (considerando que o valor da velocidade ultrapassou este limite) (International Organization for Standardization, 2014).

3.2.5 DNV-GL RU-SHIPS Pt.6 Ch.8

A Sociedade Classificadora DNV-GL desenvolveu regras para navios mercantes. Estas regras abordam várias características de um navio, desde o casco, o sistema de propulsão, o conforto da guarnição e, a que vamos abordar, as vibrações em máquinas ou em estruturas. Quando um navio está de acordo com o que a sociedade classificadora exige para estas características específicas, no certificado emitido pela sociedade classificadora aparecerá as *Class Notations* (abreviaturas/siglas especificas para cada área especifica do navio) que indicam quais a características aprovadas (Jassal, 2017).

A DNV-GL RU-SHIPS Pt.6 Ch.8 aborda as especificações para as condições de trabalho e de habitabilidade, abordando três *Class Notations*: CONF (conforto), VIBR (vibração) e SAFEFLASH (condições de trabalho em segurança em operações de vigilância a contentores) (Det Norske Veritas - Germanischer Lloyd, 2015).

A classe da vibração surge assim para fornecer aos operadores linhas de base para avaliar as condições das máquinas e estruturas, através da medição de vibrações, de forma a conseguirem detetar a presença de dano/anomalia que possa por em causa a integridade do equipamento ou da estrutura (Det Norske Veritas - Germanischer Lloyd, 2015).

Uma vez que a aplicação foi concebida com o objetivo de efetuar controlo de condição em máquinas, abordaremos apenas a aplicabilidade desta classe nas mesmas (Det Norske Veritas - Germanischer Lloyd, 2015).

O critério de avaliação da vibração baseia-se na interpretação do valor obtido após a medição, comparando o mesmo com valores estabelecidos pela sociedade classificadora, sendo que avalia a vibração em apenas dois graus: Bom ou Mau (Det Norske Veritas - Germanischer Lloyd, 2015).

Os valores estabelecidos nesta classe são específicos para cada tipo de máquina ou sistema, abordando linhas de veios, motores diesel, geradores, turbocompressores, bomba, compressores, etc. Relativamente ao equipamento de medição de vibrações, este deve apresentar os valores medidos em RMS e ter capacidade de apresentar a amplitude da vibração em aceleração, velocidade e deslocamento, dependendo dos requisitos previstos para a máquina em análise. A sociedade classificadora recomenda ainda que a medição seja feita durante pelo menos 30 segundos para que a leitura seja mais estável (Det Norske Veritas - Germanischer Lloyd, 2015).

No que diz respeito às condições de funcionamento das máquinas durante a medição, a regra dita que as máquinas devem funcionar a regimes de 100% ou, pelo menos 85 % (Det Norske Veritas - Germanischer Lloyd, 2015).

3.2.6 Seleção de valores limite e comparação da norma ISO 10816 com a regra DNV-GL RU-SHIPS Pt.6 Ch.8

Após o estudo da norma ISO 10816 e da regra DNV-GL RU- SHIPS Pt.6 Ch.8, procedeu-se à seleção dos valores limite para cada máquina suscetível de ser analisada. A seleção destes valores focou-se mais na norma ISO uma vez que a regra da DNV-GL já descrimina os valores para cada máquina. Deste modo, procedeu-se ao contacto com o GAV a fim de perceber quais eram os valores que estes utilizavam da norma ISO para cada máquina que analisam, a fim de haver concordância entre a aplicação desenvolvida e o equipamento por eles utilizados.

Por fim, procedeu-se a uma avaliação comparativa da norma ISO 10816 e da regra DNV-GL RU-SHIPS Pt.6 Ch.8, estando a mesma explícita na tabela 3.

Tabela 1 – Quadro resumo de vantagens e desvantagens da norma ISO 10816 e da regra DNV-GL RU-SHIPS Pt.6 Ch.8

Norma	Vantagens	Desvantagens
ISO 10816	Mais detalhada nos valores limites admissíveis, possuindo 4 níveis de satisfação, e uma avaliação mais completa no que diz respeito à quantificação da vibração.	Norma concebida tendo como objetivo as máquinas industriais, não considerando as condições mais extremas em que as máquinas nos navios podem estar sujeitas.
DNV-GL RU-SHIPS Pt.6 Ch.8	Apesar de ser aplicada a navios no geral, apresenta já uma maior especificação comparando com a norma ISO, uma vez que apresenta valores limites para máquinas em navios e não para máquinas industriais no geral.	Menos detalhado nos valores limite, possuindo apenas dois níveis de satisfação, e a maior parte das máquinas são apenas avaliadas na velocidade RMS da vibração.

4 Sistema de Medição e Análise de vibrações para sistema operativo Android

4.1 Motivação

Apesar de a Marinha Portuguesa já ter implementado um programa de controlo de condição tendo por base a medição e análise de vibrações (VibControlo), existem ainda algumas falhas na aplicabilidade do mesmo. Tendo em conta que:

- As medições de vibrações devem tendencialmente ser realizadas mensalmente;
- O planeamento das unidades navais nem sempre é certo, havendo alterações inopinadas, e quando é certo existe pouca disponibilidade por parte das unidades navais, por motivos de empenhamento em outros trabalhos de prioridade superior;
- A Direção de Navios, mais especificamente, o GAV, apenas tem adquiridos três equipamentos para medições de vibrações, sendo que apenas dois deles estão destinados para as guarnições efetuarem medições a bordo;
- Os navios apenas podem realizar estas análises quando estão atracados na Base Naval de Lisboa, porque está fora de questão o empréstimo do analisador durante missões a navegar uma vez que impediria o empréstimo a outras unidades navais, pelo que se torna inviável a medição e análise de vibrações a certos equipamentos, com é o caso dos da instalação propulsora (motores propulsores e chumaceiras dos veios).

A aquisição de mais equipamentos de medição de vibrações por parte da Marinha seria assim uma solução direta para o colmatar destas falhas. No entanto, como foi já constatado, os equipamentos até agora adquiridos são produtos de preço avultado, pelo que, de acordo com as limitações financeiras da Marinha, torna-se incomportável a aquisição de um sistema de medição e análise de vibrações para cada navio.

Estes fatores motivaram o desenvolvimento de uma aplicação para *smartphone* com sistema operativo *Android*, pelas seguintes razões:

- Graças ao avanço tecnológico, o *smartphone* tornou-se uma ferramenta poderosíssima, com uma capacidade de processamento bastante elevada, mesmo comparada com computadores pessoais;
- O *smartphone* acabou por se tornar num objeto "indispensável" na vida do cidadão, pelo que poderemos sempre garantir que uma grande percentagem da guarnição de um navio seja detentora de um desses dispositivos;
- Ao tornar acessível a todos os militares, torna-se possível realizar medições em qualquer altura, quer seja atracado ou a navegar, ou seja, muitos equipamentos que só podem ser monitorados a navegar (ex.: motores propulsores, caixas redutoras, chumaceiras de veio) poderão ser alvos de maior controlo;
- Um navio poderá ter mais do que um sistema de medição acessível a qualquer elemento da guarnição responsável pela manutenção das máquinas, o que potencia que no futuro a medição da vibração de uma máquina seja um hábito tão comum como medir a sua temperatura, ouvir o seu ruído ou sentir a sua vibração por apalpação.

Estes fatores aliados à existência de sensores no mercado que permitam a aquisição de dados de vibração, e que sejam compatíveis com os sistemas *Android* e ao mesmo tempo de baixo custo (valores abaixo dos 100 euros), tornam assim possível que as guarnições dos navios tenham uma forma de realizar análises de vibrações nos equipamentos de bordo.

É de salientar que com o baixo custo advém uma diminuição de capacidades destes medidores de vibrações. No entanto, será sempre possível detetar certos problemas que costumam ser frequentes em máquinas, sendo que se aconselha um estudo mais aprofundado, recorrendo a equipamentos profissionais e mais fiáveis, para que conclusões possam ser tiradas.

Concluindo, com este trabalho, será possível equipar todos os navios com uma solução satisfatoriamente fiável para medição e análise de vibrações, sendo esta de baixo custo e

representando um meio de primeira ação para a deteção e até de diagnóstico de dano em máquinas.

4.2 Configuração do Hardware

Como referido anteriormente, o sistema de medição e análise de vibrações desenvolvido trata-se de uma aplicação desenvolvida para telemóveis do tipo *smartphone* com sistema operativo *Android*.

Em temos de configurações de hardware, a aplicação desenvolvida é destinada a telemóveis com a versão mínima de sistema *Android* 5.0 também conhecida como a versão *Lollipop*. Relativamente à capacidade de processamento, uma vez que a aplicação foi concebida por forma a conseguir realizar cálculos como o algoritmo da FFT, a partir de vetores com, no máximo, 8192 elementos, aconselha-se um processador de média capacidade (por exemplo, processadores de 4 núcleos com média de 1.5 GHz por núcleo ou processadores de dois núcleos com média superior a 2 GHz por núcleo) no entanto, um processador de média-alta capacidade de processamento é mais aconselhado (processador de 4 núcleos com média de 2 GHz por núcleo). Relativamente à memória RAM, e de acordo com os últimos equipamentos de média gama lançados, os valores mais usados são os de 2 e 3 Gb, suficientes para apoiar o processador do telemóvel no que diz respeito ao processamento de dados.

Relativamente ao sensor que possibilitará a recolha de dados para a medição de vibrações, utilizou-se um acelerómetro MEMS desenvolvido pela *Phidgets*, empresa que desenvolve sensores e também outros equipamentos eletrónicos como placas de aquisição e SBC's (*Single Board Computer*), a um preço relativamente baixo. O que motivou a escolha de sensores desta empresa foi:

- Preço acessível, o que respeita o compromisso de uma solução de baixo custo de um medidor e analisador de vibrações;

- Os sensores têm já placa de aquisição incluída, sendo desnecessário a aquisição de uma placa de aquisição à parte, como normalmente acontece quando se adquire sensores de baixo custo, mas que não estão integrados numa placa de aquisição;
- A *Phidgets Inc.* fornece bibliotecas para várias linguagens de programação, desde *Python*, *LabView* e, neste caso, *Java*, bibliotecas essas que já têm incluídas as funções necessárias para que o utilizador consiga retirar dados do sensor e posteriormente analisá-los ou simplesmente visualizá-los.

Uma vez que o sensor faz-se acompanhar de um cabo USB, em que na extremidade que liga ao sensor é do tipo Mini USB A e na extremidade que irá ligar ao equipamento onde os dados serão visualizados/tratados é do tipo USB A, torna-se necessário a aquisição de um cabo USB OTG (*On-The-Go*), devendo este ser compatível com a entrada USB do telemóvel.

Estes são os elementos de hardware necessários para que o *software* (aplicação para *smartphone*) possa ser aplicado. De seguida será explicado em pormenor a aplicação desenvolvida desde o seu projeto ao produto final.

4.3 Aplicação

Nesta secção será descrita a aplicação desenvolvida no âmbito da estrutura e funcionalidades da mesma. No entanto, e por forma a abordar especificamente todo o processo de desenvolvimento, será feita uma análise evolutiva, ou seja, será explicado todo o processo de criação definido pelo desenvolvimento individual e progressivo de funções, que no final, interligadas e estruturadas, darão origem à aplicação final.

4.3.1 Android Studio

O *Android Studio* é uma ferramenta gratuita desenvolvida pela *Google* que veio possibilitar à comunidade em geral, um meio de desenvolver capacidades na área da programação e também criatividade (Figura 19).

Figura 19 – Logotipo do *software Android Studio.*

Trata-se assim de um compilador que transforma código de programação em linguagem que o telemóvel entenda e transforme numa aplicação. O *Android Studio* possibilita ao utilizador programar em uma destas três linguagens de código: *Java*, *Kotlin* e C++, tendo-se optado pelo *Java*, uma vez que se trata de uma linguagem normalmente utilizada no *Android Studio*, sendo que o apoio ao programador é mais completo em fóruns *online*.

O *Android Studio* possibilita ao programador uma abordagem à aplicação em duas vertentes: quanto à programação base da aplicação e quanto à organização da interface com o utilizador.

4.3.2 Projeto inicial

Antes de começar a desenvolver a aplicação, foi necessário primeiro definir a estrutura da aplicação, bem como as funcionalidades que esta deveria apresentar, sendo que esta deveria ser de fácil compreensão para o utilizador.

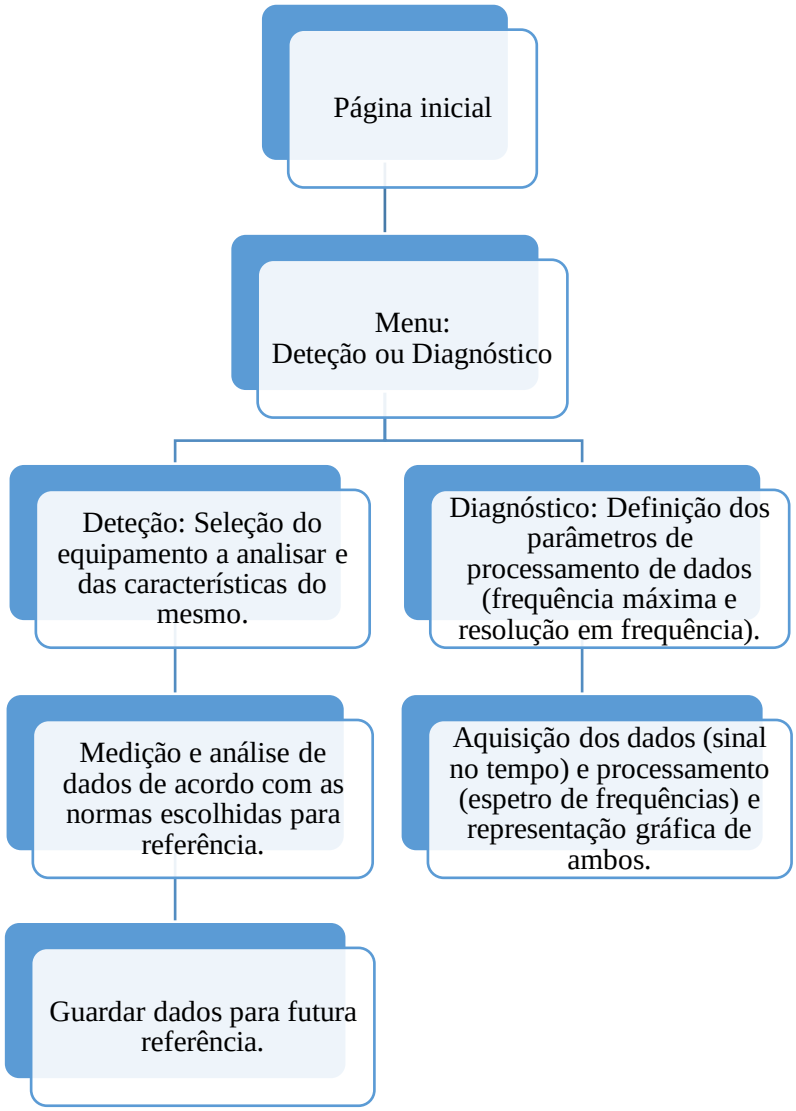


Figura 20 – Esquema explicativo da organização da aplicação desenvolvida.

Para pôr em prática este projeto foi necessário realizar certos programas secundários por forma a testar a viabilidade de recursos a utilizar no projeto final. Esses testes serão abordados de seguida, culminando com o produto final, a aplicação desenvolvida.

4.3.3 Aquisição de dados

Como referido anteriormente, o sensor utilizado foi um acelerómetro da marca Phidgets, sendo o modelo *PhidgetSpatial* 0/0/3 1041_0B.

Relativamente às características técnicas, este trata-se de um sensor MEMS, tri-axial, com um período de amostragem mínimo de 1 *ms* e máximo de 1 *s*. Apresenta um intervalo dinâmico na ordem dos ±8*g*.

Figura 21 – Visão geral do sensor usado (Phidgets Inc., 2016)

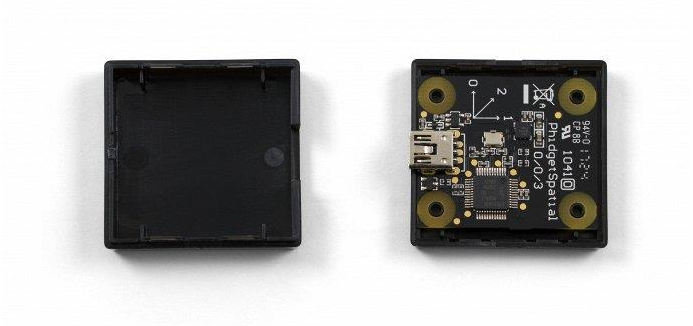


Figura 22 – Visão do hardware do sensor usado (Phidgets Inc., 2016)

A *Phidgets*, como já referido, disponibiliza bibliotecas de funções em linguagem *Java*, o que irá possibilitar ao *smartphone* adquirir os dados enviados pelo acelerómetro.

Para além das bibliotecas de funções, a *Phidgets* disponibiliza uma aplicação exemplo (*Accelerometer Example*), o que facilita ao utilizador a compreensão da estrutura do código que permitirá a aquisição dos dados do sensor.

Com os dados adquiridos, resta armazená-los num vetor, para que possam ser processados seguidamente.

4.3.4 FFT

Este foi um dos grandes desafios da programação da aplicação desenvolvida. A linguagem Java é uma linguagem pouco especializada em programação matemática, sendo mais direcionada para interface com o utilizador.

De facto, não existe nenhuma função desenvolvida pela Java para aplicar a FFT. Apesar de existirem já bibliotecas que possibilitam a implementação do algoritmo da FFT em linguagem Java, houve dificuldades em perceber como implementar essas bibliotecas. Deste modo decidiu-se programar o algoritmo para a FFT. O algoritmo programado pode ser consultado em Apêndice (Apêndice B).

Por forma a testar o algoritmo, programou-se um vetor de uma função harmónica, com amplitude e frequência conhecida, para validar o algoritmo.

4.3.5 Representação gráfica dos dados

Para a representação gráfica dos dados obtidos, tanto pelo sensor, como os dados calculados pela aplicação da FFT aos dados recolhidos, foi conseguida graças à utilização de uma biblioteca existente para programação em *Android Studio*, disponível na plataforma *GitHub*, o *MPAndroidChart*. Esta biblioteca tem vários tipos de gráficos que o programador poderá usar. Os utlizados para a aplicação desenvolvida foram o gráfico de linhas para representação do sinal no tempo, e o gráfico de barras para representação do espetro de frequências obtido através do processamento do sinal no tempo.

Para teste desta biblioteca realizou-se uma representação de uma função harmónica num gráfico de linhas e, aproveitando o cálculo da FFT da função harmónica representou-se os valores obtidos em gráfico de barras. O código e interface referente a esta aplicação teste pode ser encontrado em Apêndice (Apêndice C).

Por fim, realizou-se uma aplicação teste por forma a testar a representação gráfica dos dados obtidos pelo acelerómetro bem como os dados obtidos através do cálculo da FFT

dos dados do sensor. Para validar os resultados, fez-se a medição da vibração de um motor com frequência de rotação conhecida e com uma massa de desequilíbrio, aumentando a amplitude da vibração, sendo de esperar um espetro de frequência com uma componente bem definida na frequência de rotação e com uma amplitude significativa.

4.3.6 Registo dos dados obtidos em ficheiro .txt

Como referido anteriormente, era intenção que a aplicação tivesse a capacidade de gravar os dados recolhidos na deteção. Para isso explorou-se na página *Android Developers* funções que possibilitassem o registo de texto ao critério do utilizador, sendo que esse registo deveria estar disponível para consulta na memória do telemóvel. Após investigação, foi desenvolvida uma aplicação teste, cujo objetivo era gravar uma palavra ou frase inserida pelo utilizador sempre que este pressionasse a tecla "Guardar". O resultado final deveria ser um ficheiro *.txt*, em que cada linha iria corresponder ao texto inserido pelo utilizador em cada registo efetuado. O código e interface poderá ser consultado em Apêndice (Apêndice D).

4.3.7 Projeto Final

Nesta secção será abordada a estrutura final da aplicação bem como todos os processos presentes na mesma. Pode ser analisado em apêndice os códigos de programação associados à aplicação desenvolvida (Apêndice E a L)

No fluxograma da Figura 23 encontram-se representados os processos desde o início do programa até ao processo de aquisição de dados do sensor.

