Universidade de Aveiro Departamento de Engenharia Mecânica
2019

Bruno Alexandre Pereira Nunes

Raspberry Pi based Spectral Analyzer

Analisador espetral baseado numa solução Raspberry Pi

Universidade de Aveiro Departamento de Engenharia Mecânica
2019

Bruno Alexandre Pereira Nunes

Raspberry Pi based Spectral Analyzer

Analisador espetral baseado numa solução Raspberry Pi

Relatório de projecto apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob orientação científica de Rui António da Silva Moreira, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro e de José Paulo Oliveira Santos, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro.

o júri / the jury

Presidente / President	**Prof. Doutor Joaquim Alexandre Mendes de Pinho da Cruz** Professor Auxiliar da Universidade de Aveiro
Vogais / Committee	**Prof. Doutor Paulo Bacelar Reis Pedreiras** Professor Auxiliar da Universidade de Aveiro
	Prof. Doutor Rui António da Silva Moreira Professor Auxiliar da Universidade de Aveiro (orientador)

agradecimentos / acknowledgements

Em primeiro lugar agradeço à minha família, com especial atenção aos meus pais, por todo o apoio prestado, no meu percurso académico bem como em toda a minha vida.

Agradeço também aos meus orientadores pela ajuda, orientação e acompanhamento durante a realização deste projecto.

keywords Spectral analyzer, Raspberry Pi, vibration analysis.

abstract With the technological evolution, more and more low-cost and highly flexible development solutions are appearing on the market, making them extremely interesting tools to be used in a multitude of projects. Two of this devices, a Raspberry Pi and a audio card, were chosen in order to undertake the main objective of this project, which is the development of a low-cost spectrum analyzer to be used in vibration analysis. The main purpose of the created solution is to tackle the already commercial avaible devices that are excessively expensive and have little flexibility. The data acquisition is done using the audio card and all the signal analyzer algorithms are run by the Raspberry Pi, being both controlled through a graphic interface also developed within this project. After conducting a few tests it was possible to perceive the capabilities and limitations of the solution created, as well as some possible problems. The solution showed promising despite some of its limitations.

palavras-chave Analisador espectral, Raspberry Pi, vibrações mecânicas.

resumo Com a evolução tecnológica, estão constantemente a aparecer soluções de desenvolvimento altamente flexíveis e de baixo custo no mercado, sendo estas ferramentas extremamente interessantes para serem utilizadas numa infinidade de projectos. Duas dessas ferramentas foram escolhidas – um Raspberry Pi e uma placa de áudio – por foma a cumprir o objectivo principal deste projecto, que é o desenvolvimento de um analisador de espectro de baixo custo para ser utilizado na análise de vibrações. Um dos objectivos da solução criada é concorrer com as ferramentas disponíveis no mercado, que são excessivamente onerosas e possuem pouca flexibilidade. A aquisição de dados é feita usando a placa de áudio e todos os algoritmos implementados são executados pelo Raspberry Pi. De notar que ambos os dispositivos são controlados através de uma interface gráfica, também desenvolvida no âmbito do projecto apresentado. Após a realização de alguns testes, foi possível perceber as capacidades e as limitações da solução criada, bem como alguns possíveis problemas. A solução desenvolvida mostrou-se promissora, apesar de algumas limitações.

Contents

List of Tables

Intentionally blank page.

List of Figures

Acronyms

ADC Analog-to-Digital Converter

CPU Central Processing Unit

DAQ Data Acquisition

DFT Discrete Fourier Transform

DSP Digital Signal Processing

EQ Equalization

FFT Fast Fourier Transform

FRF Frequency-Response Function

GPIO General Purpose Input/Output

HDMI High-Definition Multimedia Interface

ICP Integrated Circuit Piezoelectric

IEPE Integrated Electronics Piezo-Electric

I/O Input/Output

LSB Least Significant Bit

PLL Phase-Lock Loop

PSD Power Spectral Density

RAM Random-Access Memory

SNR Signal-to-Noise Ratio

S/PDIF Sony/Philips Digital Interface

TRS Tip Ring Sleeve

USB Universal Serial Bus

VCO Voltage-Controlled Oscillator

Intentionally blank page.

Chapter 1

Introduction

1.1 Context

The potentialities of the development platforms, nowadays available in the market, make them into tools with high flexibility and low cost, allowing for the creation of computation and analysis solutions. On the other hand, the currently available acoustic and vibration analyzers are solutions with very little flexibility and extremely costly. Hence the idea of this project, the creation of a low cost and flexible solution for vibration analysis, using commercially available platforms, such as the Raspberry Pi. The solution should be capable of acquiring analog signal from different sensors used in impact testing.

1.2 Objectives

The aim of this project is to develop a spectral analyzer based on Raspberry Pi, to be used for vibration analysis. To do so, the Raspberry Pi should be paired with an audio card, allowing it to acquire analog signals.

For the signal acquisition, basic acquisition routines should be implemented, as well as the windowing functions, spectral functions and frequency-response functions. There should also be a frontend, allowing the user access to most of the tools used in vibration analysis.

1.3 Document outline

This project report is divided into 6 chapters, making it much easier to understand the work developed. In the following points there is a small summary of each one of this chapters:

- Chapter 1:

  The first chapter serves as an introduction to the theme and to the structure of this document, focusing on the motives and objectives of this project.

- Chapter 2:

  In this chapter there is a rundown of the necessary concepts to understand the project. This includes topics from vibration analysis, digital signal processing and specifications of the hardware used in vibration analysis.

- Chapter 3:

  In this chapter the main principles of the FFT spectrum analyzer are show, like the different types and the basic basic theory subjacent to them.

- Chapter 4:

  In this chapter the implementation of the solution developed is described, from the hardware components up to the software developed.

- Chapter 5:

  In this chapter the solution is subject to different tests in order to gain insight on its capabilities, and some initial conclusions are made.

- Chapter 6:

  Finaly, in this chapter a final conclusion of the work that has been done will be given, as well as some future work suggestions.

Chapter 2

Background theory

2.1 Vibration

The study of vibration is focused on the oscillatory motions of bodies and the forces associated with them. All bodies are capable of vibration, as long as they have mass and elasticity. Thus, being very important to take in consideration the oscillatory behavior of most engineering machines and structures during their design.

Oscillatory systems can be usually described as *linear* or *nonlinear*. The linear systems can be characterized by well defined mathematical equations and models unlike nonlinear systems.

Whenever an object is disturbed from its equilibrium position, a force will act up on it to restore it back to the initial position. If the object overshoots the equilibrium position and starts oscillating, the object is said to be vibrating. There are two main classes of vibrations, the free and the forced. A system is under free vibration when external impressed forces are not present, and the system oscillates only under the action of forces inherent in the system itself. This system will vibrate at one or more *natural frequencies*, which are properties determined by its mass and stiffness distribution.

It is called forced vibration when the system vibrates under the excitation of external forces. If the excitation is oscillatory the system vibrates at the excitation frequency. Extreme oscillations may occur when a condition called *resonance* is encountered. This happens when the frequency of excitation coincides with one of the natural frequencies of the system.

A oscillatory movement that keeps repeating itself after a certain interval of time is called periodic motion. This interval is referred to as the period of the vibration, T. The frequency of a vibration, is equal to the inverse of the period, $f = \frac{1}{T}$, and usually is represented in Hertz (Hz). Another characteristic is the amplitude, which represents the maximum displacement. A harmonic function is the simplest type of periodic motion as shown in Figure 2.1.

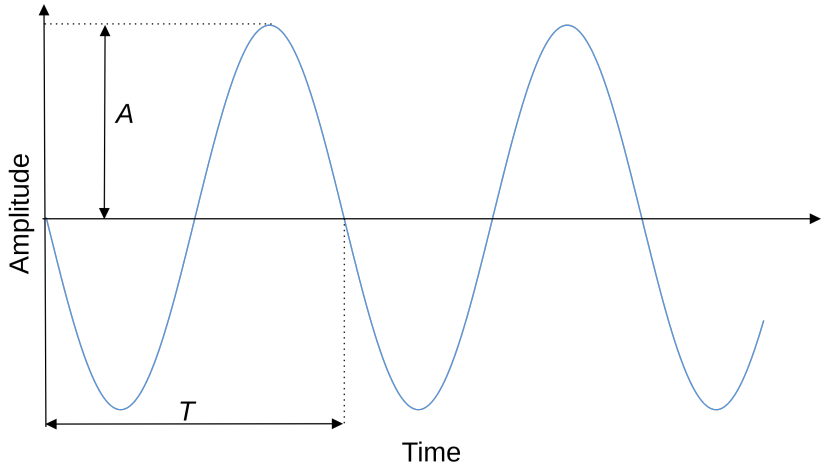


Figure 2.1: Harmonic function

A harmonic function and all of its characteristics can be expressed by the following equation:

$$x(t) = A\sin(\omega t) \tag{2.1}$$

where

x = displacement response

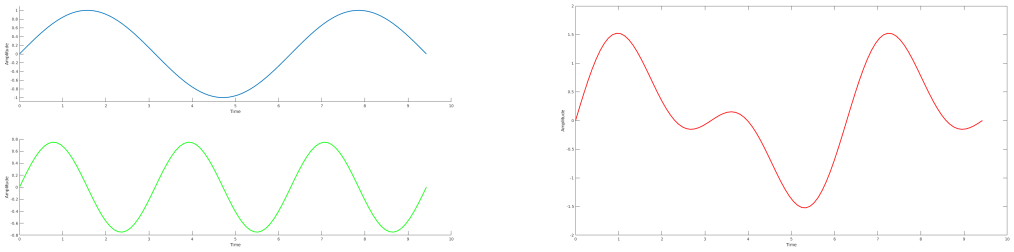
A = amplitude

ω = angular frequency

t = time

The angular frequency of a sinusoid is represented by $\omega = 2\pi f$ $(\mathrm{rad\,s^{-1}})$.