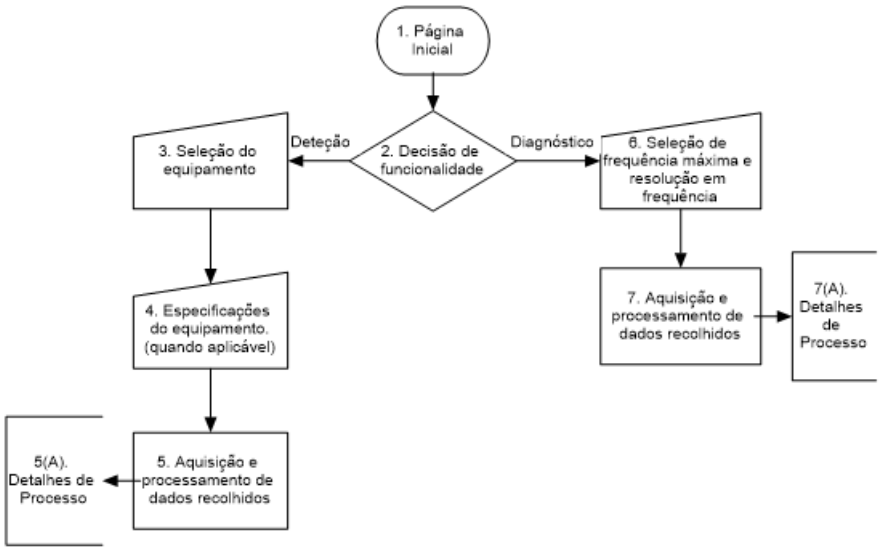


Figura 23 – Fluxograma 1, referente aos processos desde o início do programa até ao processo de aquisição de dados do sensor.

O programa inicia com uma página inicial, onde se encontra nome e logotipo da *app*, uma breve explicação do objetivo da aplicação e os autores da mesma (ponto 1 do fluxograma 1). A aplicação prossegue assim que o utilizador premir a tecla “Iniciar” (Figura 24).

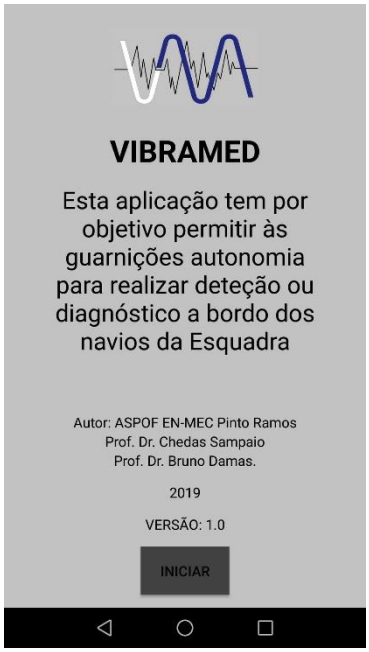


Figura 24 – Página inicial da aplicação.

De seguida o utilizador terá que decidir se quer fazer uma deteção ou um diagnóstico (Figura 25) (ponto 2 fluxograma 1).

Figura 25 – Página de decisão entre deteção e diagnóstico (ponto 2 fluxograma 1).

Se o utilizador decidir que pretende fazer uma deteção, terá que de seguida escolher o equipamento que pretende ter por base na análise de vibrações (Figura 26 e Figura 27)(exemplo: se pretender analisar o compressor de um grupo de Ar Condicionado, deverá selecionar a opção “Compressor Alternativo”) (ponto 3 do fluxograma 1).

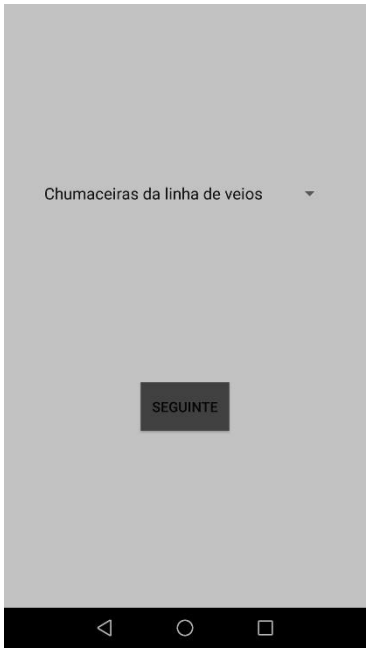


Figura 26 – Página de seleção do equipamento a analisar na deteção (ponto 3 fluxograma 1).

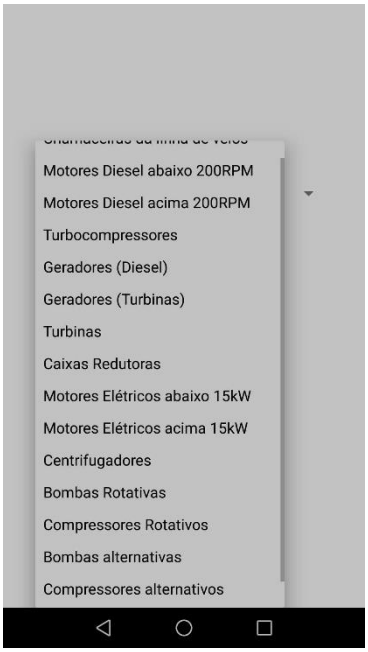


Figura 27 – Lista de grande parte dos equipamentos passíveis de ser analisados.

Após a seleção do equipamento, o utilizador poderá ter que fornecer informações mais detalhadas acerca do equipamento, dependendo do equipamento que pretende analisar (Figura 28) (exemplos: tipos de apoios, rotações do equipamento, potência) (ponto 4 do fluxograma 1).

Figura 28 – Exemplo de página onde o utilizador define informações mais detalhadas (exemplo referente às bombas rotativas) (ponto 4 fluxograma 1)

A partir de aqui, a aplicação já tem todas as informações que necessita para proceder à deteção. Esta fase (ponto 5 do fluxograma 1) será abordada mais à frente (ponto 5(A) do fluxograma 1).

Se o utilizador decidir que pretende fazer um diagnóstico, terá que escolher qual a frequência máxima do sinal e a resolução em frequência (ponto 6 do fluxograma 1) (Figura 29, Figura 30 e Figura 31). Estas escolhas irão influenciar no período de amostragem do sensor e no número de amostras recolhidas.

Figura 29 - Página de seleção da frequência máxima e da resolução em frequência (ponto 5 fluxograma 1).

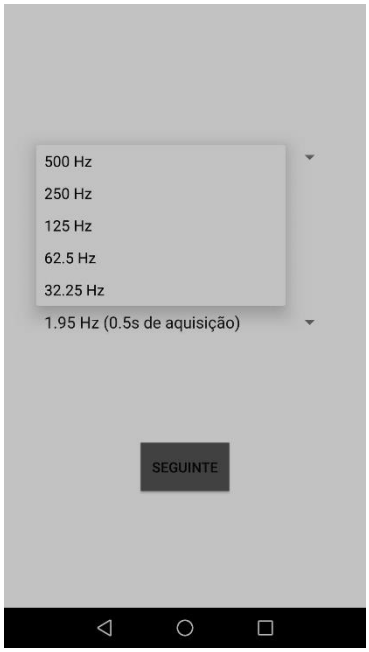


Figura 30 – Lista de frequências máximas do sinal disponíveis ao utilizador.

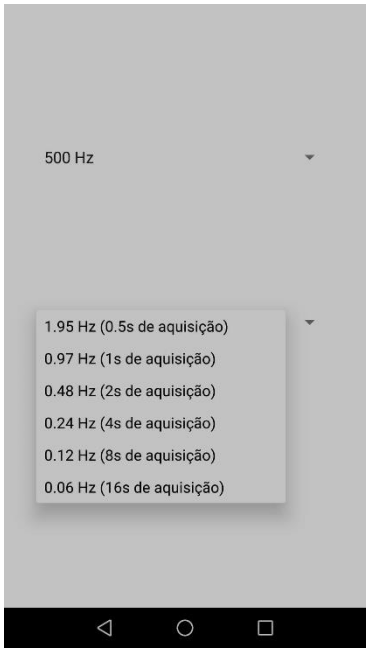


Figura 31 - Lista de resoluções em frequência do sinal disponíveis ao utilizador.

A partir daqui, a aplicação já tem todas as informações que necessita para proceder à aquisição de dados para o utilizador poder fazer diagnóstico. Esta fase (ponto 6 do fluxograma 1) será abordada mais á frente (ponto 6(A) do fluxograma 1).

Agora será abordado um segundo fluxograma (Figura 32), que representa os processos desde o momento em que a aplicação está pronta a fazer uma deteção até ao processo de gravação dos dados da deteção em memória.

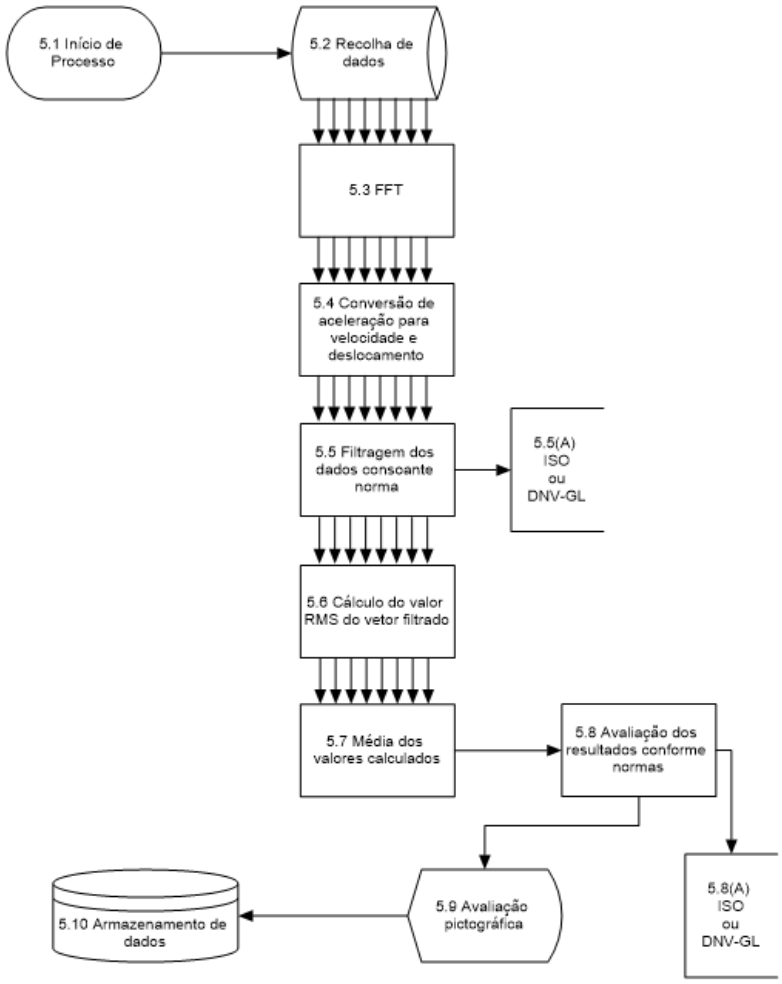


Figura 32 – Fluxograma 2, referente aos processos entre a recolha de dados para a deteção até ao armazenamento de dados.

Depois de todos dados referentes ao equipamento alvo de deteção estarem definidos, o programa está pronto para recolher dados do sensor. Assim que o sensor estiver ligado ao *smartphone* e o utilizador pressionar o botão "Medir", o programa irá começar a recolher dados de aceleração, convertendo-os de *g* para m/s^2, organizando-os num vetor com 32768 pontos. A taxa de recolha de dados do acelerómetro será de 1*ms*/amostra, pelo que o processo de recolha de dados (ponto 5.2 do fluxograma 2) deverá demorar cerca de 32 *s* (Figura 33).

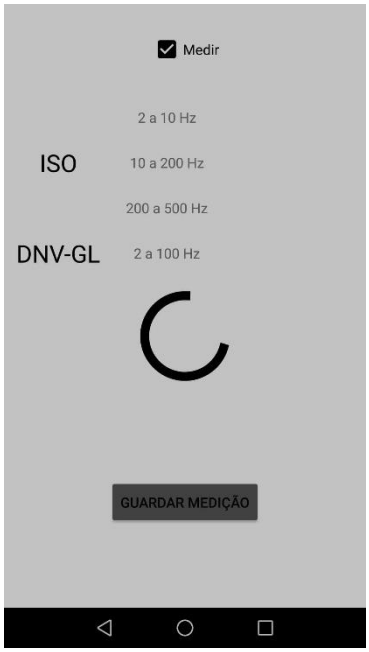


Figura 33 – Aspeto da aplicação, quando esta se encontra a adquirir dados para a deteção.

Assim que a recolha de dados estiver completa, o programa dividirá o vetor original de 32768 pontos em 8 vetores de 4096 pontos. Deste modo, será possível realizar uma média linear de espetros por forma a obter valores mais estáveis.

Após a divisão em 8 vetores, aplica-se o algoritmo da FFT programado a cada um dos vetores (ponto 5.3 do fluxograma 2). Neste passo, o algoritmo da FFT irá receber um vetor de números reais (sinal no tempo da aceleração), e fornecerá um vetor de números complexos (sinal em frequência da aceleração).

A partir daqui o programa fará a conversão dos valores de aceleração para velocidade e deslocamento, recorrendo às fórmulas (9) e (10) anteriormente abordadas (ponto 5.4 do fluxograma 2). Assim teremos vinte e quatro vetores de números complexos, oito de aceleração, oito de velocidade e oito de deslocamento, todos referentes ao sinal em frequência. Agora que a conversão de unidades foi feita, o programa transforma os vetores de números complexos em vetores de números reais, calculando o valor absoluto de cada número complexo..

Como pudemos verificar nas tabelas de compilação dos critérios das normas para os valores de alarme da vibração, dependendo do equipamento, utilizam-se larguras de banda diferentes. Deste modo, o programa irá efetuar uma filtragem em frequência de acordo com o equipamento em análise e com os critérios que as normas ditam. Essa filtragem é feita igualando a zero todos os valores dos espetros de frequência, fora do intervalo de frequências a analisar (ponto 5.5 e 5.5(A) do fluxograma 2).

Assim que os vetores de aceleração, velocidade e deslocamento estiverem filtrados em frequência de acordo com as normas programadas, o programa executa o cálculo do valor RMS, aplicando a equação (8) aos vetores filtrados (ponto 5.6 do fluxograma 2).

Já com o RMS calculado, teremos oito valores referentes a cada grandeza medida, seja aceleração, velocidade ou deslocamento (dependendo dos critérios das normas). O programa executará a média desses oito valores, por forma a obter-se valores mais estáveis (ponto 5.7 do fluxograma 2).

Com as médias calculadas, o programa tem já os valores RMS das grandezas a analisar. De acordo com os valores limite definidos pelas normas, enquadra os valores RMS calculados segundo esses valores limite (ponto 5.8 e 5.8(A) do fluxograma 2). De seguida avaliará os valores seguindo um código de cores (ponto 5.9 do fluxograma 2):

- Para a avaliação segundo a norma ISO 10816, utilizou-se um código de quatro cores: verde escuro para “BOM”, verde claro para “ACEITÁVEL”, amarelo para “INSATISFATÓRIO” e vermelho para “INACEITÁVEL”;
- Para a avaliação segundo a norma DNV-GL RU-SHIPS Pt.6 Ch.8, utilizou-se um código de duas cores: verde escuro para “BOM” e vermelho para “INACEITÁVEL”;

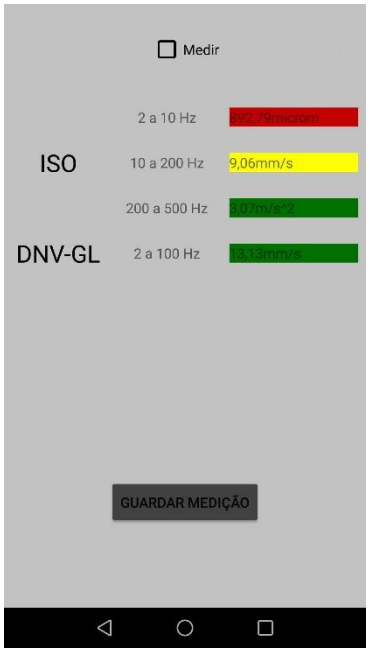


Figura 34 – Exemplo de resultados obtidos numa deteção.

Já com a deteção feita, o utilizador tem a possibilidade de guardar os dados medidos, o que poderá ser útil se quiser fazer uma análise de tendência para cada equipamento analisado. Para isso o utilizador deverá premir o botão "Guardar Medição" (Figura 34).

Ao premir o botão "Guardar Medição", uma nova página aparecerá, onde o utilizador deverá preencher alguns campos que irão identificar o navio, operador, máquina medida, condições de medição e motivos de medição, como podemos verificar na figura 35. Assim que os campos estiverem preenchidos, o utilizador poderá armazenar os dados da deteção na memória do dispositivo (ponto 5.10 do fluxograma 2), premindo o botão "Guardar" (Figura 35).

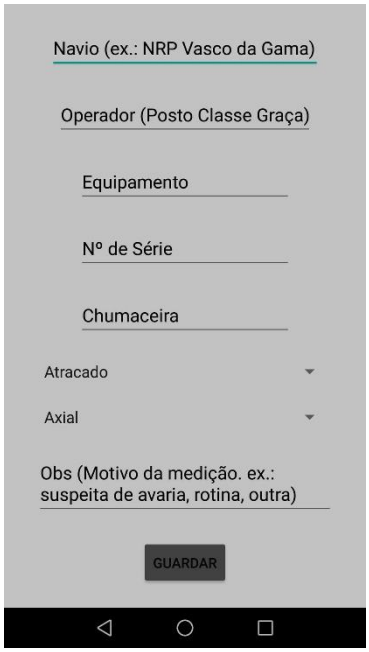


Figura 35 – Campos a preencher pelo utilizador para registo dos resultados de uma deteção.

Assim que o programa tiver informação para guardar os dados, irá gerar um texto, como o da Figura 36, que irá estar disponível num documento de extensão *.txt*, guardado em memória com o nome "Deteção". Será neste documento onde serão guardados todos os dados referentes às deteções realizadas. Deste modo, depois de efetuar a análise de vibrações a um ou mais equipamentos, o utilizador poderá exportar esse documento para um computador e analisar os resultados posteriormente.

Avaliação DNV 1: Bom
Valor DNV 2: 0,29mm/s
Avaliação DNV 2: Bom

****
****

Info Navio
Data/hora: 17/06/2019 22:21:19
Navio: NRP Figueira Da Foz
Operador: ASPOF Pinto Ramos
Situação: A Navegar

Info Medição
Equipamento: Motor Compressor de Ar
Nº de Série: 8986868758
Chumaceira: 2
Direção: Axial
Observações: Rotina

Avaliação
Valor ISO 1: 16,36microm
Avaliação ISO 1: Bom
Valor ISO 2: 1,49mm/s
Avaliação ISO 2: Satisfatório
Valor DNV 1: 627,96microm
Avaliação DNV 1: Mau
Valor DNV 2: 6,57mm/s
Avaliação DNV 2: Mau

****
****

Info Navio

Figura 36 – Exemplo de um registo de uma deteção realizada e resultados obtidos.

Por fim, será abordado um terceiro fluxograma (Figura 37), que representa os processos desde o momento em que a aplicação está pronta a fazer um diagnóstico até ao processo de apresentação do espetro de frequências ao utilizador.

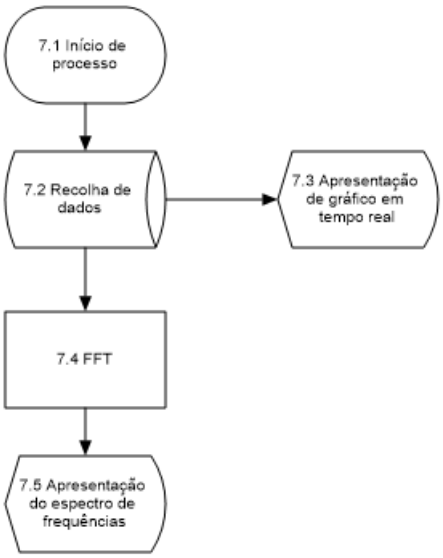


Figura 37 - Fluxograma 3, referente aos processos entre a recolha de dados para o diagnóstico até à apresentação do espetro de frequências ao utilizador.

Após a definição da aquisição de dados do sensor (frequência máxima do sinal e resolução em frequência) por parte do utilizador, o programa está pronto a efetuar a medição.