As opposed to the simple theoretical vibration curve shown in Figure 2.1, most data gathered for vibration analysis is extremely complex. This is due to the different sources of vibration, with each source generating its own spectral content, being essentially added and displayed as a composite profile as can be seen in Figure 2.2.



(a) Two sin waves

(b) Complex waveform

Figure 2.2: Simple and complex waveforms.

-

2.2 Fourier Series

As previously stated, it is frequent for vibrations of different frequencies to exist simultaneously, like for example, the vibration of a violin string, which is composed of the fundamental frequency f and all its harmonics, *2f*, *3f*, and so forth [2].

Jean Fourier [11] demonstrated that any periodic function can be represented by a series of sines and cosines as represented in Equation (2.2).

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n cos(nx) + \sum_{n=1}^{\infty} b_n sin(nx) \tag{2.2}$$

where the coefficients a_0, a_n, and b_n are related to $f(x)$ by definite integrals:

$$a_n = \frac{1}{\pi}\int_0^{2\pi} f(x)cos(nx)dx, \quad n = 0, 1, 2, ...,$$

$$b_n = \frac{1}{\pi}\int_0^{2\pi} f(x)sin(nx)dx, \quad n = 1, 2, ...,$$

which are dependent to the requirement that the integrals exist. To make Equation (2.2) valid, it is imposed that $f(x)$ have only a finite number of discontinuities and only a finite number of extreme values in the correspondent interval. These conditions are sufficient but not necessary, being known as Dirichlet conditions [12].

Equation (2.2) can be rewriten if the terms $cos(nx)$ and $sin(nx)$ are represented in the exponential form:

$$cos(nx) = \frac{1}{2}(e^{inx} + e^{-inx})$$

$$sin(nx) = -\frac{1}{2}(e^{inx} - e^{-inx})$$

So, substituting the terms above in Equation (2.2), the following is obtained

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx}, \tag{2.3}$$

in which

$$c_n = \frac{1}{2}(a_n - ib_n), \quad c_{-n} = \frac{1}{2}(a_n + ib_n), \quad n > 0,$$

and

$$c_0 = \frac{1}{2}a_0.$$

When the Fourier series coefficients are plotted against frequency, a series of discrete lines are obtained, this is called the *Fourier spectrum*. Usually the values that are plotted are the absolute ones $|2c_n| = \sqrt{a_n^2 + b_n^2}$ and the phase $\phi_n = \tan^{-1}\left(\frac{b_n}{a_n}\right)$, this can be seen in Figure 2.3.

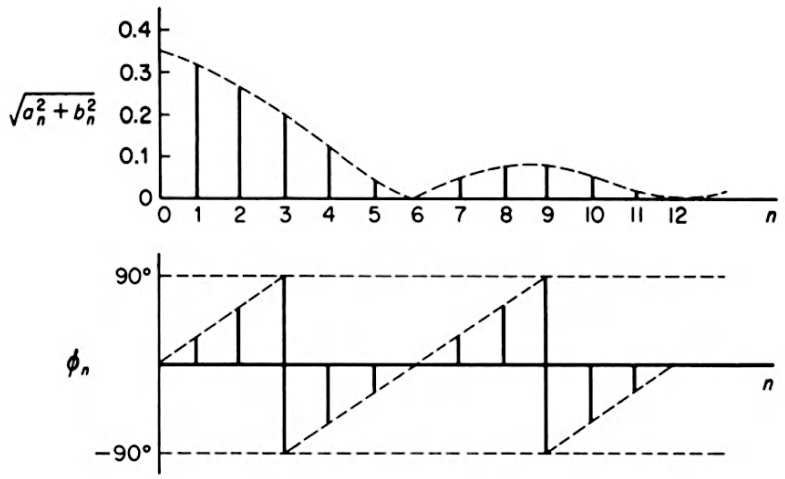


Figure 2.3: Example of Fourier spectrum of a wave [2].

Harmonic analysis is easily and efficiently carried out thanks to the FFT algorithm [2].