Assim que o utilizador pressionar o botão medir, a aplicação começa a adquirir dados do sensor, segundo um período de amostragem que irá depender da frequência máxima selecionada. O tempo total de aquisição de dados depende da resolução em frequência selecionada pelo utilizador, uma vez que esta influencia a quantidade de valores que será adquirida. Os dados adquiridos serão armazenados num vetor (ponto 7.2 do fluxograma 3), sendo que sempre que um valor é adicionado ao vetor, esse mesmo valor é representado graficamente, fazendo com que o utilizador tenha uma perspetiva em tempo real da variação da aceleração no sinal no tempo (ponto 7.3 do fluxograma 3).

Com todos os dados do sensor já adquiridos, o programa aplica ao vetor dos dados de aceleração no tempo o algoritmo da FFT (ponto 7.4 do fluxograma 3). Tal como na deteção, o resultado da aplicação da FFT é um vetor de números complexos, sendo que o programa irá transformá-lo num vetor de números reais, calculando o dobro do módulo de cada valor complexo. De seguida o programa apresenta esse vetor num gráfico de barras, que irá representar o espetro de frequências do sinal medido (ponto 7.5 do fluxograma 3).

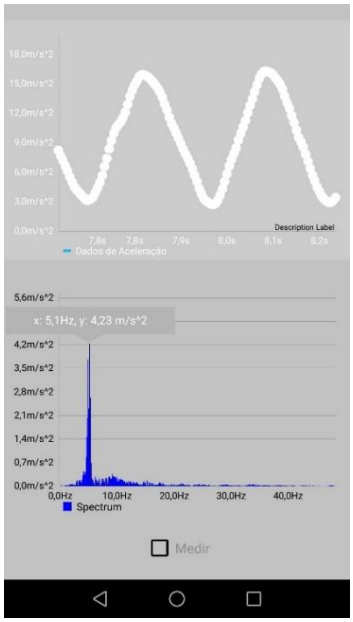


Figura 38 – Exemplo de um diagnóstico realizado com a aplicação, com a representação gráfica do sinal no tempo e do espetro de frequências.

4.3.8 Proposta de Estrutura Protetora do Sensor

Como foi possível verificar nas figuras 21 e 22, o sensor usado já inclui uma caixa protetora do sensor. No entanto esta caixa não se torna adequada, uma vez que apresenta folgas no seu interior, podendo causar erros de medição.

Deste modo foi desenvolvido um protótipo de estrutura protetora, recorrendo ao *software SolidWorks* (Figura 39), semelhante à original, mas que elimina as folgas existentes, uma vez que será fechada recorrendo a quatro parafusos, ao invés de ser somente de encaixe como é o caso da estrutura original. Para além desta alteração o protótipo sugerido inclui um entalhe circular onde será possível colar um íman de neodímio, com grande poder magnético, o que irá reduzir bastante os erros de leitura.

Para além das modificações vantajosas acima apresentadas, esta estrutura pode ser concebida recorrendo à tecnologia de impressão 3D, tecnologia essa já disponível na Marinha, evitando custos acrescidos (Figura 40).

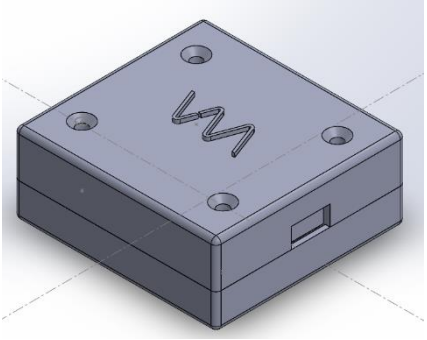


Figura 39 – Projeto desenvolvido em *SolidWorks* para o protótipo de estrutura protetora.

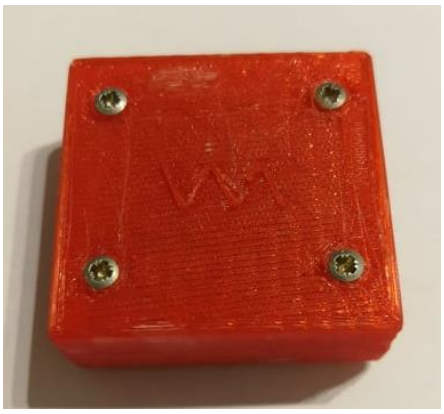


Figura 40 – Protótipo da estrutura protetora produzido recorrendo à tecnologia de impressão 3D.

4.4 Testes e Resultados

Os testes à aplicação são uma etapa essencial para validar a viabilidade da utilização da mesma nos navios da Marinha Portuguesa.

Uma maneira direta de realizar esses testes é a de comparar valores medidos com equipamentos profissionais. Uma vez que a Marinha possui equipamentos profissionais de medição e análise de vibrações, conseguimos garantir termo de comparação. No entanto, existem testes que devem ser feitos antes de testar a aplicação final.

Neste caso, todas as aplicações de teste elaboradas, como é o caso da aplicação de teste ao algoritmo da FFT anteriormente referida e da aplicação de teste da representação gráfica tanto de função harmónica e respetivo espetro de frequências recorrendo à aplicação da FFT, foram essenciais para uma validação progressiva de todos os processos desenvolvidos para a aplicação.

Para além disso, foi testada a aplicação medindo a vibração num motor elétrico (Figura 41) com massa de desequilíbrio e frequência de rotação conhecida, por forma a verificar se a aplicação conseguia, através do seu espetro de frequências detetar o pico à frequência de rotação, sintoma esse característico de uma máquina rotativa com desequilíbrio.

Figura 41 – Motor elétrico com massa de desequilíbrio, alvo de medição para testes à aplicação.

Após os testes intermédios à aplicação, procedeu-se aos testes finais (Figura 42), em que se comparou os valores medidos, durante uma deteção, com os valores medidos por um equipamento do GAV. É importante realçar o facto de os testes terem sido feitos apenas comparando os resultados obtidos pela análise segundo a norma ISO 10816, uma vez que o GAV apenas faz as deteções segundo os critérios da ISO.

No entanto, estes testes não foram conclusivos pelas seguintes razões:

- A agenda do GAV é bastante preenchida, tornando-se difícil a realização de testes, comparando a aplicação desenvolvida com o equipamento deles. Apesar disso ainda foram realizados alguns testes, recorrendo à análise de vibrações

de um motor elétrico de um grupo de Ar Condicionado, da classe Viana do Castelo;

- Com a comparação dos resultados obtidos, notaram-se certas semelhanças nos valores obtidos por ambos os equipamentos de medição e análise. No entanto, devido a dúvidas na parametrização da norma de ambas as partes, tornou-se impossível retirar conclusões sólidas acerca dos resultados.

Figura 42 – Realização de testes à aplicação, comparando o equipamento do GAV com a aplicação desenvolvida.

Concluindo, testes futuros são necessários para validar definitivamente a viabilidade do uso da aplicação a bordo das unidades navais, sendo que as perspetivas são bastante animadoras.

5 Conclusão

Esta dissertação focou-se na conceção de uma solução de baixo custo para medir e analisar vibrações a bordo dos navios de guerra da Marinha Portuguesa. Esta solução tem por objetivo fazer frente às dificuldades que as guarnições têm atualmente em realizar a análise de vibrações em máquinas a bordo dos navios da Marinha, sendo que essas dificuldades advêm da falta de equipamentos, o custo inerente à aquisição de mais equipamentos e da necessidade que os navios têm de possuir um meio próprio de medição e análise de vibrações sem terem de depender de outras unidades em terra, que apenas podem facultar esses meios quando o navio está atracado na base.

Deste modo projetou-se uma aplicação para *smartphones* com sistema operativo *Android* que utiliza como sensor para aquisição de dados de vibração um acelerómetro MEMS da marca *Phidgets,* que apresenta um custo reduzido características satisfatórias para o objetivo previsto para o medidor de vibrações proposto.

Esta aplicação foi desenvolvida por forma a dar ao utilizador a hipótese de realizar deteção diagnóstico. Ao fazer deteção recorrendo à aplicação desenvolvida, o utilizador irá comparar os níveis de vibração da máquina a analisar com os valores limite estabelecidos pela norma ISO 10816 e pela DNV-GL RU-SHIPS Pt.6 Ch.8, sendo que o utilizador terá a possibilidade de fazer deteção a vários tipos de máquinas, uma vez que a aplicação foi programada de forma a aplicar diferentes valores de alarme, dependendo da máquina selecionada pelo operador e dos valores estabelecidos pelas normas. Para além disso, caso o utilizador pretenda guardar os resultados da deteção para futura referência ou caso pretenda fazer uma análise de tendência, a funcionalidade também está disponível.

Ao fazer diagnóstico, o utilizador tem a possibilidade de definir os parâmetros da aquisição de dados, decidindo qual a frequência máxima que pretende observar no sinal e a resolução em frequência, podendo de seguida analisar em pormenor o sinal no tempo e o respetivo espetro de frequências da vibração medida, graças à representação gráfica dos mesmos.

Durante a conceção da aplicação, foram realizados vários testes de forma a validar vários processos desenvolvidos. Estes testes revelaram-se importante para a correção de erros de programação que poderiam influenciar negativamente os resultados que fossem obtidos quando a aplicação fosse utilizada para fazer deteção ou diagnóstico.

Para além destes testes intermédios, fizeram-se alguns testes finais à aplicação, comparando os valores obtidos pela aplicação numa deteção com os valores obtidos por um equipamento profissional do GAV. No entanto, os mesmos foram insuficientes para se considerar os resultados obtidos como conclusivos, mesmo estes tendo sido favoráveis. De facto, o elevado emprego operacional da equipa do GAV tornou-se um entrave para o encontro de ambas as partes a fim de realizar testes à aplicação.

Concluindo, a aplicação desenvolvida, poderá revelar-se um meio importantíssimo para os navios conseguirem implementar um bom programa de manutenção preventiva recorrendo ao controlo de condição, uma vez que o navio começará a ter mais autonomia no que diz respeito à medição e análise de vibrações, não dependendo exclusivamente do GAV e dos seus equipamentos para o fazer. Inerente a esta maior autonomia, certos equipamentos, como é o caso dos da instalação propulsora, poderão começar a ser alvo de controlo de condição, uma vez que a guarnição do navio começará a ter autonomia para o fazer quando estiver a navegar. Para além disso, a deteção, que era feita tendo por base apenas os valores limite estabelecidos pela norma ISO 10816, começará a ser feita também pelos limites definidos pela sociedade classificadora DNV-GL enunciados na norma DNV-GL RU-SHIPS Pt.6 Ch.8. É importante realçar que esta solução para a medição de vibrações não invalida o apoio especializado do GAV, recorrendo a equipamentos profissionais. Revela-se sim um meio de primeira ação para a deteção e diagnóstico de dano em máquinas quando o acesso a equipamentos profissionais não é possível ou é limitado.

5.1 Lições aprendidas

Este projeto revelou-se bastante proveitoso para formação do signatário, uma vez que fomentou o desenvolvimento de competências de programação, principalmente pela ne-

cessidade que existiu em aprender uma linguagem nova, tendo em conta as bases adquiridas no percurso académico na Escola Naval. Adicionalmente, permitiu a abertura de novas portas para o desenvolvimento de futuros projetos que envolvam o uso da ferramenta versátil que é o *smartphone* e que possibilitem à Marinha Portuguesa um maior desenvolvimento a nível tecnológico. Esta dissertação permitiu igualmente a consolidação de conhecimentos adquiridos ao longo de cinco anos de formação académica na Escola Naval.

5.2 Projetos futuros

Apesar de todo o trabalho desenvolvido, considera-se que existem certos aspetos que podem ser melhorados, tendo por base as necessidades da Marinha. Apresenta-se assim uma série de sugestões de projetos futuros:

- Alargar a aplicabilidade desta *app* a *smartphones* com sistema operativo IOS, da marca *Apple*, tornando possível a todos os militares da Marinha o acesso à aplicação, independentemente do sistema operativo que usem;
- Integrar na aplicação os critérios de conforto e ruído e vibrações no corpo humano, de forma a poder avaliar as condições de habitabilidade e de trabalho a bordo dos navios da Marinha Portuguesa, tendo por base os valores de referência estabelecidos pela ISO e Sociedades Classificadoras;
- Integrar na aplicação valores limite para as vibrações de máquinas que tenham sido estabelecidos por outras Sociedades Classificadoras.
- Integrar na *app* outros critérios de deteção, como por exemplo, a variação relativa a um valor de referência, indicador de máquina saudável.
- Integrar na aplicação recomendações de procedimentos de medição, significados do código de cores implementado na deteção e outras informações úteis.

6 Bibliografia

Archambault, R. (2003). The Past, Present and Future of Vibration Analysis. Em B. d. International Measurement Solutions (Ed.), *Conference of the Canadian Machinery Vibration Association (CMVA).* Halifax, Nova Scotia, Canada. Obtido em 22 de Maio de 2019, de http://environmentaltestanddesign.com/past-present-future-vibration-analysis/

Armada, Almirante Chefe de Estado-Maior da. (3 de Maio de 2016). Despacho do Almirante Chefe de Estado-Maior da Armada n.º 38/2016. *Regulamento Interno da Direção de Navios*.

Cooley, J. W., Lewis, P. A., & Welch, P. D. (1967). Historical Notes on the Fast Fourier Transform. *IEEE Trans. Audio Electroacoust* (pp. 76-79). Yorktown Heghts, New York: IBM Research Center.

Det Norske Veritas - Germanischer Lloyd. (Outubro de 2015). Rules for classification: Ships. *Part 6: Aditional Class Notations. Chapter 8 Living and Working Conditions*.

Det Norske Veritas - Germanischer Lloyd. (2019). *About DNV-GL*. Obtido em 30 de Julho de 2019, de DNV-GL: https://www.dnvgl.com/about/index.html

Digiducer. (2019). *333D01 USB Digital Accelerometer*. Obtido em 19 de Junho de 2019, de Digiducer, Digital-Ready USB Accelerometre: https://digiducer.com/

Direção de Navios. (2015). Documento Iniciador de Projeto. *Projeto VibControlo*.

Estado-Maior da Armada. (7 de Fevereiro de 1997). Instruções para a Organização da Manutenção das Unidades Navais e Outros Meios de Acção Naval. *ILA 5 (A)*.

Instituto Português da Qualidade. (Setembro de 2007). NP EN 13306. *Terminologia da Manutenção*. Caparica, Portugal.

International Association of Classification Societies. (Junho de 2011). *Classification Socities - What, Why and How*. Londres, Reino Unido: IACS.

International Association of Classification Societies. (2019). *About IACS*. (IACS, Editor) Obtido em 2019 de Julho de 2019, de IACS: http://www.iacs.org.uk/about/

International Organization for Standardization. (1995). ISO 10816-1. *Mechanical Vibration - Evaluation of Machine Vibration by Measurements on Non-Rotating Parts. Part 1: General Guidelines*.

International Organization for Standardization. (15 de Dezembro de 1995). ISO 10816-6. *Mechanical Vibration - Evaluation of Machine Vibration by Measurements on Non-Rotating Parts. Part 6: Rciprocatinhg Machines with power ratings above 100 KW*.

International Organization for Standardization. (15 de Julho de 1996). ISO 7919-1. *Mechanical vibration of non reciprocating machines - Measurements on rotating shafts and evaluation criteria. Part 1: General Guidelines*.

International Organization for Standardization. (1 de Fevereiro de 2009). ISO 10816-3. *Mechanical Vibration - Evaluation of Machine Vibration by Measurements on Non-Rotating Parts. Part 3: Insdustrial Machines with Nominal Power above 15 kW and Nominal Speeds between 120 r/min and 15 000 r/min when measured in situ*.

International Organization for Standardization. (1 de Outubro de 2009). ISO 10816-4. *Mechanical Vibration - Evaluation of Machine Vibration by Measurements on Non-Rotating Parts. Part 4: Gas Turbine sets with fluid-film bearings*.

International Organization for Standardization. (1 de Fevereiro de 2009). ISO 10816-7. *Mechanical Vibration - Evaluation of Machine Vibration by Measurements on Non-Rotating Parts. Part 7: Rotodynamic pumps for industrial applications, including measurements on rotating shafts*.

International Organization for Standardization. (1 de Julho de 2014). ISO 10816-8. *Mechanical Vibration - Evaluation of Machine Vibration by Measurements on Non-Rotating Parts. Part 8: Reciprocating compressor systems*.

International Organization for Standardization. (s.d.). *About Us*. Obtido em 29 de Julho de 2019, de International Organization for Standardization: https://www.iso.org/about-us.html

International Organization for Standardization. (s.d.). *Benefits of ISO standards*. Obtido em 31 de Julho de 2019, de International Organization for Standardization: https://www.iso.org/benefits-of-standards.html

Itnoovate. (10 de Junho de 2019). *Powerful, Affordable Vibration Monitoring and Analysis*. Obtido de Vib Cloud: http://www.vib.cloud/

Jassal, R. (6 de Junho de 2017). *5 Classification Society Terms You Need to Understand Now*. Obtido em 31 de Julho de 2019, de My Sea Time: https://www.myseatime.com/blog/detail/classification-society-terms

Marinha Portuguesa. (2019). *A Missão*. Obtido em 30 de Julho de 2019, de Marinha: https://www.marinha.pt/pt/a-marinha/Paginas/missao.aspx

Mundinstal - Técnicas de Instalações Elétricas, Lda. (2014). *Serviços de Termografia*. Obtido em 29 de Julho de 2019, de Mundinstal: http://www.mundinstal.pt/termografia.html

Phidgets Inc. (2016). *Phidgets*. Obtido em 20 de Junho de 2019, de PhidgetSpatial 0/0/3 Basic: https://www.phidgets.com/?tier=3&catid=10&pcid=8&prodid=1022

Rao, B. K. (1998). *Handbook of Condition Monitoring: Techniques and Methodology*. Tullamore, Ireland: Chapman & Hall.

Rao, S. S. (2011). *Mechanical Vibrations* (5ª ed.). Upper Saddle River: Prentice Hall.

Sampaio, R. P. (2017). O Papel do Controlo de Vibrações Mecânicas . *Anais do Clube Militar Naval*, 159-213.

SKF. (s.d.). *Indicadores de máquina*. Obtido em 9 de Junho de 2019, de Web Site de SKF Portugal: https://www.skf.com/pt/products/condition-monitoring/basic-condition-monitoring-products/vibration-measurement-tools/static-vibration-sensor/index.html

SKF. (s.d.). *QuickCollect Sensor*. Obtido em 9 de Junho de 2019, de Web Site de SKF Portugal: https://www.skf.com/pt/products/condition-monitoring/basic-condition-monitoring-products/vibration-measurement-tools/quickcollect-sensor/index.html

SKF. (s.d.). *SKF MCA (Machine Condition Advisor)*. Obtido em 9 de Junho de 2019, de Web Site de SKF Portugal: https://www.skf.com/pt/products/condition-monitoring/basic-condition-monitoring-products/vibration-measurement-tools/basic-handheld-vibration-sensor/index.html

VTT Technical Research Centre of Finland. (2006). Prognostics for industrial machinery availability. Final Seminar. *VTT SYMPOSIUM 243* (p. 6). Helsinki: Edita Prima Oy.