2.3 Fourier transform

The Fourier transform is represented by the following equation:

$$g(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(t) e^{i\omega t} dt \tag{2.4}$$

If the exponential part in Equation (2.4) is rewriten in terms of the sine and cosine then, taking in consideration the restrictions to the function assuming x to either be even or odd, the following integral transforms is obtained:

$$g_c(\omega) = \sqrt{\frac{2}{\pi}} \int_0^{\infty} f(t) cos(\omega t) dt \tag{2.5}$$

$$g_s(\omega) = \sqrt{\frac{2}{\pi}} \int_0^{\infty} f(t) sin(\omega t) dt \tag{2.6}$$

These formulas are, respectively, the Fourier cosine and Fourier sine transforms. They are really useful for extracting information from waves, especially when phase is involved [12].

The equations above are of extreme importance in Digital Signal Processing (DSP) and vibration analysis, particularly Equation (2.4) since it is the base of the FFT algorithm.

There are four types of Fourier transforms, and they are divided accordingly to the signal type being study as described in Figure 2.4. All of these four classes of signals extend to positive and negative infinity. A signal can be either *periodic* or *aperiodic*, if it repeats itself after a period or not, respectively. And it can be *continuous* or *discrete*. So, with the combination of these two main categories the aperiodic-continuous is obtained, which include, for example, the Gaussian curve. The Fourier transform for this type of signal is called the Fourier transform. Then there are the periodic-continuous, like sine waves and square waves, that the transform is referred to as Fourier

series. After the continuous signals, there are the discrete ones. The aperiodic-discrete are signals defined only at discrete points, between positive and negative infinity, and do not repeat themselves. The Fourier transform for these signals is called discrete time Fourier transform. And finally, there are the periodic-discrete signals, which, by opposition to the ones before, do repeat themselves in a periodic fashion. The Fourier transform for this type is often referred to as DFT [3].

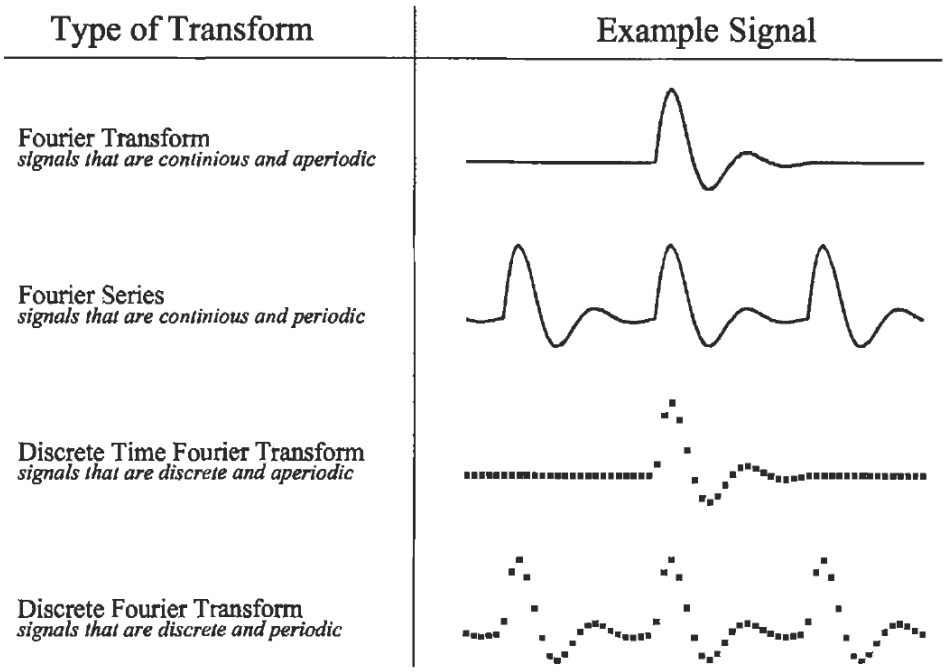


Figure 2.4: Illustration of the four types of Fourier transforms [3].

So, since it is required an infinity number of sinusoids to synthesize a signal that is aperiodic, this makes it impossible to compute the discrete time Fourier transform using a computer algorithm. And since computer only run discrete signals, the only type of Fourier transform used in DSP is the DFT [3].

2.4 DFT

The DFT changes an N point input signal into two $N/2+1$ point output signals. The input signal is the one to be decomposed, while the two output signals contain the amplitudes of the component sine and cosine waves, as it is observed in Figure 2.5. The input signal is in the time domain and it consists of N points going from 0 to $N-1$. Contrarily, in the frequency domain, two signals are produce by the DFT. The real part, $ReX[\,]$, and the imaginary part, $ImX[\,]$.