Apêndices

Apêndice A – Tabelas dos valores limite selecionados da norma ISO 10816 e da regra DNV-GL RU-SHIPS Pt.6 Ch.8

Tabela 2 – Tabela de valores limites utilizados na programação da aplicação, para avaliação da vibração segundo a norma ISO 10816 organizados por equipamentos existentes a bordo (ISO 10816-1) (ISO 10816-3) (ISO 10816-4) (ISO 10816-6) (ISO 10816-7) (ISO 10816-8)

	ISO 10816		
	Parâmetros	Frequência	RMS
Chumaceiras de linhas de veio	Velocidade (*mm/s*)	10 – 500 Hz	A/B = 1,8 *mm/s* B/C = 4,5 *mm/s* C/D = 11,2 *mm/s*
Motores Diesel < 200 RPM	Delocamento (μm)	2 – 10 Hz	A/B = 44,8 μm B/C = 113 μm C/D = 178 μm
	Velocidade (*mm/s*)	10 – 250 Hz	A/B = 2,82 *mm/s* B/C = 7,07 *mm/s* C/D = 11,2 *mm/s*
	Aceleração (m/s^2)	250 – 500 Hz	A/B = 4,42 m/s^2 B/C = 11,1 m/s^2 C/D = 17,6 m/s^2
Motores Diesel > 200 RPM	Delocamento (μm)	2 – 10 Hz	Propulsão: A/B = 283 μm B/C = 448 μm C/D = 710 μm

			Grupos Eleterogéneos: A/B = 113 μm B/C = 283 μm C/D = 448 μm
	Velocidade (*mm/s*)	10 – 250 Hz	Propulsão: A/B = 17,8 *mm/s* B/C = 28,2 *mm/s* C/D = 44,6 *mm/s*
			Grupos Eleterogéneos: A/B = 7,07 *mm/s* B/C = 17,8 *mm/s* C/D = 28,2 *mm/s*
	Aceleração (m/s^2)	250 – 500 Hz	Propulsão: A/B = 27,9 m/s^2 B/C = 44,2 m/s^2 C/D = 70,1 m/s^2
			Grupos Eleterogéneos: A/B = 11,1 m/s^2 B/C = 27,9 m/s^2 C/D = 44,2 m/s^2
Turbinas a Gás	Velocidade (*mm/s*)	10 – 500 Hz	A/B = 4,5 *mm/s* B/C = 9,3 *mm/s* C/D = 14,7 *mm/s*
Turbo-Geradores	Delocamento (μm)	10 – 500 Hz	Apoios Rígidos: A/B = 29 μm B/C = 57 μm C/D = 90 μm
			Apoios flexíveis:

			A/B = 45 μm B/C = 90 μm C/D = 140 μm
	Velocidade (*mm/s*)	10 – 500 Hz	Apoios rígidos: A/B = 2,3 *mm/s* B/C = 4,5 *mm/s* C/D = 7,1 *mm/s*
			Apoios flexíveis: A/B = 3,5 *mm/s* B/C = 7,1 *mm/s* C/D = 11,0 *mm/s*
Geradores Diesel	Delocamento (μm)	10 – 500 Hz	Apoios Rígidos: A/B = 29 μm B/C = 57 μm C/D = 90 μm
			Apoios flexíveis: A/B = 45 μm B/C = 90 μm C/D = 140 μm
	Velocidade (*mm/s*)	10 – 500 Hz	Apoios rígidos: A/B = 2,3 *mm/s* B/C = 4,5 *mm/s* C/D = 7,1 *mm/s*
			Apoios flexíveis: A/B = 3,5 *mm/s* B/C = 7,1 *mm/s* C/D = 11 *mm/s*
Bombas Alternativas	Velocidade (*mm/s*)	10 – 500 Hz	A/B = 1,12 *mm/s*

			B/C = 2,8 *mm/s* C/D = 7,1 *mm/s*
Bombas rotativas	Velocidade (*mm/s*)	10 – 500 Hz	(Abaixo de 200kW) Categoria I A/B = 2,5 *mm/s* B/C = 4 *mm/s* C/D = 6,6 *mm/s*
			(Acima de 200kW) Categoria I A/B = 3,5 *mm/s* B/C = 5 *mm/s* C/D = 7,6 *mm/s*
			(Abaixo de 200kW) Categoria II A/B = 3,2 *mm/s* B/C = 5,1 *mm/s* C/D = 8,5 *mm/s*
			(Acima de 200kW) Categoria II A/B = 4,2 *mm/s* B/C = 6,1 *mm/s* C/D = 9,5 *mm/s*
	Deslocamento (μm)	0,5× RPM; 1× RPM; 2× RPM	A/B = 50 μm B/C = 80 μm C/D = 130 μm

Compressores Alternativos	Delocamento (μm)	2 – 10 Hz	Base A/B = 32 μm B/C = 48 μm C/D = 72 μm
			Armação A/B = 84 μm B/C = 127 μm C/D = 151 μm
			Cilindro (Radial) A/B = 139 μm B/C = 207 μm C/D = 310 μm
			Cilindro (Axial) A/B = 170 μm B/C = 255 μm C/D = 382 μm
			Apoios e Canalização A/B = 202 μm B/C = 302 μm C/D = 454 μm
	Velocidade (*mm/s*)	10 – 200 Hz	Base A/B = 2 *mm/s* B/C = 3 *mm/s* C/D = 4,5 *mm/s*
			Armação A/B = 5,3 *mm/s* B/C = 8 *mm/s* C/D = 12 *mm/s*

			Cilindro (Radial) A/B = 8,7 *mm/s* B/C = 13 *mm/s* C/D = 19,5 *mm/s*
			Cilindro (Axial) A/B = 10,7 *mm/s* B/C = 16 *mm/s* C/D = 24 *mm/s*
			Apoios e Canalização A/B = 12,7 *mm/s* B/C = 19 *mm/s* C/D = 28.5 *mm/s*
	Aceleração (m/s^2)	200 – 500 Hz	Base A/B = 2,5 m/s^2 B/C = 3,8 m/s^2 C/D = 5,7 m/s^2
			Armação A/B = 6,7 m/s^2 B/C = 10,1 m/s^2 C/D = 15,1 m/s^2
			Cilindro (Radial) A/B = 10,9 m/s^2 B/C = 16,3 m/s^2 C/D = 24,5 m/s^2
			Cilindro (Axial) A/B = 13,5 m/s^2 B/C = 20,1 m/s^2

			C/D = 30,2 m/s^2
			Apoios e Canalização A/B = 16 m/s^2 B/C = 23,9 m/s^2 C/D = 35,8 m/s^2
Compressors rotativos	Delocamento (μm)	10 – 500 Hz	Apoios Rígidos: A/B = 22 μm B/C = 45 μm C/D = 71 μm
			Apoios flexíveis: A/B = 37 μm B/C = 71 μm C/D = 113 μm
	Velocidade (*mm/s*)	10 – 500 Hz	Apoios rígidos: A/B = 1,4 *mm/s* B/C = 2,8 *mm/s* C/D = 4,5 *mm/s*
			Apoios flexíveis: A/B = 2,3 *mm/s* B/C = 4,5 *mm/s* C/D = 7,1 *mm/s*
Motores elétricos < 15 kW	Velocidade (*mm/s*)	10 – 500 Hz	A/B = 0,71 *mm/s* B/C = 1,8 *mm/s* C/D = 4,5 *mm/s*
Motores elétricos > 15 kW	Delocamento (μm)	10 – 500 Hz	(Entre 15 e 300kW) Apoios Rígidos: A/B = 22 μm B/C = 45 μm

			C/D = 71 μm
			(Entre 15 e 300kW) Apoios flexíveis: A/B = 37 μm B/C = 71 μm C/D = 113 μm
			(Entre 300kW e 50 MW) Apoios Rígidos: A/B = 29 μm B/C = 57 μm C/D = 90 μm
			(Entre 300kW e 50 MW) Apoios flexíveis: A/B = 45 μm B/C = 90 μm C/D = 140 μm
	Velocidade (*mm/s*)	10 – 500 Hz	(Entre 15 e 300kW) Apoios rígidos: A/B = 1,4 *mm/s* B/C = 2,8 *mm/s* C/D = 4,5 *mm/s*
			(Entre 15 e 300kW) Apoios flexíveis: A/B = 2,3 *mm/s* B/C = 4,5 *mm/s* C/D = 7,1 *mm/s*

			(Entre 300kW e 50 MW) Apoios rígidos: A/B = 2,3 *mm/s* B/C = 4,5 *mm/s* C/D = 7,1 *mm/s*
			(Entre 300kW e 50 MW) Apoios flexíveis: A/B = 3,5 *mm/s* B/C = 7,1 *mm/s* C/D = 11,0 *mm/s*
Caixas redutoras	Velocidade (*mm/s*)	10 – 500 Hz	A/B = 1,8 *mm/s* B/C = 4,5 *mm/s* C/D = 11,2 *mm/s*
Turbocompressores	Velocidade (*mm/s*)	10 – 500 Hz	A/B = 0,71 *mm/s* B/C = 1,8 *mm/s* C/D = 4,5 *mm/s*
Centrifugadores	Velocidade (*mm/s*)	10 – 500 Hz	Abaixo de 15kW A/B = 0,71 *mm/s* B/C = 1,8 *mm/s* C/D = 4,5 *mm/s*
			Acima de 15kW A/B = 1,12 *mm/s* B/C = 2,8 *mm/s* C/D = 7,1 *mm/s*
Ventoinhas	Velocidade (*mm/s*)	10 – 500 Hz	A/B = 0,71 *mm/s* B/C = 1,8 *mm/s* C/D = 4,5 *mm/s*

Tabela 3 - Tabela de valores limites utilizados na programação da aplicação, para avaliação da vibração segundo a norma DNV-GL RU-SHIP Pt.6 Ch.8 organizados por equipamentos existentes a bordo (Det Norske Veritas - Germanischer Lloyd, 2015)

	DNV-GL RU-SHIPS Pt.6 Ch.8		
	Parâmetros	Frequência	RMS
Chumaceiras de linhas de veio	Velocidade (*mm/s)*	1 – 200 Hz	Perigo 5 *mm/s*
Motores Diesel < 200 RPM	Deslocamento (μm)	1 – 200 Hz	Vertical e Axial Perigo 1000 μm
			Transversal Perigo 1500 μm
	Velocidade (*mm/s)*	1 – 200 Hz	Vertical e Axial Perigo 10 *mm/s*
			Transversal Perigo 25 *mm/s*
Motores Diesel > 200 RPM	Velocidade (*mm/s)*	4 – 200 Hz	Apoios Rigidos Perigo 15 *mm/s*
			Apoios Flexiveis Perigo 25 *mm/s*
Turbinas a Gás	Velocidade (*mm/s)*	4 – 500 Hz	Perigo 7 *mm/s*
Turbo-Geradores	Velocidade (*mm/s)*	4 – 500 Hz	Perigo 7 *mm/s*
Geradores Diesel	Velocidade (*mm/s)*	4 – 200 Hz	Perigo 18 *mm/s*
Bombas Alternativas	Velocidade (*mm/s)*	4 – 500 Hz	Perigo 30 *mm/s*

Bombas rotativas	Velocidade (*mm/s)*	4 – 200 Hz	Perigo 7 *mm/s*
Compressores Alternati-vos	Velocidade (*mm/s)*	4 – 500 Hz	Perigo 30 *mm/s*
Compressors rotativos	Velocidade (*mm/s)*	4 – 200 Hz	Apoios Rigidos Perigo 7 *mm/s*
			Apoios Flexiveis Perigo 10 *mm/s*
Motores elétricos	Velocidade (*mm/s)*	4 – 200 Hz	Perigo 7 *mm/s*
Caixas redutoras	Velocidade (*mm/s)*	4 – 500 Hz	Perigo 7 *mm/s*
Turbocompressores	Velocidade (*mm/s)*	4 – 200 Hz	Abaixo de 5MW(1) Perigo 45 *mm/s*
			Entre 5 a 10MW(1) Perigo 50 *mm/s*
			Acima de 10MW(1) Perigo 55 *mm/s*
	Aceleração (m/s^2)	4 – 200 Hz	Abaixo de 5MW(1) 24,5 m/s^2
			Entre 5 a 10MW(1) 19,6 m/s^2
			Acima de 10MW(1) 14,7 m/s^2
Centrifugadores	Velocidade (*mm/s)*	4 – 200 Hz	Perigo 12 *mm/s*

Ventoinhas	Velocidade (*mm/s)*	4 – 200 Hz	Perigo 12 *mm/s*
(1) Os valores limites definidos para os turbocompressores dependem da potência combinada do grupo de cilindros associada ao turbocompressor sujeito a avaliação.			

Apêndice B – Algoritmo da FFT programado

Este código consiste numa função isolada. Essa função precisa de 4 elementos de entrada:

- *n* que é o número de pontos do vetor real;
- *m* que é uma constante que prova que o número de pontos do vetro real é uma potência de 2;
- *x*[] que é um vetor de números reais (vetor de entrada) com n pontos;
- *xx*[] que é um vetor de números complexos (entra com todos os pontos iguais a 0) com n/2 pontos.

```
public void fft (int n , int m , double[] x , Complex[] xx){
    int k , i , k1 ,k2 , Nf, j , j1 , j2 , e1 , e2 , o1 , o2, n1, nk,
ei, ej, N2;
    double th   , ag , wr , wi ,  dr , di, t, ar, ai, br, bi, s, c,
cr, ci ;


    k2 = 1;
    Nf=n/2;

    for ( k = 1; k<=m ; k++){
        k2 = 2*k2;
        k1=k2/2;
        th=2*(Math.PI/Nf)*k1;
        nk = Nf/k2;
        n1=Nf/k1;
        for ( i = 1 ; i<=nk ; i++){
            ag = (i - 1)*th;
            wr = Math.cos(ag);
            wi = - Math.sin(ag);
            for (j=n1 ; j<=Nf ; j=j + n1){
                j1 = j - n1 + i - 1;
                j2 = j1 + nk;
                e1 = 2*j1;
                e2 = 2*j2;
                o1 = e1 + 1;
                o2 = e2 + 1;
                dr = x[e1] - x[e2];
                di = x[o1] - x[o2];
                x[e1] = x[e1] + x[e2];
                x[o1] = x[o1] + x[o2];
                x[e2] = wr*dr - wi*di;
                x[o2] = wi*dr + wr*di;
            }
        }
    }


    j = 0;
    for (i=0 ; i<=Nf - 2; i++){
```

```
        if (i<j){
            ei = 2*i;
            ej = 2*j;
            t = x[ei];
            x[ei] = x[ej];
            x[ej]=t;
            t= x[ei + 1];
            x[ei + 1]=x[ej + 1];
            x[ej + 1]=t;
        }
        k=Nf/2;
        while (k<=j){
            j=j - k;
            k=k/2;
        }
        j=j+k;
    }


    t = x[0]+x[1];
    x[1]=x[0]-x[1];
    x[0]=t;
    x[Nf + 1]=-x[Nf + 1];
    N2 = 2*Nf;
    for (k=1 ; k<=(Nf/2) - 1; k++){
        k2=2*k;
        nk=N2-k2;
        ar=x[k2]+x[nk];
        ai=x[k2 + 1] - x[nk + 1];
        br=x[nk]-x[k2];
        bi= -x[nk + 1] - x[k2 + 1];
        s = Math.sin(k*(Math.PI/Nf));
        c = Math.cos(k*(Math.PI/Nf));
        cr = br*s-bi*c;
        ci = br*c+bi*s;
        x[k2]=(ar+cr)/2;
        x[k2 + 1]=(ai+ci)/2;
        x[nk]= (ar-cr)/2;
        x[nk + 1]=(ci-ai)/2;

    }

    for (i=0; i<n ; i=i + 2){
        xx[i/2]= new Complex(x[i]/Nf , x[i+1]/Nf);
    }
    xx[0]=new Complex(x[0]/(2*Nf) , x[1]/(2*Nf));
}
```

Apêndice C – Código da aplicação teste referente à representação gráfica de uma função harmonica e respetivo espetro de frequências

```
package com.example.app1.graph_fft_teste;

import android.graphics.Color;
import android.graphics.Typeface;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.components.XAxis;
import com.github.mikephil.charting.components.YAxis;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.github.mikephil.charting.highlight.Highlight;
import com.github.mikephil.charting.interfaces.datasets.ILineDataSet;
import com.github.mikephil.charting.listener.OnChartValueSelectedLis-
tener;

import java.util.Arrays;

public class MainActivity extends AppCompatActivity implements On-
ChartValueSelectedListener {

    private LineChart grafico1;
    private LineChart grafico2;



    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        grafico1 = findViewById(R.id.grafico1);
        grafico2 = findViewById(R.id.grafico2);
        //definições grafico 1
        grafico1.setOnChartValueSelectedListener(this);
        grafico1.getDescription().setEnabled(true);
        grafico1.setTouchEnabled(true);
        grafico1.setDragEnabled(true);
        grafico1.setScaleEnabled(true);
        grafico1.setDrawGridBackground(false);
        grafico1.setPinchZoom(true);
        grafico1.setBackgroundColor(Color.YELLOW);
        LineData pontos1 = new LineData();
        pontos1.setValueTextColor(Color.WHITE);
        grafico1.setData(pontos1);
        //definições grafico 2
        grafico2.setOnChartValueSelectedListener(this);
        grafico2.getDescription().setEnabled(true);
        grafico2.setTouchEnabled(true);
```

```
        grafico2.setDragEnabled(true);
        grafico2.setScaleEnabled(true);
        grafico2.setDrawGridBackground(false);
        grafico2.setPinchZoom(true);
        grafico2.setBackgroundColor(Color.LTGRAY);
        LineData pontos2 = new LineData();
        pontos2.setValueTextColor(Color.WHITE);
        grafico2.setData(pontos2);


        //definições eixos grafico 1
        XAxis x1 = grafico1.getXAxis();
        x1.setPosition(XAxis.XAxisPosition.BOTTOM);
        x1.setAxisMaximum(600f);
        x1.setAxisMinimum(0f);
        x1.setTypeface(Typeface.DEFAULT);
        x1.setTextColor(Color.BLACK);
        x1.setDrawGridLines(false);
        x1.setAvoidFirstLastClipping(true);
        x1.setEnabled(true);

        YAxis y1 = grafico1.getAxisLeft();
        y1.setTypeface(Typeface.DEFAULT);
        y1.setTextColor(Color.BLACK);
        y1.setAxisMaximum(10f);
        y1.setAxisMinimum(-10f);
        y1.setDrawGridLines(false);

        YAxis rightAxis1 = grafico1.getAxisRight();
        rightAxis1.setEnabled(false);



        //definições eixos grafico 2
        XAxis x2 = grafico2.getXAxis();
        x2.setPosition(XAxis.XAxisPosition.BOTTOM);
        x2.setAxisMaximum(600f);
        x2.setAxisMinimum(0f);
        x2.setTypeface(Typeface.DEFAULT);
        x2.setTextColor(Color.WHITE);
        x2.setDrawGridLines(false);
        x2.setAvoidFirstLastClipping(true);
        x2.setEnabled(true);
        x2.setGranularity(0.5f);
        x2.setGranularityEnabled(true);

        YAxis y2 = grafico2.getAxisLeft();
        y2.setTypeface(Typeface.DEFAULT);
        y2.setTextColor(Color.WHITE);
        y2.setAxisMaximum(10f);
        y2.setAxisMinimum(0f);
        y2.setDrawGridLines(false);

        YAxis rightAxis2 = grafico2.getAxisRight();
        rightAxis2.setEnabled(false);

        int n = 512;
        double[] teste = new double[n];
```

```
        float T = 2;
        float dt = T / n;
        for (int i = 0; i < n ; i++ ){
            teste[i]= (6* Math.cos(2*Math.PI*5.5*i*dt + 2.5));
        }

        int m = (int) (Math.log(n)/Math.log(2));
        m = m - 1;
        Complex[] fftteste = new Complex[n/2];

        //Primeiro inserir no grafico e depois implementar a FFT
        addEntryGrafico1(teste,dt);
        fft(n,m,teste,fftteste);

        double[] fim = new double[n/2];
        for (int i =0 ; i< n/2 ; i++){
            fim[i]= fftteste[i].abs();
        }

        addEntryGrafico2(fim,T);


    }

    public void fft (int n , int m , double[] x , Complex[] xx){
        int k , i , k1 ,k2 , Nf, j , j1 , j2 , e1 , e2 , o1 , o2, n1,
nk, ei, ej, N2;
        double th   , ag , wr , wi ,  dr , di, t, ar, ai, br, bi, s,
c, cr, ci ;

        k2 = 1;
        Nf=n/2;

        for ( k = 1; k<=m ; k++){
            k2 = 2*k2;
            k1=k2/2;
            th=2*(Math.PI/Nf)*k1;
            nk = Nf/k2;
            n1=Nf/k1;
            for ( i = 1 ; i<=nk ; i++){
                ag = (i - 1)*th;
                wr = Math.cos(ag);
                wi = - Math.sin(ag);
                for (j=n1 ; j<=Nf ; j=j + n1){
                    j1 = j - n1 + i - 1;
                    j2 = j1 + nk;
                    e1 = 2*j1;
                    e2 = 2*j2;
                    o1 = e1 + 1;
                    o2 = e2 + 1;
                    dr = x[e1] - x[e2];
                    di = x[o1] - x[o2];
                    x[e1] = x[e1] + x[e2];
                    x[o1] = x[o1] + x[o2];
                    x[e2] = wr*dr - wi*di;
                    x[o2] = wi*dr + wr*di;
                }
            }
```

```
        }


        j = 0;
        for (i=0 ; i<=Nf - 2; i++){
            if (i<j){
                ei = 2*i;
                ej = 2*j;
                t = x[ei];
                x[ei] = x[ej];
                x[ej]=t;
                t= x[ei + 1];
                x[ei + 1]=x[ej + 1];
                x[ej + 1]=t;
            }
            k=Nf/2;
            while (k<=j){
                j=j - k;
                k=k/2;
            }
            j=j+k;
        }



        t = x[0]+x[1];
        x[1]=x[0]-x[1];
        x[0]=t;
        x[Nf + 1]=-x[Nf + 1];
        N2 = 2*Nf;
        for (k=1 ; k<=(Nf/2) - 1; k++){
            k2=2*k;
            nk=N2-k2;
            ar=x[k2]+x[nk];
            ai=x[k2 + 1] - x[nk + 1];
            br=x[nk]-x[k2];
            bi= -x[nk + 1] - x[k2 + 1];
            s = Math.sin(k*(Math.PI/Nf));
            c = Math.cos(k*(Math.PI/Nf));
            cr = br*s-bi*c;
            ci = br*c+bi*s;
            x[k2]=(ar+cr)/2;
            x[k2 + 1]=(ai+ci)/2;
            x[nk]= (ar-cr)/2;
            x[nk + 1]=(ci-ai)/2;

        }

        for (i=0; i<n ; i=i + 2){
            xx[i / 2] = new Complex(x[i] / Nf, x[i + 1] / Nf);
        }
        xx[0]=new Complex(x[0]/(2*Nf) , x[1]/(2*Nf));
    }

    public class Complex {
        public final double re;
        public final double im;
```

```
        public Complex(double r, double i) {
            re = r;
            im = i;
        }

        public String toString() {
            if (im == 0) return re + "";
            if (re == 0) return im + "i";
            if (im < 0) return re + " - " + (-im) + "i";
            return re + " + " + im + "i";
        }

        public double abs(){
            return Math.hypot(re , im);
        }

    }



    private void addEntryGrafico1 (double[] teste2, float dt){


        LineData pontos1 = grafico1.getData();

        if (pontos1 != null) {

            ILineDataSet set1 = pontos1.getDataSetByIndex(0);

            if (set1 == null){
                set1 = createSet1();
                pontos1.addDataSet(set1);
            }


            for (int i =0; i< teste2.length ; i++){
                pontos1.addEntry(new Entry((float) i*dt, (float)
teste2[i] ), 0);
                pontos1.notifyDataChanged();

            }

            grafico1.notifyDataSetChanged();
            grafico1.setVisibleXRangeMaximum(teste2.length*dt/3);

        }

    }


    private LineDataSet createSet1 (){
        LineDataSet set1 = new LineDataSet(null , "Experiencia de Grá-
fico");
        set1.setAxisDependency(YAxis.AxisDependency.LEFT);
        set1.setColor(Color.BLUE);
        set1.setCircleColor(Color.LTGRAY);
```

```
        set1.setLineWidth(1f);
        set1.setCircleRadius(5f);
        set1.setFillAlpha(150);
        set1.setFillColor(Color.BLUE);
        set1.setHighLightColor(Color.RED);
        set1.setValueTextColor(Color.GREEN);
        set1.setValueTextSize(15f);
        set1.setDrawValues(false);
        return set1;

    }

    private void addEntryGrafico2 (double[] fftteste , float T){

        LineData pontos2 = grafico2.getData();


        if (pontos2 != null) {

            ILineDataSet set2 = pontos2.getDataSetByIndex(0);

            if (set2 == null){
                set2 = createSet2();
                pontos2.addDataSet(set2);
            }


            for (int i =0; i< fftteste.length ; i++){
                pontos2.addEntry(new Entry(i * 1/T, (float)
Math.abs(fftteste[i]) ), 0);
                pontos2.notifyDataChanged();

            }


            grafico2.notifyDataSetChanged();
            grafico2.setVisibleXRangeMaximum(20);

        }

    }


    private LineDataSet createSet2 (){
        LineDataSet set2 = new LineDataSet(null , "Experiencia de Grá-
fico");
        set2.setAxisDependency(YAxis.AxisDependency.LEFT);
        set2.setColor(Color.BLUE);
        set2.setCircleColor(Color.BLACK);
        set2.setLineWidth(1f);
        set2.setCircleRadius(5f);
        set2.setFillAlpha(150);
        set2.setFillColor(Color.BLUE);
        set2.setHighLightColor(Color.RED);
        set2.setValueTextColor(Color.GREEN);
        set2.setValueTextSize(15f);
        set2.setDrawValues(false);
        return set2;
```

```
    }


    @Override
    public void onValueSelected(Entry e, Highlight h) {

    }

    @Override
    public void onNothingSelected() {

    }
}
```

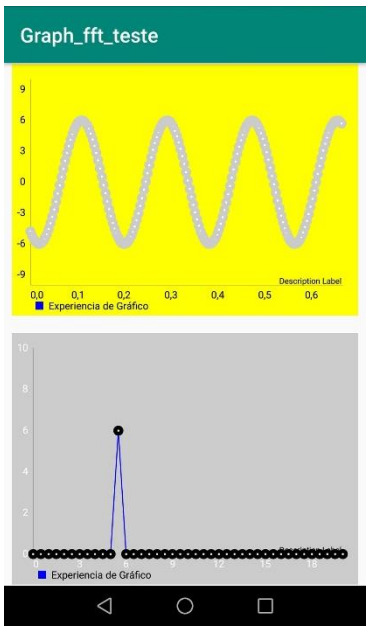


Figura 43 – Interface da aplicação teste referente à representação gráfica de uma função harmonica e respetivo espetro de frequências.

Apêndice D – Código da aplicação teste referente ao armazenamento de dados em ficheiro .txt

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class MainActivity extends AppCompatActivity {

    public EditText data;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        data = findViewById(R.id.data);
    }

    public void save (View view) {
        try {
            DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy
HH:mm:ss");
            Date date = new Date();
            String datedate = (dateFormat.format(date));
            String datadata = data.getText().toString();
            File file = new File(this.getExternalFilesDir(null), "tes-
tfile.txt");
            OutputStreamWriter escritor = new OutputStreamWriter(new
FileOutputStream(file , true));
            escritor.write(datadata + "\\" + datedate + "\r\n");
            escritor.close();
        }catch (IOException e){
            e.printStackTrace();
        }
    }
}
```

Figura 44 – Interface da aplicação teste referente ao armazenamento de dados em ficheiro *.txt*.

Apêndice E – Código da página inicial da aplicação final

```
package com.example.app1.projeto_final_1;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

    }

    public void go (View view) {
        startActivity(new Intent(getApplicationContext(), Menu1Acti-
vity.class));
    }
}
```

Apêndice F – Código do menu inicial da aplicação final

```
package com.example.app1.projeto_final_1;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

import com.example.app1.projeto_final_1.Diagnóstico.Menu_Diagnostico;
import com.example.app1.projeto_final_1.Medição.MenuMedicao;

public class Menu1Activity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu1);
    }

    public void medicaogo(View view) {
        startActivity(new Intent(getApplicationContext(), MenuMedi-
cao.class));
    }

    public void diagnósticogo(View view){
        startActivity(new Intent(getApplicationContext(), Menu_Diag-
nostico.class));
    }
}
```

Apêndice G – Código do menu da deteção da aplicação final

```
package com.example.app1.projeto_final_1.Medição;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;

import com.example.app1.projeto_final_1.Medição.Details.BombasRotati-
vas_Menu;
import com.example.app1.projeto_final_1.Medição.Details.Centrifugado-
res_Menu;
import com.example.app1.projeto_final_1.Medição.Details.Compressore-
sAlternativos_Menu;
import com.example.app1.projeto_final_1.Medição.Details.Compressores-
Rotativos_Menu;
import com.example.app1.projeto_final_1.Medição.Details.GeradoresDie-
sel_Menu;
import com.example.app1.projeto_final_1.Medição.Details.GeradoresTur-
binas_Menu;
import com.example.app1.projeto_final_1.Medição.Details.Motores-
Mais200_Menu;
import com.example.app1.projeto_final_1.Medição.Details.MotoresMe-
nos200_Menu;
import com.example.app1.projeto_final_1.Medição.Details.Motores-
mais15_Menu;
import com.example.app1.projeto_final_1.Medição.Medição.Medicao_Bomba-
sAlternativas;
import com.example.app1.projeto_final_1.Medição.Medição.Medicao_Cai-
xasRedutoras;
import com.example.app1.projeto_final_1.Medição.Medição.Medicao_Li-
nhasVeio;
import com.example.app1.projeto_final_1.Medição.Medição.Medicao_Moto-
resmenos15;
import com.example.app1.projeto_final_1.Medição.Medição.Medicao_Turbi-
nas;
import com.example.app1.projeto_final_1.Medição.Medição.Medicao_Tur-
bocompressores;
import com.example.app1.projeto_final_1.Medição.Medição.Medicao_Vento-
inhas;
import com.example.app1.projeto_final_1.R;

public class MenuMedicao extends AppCompatActivity implements Adapter-
View.OnItemSelectedListener {

    public int indexMaquinas;
    public Spinner spinnerMaquinas;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu_medicao);

        spinnerMaquinas = findViewById(R.id.spinnerMaquinas);
```

```
        ArrayAdapter<CharSequence> adaptermaquinas = ArrayAdapter.cre-
ateFromResource(this,
                R.array.tipos_maquina, android.R.layout.simple_spin-
ner_item);


        adaptermaquinas.setDropDownViewResource(android.R.layout.sim-
ple_spinner_dropdown_item);


        spinnerMaquinas.setAdapter(adaptermaquinas);



        spinnerMaquinas.setOnItemSelectedListener(this);

    }




    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
        indexMaquinas = spinnerMaquinas.getSelectedItemPosition();
    }


    public void Continue(View view){

        switch (indexMaquinas){
            case 0:
                startActivity(new Intent(getApplicationContext(), Me-
dicao_LinhasVeio.class));
                break;
            case 1:
                startActivity(new Intent(getApplicationContext(), Mo-
toresMenos200_Menu.class));
                break;
            case 2:
                startActivity(new Intent(getApplicationContext(), Mo-
toresMais200_Menu.class));
                break;
            case 3:
                startActivity(new Intent(getApplicationContext(), Me-
dicao_Turbocompressores.class));
                break;
            case 4:
                startActivity(new Intent(getApplicationContext(), Ge-
radoresDiesel_Menu.class));
                break;
            case 5:
                startActivity(new Intent(getApplicationContext(), Ge-
radoresTurbinas_Menu.class));
                break;
            case 6:
```

```
                startActivity(new Intent(getApplicationContext(), Me-
dicao_Turbinas.class));
                break;
            case 7:
                startActivity(new Intent(getApplicationContext(), Me-
dicao_CaixasRedutoras.class));
                break;
            case 8:
                startActivity(new Intent(getApplicationContext(), Me-
dicao_Motoresmenos15.class));
                break;
            case 9:
                startActivity(new Intent(getApplicationContext(), Mo-
toresmais15_Menu.class));
                break;
            case 10:
                startActivity(new Intent(getApplicationContext(), Cen-
trifugadores_Menu.class));
                break;
            case 11:
                startActivity(new Intent(getApplicationContext(), Bom-
basRotativas_Menu.class));
                break;
            case 12:
                startActivity(new Intent(getApplicationContext(), Com-
pressoresRotativos_Menu.class));
                break;
            case 13:
                startActivity(new Intent(getApplicationContext(), Me-
dicao_BombasAlternativas.class));
                break;
            case 14:
                startActivity(new Intent(getApplicationContext(), Com-
pressoresAlternativos_Menu.class));
                break;
            case 15:
                startActivity(new Intent(getApplicationContext(), Me-
dicao_Ventoinhas.class));
                break;
        }
    }


    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }
}
```

Apêndice H – Código de menu de detalhes de funcionamento de um quipamento (exemplo de bombas rotativa)

```
package com.example.app1.projeto_final_1.Medição.Details;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

import com.example.app1.projeto_final_1.Medição.Medição.Medicao_Bom-
basRotativas;
import com.example.app1.projeto_final_1.R;

import java.util.Objects;

public class BombasRotativas_Menu extends AppCompatActivity implements
AdapterView.OnItemSelectedListener {

    public EditText RotText;
    public Spinner spinnercategorias;
    public int indexcategorias;
    public int rotacoes;
    public String rotacoesstring;
    public Spinner spinnerpotencia;
    public int indexpotencia;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_bombas_rotativas__menu);

        RotText = findViewById(R.id.RotText);
        spinnercategorias = findViewById(R.id.spinnerapoios);
        spinnerpotencia = findViewById(R.id.spinnerpotencia);

        ArrayAdapter<CharSequence> adaptercategorias = ArrayAdap-
ter.createFromResource(this,
                R.array.categoria, android.R.layout.simple_spin-
ner_item);


        adaptercategorias.setDropDownViewResource(an-
droid.R.layout.simple_spinner_dropdown_item);


        spinnercategorias.setAdapter(adaptercategorias);

        spinnercategorias.setOnItemSelectedListener(this);
```

```
        ArrayAdapter<CharSequence> adapterpotencia = ArrayAdapter.cre-
ateFromResource(this,
                R.array.potencia_bombarot, android.R.layout.sim-
ple_spinner_item);


        adapterpotencia.setDropDownViewResource(android.R.layout.sim-
ple_spinner_dropdown_item);


        spinnerpotencia.setAdapter(adapterpotencia);

        spinnerpotencia.setOnItemSelectedListener(this);




    }

    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
        indexcategorias = spinnercategorias.getSelectedItemPosition();
        indexpotencia = spinnerpotencia.getSelectedItemPosition();
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }

    public void medirbomrot (View view) {

        rotacoesstring = RotText.getText().toString();

        if (Objects.equals(rotacoesstring, "")){

            Toast toast = Toast.makeText(this , "Insira as rotaçãoes
da bomba!" , Toast.LENGTH_LONG);
            toast.show();
        }else {
            rotacoes = Integer.parseInt(rotacoesstring);
            Intent intent = new Intent(BombasRotativas_Menu.this , Me-
dicao_BombasRotativas.class);
            intent.putExtra("rotacoes" , rotacoes);
            intent.putExtra("indexcategorias" , indexcategorias);
            intent.putExtra("indexpotencia" , indexpotencia);
            startActivity(intent);
        }

    }
}
```

Apêndice I – Código de aquisição e análise de dados do sensor para deteção (exemplo de deteção para bombas rotativas)

```
package com.example.app1.projeto_final_1.Medição.Medição;

import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.ProgressBar;
import android.widget.TextView;

import com.example.app1.projeto_final_1.Medição.Details.BombasRotati-
vas_Menu;
import com.example.app1.projeto_final_1.Medição.Save.Save_BombasRota-
tivas;
import com.example.app1.projeto_final_1.R;
import com.phidget22.Accelerometer;
import com.phidget22.AccelerometerAccelerationChangeEvent;
import com.phidget22.AccelerometerAccelerationChangeListener;
import com.phidget22.AttachEvent;
import com.phidget22.AttachListener;
import com.phidget22.DetachEvent;
import com.phidget22.DetachListener;
import com.phidget22.Phidget;
import com.phidget22.PhidgetException;

import java.text.DecimalFormat;

public class Medicao_BombasRotativas extends BombasRotativas_Menu {

    public CheckBox aquisicao;
    public TextView ISOvalue;
    public TextView DNVvalue1;
    public Accelerometer accel1;
    public double[] accelvaluesraw = new double[32768];
    public int dInt;
    public ProgressBar progressBar;
    public int counter = 0;
    public Complex[] xx =new  Complex[(2048)];
    public Complex[] xxvel =new  Complex[(2048)];
    public Complex[] xxdesloc =new  Complex[(2048)];
    public double[] fftaccel = new double[2048];
    public double [] fftvel = new double[2048];
    public double [] fftdesloc = new double[2048];
    public double sumaccel = 0;
    public double sumvel = 0;
    public double sumdesloc = 0;
    public double rmsaccel;
    public double rmsvel;
    public double rmsdesloc;
    public double RMSvel = 0;
    public int rotacoes;
```

```
    public TextView ad1;
    public TextView ad2;
    public TextView ad3;
    public TextView ad4;
    public TextView ISOad05;
    public TextView ISOad1;
    public TextView ISOad2;
    public double pp05;
    public double pp1;
    public double pp2;
    public double rot;
    public double passo = 0.244140625;
    public double firstk;
    public int finalk;
    public int finalk05;

    public double[] accelvalues1 = new double[4096];
    public double[] accelvalues2 = new double[4096];
    public double[] accelvalues3 = new double[4096];
    public double[] accelvalues4 = new double[4096];
    public double[] accelvalues5 = new double[4096];
    public double[] accelvalues6 = new double[4096];
    public double[] accelvalues7 = new double[4096];
    public double[] accelvalues8 = new double[4096];
    public double[] accelvalues = new double[4096];
    public Complex[] xx1 =new Complex[(2048)];
    public Complex[] xx2=new Complex[(2048)];
    public Complex[] xx3 =new Complex[(2048)];
    public Complex[] xx4 =new Complex[(2048)];
    public Complex[] xx5 =new Complex[(2048)];
    public Complex[] xx6 =new Complex[(2048)];
    public Complex[] xx7 =new Complex[(2048)];
    public Complex[] xx8 =new Complex[(2048)];
    public Complex[] xxvel1 =new  Complex[2048];
    public Complex[] xxvel2 =new  Complex[2048];
    public Complex[] xxvel3 =new  Complex[2048];
    public Complex[] xxvel4 =new  Complex[2048];
    public Complex[] xxvel5 =new  Complex[2048];
    public Complex[] xxvel6 =new  Complex[2048];
    public Complex[] xxvel7 =new  Complex[2048];
    public Complex[] xxvel8 =new  Complex[2048];
    public Complex[] xxdesloc1 =new  Complex[2048];
    public Complex[] xxdesloc2 =new  Complex[2048];
    public Complex[] xxdesloc3 =new  Complex[2048];
    public Complex[] xxdesloc4 =new  Complex[2048];
    public Complex[] xxdesloc5 =new  Complex[2048];
    public Complex[] xxdesloc6 =new  Complex[2048];
    public Complex[] xxdesloc7 =new  Complex[2048];
    public Complex[] xxdesloc8 =new  Complex[2048];
    public double sumaccel1 = 0;
    public double sumaccel2 = 0;
    public double sumaccel3 = 0;
    public double sumaccel4 = 0;
    public double sumaccel5 = 0;
    public double sumaccel6 = 0;
    public double sumaccel7 = 0;
    public double sumaccel8 = 0;
```

```
    public double SUMvel1 = 0;
    public double SUMvel2 = 0;
    public double SUMvel3 = 0;
    public double SUMvel4 = 0;
    public double SUMvel5 = 0;
    public double SUMvel6 = 0;
    public double SUMvel7 = 0;
    public double SUMvel8 = 0;
    public double sumvel1 = 0;
    public double sumvel2 = 0;
    public double sumvel3 = 0;
    public double sumvel4 = 0;
    public double sumvel5 = 0;
    public double sumvel6 = 0;
    public double sumvel7 = 0;
    public double sumvel8 = 0;
    public double sumdesloc1 = 0;
    public double sumdesloc2 = 0;
    public double sumdesloc3 = 0;
    public double sumdesloc4 = 0;
    public double sumdesloc5 = 0;
    public double sumdesloc6 = 0;
    public double sumdesloc7 = 0;
    public double sumdesloc8 = 0;
    public double RMSvel1 ;
    public double RMSvel2 ;
    public double RMSvel3 ;
    public double RMSvel4 ;
    public double RMSvel5 ;
    public double RMSvel6 ;
    public double RMSvel7 ;
    public double RMSvel8 ;
    public double rmsvel1 ;
    public double rmsvel2 ;
    public double rmsvel3 ;
    public double rmsvel4 ;
    public double rmsvel5 ;
    public double rmsvel6 ;
    public double rmsvel7 ;
    public double rmsvel8 ;
    public double rmsaccel1 ;
    public double rmsaccel2 ;
    public double rmsaccel3 ;
    public double rmsaccel4 ;
    public double rmsaccel5 ;
    public double rmsaccel6 ;
    public double rmsaccel7 ;
    public double rmsaccel8 ;
    public double rmsdesloc1 ;
    public double rmsdesloc2 ;
    public double rmsdesloc3 ;
    public double rmsdesloc4 ;
    public double rmsdesloc5 ;
    public double rmsdesloc6 ;
    public double rmsdesloc7 ;
    public double rmsdesloc8 ;
    public double[] fftaccel1 = new double[2048];
    public double[] fftaccel2 = new double[2048];
```