The DFT has two forms, the Forward DFT (also known as analysis/decomposition) and the Inverse DFT (also known as synthesis). The Forward DFT transforms the signal in the time domain to the frequency domain. On the other hand, the Inverse DFT transforms from the frequency domain to the time domain. This is possible because, even though they are different domains, they contain exactly the same information.

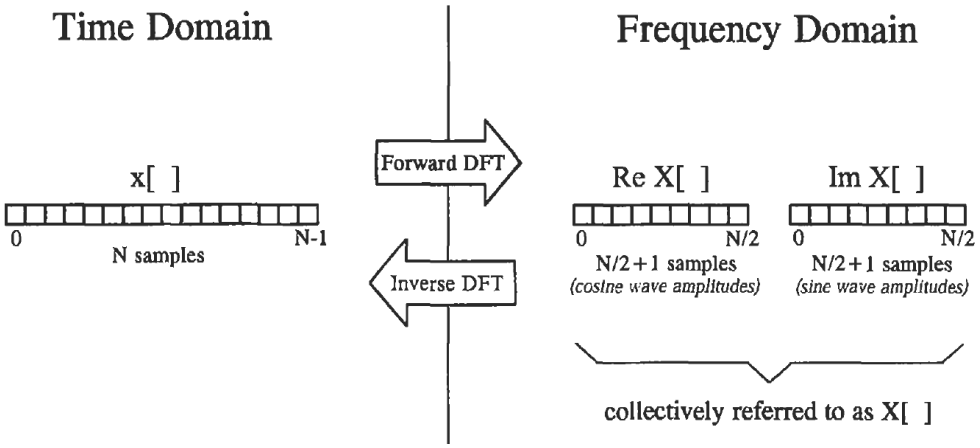


Figure 2.5: The real DFT terminology [3].

The equations for the analysis and the synthesis are the following:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad 0 \leq k \leq N-1 \tag{2.7}$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn}, \quad 0 \leq n \leq N-1 \tag{2.8}$$

Taking in consideration the Euler's formula, $e^{jx} = cos(x) + jsin(x)$, and applying it to W_N^{kn}, we have $W_N^{kn} = cos(2\pi kn/N) - jsin(2\pi kn/N)$ [4].

So, substituting in Equation (2.8) with Euler's formula and the real and imaginary notation mention before, the following representation of the synthesis appears:

$$x[i] = \sum_{k=0}^{N/2} Re\bar{X}[k] cos(2\pi ki/N) + \sum_{k=0}^{N/2} Im\bar{X}[k] sin(2\pi ki/N) \tag{2.9}$$

where

$$Re\bar{X}[k] = \frac{ReX[k]}{N/2}$$

$$Im\bar{X}[k] = -\frac{ImX[k]}{N/2}$$

This is because the amplitudes needed for synthesis are slightly different from the frequency domain of a signal, hence the different notation. This is true except for the following special cases:

$$Re\bar{X}[0] = \frac{ReX[0]}{N}$$

$$Re\bar{X}[N/2] = \frac{ReX[N/2]}{N}$$

Something similar could also be done for the analysis equation [3].

2.5 FFT

The FFT is based upon the complex DFT algorithm, which is a more elaborate version of the one mention in Section 2.4, being capable of computing the DFT of a finite-duration sequence [4]. The responsibility for bringing the FFT to the world is given to J.W.Cooley and J.W.Tukey [13], although others had discovered this technique many years before.

As said above, the FFT algorithm uses the Complex DFT instead of the Real DFT. As can be seen in Figure 2.6, the real DFT takes an N point signal, in the time domain, and creates two $N/2+1$ point signals, in the frequency domain. On the other hand, the complex DFT takes two N point time domain signals, and returns two N point signals in the frequency domain. The squares that are cross-hatched represent the values that

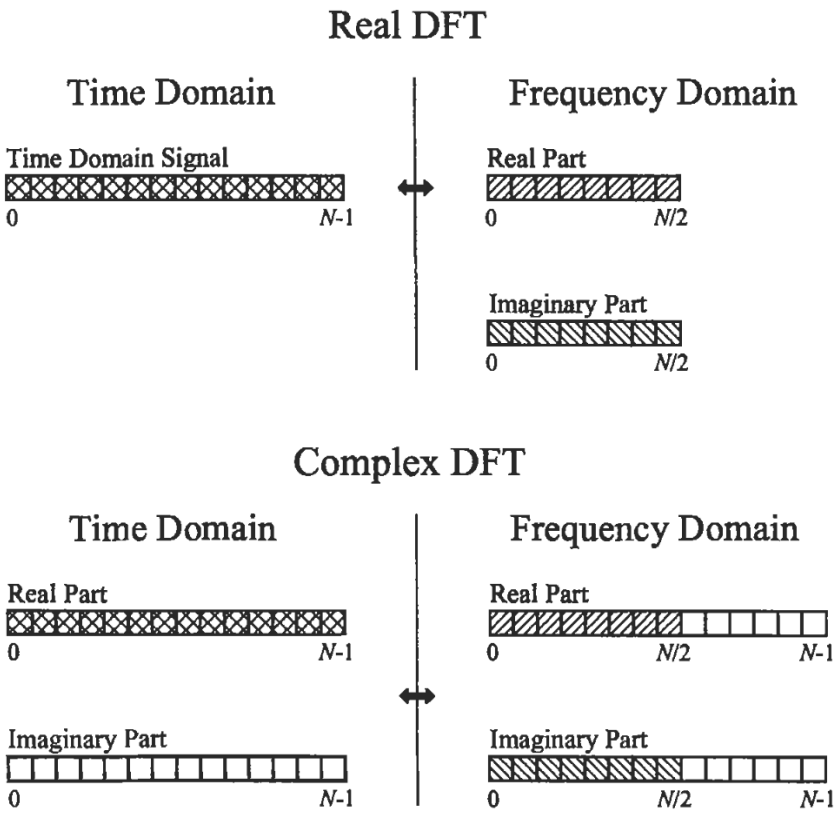


Figure 2.6: Comparison of the real and complex DFT [3].

are common to the two transforms [3].

In the complex notation, a signal is composed of N complex points, and each of these points is made up of two numbers, one the real part and the other the imaginary part. In this notation, the time and the frequency domains each contain one of these signals. So for example, the complex sample $X[13]$ is the combination of $ReX[13]$, the real part, and $ImX[13]$, the imaginary part.

The FFT algorithm consists of three steps, as shown in Figure 2.7: the decomposition of an N point time domain signal into N signals each consisting of a single point; the calculus of the spectrum of each point; and the synthesis of a single frequency spectrum,

taking as input the N frequency spectra [3].

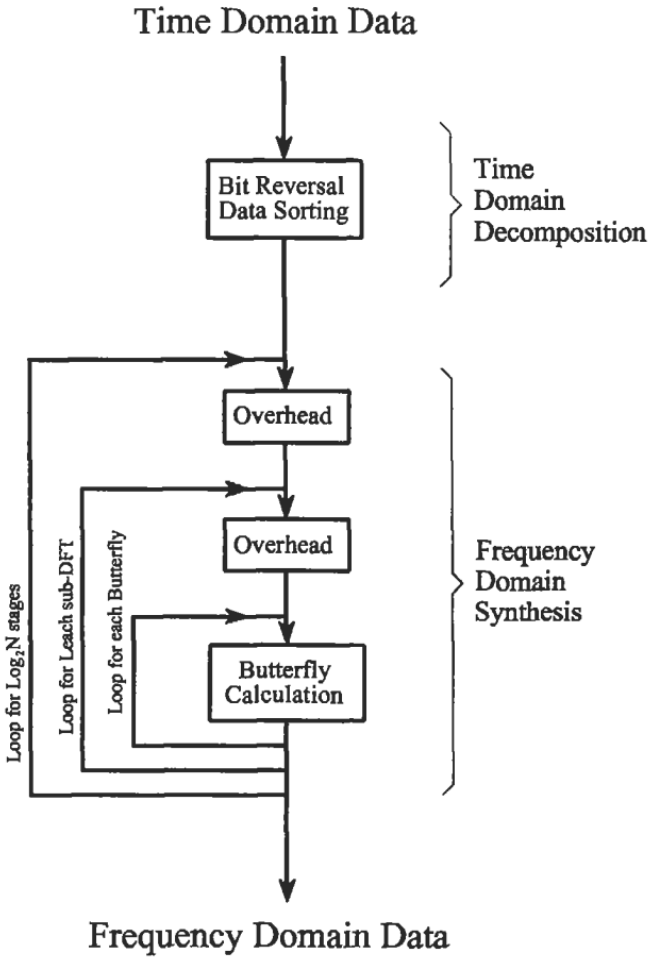


Figure 2.7: Flow diagram of the FFT [3].

2.6 ADC

An ADC is a system that converts an analog (continuous time) signal into a digital signal, the latter one being a sequence of finite-precision samples. This converter is an electronic device that converts an input voltage or current amplitude into a digital number (binary code) representing a quantized value closest to the one in the input. An ADC is usually paired with an external clock, making it possible to start and complete a conversion every T seconds. Nonetheless, since the conversion is not instantaneous, a high-performance ADC system generally includes a sample-and-hold buffer. The sole purpose of the sample-and-hold buffer is to maintain the sample value until it is quantized by the A/D converter [4].