```
    public double[] fftaccel3 = new double[2048];
    public double[] fftaccel4 = new double[2048];
    public double[] fftaccel5 = new double[2048];
    public double[] fftaccel6 = new double[2048];
    public double[] fftaccel7 = new double[2048];
    public double[] fftaccel8 = new double[2048];
    public double[] fftvel1 = new double[2048];
    public double[] fftvel2 = new double[2048];
    public double[] fftvel3 = new double[2048];
    public double[] fftvel4 = new double[2048];
    public double[] fftvel5 = new double[2048];
    public double[] fftvel6 = new double[2048];
    public double[] fftvel7 = new double[2048];
    public double[] fftvel8 = new double[2048];
    public double[] fftdesloc1 = new double[2048];
    public double[] fftdesloc2 = new double[2048];
    public double[] fftdesloc3 = new double[2048];
    public double[] fftdesloc4 = new double[2048];
    public double[] fftdesloc5 = new double[2048];
    public double[] fftdesloc6 = new double[2048];
    public double[] fftdesloc7 = new double[2048];
    public double[] fftdesloc8 = new double[2048];


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_medicao__bombas_rotativas);

        aquisicao = findViewById(R.id.aquisicao);
        ISOvalue = findViewById(R.id.ISOvalue1);
        DNVvalue1 = findViewById(R.id.DNVvalue1);
        progressBar = findViewById(R.id.progressBar);
        ad1 = findViewById(R.id.ad1);
        ad2 = findViewById(R.id.ad2);
        ad3 = findViewById(R.id.ad3);
        ad4 = findViewById(R.id.ad4);
        ISOad05 = findViewById(R.id.ISOad05);
        ISOad1 = findViewById(R.id.ISOad1);
        ISOad2 = findViewById(R.id.ISOad2);

        ad1.setVisibility(View.INVISIBLE);
        ad2.setVisibility(View.INVISIBLE);
        ad3.setVisibility(View.INVISIBLE);
        ad4.setVisibility(View.INVISIBLE);
        ISOad05.setVisibility(View.INVISIBLE);
        ISOad1.setVisibility(View.INVISIBLE);
        ISOad2.setVisibility(View.INVISIBLE);

        progressBar.setVisibility(View.INVISIBLE);
        aquisicao.setEnabled(false);

        Intent intent = getIntent();
        rotacoes = intent.getIntExtra("rotacoes" , 0);
        indexcategorias = intent.getIntExtra("indexcategorias" , 0);


        try {
```

```
            accel1 = new Accelerometer();
            if (getPackageManager().hasSystemFeature(PackageMana-
ger.FEATURE_USB_HOST))
                com.phidget22.usb.Manager.Initialize(this);


            accel1.addAttachListener(new AttachListener() {
                public void onAttach(final AttachEvent attachEvent) {
                    AttachEventHandler handler2 = new AttachEventHand-
ler(accel1);
                    synchronized (handler2) {
                        runOnUiThread(handler2);
                        try {
                            handler2.wait();
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
            });

            accel1.addDetachListener(new DetachListener() {
                public void onDetach(final DetachEvent detachEvent) {
                    DetachEventHandler handler2 = new DetachEventHand-
ler(accel1);
                    synchronized (handler2) {
                        runOnUiThread(handler2);
                        try {
                            handler2.wait();
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
            });

            accel1.addAccelerationChangeListener(new AccelerometerAc-
celerationChangeListener() {
                @Override
                public void onAccelerationChange(AccelerometerAccele-
rationChangeEvent accelerationChangeEvent) {
                    AccelerometerAccelerationChangeEventHandler hand-
ler2 = new AccelerometerAccelerationChangeEventHandler(accel1, accele-
rationChangeEvent);
                    runOnUiThread(handler2);
                }
            });

            accel1.open();


        } catch (PhidgetException pe) {
            pe.printStackTrace();
        }
    }

    public void dataIntervalChangeListener() {
        try {
```

```
            dInt = 1;
            accel1.setDataInterval(dInt);

        } catch (PhidgetException e) {
            e.printStackTrace();
        }
    }


    class AttachEventHandler implements Runnable {
        Phidget accel1;


        public AttachEventHandler(Phidget accel1) {
            this.accel1 = accel1;
        }

        public void run() {

            aquisicao.setEnabled(true);


            synchronized (this) {
                this.notify();
            }

        }

    }

    class DetachEventHandler implements Runnable {
        Phidget accel1;

        public DetachEventHandler(Phidget accel1) {
            this.accel1 = accel1;
        }

        @Override
        public void run() {
            aquisicao.setEnabled(false);

            synchronized (this) {
                this.notify();
            }
        }
    }

    class AccelerometerAccelerationChangeEventHandler implements Run-
nable {

        Phidget accel1;
        AccelerometerAccelerationChangeEvent accelerationChangeEvent;


        public AccelerometerAccelerationChangeEventHandler(Phidget ac-
cel1, AccelerometerAccelerationChangeEvent accelerationChangeEvent) {
            this.accel1 = accel1;
            this.accelerationChangeEvent = accelerationChangeEvent;
```

```
        }


        @Override
        public void run() {

            if (aquisicao.isChecked()){
                dataIntervalChangeListener();

                progressBar.setVisibility(View.VISIBLE);

                accelvaluesraw[counter]=accelerationChangeEvent.getAc-
celeration()[2] * 9.81;

                counter=counter+1;

                int m = (int) (Math.log(accelvalues.len-
gth)/Math.log(2));
                m = m - 1;


                if (counter == 32768) {
                    aquisicao.setChecked(false);
                    progressBar.setVisibility(View.INVISIBLE);
                    counter = 0;

                    for (int i = 0 ; i < 4096 ; i++){
                        accelvalues1 [i] = accelvaluesraw [i];
                        accelvalues2 [i] = accelvaluesraw [i + 4096];
                        accelvalues3 [i] = accelvaluesraw [i + 8192];
                        accelvalues4 [i] = accelvaluesraw [i + 12288];
                        accelvalues5 [i] = accelvaluesraw [i + 16384];
                        accelvalues6 [i] = accelvaluesraw [i + 20480];
                        accelvalues7 [i] = accelvaluesraw [i + 24576];
                        accelvalues8 [i] = accelvaluesraw [i + 28672];
                    }

                    fft(accelvalues.length, m, accelvalues1, xx1);
                    fft(accelvalues.length, m, accelvalues2, xx2);
                    fft(accelvalues.length, m, accelvalues3, xx3);
                    fft(accelvalues.length, m, accelvalues4, xx4);
                    fft(accelvalues.length, m, accelvalues5, xx5);
                    fft(accelvalues.length, m, accelvalues6, xx6);
                    fft(accelvalues.length, m, accelvalues7, xx7);
                    fft(accelvalues.length, m, accelvalues8, xx8);



                    Complex j = new Complex(0 , -1);

                    for (int i = 1; i < xx.length; i++) {
                        fftaccel1[i] = xx1[i].abs();
                        xxvel1[i] =(xx1[i].scale(1 / (2 * Math.PI *(i
/ (dInt * 0.001 * 4096))))).times(j);
                        xxdesloc1[i] = xx1[i].scale(1 / ((2 * Math.PI
* i / (dInt * 0.001 * 4096))* (2 * Math.PI * i / (dInt * 0.001 *
```

```
4096))) * -1);
                        fftvel1[i] = xxvel1[i].abs()*1000;
                        fftdesloc1[i] = xxdesloc1[i].abs()*1000000;

                        fftaccel2[i] = xx2[i].abs();
                        xxvel2[i] =(xx2[i].scale(1 / (2 * Math.PI * (i
/ (dInt * 0.001 * 4096))))).times(j);
                        xxdesloc2[i] = xx2[i].scale(1 / ((2 * Math.PI
* i / (dInt * 0.001 * 4096))* (2 * Math.PI * i / (dInt * 0.001 *
4096))) * -1);
                        fftvel2[i] = xxvel2[i].abs()*1000;
                        fftdesloc2[i] = xxdesloc2[i].abs()*1000000;

                        fftaccel3[i] = xx3[i].abs();
                        xxvel3[i] =(xx3[i].scale(1 / (2 * Math.PI * (i
/ (dInt * 0.001 * 4096))))).times(j);
                        xxdesloc3[i] = xx3[i].scale(1 / ((2 * Math.PI
* i / (dInt * 0.001 * 4096))* (2 * Math.PI * i / (dInt * 0.001 *
4096))) * -1);
                        fftvel3[i] = xxvel3[i].abs()*1000;
                        fftdesloc3[i] = xxdesloc3[i].abs()*1000000;

                        fftaccel4[i] = xx4[i].abs();
                        xxvel4[i] =(xx4[i].scale(1 / (2 * Math.PI * (i
/ (dInt * 0.001 * 4096))))).times(j);
                        xxdesloc4[i] = xx4[i].scale(1 / ((2 * Math.PI
* i / (dInt * 0.001 * 4096))* (2 * Math.PI * i / (dInt * 0.001 *
4096))) * -1);
                        fftvel4[i] = xxvel4[i].abs()*1000;
                        fftdesloc4[i] = xxdesloc4[i].abs()*1000000;

                        fftaccel5[i] = xx5[i].abs();
                        xxvel5[i] =(xx5[i].scale(1 / (2 * Math.PI * (i
/ (dInt * 0.001 * 4096))))).times(j);
                        xxdesloc5[i] = xx5[i].scale(1 / ((2 * Math.PI
* i / (dInt * 0.001 * 4096))* (2 * Math.PI * i / (dInt * 0.001 *
4096))) * -1);
                        fftvel5[i] = xxvel5[i].abs()*1000;
                        fftdesloc5[i] = xxdesloc5[i].abs()*1000000;

                        fftaccel6[i] = xx6[i].abs();
                        xxvel6[i] =(xx6[i].scale(1 / (2 * Math.PI * (i
/ (dInt * 0.001 * 4096))))).times(j);
                        xxdesloc6[i] = xx6[i].scale(1 / ((2 * Math.PI
* i / (dInt * 0.001 * 4096))* (2 * Math.PI * i / (dInt * 0.001 *
4096))) * -1);
                        fftvel6[i] = xxvel6[i].abs()*1000;
                        fftdesloc6[i] = xxdesloc6[i].abs()*1000000;

                        fftaccel7[i] = xx7[i].abs();
                        xxvel7[i] =(xx7[i].scale(1 / (2 * Math.PI * (i
/ (dInt * 0.001 * 4096))))).times(j);
                        xxdesloc7[i] = xx7[i].scale(1 / ((2 * Math.PI
* i / (dInt * 0.001 * 4096))* (2 * Math.PI * i / (dInt * 0.001 *
4096))) * -1);
                        fftvel7[i] = xxvel7[i].abs()*1000;
                        fftdesloc7[i] = xxdesloc7[i].abs()*1000000;
```

```
                        fftaccel8[i] = xx8[i].abs();
                        xxvel8[i] =(xx8[i].scale(1 / (2 * Math.PI * (i
/ (dInt * 0.001 * 4096))))).times(j);
                        xxdesloc8[i] = xx8[i].scale(1 / ((2 * Math.PI
* i / (dInt * 0.001 * 4096))* (2 * Math.PI * i / (dInt * 0.001 *
4096))) * -1);
                        fftvel8[i] = xxvel8[i].abs()*1000;
                        fftdesloc8[i] = xxdesloc8[i].abs()*1000000;

                    }



                    for (int i = 41 ; i< xx.length; i++){
                        sumaccel1 = sumaccel1 + (fftaccel1[i] * fftac-
cel1[i]);
                        sumvel1 = sumvel1 + (fftvel1[i] * fftvel1[i]);
                        sumdesloc1 = sumdesloc1 + (fftdesloc1[i] *
fftdesloc1[i]);

                        sumaccel2 = sumaccel2 + (fftaccel2[i] * fftac-
cel2[i]);
                        sumvel2 = sumvel2 + (fftvel2[i] * fftvel2[i]);
                        sumdesloc2 = sumdesloc2 + (fftdesloc2[i] *
fftdesloc2[i]);

                        sumaccel3 = sumaccel3 + (fftaccel3[i] * fftac-
cel3[i]);
                        sumvel3 = sumvel3 + (fftvel3[i] * fftvel3[i]);
                        sumdesloc3 = sumdesloc3 + (fftdesloc3[i] *
fftdesloc3[i]);

                        sumaccel4 = sumaccel4 + (fftaccel4[i] * fftac-
cel4[i]);
                        sumvel4 = sumvel4 + (fftvel4[i] * fftvel4[i]);
                        sumdesloc4 = sumdesloc4 + (fftdesloc4[i] *
fftdesloc4[i]);

                        sumaccel5 = sumaccel5 + (fftaccel5[i] * fftac-
cel5[i]);
                        sumvel5 = sumvel5 + (fftvel5[i] * fftvel5[i]);
                        sumdesloc5 = sumdesloc5 + (fftdesloc5[i] *
fftdesloc5[i]);

                        sumaccel6 = sumaccel6 + (fftaccel6[i] * fftac-
cel6[i]);
                        sumvel6 = sumvel6 + (fftvel6[i] * fftvel6[i]);
                        sumdesloc6 = sumdesloc6 + (fftdesloc6[i] *
fftdesloc6[i]);

                        sumaccel7 = sumaccel7 + (fftaccel7[i] * fftac-
cel7[i]);
                        sumvel7 = sumvel7 + (fftvel7[i] * fftvel7[i]);
                        sumdesloc7 = sumdesloc7 + (fftdesloc7[i] *
fftdesloc7[i]);

                        sumaccel8 = sumaccel8 + (fftaccel8[i] * fftac-
cel8[i]);
```

```
                        sumvel8 = sumvel8 + (fftvel8[i] * fftvel8[i]);
                        sumdesloc8 = sumdesloc8 + (fftdesloc8[i] *
fftdesloc8[i]);
                    }

                    for (int i = 16; i < 810; i++){

                        SUMvel1 = SUMvel1 + (fftvel1[i] * fftvel1[i]);
                        SUMvel2 = SUMvel2 + (fftvel2[i] * fftvel2[i]);
                        SUMvel3 = SUMvel3 + (fftvel3[i] * fftvel3[i]);
                        SUMvel4 = SUMvel4 + (fftvel4[i] * fftvel4[i]);
                        SUMvel5 = SUMvel5 + (fftvel5[i] * fftvel5[i]);
                        SUMvel6 = SUMvel6 + (fftvel6[i] * fftvel6[i]);
                        SUMvel7 = SUMvel7 + (fftvel7[i] * fftvel7[i]);
                        SUMvel8 = SUMvel8 + (fftvel8[i] * fftvel8[i]);
                    }

                    rmsaccel1 = Math.sqrt(sumaccel1/2);
                    rmsaccel2 = Math.sqrt(sumaccel2/2);
                    rmsaccel3 = Math.sqrt(sumaccel3/2);
                    rmsaccel4 = Math.sqrt(sumaccel4/2);
                    rmsaccel5 = Math.sqrt(sumaccel5/2);
                    rmsaccel6 = Math.sqrt(sumaccel6/2);
                    rmsaccel7 = Math.sqrt(sumaccel7/2);
                    rmsaccel8 = Math.sqrt(sumaccel8/2);

                    rmsvel1 = Math.sqrt(sumvel1/2);
                    rmsvel2 = Math.sqrt(sumvel2/2);
                    rmsvel3 = Math.sqrt(sumvel3/2);
                    rmsvel4 = Math.sqrt(sumvel4/2);
                    rmsvel5 = Math.sqrt(sumvel5/2);
                    rmsvel6 = Math.sqrt(sumvel6/2);
                    rmsvel7 = Math.sqrt(sumvel7/2);
                    rmsvel8 = Math.sqrt(sumvel8/2);

                    rmsdesloc1 = Math.sqrt(sumdesloc1/2);
                    rmsdesloc2 = Math.sqrt(sumdesloc2/2);
                    rmsdesloc3 = Math.sqrt(sumdesloc3/2);
                    rmsdesloc4 = Math.sqrt(sumdesloc4/2);
                    rmsdesloc5 = Math.sqrt(sumdesloc5/2);
                    rmsdesloc6 = Math.sqrt(sumdesloc6/2);
                    rmsdesloc7 = Math.sqrt(sumdesloc7/2);
                    rmsdesloc8 = Math.sqrt(sumdesloc8/2);

                    rmsaccel = (rmsaccel1 +rmsaccel2+rmsaccel3+rmsac-
cel4+rmsaccel5+rmsaccel6+rmsaccel7+rmsaccel8)/8;

                    rmsvel =
(rmsvel1+rmsvel2+rmsvel3+rmsvel4+rmsvel5+rmsvel6+rmsvel7+rmsvel8)/8;

                    rmsdesloc = (rmsdesloc1+rmsdesloc2+rmsdesloc3+rms-
desloc4+rmsdesloc5+rmsdesloc6+rmsdesloc7+rmsdesloc8)/8;

                    RMSvel =
(RMSvel1+RMSvel2+RMSvel3+RMSvel4+RMSvel5+RMSvel6+RMSvel7+RMSvel8)/8;


                    DecimalFormat df = new DecimalFormat("##.##");
```

```
                    ISOvalue.setText(df.format(rmsvel) + "mm/s");
                    DNVvalue1.setText(df.format(RMSvel) + "mm/s");


                    if (indexcategorias == 0 && indexpotencia == 1){
                        if (rmsvel >= 2.5  && rmsvel < 4){
                            ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.lightgreen));
                        }else if (rmsvel >= 4 && rmsvel < 6.6){
                            ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.yellow));
                        }else if (rmsvel >= 6.6){
                            ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.red));
                        }else ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.darkgreen));
                    }else if (indexcategorias == 0 && indexpotencia ==
0){
                        if (rmsvel >= 3.5 && rmsvel < 5){
                            ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.lightgreen));
                        }else if (rmsvel >= 5 && rmsvel < 7.6){
                            ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.yellow));
                        }else if (rmsvel >= 7.6){
                            ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.red));
                        }else ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.darkgreen));
                    }else if (indexcategorias == 1 && indexpotencia ==
1){
                        if (rmsvel >= 3.2 && rmsvel < 5.1){
                            ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.lightgreen));
                        }else if (rmsvel >= 5.1 && rmsvel < 8.5){
                            ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.yellow));
                        }else if (rmsvel >= 8.5){
                            ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.red));
                        }else ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.darkgreen));
                    }else if (indexcategorias == 1 && indexpotencia ==
0){
                        if (rmsvel >= 3.2 && rmsvel < 5.1){
                            ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.lightgreen));
                        }else if (rmsvel >= 5.1 && rmsvel < 8.5){
                            ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.yellow));
                        }else if (rmsvel >= 8.5){
                            ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.red));
                        }else ISOvalue.setBackgroundColor(getResour-
ces().getColor(R.color.darkgreen));
                    }
```

```
                    if (RMSvel >= 7){
                        DNVvalue1.setBackgroundColor(getResour-
ces().getColor(R.color.red));
                    }else DNVvalue1.setBackgroundColor(getResour-
ces().getColor(R.color.darkgreen));

                    if (rotacoes < 600){
                        ad1.setVisibility(View.VISIBLE);
                        ad2.setVisibility(View.VISIBLE);
                        ad3.setVisibility(View.VISIBLE);
                        ad4.setVisibility(View.VISIBLE);
                        ISOad05.setVisibility(View.VISIBLE);
                        ISOad1.setVisibility(View.VISIBLE);
                        ISOad2.setVisibility(View.VISIBLE);
                        rot = rotacoes * 0.0166666666666666667;
                        firstk = rot / passo;

                        for (int i = 1 ; i <= 42 ; i++){
                            if (Math.abs(i - firstk) < 0.5){
                                finalk = i;
                                if (finalk%2 != 0){
                                    finalk05 = finalk + 1;
                                }
                            }
                        }