An ADC is mainly characterized by its bandwidth and Signal-to-Noise Ratio (SNR). The bandwidth is primarily defined by its sampling rate. On the other hand, the SNR is determined by many factors, such as the resolution, linearity, accuracy, jitter and

aliasing. The resolution of the converter indicates, over the range of analog values at the input, the number of discrete values it can output. It also determines the maximum average SNR for an ideal ADC (without oversampling) and therefore the magnitude of the quantization error.

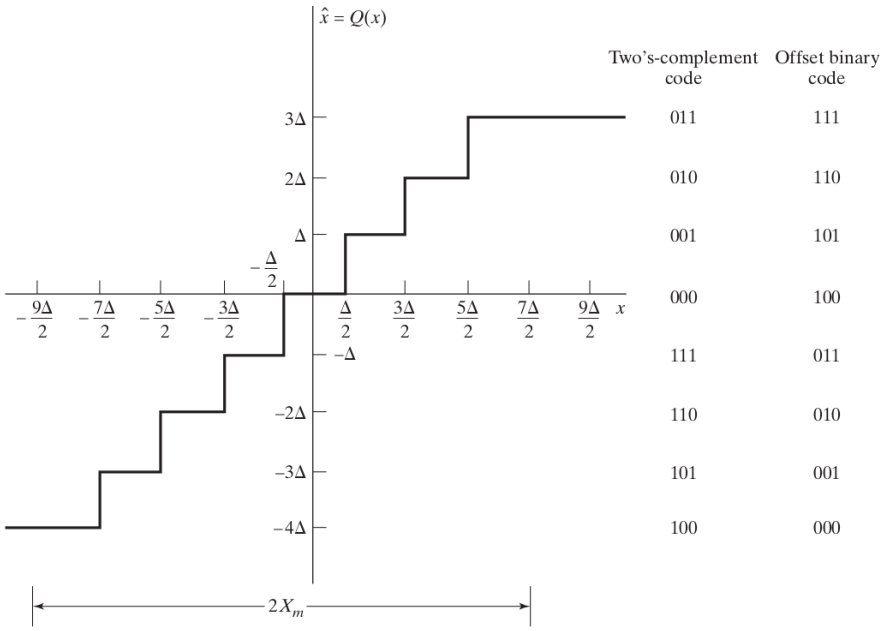


Figure 2.8: Common quantizer used in A/D conversion [4].

From Figure 2.8, it is possible to state that the step size of the quantizer is given by the following equation:

$$\Delta = \frac{2 * X_m}{2^{B+1}} = \frac{X_m}{2^B}, \tag{2.10}$$

with X_m being the full scale voltage range (also known as 'span'), 2^B as the number of voltage intervals and B the ADC's resolution in bits. In Figure 2.8, it can also be seen that it has 8 levels of quantization, and therefore can be labelled by a binary code of 3 bits, as seen on the right side of the Figure 2.8 [4].

One of the effects of quantization is that any sample, in the digitized signal, can have a maximum error of $\pm\frac{1}{2}$ Least Significant Bit (LSB). With that in mind, the digital output is equivalent to the input, plus a quantization error, this being an important feature since it appears like random noise [3].

2.7 Power Spectrum

The power spectrum represents how the power is spread over the frequency range. Even though the major application of the power spectrum is with random signals, it also can be used with periodic waveform [14].

2.8 Power Spectral Density

Power spectra are usually left in its discrete form, normally being the appropriate format for periodic data. On the other hand, for random data, since T is slightly longer and the spacing of the spectral lines ($\frac{1}{T}$ Hz) is much closer, it is common to express the power as a continuous Power Spectral Density (PSD) curve. As seen in Equation (2.11), the PSD, at a given discrete frequency $f_{(n)}$, is given by the division of the power at that frequency by the bandwitdth over which it is distributed δf:

$$S(f_{(n)}) = \frac{\frac{1}{2}(a_n^2 + b_n^2)}{\delta f} = \frac{T}{2}(a_n^2 + b_n^2), \tag{2.11}$$

with $\delta f = \frac{1}{T}$. The units of the PSD are given by the square of the physical quantity per unit of frequency, for example $\frac{V^2}{Hz}$, $\frac{g^2}{Hz}$ and so forth. One major advantage of the PSD is that it is independent of the frequency intervals at which it is computed, making it quite easy to compare plots obtained from different sources or even by different methods [14].