                        pp05 = ((fftdesloc1[finalk05 / 2] *2)+
(fftdesloc2[finalk05 / 2] *2)+(fftdesloc3[finalk05 / 2] *2)+ (fftdes-
loc4[finalk05 / 2] *2)
                        +(fftdesloc5[finalk05 / 2] *2)+(fftdesloc6[fi-
nalk05 / 2] *2)+(fftdesloc7[finalk05 / 2] *2)+(fftdesloc8[finalk05 /
2] *2))/8;
                        pp1 = ((fftdesloc1[finalk] *2)+(fftdesloc2[fi-
nalk] *2)+(fftdesloc3[finalk] *2)+(fftdesloc4[finalk] *2)+(fftdes-
loc5[finalk] *2)+(fftdesloc6[finalk] *2)+(fftdesloc7[finalk]
*2)+(fftdesloc8[finalk] *2))/8;
                        pp2 = ((fftdesloc1[finalk * 2] *2)+(fftdes-
loc2[finalk * 2] *2)+(fftdesloc3[finalk * 2] *2)+(fftdesloc4[finalk *
2] *2)+(fftdesloc5[finalk * 2] *2)+(fftdesloc6[finalk * 2]
*2)+(fftdesloc7[finalk * 2] *2)+(fftdesloc8[finalk * 2] *2))/8;


                        ISOad05.setText(df.format(pp05) + "microm");
                        ISOad1.setText(df.format(pp1) + "microm");
                        ISOad2.setText(df.format(pp2) + "microm");

                        if (pp05 >= 50 && pp05 < 80){
                            ISOad05.setBackgroundColor(getResour-
ces().getColor(R.color.lightgreen));
                        }else if (pp05 >= 80 && pp05 < 130){
                            ISOad05.setBackgroundColor(getResour-
ces().getColor(R.color.yellow));
                        }else if (pp05 >= 130){
                            ISOad05.setBackgroundColor(getResour-
ces().getColor(R.color.red));
                        }else ISOad05.setBackgroundColor(getResour-
ces().getColor(R.color.darkgreen));
```

```
                        if (pp1 >= 50 && pp1 < 80){
                            ISOad1.setBackgroundColor(getResour-
ces().getColor(R.color.lightgreen));
                        }else if (pp1 >= 80 && pp1 < 130){
                            ISOad1.setBackgroundColor(getResour-
ces().getColor(R.color.yellow));
                        }else if (pp1 >= 130){
                            ISOad1.setBackgroundColor(getResour-
ces().getColor(R.color.red));
                        }else ISOad1.setBackgroundColor(getResour-
ces().getColor(R.color.darkgreen));

                        if (pp2 >= 50 && pp2 < 80){
                            ISOad2.setBackgroundColor(getResour-
ces().getColor(R.color.lightgreen));
                        }else if (pp2 >= 80 && pp2 < 130){
                            ISOad2.setBackgroundColor(getResour-
ces().getColor(R.color.yellow));
                        }else if (pp2 >= 130){
                            ISOad2.setBackgroundColor(getResour-
ces().getColor(R.color.red));
                        }else ISOad2.setBackgroundColor(getResour-
ces().getColor(R.color.darkgreen));
                    }

                }
            }

        }

        public  void fft (int n , int m , double[] x , Complex[] xx){
            int k , i , k1 ,k2 , Nf, j , j1 , j2 , e1 , e2 , o1 , o2,
n1, nk, ei, ej, N2;
            double th   , ag , wr , wi ,  dr , di, t, ar, ai, br, bi,
s, c, cr, ci ;

            k2 = 1;
            Nf=n/2;

            for ( k = 1; k<=m ; k++){
                k2 = 2*k2;
                k1=k2/2;
                th=2*(Math.PI/Nf)*k1;
                nk = Nf/k2;
                n1=Nf/k1;
                for ( i = 1 ; i<=nk ; i++){
                    ag = (i - 1)*th;
                    wr = Math.cos(ag);
                    wi = - Math.sin(ag);
                    for (j=n1 ; j<=Nf ; j=j + n1){
                        j1 = j - n1 + i - 1;
                        j2 = j1 + nk;
                        e1 = 2*j1;
                        e2 = 2*j2;
                        o1 = e1 + 1;
                        o2 = e2 + 1;
                        dr = x[e1] - x[e2];
                        di = x[o1] - x[o2];
```

```
                x[e1] = x[e1] + x[e2];
                x[o1] = x[o1] + x[o2];
                x[e2] = wr*dr - wi*di;
                x[o2] = wi*dr + wr*di;
            }
        }
    }


    j = 0;
    for (i=0 ; i<=Nf - 2; i++){
        if (i<j){
            ei = 2*i;
            ej = 2*j;
            t = x[ei];
            x[ei] = x[ej];
            x[ej]=t;
            t= x[ei + 1];
            x[ei + 1]=x[ej + 1];
            x[ej + 1]=t;
        }
        k=Nf/2;
        while (k<=j){
            j=j - k;
            k=k/2;
        }
        j=j+k;
    }



    t = x[0]+x[1];
    x[1]=x[0]-x[1];
    x[0]=t;
    x[Nf + 1]=-x[Nf + 1];
    N2 = 2*Nf;
    for (k=1 ; k<=(Nf/2) - 1; k++){
        k2=2*k;
        nk=N2-k2;
        ar=x[k2]+x[nk];
        ai=x[k2 + 1] - x[nk + 1];
        br=x[nk]-x[k2];
        bi= -x[nk + 1] - x[k2 + 1];
        s = Math.sin(k*(Math.PI/Nf));
        c = Math.cos(k*(Math.PI/Nf));
        cr = br*s-bi*c;
        ci = br*c+bi*s;
        x[k2]=(ar+cr)/2;
        x[k2 + 1]=(ai+ci)/2;
        x[nk]= (ar-cr)/2;
        x[nk + 1]=(ci-ai)/2;

    }

    for (i=0; i<n ; i=i + 2){
        xx[i / 2] = new Complex(x[i] / Nf, x[i + 1] / Nf);
    }
    xx[0]=new Complex(x[0]/(2*Nf) , x[1]/(2*Nf));
```

```
        }
    }

    public class Complex {
        public final double re;
        public final double im;



        public Complex(double r, double i) {
            re = r;
            im = i;
        }

        public String toString() {
            if (im == 0) return re + "";
            if (re == 0) return im + "i";
            if (im < 0) return re + " - " + (-im) + "i";
            return re + " + " + im + "i";
        }

        public double abs(){
            return Math.hypot(re , im);
        }
        public Complex conjugate() {
            return new Complex(re, -im);
        }
        public Complex scale(double alpha) {
            return new Complex(alpha * re, alpha * im);
        }
        public Complex times(Complex b) {
            Complex a = this;
            double real = a.re * b.re - a.im * b.im;
            double imag = a.re * b.im + a.im * b.re;
            return new Complex(real, imag);
        }

        public Complex plus(Complex b) {
            Complex a = this;
            double real = a.re + b.re;
            double imag = a.im + b.im;
            return new Complex(real, imag);
        }

        public Complex minus(Complex b) {
            Complex a = this;
            double real = a.re - b.re;
            double imag = a.im - b.im;
            return new Complex(real, imag);
        }

    }

    public void guardar (View view){
        Intent intent = new Intent(Medicao_BombasRotativas.this ,
Save_BombasRotativas.class);
        intent.putExtra("rmsvel" , rmsvel);
```

```
        intent.putExtra("RMSvel" , RMSvel);
        if (rotacoes < 600){
            intent.putExtra("pp05" , pp05);
            intent.putExtra("pp1" , pp1);
            intent.putExtra("pp2" , pp2);
        }
        intent.putExtra("rotacoes" , rotacoes);
        intent.putExtra("indexcategorias" , indexcategorias);
        intent.putExtra("indexpotencia" , indexpotencia);
        startActivity(intent);
    }


    @Override
    protected void onDestroy() {
        super.onDestroy();


        try {

            if (getPackageManager().hasSystemFeature(PackageMana-
ger.FEATURE_USB_HOST))
                com.phidget22.usb.Manager.Uninitialize();


            accel1.removeAttachListener( new AttachListener() {
                public void onAttach(final AttachEvent attachEvent) {
                    AttachEventHandler handler2 = new AttachEventHand-
ler(accel1);
                    synchronized (handler2) {
                        runOnUiThread(handler2);
                        try {
                            handler2.wait();
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
            });

            accel1.removeDetachListener(new DetachListener() {
                public void onDetach(final DetachEvent detachEvent) {
                    DetachEventHandler handler2 = new DetachEventHand-
ler(accel1);
                    synchronized (handler2) {
                        runOnUiThread(handler2);
                        try {
                            handler2.wait();
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
            });

            accel1.removeAccelerationChangeListener(new Accelerometer-
rAccelerationChangeListener() {
                @Override
```

```
                public void onAccelerationChange(AccelerometerAccele-
rationChangeEvent accelerationChangeEvent) {
                    AccelerometerAccelerationChangeEventHandler hand-
ler2 = new AccelerometerAccelerationChangeEventHandler(accel1, accele-
rationChangeEvent);
                    runOnUiThread(handler2);
                }
            });

            accel1.close();


        } catch (PhidgetException pe) {
            pe.printStackTrace();
        }
    }
}
```

Apêndice J – Código do armazenamento dos dados de deteção obtidos (exemplo de armazenamento para deteção para bombas rotativas)

```
package com.example.app1.projeto_final_1.Medição.Save;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

import com.example.app1.projeto_final_1.Medição.Medição.Medicao_Bom-
basRotativas;
import com.example.app1.projeto_final_1.R;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.text.DateFormat;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class Save_BombasRotativas extends Medicao_BombasRotativas {

    public EditText NavioText;
    public EditText OperadorText;
    public EditText EquipamentoText;
    public EditText SerieText;
    public EditText ChumaceiraText;
    public Spinner spinnerdir;
    public Spinner spinnersituacao;
    public EditText ObsText;
    public int indexsituacao;
    public int indexdir;
    public String[] sit = new String[2];
    public String[] dir = new String[3];
    public String avalISO;
    public String avalDNV;
    public String avalISOad05;
    public String avalISOad1;
    public String avalISOad2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_save__bombas_rotativas);

        NavioText = findViewById(R.id.NavioText);
        OperadorText = findViewById(R.id.OperadorText);
```

```
        EquipamentoText = findViewById(R.id.EquipamentoText);
        SerieText = findViewById(R.id.SerieText);
        ChumaceiraText = findViewById(R.id.ChumaceiraText);
        spinnerdir = findViewById(R.id.spinnerdir);
        spinnersituacao = findViewById(R.id.spinnersituacao);
        ObsText = findViewById(R.id.ObsText);

        Intent intent = getIntent();
        rmsvel = intent.getDoubleExtra("rmsvel" , 0);
        RMSvel = intent.getDoubleExtra("RMSvel" , 0);
        rotacoes = intent.getIntExtra("rotacoes" , 0);
        pp05 = intent.getDoubleExtra("pp05" , 0);
        pp1 = intent.getDoubleExtra("pp1" , 0);
        pp2 = intent.getDoubleExtra("pp2" , 0);
        indexcategorias = intent.getIntExtra("indexcategorias" , 0);
        indexpotencia = intent.getIntExtra("indexpotencia" , 0);

        ArrayAdapter<CharSequence> adaptersituacao = ArrayAdapter.cre-
ateFromResource(this,
                R.array.situacao, android.R.layout.simple_spin-
ner_item);


        adaptersituacao.setDropDownViewResource(android.R.layout.sim-
ple_spinner_dropdown_item);


        spinnersituacao.setAdapter(adaptersituacao);

        spinnersituacao.setOnItemSelectedListener(this);

        ArrayAdapter<CharSequence> adapterdirecao = ArrayAdapter.crea-
teFromResource(this,
                R.array.direção, android.R.layout.simple_spin-
ner_item);


        adapterdirecao.setDropDownViewResource(android.R.layout.sim-
ple_spinner_dropdown_item);


        spinnerdir.setAdapter(adapterdirecao);

        spinnerdir.setOnItemSelectedListener(this);

        sit = getResources().getStringArray(R.array.situacao);
        dir = getResources().getStringArray(R.array.direção);

        if (indexcategorias == 0 && indexpotencia == 1){
            if (rmsvel >= 2.5  && rmsvel < 4){
                avalISO = "Satisfatório" ;
            }else if (rmsvel >= 4 && rmsvel < 6.6){
                avalISO = "Insatisfatório" ;
            }else if (rmsvel >= 6.6){
                avalISO = "Inaceitável" ;
            }else avalISO = "Bom" ;
        }else if (indexcategorias == 0 && indexpotencia == 0){
            if (rmsvel >= 3.5 && rmsvel < 5){
```

```
                avalISO = "Satisfatório" ;
            }else if (rmsvel >= 5 && rmsvel < 7.6){
                avalISO = "Insatisfatório" ;
            }else if (rmsvel >= 7.6){
                avalISO = "Inaceitável" ;
            }else avalISO = "Bom" ;
        }else if (indexcategorias == 1 && indexpotencia == 1){
            if (rmsvel >= 3.2 && rmsvel < 5.1){
                avalISO = "Satisfatório" ;
            }else if (rmsvel >= 5.1 && rmsvel < 8.5){
                avalISO = "Insatisfatório" ;
            }else if (rmsvel >= 8.5){
                avalISO = "Inaceitável" ;
            }else avalISO = "Bom" ;
        }else if (indexcategorias == 1 && indexpotencia == 0){
            if (rmsvel >= 3.2 && rmsvel < 5.1){
                avalISO = "Satisfatório" ;
            }else if (rmsvel >= 5.1 && rmsvel < 8.5){
                avalISO = "Insatisfatório" ;
            }else if (rmsvel >= 8.5){
                avalISO = "Inaceitável" ;
            }else avalISO = "Bom" ;
        }

        if (RMSvel >= 7 ){
            avalDNV = "Mau";
        }else avalDNV = "Bom";

        if (pp05 >= 50 && pp05 < 80){
            avalISOad05 = "Satisfatório" ;
        }else if (pp05 >= 80 && pp05 < 130){
            avalISOad05 = "Insatisfatório" ;
        }else if (pp05 >= 130){
            avalISOad05 = "Inaceitável" ;
        }else avalISOad05 = "Bom" ;

        if (pp1 >= 50 && pp1 < 80){
            avalISOad1 = "Satisfatório" ;
        }else if (pp1 >= 80 && pp1 < 130){
            avalISOad1 = "Insatisfatório" ;
        }else if (pp1 >= 130){
            avalISOad1 = "Inaceitável" ;
        }else avalISOad1 = "Bom" ;

        if (pp2 >= 50 && pp2 < 80){
            avalISOad2 = "Satisfatório" ;
        }else if (pp2 >= 80 && pp2 < 130){
            avalISOad2 = "Insatisfatório" ;
        }else if (pp2 >= 130){
            avalISOad2 = "Inaceitável" ;
        }else avalISOad2 = "Bom" ;

    }

    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
        indexdir = spinnerdir.getSelectedItemPosition();
```

```
        indexsituacao = spinnersituacao.getSelectedItemPosition();
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {

    }

    public void savedata (View view){
        try {
            DecimalFormat df = new DecimalFormat("##.##");
            DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy
HH:mm:ss");
            Date date = new Date();
            String datedate = (dateFormat.format(date));
            String navio = NavioText.getText().toString();
            String operador = OperadorText.getText().toString();
            String equipamento = EquipamentoText.getText().toString();
            String serie = SerieText.getText().toString();
            String chumaceira = ChumaceiraText.getText().toString();
            String obs = ObsText.getText().toString();
            File file = new File(this.getExternalFilesDir(null), "De-
teção.txt");
            OutputStreamWriter escritor = new OutputStreamWriter(new
FileOutputStream(file , true));
            escritor.write("Info Navio" + "\r\n"
                    + "Data/hora: " + datedate + "\r\n"
                    + "Navio: " + navio + "\r\n"
                    + "Operador: " + operador + "\r\n"
                    + "Situação: " + sit[indexsituacao] + "\r\n"
                    + "\r\n"
                    + "Info Medição" + "\r\n"
                    + "Equipamento: " + equipamento + "\r\n"
                    + "N° de Série: " + serie + "\r\n"
                    + "Chumaceira: " + chumaceira + "\r\n"
                    + "Direção: " + dir[indexdir] + "\r\n"
                    + "Observações: " + obs + "\r\n"
                    + "\r\n"
                    + "Avaliação" + "\r\n"
                    + "Valor ISO: " + df.format(rmsvel) + "mm/s" +
"\r\n"
                    + "Avaliação ISO: " + avalISO + "\r\n"
                    + "Valor DNV: " + df.format(RMSvel) + "mm/s" +
"\r\n"
                    + "Avaliação DNV: " + avalDNV + "\r\n");
            if (rotacoes < 600){
                escritor.write("\r\n"
                        + "Avaliação adicional (RPM < 600)" + "\r\n"
                        + "Valor ISO 0.5x RPM: " + df.format(pp05) +
"microm" + "\r\n"
                        + "Avaliação ISO 0.5x RPM: " + avalISOad05 +
"\r\n"
                        + "Valor ISO 1x RPM: " + df.format(pp1) + "mi-
crom" + "\r\n"
                        + "Avaliação ISO 1x RPM: " + avalISOad1 +
"\r\n"
                        + "Valor ISO 2x RPM: " + df.format(pp2) + "mi-
crom" + "\r\n"
```

```
                        + "Avaliação ISO 2x RPM: " + avalISOad2 +
"\r\n"
                        + "
**********************************************************************
***************** " + "\r\n"
                        + "\r\n");
            }else escritor.write("
**********************************************************************
***************** " + "\r\n"
                    + "\r\n");
            escritor.close();
        }catch (IOException e){
            e.printStackTrace();
        }
        Toast toast = Toast.makeText(this , "A sua medição foi guar-
dada com sucesso" , Toast.LENGTH_LONG);
        toast.show();
    }
}
```

Apêndice K – Código do menu de diagnóstico da aplicação final

```
package com.example.app1.projeto_final_1.Diagnóstico;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Switch;

import com.example.app1.projeto_final_1.R;

public class Menu_Diagnostico extends AppCompatActivity  {

    public Spinner spinner_numeropontos;
    public int indexnumeropontos;
    public Spinner spinner_df;
    public int indexdf;
    public int dInt;
    public String dI;
    public String[] npt = new String[4];
    public int numpt;
    public String numpt1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu__diagnostico);

        spinner_numeropontos = findViewById(R.id.spinner_numeropon-
tos);
        spinner_df = findViewById(R.id.spinner_df);


        ArrayAdapter<CharSequence> adapternumeropontos = ArrayAdap-
ter.createFromResource(this,
                R.array.freq_max , android.R.layout.simple_spin-
ner_item);


        adapternumeropontos.setDropDownViewResource(an-
droid.R.layout.simple_spinner_dropdown_item);


        spinner_numeropontos.setAdapter(adapternumeropontos);



        spinner_numeropontos.setOnItemSelectedListener(new Adapter-
View.OnItemSelectedListener() {
            @Override
```

```
            public void onItemSelected(AdapterView<?> parent, View
view, int position, long id) {
                indexnumeropontos = spinner_numeropontos.getSelectedI-
temPosition();
                if (indexnumeropontos == 0){
                    ArrayAdapter<CharSequence> adapterdf = ArrayAdap-
ter.createFromResource(getApplicationContext(),
                            R.array.df_500 , android.R.layout.sim-
ple_spinner_item);


                    adapterdf.setDropDownViewResource(an-
droid.R.layout.simple_spinner_dropdown_item);


                    spinner_df.setAdapter(adapterdf);

                    spinner_df.setOnItemSelectedListener(new Adapter-
View.OnItemSelectedListener() {
                        @Override
                        public void onItemSelected(AdapterView<?> pa-
rent, View view, int position, long id) {
                            indexdf = spinner_df.getSelectedItemPosi-
tion();
                        }

                        @Override
                        public void onNothingSelected(AdapterView<?>
parent) {

                        }
                    });
                }else if (indexnumeropontos == 1){
                    ArrayAdapter<CharSequence> adapterdf = ArrayAdap-
ter.createFromResource(getApplicationContext(),
                            R.array.df_250 , android.R.layout.sim-
ple_spinner_item);


                    adapterdf.setDropDownViewResource(an-
droid.R.layout.simple_spinner_dropdown_item);


                    spinner_df.setAdapter(adapterdf);
                    spinner_df.setOnItemSelectedListener(new Adapter-
View.OnItemSelectedListener() {
                        @Override
                        public void onItemSelected(AdapterView<?> pa-
rent, View view, int position, long id) {
                            indexdf = spinner_df.getSelectedItemPosi-
tion();
                        }

                        @Override
                        public void onNothingSelected(AdapterView<?>
parent) {

                        }
```

```
                    });
                }else if (indexnumeropontos == 2) {
                    ArrayAdapter<CharSequence> adapterdf = ArrayAdap-
ter.createFromResource(getApplicationContext(),
                            R.array.df_125, android.R.layout.sim-
ple_spinner_item);


                    adapterdf.setDropDownViewResource(an-
droid.R.layout.simple_spinner_dropdown_item);


                    spinner_df.setAdapter(adapterdf);
                    spinner_df.setOnItemSelectedListener(new Adapter-
View.OnItemSelectedListener() {
                        @Override
                        public void onItemSelected(AdapterView<?> pa-
rent, View view, int position, long id) {
                            indexdf = spinner_df.getSelectedItemPosi-
tion();
                        }

                        @Override
                        public void onNothingSelected(AdapterView<?>
parent) {

                        }
                    });
                }else if (indexnumeropontos == 3) {
                    ArrayAdapter<CharSequence> adapterdf = ArrayAdap-
ter.createFromResource(getApplicationContext(),
                            R.array.df_62_5, android.R.layout.sim-
ple_spinner_item);


                    adapterdf.setDropDownViewResource(an-
droid.R.layout.simple_spinner_dropdown_item);


                    spinner_df.setAdapter(adapterdf);
                    spinner_df.setOnItemSelectedListener(new Adapter-
View.OnItemSelectedListener() {
                        @Override
                        public void onItemSelected(AdapterView<?> pa-
rent, View view, int position, long id) {
                            indexdf = spinner_df.getSelectedItemPosi-
tion();
                        }

                        @Override
                        public void onNothingSelected(AdapterView<?>
parent) {

                        }
                    });
                }else if (indexnumeropontos == 4) {
                    ArrayAdapter<CharSequence> adapterdf = ArrayAdap-
ter.createFromResource(getApplicationContext(),
```

```
                            R.array.df_32_25, android.R.layout.sim-
ple_spinner_item);


                    adapterdf.setDropDownViewResource(an-
droid.R.layout.simple_spinner_dropdown_item);


                    spinner_df.setAdapter(adapterdf);
                    spinner_df.setOnItemSelectedListener(new Adapter-
View.OnItemSelectedListener() {
                        @Override
                        public void onItemSelected(AdapterView<?> pa-
rent, View view, int position, long id) {
                            indexdf = spinner_df.getSelectedItemPosi-
tion();
                        }

                        @Override
                        public void onNothingSelected(AdapterView<?>
parent) {

                        }
                    });
                }
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

    }



    public void next(View view) {
        switch (indexnumeropontos){
            case 0:
                dInt = 1;
                break;
            case 1:
                dInt = 2;
                break;
            case 2:
                dInt = 4;
                break;
            case 3:
                dInt = 8;
                break;
            case 4:
                dInt = 16;
                break;
        }

        switch (indexdf){
            case 0:
```

```
                numpt = 512;
                break;
            case 1:
                numpt = 1024;
                break;
            case 2:
                numpt = 2048;
                break;
            case 3:
                numpt = 4096;
                break;
            case 4:
                numpt = 8192;
                break;
            case 5:
                numpt = 16384;
                break;
        }
        Intent intent = new Intent(Menu_Diagnostico.this , Medicao_Di-
agnostico.class);
        intent.putExtra("numpt" , numpt);
        intent.putExtra("dInt" , dInt);
        startActivity(intent);
    }
}
```

Apêndice L – Código de aquisição e análise de dados do sensor para diagnóstico

```
package com.example.app1.projeto_final_1.Diagnóstico;

import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.graphics.RectF;
import android.graphics.Typeface;
import android.os.Bundle;
import android.util.Log;
import android.widget.CheckBox;
import com.example.app1.projeto_final_1.R;
import com.github.mikephil.charting.charts.BarChart;
import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.components.Legend;
import com.github.mikephil.charting.components.XAxis;
import com.github.mikephil.charting.components.YAxis;
import com.github.mikephil.charting.data.BarData;
import com.github.mikephil.charting.data.BarDataSet;
import com.github.mikephil.charting.data.BarEntry;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.github.mikephil.charting.highlight.Highlight;
import com.github.mikephil.charting.interfaces.datasets.IBarDataSet;
import com.github.mikephil.charting.interfaces.datasets.ILineDataSet;
import com.github.mikephil.charting.listener.OnChartValueSelectedLis-
tener;
import com.github.mikephil.charting.utils.ColorTemplate;
import com.github.mikephil.charting.utils.MPPointF;
import com.phidget22.Accelerometer;
import com.phidget22.AccelerometerAccelerationChangeEvent;
import com.phidget22.AccelerometerAccelerationChangeListener;
import com.phidget22.AttachEvent;
import com.phidget22.AttachListener;
import com.phidget22.DetachEvent;
import com.phidget22.DetachListener;
import com.phidget22.Phidget;
import com.phidget22.PhidgetException;
import java.util.ArrayList;

import static android.graphics.Typeface.DEFAULT;

public class Medicao_Diagnostico extends Menu_Diagnostico implements
OnChartValueSelectedListener {

    public Accelerometer accel;
    public LineChart grafico;
    public double[] acceldata;
    public int counter = 0;
    public CheckBox aquisicao;
    public BarChart graficofft;
    public Complex[] accelcomplex;
    public double [] fftaccel;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_medicao__diagnostico);

    grafico = findViewById(R.id.grafico);
    graficofft = findViewById(R.id.graficofft);
    aquisicao = findViewById(R.id.aquisicao);

    aquisicao.setEnabled(false);


    Intent intent = getIntent();
    numpt = intent.getIntExtra("numpt" , 0);
    dInt = intent.getIntExtra("dInt" , 0);
    acceldata = new double[numpt];
    accelcomplex = new Complex[numpt/2];
    fftaccel = new double[numpt/2];


    //grafico accel definitions
    grafico.setOnChartValueSelectedListener(this);
    grafico.getDescription().setEnabled(true);
    grafico.setTouchEnabled(true);
    grafico.setDragEnabled(true);
    grafico.setScaleEnabled(true);
    grafico.setDrawGridBackground(false);

    grafico.setPinchZoom(true);
    grafico.setBackgroundColor(Color.LTGRAY);

    LineData acceldata = new LineData();
    acceldata.setValueTextColor(Color.BLACK);

    grafico.setData(acceldata);

    Legend l = grafico.getLegend();
    l.setForm(Legend.LegendForm.LINE);
    l.setTypeface(DEFAULT);
    l.setTextColor(Color.WHITE);

    MyValueFormatter x1formatter = new MyValueFormatter("s");

    XAxis x1 = grafico.getXAxis();
    x1.setTypeface(DEFAULT);
    x1.setTextColor(Color.WHITE);
    x1.setDrawGridLines(false);
    x1.setValueFormatter(x1formatter);
    x1.setAvoidFirstLastClipping(true);
    x1.setEnabled(true);
    x1.setPosition(XAxis.XAxisPosition.BOTTOM);

    MyValueFormatter y1formatter = new MyValueFormatter("m/s^2");

    YAxis y1 = grafico.getAxisLeft();
    y1.setTypeface(DEFAULT);
    y1.setTextColor(Color.WHITE);
```

```
        y1.setValueFormatter(y1formatter);
        y1.setAxisMinimum(0f);
        y1.setAxisMaximum(20f);
        y1.setDrawGridLines(false);

        YAxis yright = grafico.getAxisRight();
        yright.setEnabled(false);

        //--------------------------------------------------------------
---------------------------
        //grafico fft definitions

        graficofft.getDescription().setEnabled(false);


        graficofft.setPinchZoom(true);

        graficofft.setDrawBarShadow(false);
        graficofft.setDrawGridBackground(false);

        MyValueFormatter xformatter = new MyValueFormatter("Hz");

        XAxis xAxis = graficofft.getXAxis();
        xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
        xAxis.setValueFormatter(xformatter);
        xAxis.setTypeface(DEFAULT);
        xAxis.setDrawGridLines(false);
        xAxis.setGranularity(0.1f);

        MyValueFormatter yformatter = new MyValueFormatter("m/s^2");

        YAxis leftAxis = graficofft.getAxisLeft();
        leftAxis.setTypeface(DEFAULT);
        leftAxis.setValueFormatter(yformatter);
        leftAxis.setLabelCount(8, false);
        leftAxis.setPosition(YAxis.YAxisLabelPosition.OUTSIDE_CHART);
        leftAxis.setSpaceTop(15f);
        leftAxis.setAxisMinimum(0f);
        leftAxis.setAxisMaximum(15f);

        YAxis rightAxis = graficofft.getAxisRight();
        rightAxis.setEnabled(false);

        Legend l1 = graficofft.getLegend();
        l1.setVerticalAlignment(Legend.LegendVertica-
lAlignment.BOTTOM);
        l1.setHorizontalAlignment(Legend.LegendHorizonta-
lAlignment.LEFT);
        l1.setOrientation(Legend.LegendOrientation.HORIZONTAL);
        l1.setDrawInside(false);
        l1.setForm(Legend.LegendForm.SQUARE);
        l1.setFormSize(9f);
        l1.setTextSize(11f);
        l1.setXEntrySpace(4f);


        XYMarkerView mv = new XYMarkerView(this, xformatter);
        mv.setChartView(graficofft); // For bounds control
```

```
        graficofft.setMarker(mv); // Set the marker to the chart


        try {
            accel = new Accelerometer();
            if (getPackageManager().hasSystemFeature(PackageMana-
ger.FEATURE_USB_HOST))
                com.phidget22.usb.Manager.Initialize(this);


            accel.addAttachListener(new AttachListener() {
                public void onAttach(final AttachEvent attachEvent) {
                    AttachEventHandler handler1 = new AttachEventHand-
ler(accel);
                    synchronized (handler1) {
                        runOnUiThread(handler1);
                        try {
                            handler1.wait();
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
            });

            accel.addDetachListener(new DetachListener() {
                public void onDetach(final DetachEvent detachEvent) {
                    DetachEventHandler handler1 = new DetachEventHand-
ler(accel);
                    synchronized (handler1) {
                        runOnUiThread(handler1);
                        try {
                            handler1.wait();
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
            });

            accel.addAccelerationChangeListener(new AccelerometerAcce-
lerationChangeListener() {
                @Override
                public void onAccelerationChange(AccelerometerAccele-
rationChangeEvent accelerationChangeEvent) {
                    AccelerometerAccelerationChangeEventHandler hand-
ler1 = new AccelerometerAccelerationChangeEventHandler(accel, accele-
rationChangeEvent);
                    runOnUiThread(handler1);
                }
            });


            accel.open();


        } catch (PhidgetException pe) {
            pe.printStackTrace();
```

```
        }
    }

    public void dataIntervalChangeListener() {
        try {

            accel.setDataInterval(dInt);

        } catch (PhidgetException e) {
            e.printStackTrace();
        }
    }

    class AttachEventHandler implements Runnable {
        Phidget accel;


        public AttachEventHandler(Phidget accel) {
            this.accel = accel;
        }

        public void run() {

            aquisicao.setEnabled(true);


            synchronized (this) {
                this.notify();
            }

        }

    }

    class DetachEventHandler implements Runnable {
        Phidget accel;

        public DetachEventHandler(Phidget accel) {
            this.accel = accel;
        }

        @Override
        public void run() {
            aquisicao.setEnabled(false);

            synchronized (this) {
                this.notify();
            }
        }
    }

    class AccelerometerAccelerationChangeEventHandler implements Run-
nable {

        Phidget accel;
        AccelerometerAccelerationChangeEvent accelerationChangeEvent;
```

```
        public AccelerometerAccelerationChangeEventHandler(Phidget ac-
cel, AccelerometerAccelerationChangeEvent accelerationChangeEvent) {
            this.accel = accel;
            this.accelerationChangeEvent = accelerationChangeEvent;


        }


        @Override
        public void run() {

            if (aquisicao.isChecked()){
                dataIntervalChangeListener();


                acceldata[counter]=accelerationChangeEvent.getAccele-
ration()[2] * 9.81;
                addEntry(acceldata);
                counter=counter+1;

                int m = (int) (Math.log(acceldata.len-
gth)/Math.log(2));
                m = m - 1;


                if (counter == numpt) {
                    aquisicao.setChecked(false);
                    counter = 0;



                    fft(acceldata.length, m, acceldata, accelcomplex);




                    for (int i = 1; i < accelcomplex.length; i++) {
                        fftaccel[i] = accelcomplex[i].abs();

                    }


                    setData(fftaccel.length,fftaccel);

                }
            }

        }

        public  void fft (int n , int m , double[] x , Complex[] xx){
            int k , i , k1 ,k2 , Nf, j , j1 , j2 , e1 , e2 , o1 , o2,
n1, nk, ei, ej, N2;
            double th   , ag , wr , wi ,  dr , di, t, ar, ai, br, bi,
s, c, cr, ci ;

            k2 = 1;
            Nf=n/2;
```

```
for ( k = 1; k<=m ; k++){
    k2 = 2*k2;
    k1=k2/2;
    th=2*(Math.PI/Nf)*k1;
    nk = Nf/k2;
    n1=Nf/k1;
    for ( i = 1 ; i<=nk ; i++){
        ag = (i - 1)*th;
        wr = Math.cos(ag);
        wi = - Math.sin(ag);
        for (j=n1 ; j<=Nf ; j=j + n1){
            j1 = j - n1 + i - 1;
            j2 = j1 + nk;
            e1 = 2*j1;
            e2 = 2*j2;
            o1 = e1 + 1;
            o2 = e2 + 1;
            dr = x[e1] - x[e2];
            di = x[o1] - x[o2];
            x[e1] = x[e1] + x[e2];
            x[o1] = x[o1] + x[o2];
            x[e2] = wr*dr - wi*di;
            x[o2] = wi*dr + wr*di;
        }
    }
}


j = 0;
for (i=0 ; i<=Nf - 2; i++){
    if (i<j){
        ei = 2*i;
        ej = 2*j;
        t = x[ei];
        x[ei] = x[ej];
        x[ej]=t;
        t= x[ei + 1];
        x[ei + 1]=x[ej + 1];
        x[ej + 1]=t;
    }
    k=Nf/2;
    while (k<=j){
        j=j - k;
        k=k/2;
    }
    j=j+k;
}



t = x[0]+x[1];
x[1]=x[0]-x[1];
x[0]=t;
x[Nf + 1]=-x[Nf + 1];
N2 = 2*Nf;
for (k=1 ; k<=(Nf/2) - 1; k++){
    k2=2*k;
```

```
                nk=N2-k2;
                ar=x[k2]+x[nk];
                ai=x[k2 + 1] - x[nk + 1];
                br=x[nk]-x[k2];
                bi= -x[nk + 1] - x[k2 + 1];
                s = Math.sin(k*(Math.PI/Nf));
                c = Math.cos(k*(Math.PI/Nf));
                cr = br*s-bi*c;
                ci = br*c+bi*s;
                x[k2]=(ar+cr)/2;
                x[k2 + 1]=(ai+ci)/2;
                x[nk]= (ar-cr)/2;
                x[nk + 1]=(ci-ai)/2;

            }

            for (i=0; i<n ; i=i + 2){
                xx[i / 2] = new Complex(x[i] / Nf, x[i + 1] / Nf);
            }
            xx[0]=new Complex(x[0]/(2*Nf) , x[1]/(2*Nf));


        }


        public void addEntry(double[] x){
            LineData acceldata = grafico.getData();
            if (acceldata != null) {
                ILineDataSet set = acceldata.getDataSetByIndex(0);

                if (set == null) {
                    set = createSet();
                    acceldata.addDataSet(set);
                }


                acceldata.addEntry(new Entry((float) (coun-
ter*dInt*0.001 ), (float) x[counter] ), 0);
                acceldata.notifyDataChanged();

                grafico.notifyDataSetChanged();
                grafico.setVisibleXRangeMaximum((float)(120 * dInt
*0.001));
                grafico.moveViewToX(acceldata.getEntryCount());

            }
        }

        private LineDataSet createSet() {
            LineDataSet set = new LineDataSet(null,"Dados de Acelera-
ção");
            set.setAxisDependency(YAxis.AxisDependency.LEFT);
            set.setColor(ColorTemplate.getHoloBlue());
            set.setCircleColor(Color.WHITE);
            set.setLineWidth(2f);
            set.setCircleRadius(5f);
            set.setFillAlpha(65);
            set.setFillColor(ColorTemplate.getHoloBlue());
```

```
            set.setHighLightColor(Color.YELLOW);
            set.setValueTextColor(Color.WHITE);
            set.setValueTextSize(10f);
            set.setDrawValues(false);
            return set;
        }

        private void setData(int count, double [] fim){
            ArrayList<BarEntry> fftvalues = new ArrayList<>();

            for (int i = 1 ; i < count ; i++ ){
                fftvalues.add(new BarEntry( (float) (i /(dInt * 0.001
* acceldata.length)),(float) fim[i]));
            }

            BarDataSet set1;


            set1 = new BarDataSet(fftvalues , "Spectrum");

            set1.setDrawIcons(false);


            set1.setColor(Color.BLUE);

            ArrayList<IBarDataSet> dataSets = new ArrayList<>();
            dataSets.add(set1);

            BarData data = new BarData(dataSets);

            data.setValueTextSize(10f);
            data.setValueTypeface(Typeface.DEFAULT);
            data.setBarWidth((float) (1/(dInt*0.001*(acceldata.len-
gth+10))));

            graficofft.setData(data);


        }
    }

    public class Complex {
        public final double re;
        public final double im;


        public Complex(double r, double i) {
            re = r;
            im = i;
        }

        public String toString() {
            if (im == 0) return re + "";
            if (re == 0) return im + "i";
            if (im < 0) return re + " - " + (-im) + "i";
            return re + " + " + im + "i";
        }
```

```
        public double abs(){
            return Math.hypot(re , im);
        }
        public Complex conjugate() {
            return new Complex(re, -im);
        }
        public Complex scale(double alpha) {
            return new Complex(alpha * re, alpha * im);
        }
        public Complex times(Complex b) {
            Complex a = this;
            double real = a.re * b.re - a.im * b.im;
            double imag = a.re * b.im + a.im * b.re;
            return new Complex(real, imag);
        }

        public Complex plus(Complex b) {
            Complex a = this;
            double real = a.re + b.re;
            double imag = a.im + b.im;
            return new Complex(real, imag);
        }

        public Complex minus(Complex b) {
            Complex a = this;
            double real = a.re - b.re;
            double imag = a.im - b.im;
            return new Complex(real, imag);
        }

    }


    @Override
    protected void onDestroy() {
        super.onDestroy();


        try {

            if (getPackageManager().hasSystemFeature(PackageMana-
ger.FEATURE_USB_HOST))
                com.phidget22.usb.Manager.Uninitialize();


            accel.removeAttachListener( new AttachListener() {
                public void onAttach(final AttachEvent attachEvent) {
                    AttachEventHandler handler1 = new AttachEventHand-
ler(accel);
                    synchronized (handler1) {
                        runOnUiThread(handler1);
                        try {
                            handler1.wait();
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
```

```
            });

            accel.removeDetachListener(new DetachListener() {
                public void onDetach(final DetachEvent detachEvent) {
                    DetachEventHandler handler1 = new DetachEventHand-
ler(accel);
                    synchronized (handler1) {
                        runOnUiThread(handler1);
                        try {
                            handler1.wait();
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
            });

            accel.removeAccelerationChangeListener(new Accelerome-
rAccelerationChangeListener() {
                @Override
                public void onAccelerationChange(AccelerometerAccele-
rationChangeEvent accelerationChangeEvent) {
                    AccelerometerAccelerationChangeEventHandler hand-
ler1 = new AccelerometerAccelerationChangeEventHandler(accel, accele-
rationChangeEvent);
                    runOnUiThread(handler1);
                }
            });

            accel.close();


        } catch (PhidgetException pe) {
            pe.printStackTrace();
        }
    }

    private final RectF onValueSelectedRectF = new RectF();

    @Override
    public void onValueSelected(Entry e, Highlight h) {

        if (e == null)
            return;

        RectF bounds = onValueSelectedRectF;
        graficofft.getBarBounds((BarEntry) e, bounds);
        MPPointF position = graficofft.getPosition(e, YAxis.AxisDepen-
dency.LEFT);

        Log.i("bounds", bounds.toString());
        Log.i("position", position.toString());

        Log.i("x-index",
                "low: " + graficofft.getLowestVisibleX() + ", high: "
                        + graficofft.getHighestVisibleX());

        MPPointF.recycleInstance(position);
```

```
    }

    @Override
    public void onNothingSelected() {

    }
}
```