A power spectrum can be represented in terms of the time derivative or integral. So, given a random displacement, $x(t)$, and computing the PSD function to it, the PSD functions for velocity, $\dot{x}(t)$, and acceleration, $\ddot{x}(t)$, are easily found. This also works if the initial signal is either velocity or acceleration. So, taking Equation (2.11), the representation of the PSD for displacement, velocity and acceleration is, respectively:

$$S_x(f_{(n)}) = \frac{T(a_n^2 + b_n^2)}{2} \tag{2.12}$$

$$S_{\dot{x}}(f_{(n)}) = (2\pi f_{(n)})^2 \frac{T(a_n^2 + b_n^2)}{2} = (2\pi f_{(n)})^2 S_x(f_{(n)}) \tag{2.13}$$

$$S_{\ddot{x}}(f_{(n)}) = (2\pi f_{(n)})^4 \frac{T(a_n^2 + b_n^2)}{2} = (2\pi f_{(n)})^4 S_x(f_{(n)}) \tag{2.14}$$

2.9 Cross-Power Spectrum

Altough the cross-power spectrum is not often used on its own, it is incredible important to FFT analyzers, used to compute transfer functions and coherence. The cross-power spectrum, represented by G_{xy}, is defined as taking the PSD of two signals separately and multiplying the result:

$$G_{xy}(f) = S_x(f)S_y^*(f).^1 \tag{2.15}$$

The auto-power spectrum can be obtained by usind the cross-power spectrum function when the input and output signals are the same, $x = y$.

2.10 Correlation

Correlation can be described as the measure of similarity between two signals, usually computed at a specified time offset [8]. Correlation functions are normally used to describe patterns and dynamic behavior of vibrating systems and vibration signals [1].

[1] The * symbol is used to identify the complex conjugate.

The degree of linear dependence between two data sets, x_k and y_k, can be assessed using the concepts of covariance and correlation. The correlation coefficient R is described as:

$$R = \frac{\sum_k x_k y_k}{\left(\sum_k x_k^2\right)^{1/2} \left(\sum_k y_k^2\right)^{1/2}} \tag{2.16}$$

If this notion is extended to time-history data, the covariance function between the time data $x(t)$ and $y(t)$ is defined as:

$$C_{xy}(\tau) = E\left[(x(t) - \mu_x)\,(y(t+\tau) - \mu_y)\right] \tag{2.17}$$

where τ defined the time shift, E referes to the expecations, $\mu_x = E[x(t)]$ and $\mu_y = E[y(t)]$ [1].

2.11 Cross-correlation

Taking into consideration the statements refered in Section 2.10, the definition of cross-correlation appears as a measure of the similarity between two signals, as a function of the time shift between the two [8]. For zero-mean functions, the cross-correlation may be described as:

$$R_{x_y}(\tau) = E[x(t)y(t+\tau)] \tag{2.18}$$

which in the case of continuous functions would be computed as:

$$R_{xy}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_0^T x(t)y(t+\tau)dt \tag{2.19}$$

The main application for cross-correlation is the determination of time delays between signals [1, 8].

2.12 Autocorrelation

Autocorrelation is also described by Equation (2.19), but for the special case where $x(t) = y(t)$, therefore autocovariance and autocorrelation are defined as $C_{xx}(\tau)$ and $R_{xx}(\tau)$, respectively [1].

So, the autocorrelation function is a special time average, defined by:

$$R_{xx}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_0^T x(t)x(t+\tau)dt \tag{2.20}$$

This function can be used to improve the SNR of periodic signals [8].

2.13 Frequency Response Function

Many of the pratical problems that involve the response of a system to a random excitation can be solved taking in consideration a single random input function. This can be straighforward, with two main steps: the power spectrum of the input and the FRF of the system.

The FRF, normally denoted by $H(\omega)$ or $H(f)$, depending whether it is expressed, respectively, in terms of rad/s or Hz, is the ratio of the stedy-state response of a system to an sinusoidal input, which can be of different types, like a force or an imposed displacement. The FRF is normally expressed in complex form, but it can also be expressed in magnitude and phase form. The FRF of a system is sometimes referred to as the system transfer function, but this is only true if the response is expressed in Laplace notation. Considering that if s is replaced by iw, the transfer function becomes the FRF ($H(\omega)$, in the complex notation [14].

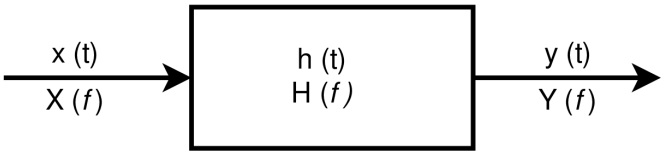


Figure 2.9: Block diagram of a linear system.

In Figure 2.9, it is possible to observe the block diagram of a linear system, with $x(t)$ as input, $y(t)$ as the output, $h(t)$ as the impulse response and $H(f)$ as the frequency response. This system is considered time-invariant if the coefficients of the system of differential equations, which define the system, do not change with time.

Taking the assumption that the system under analysis is linear and time-invariant, the frequency response can be estimated experimentally, as it is represented in the following equation:

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(u)h(t-u)du. \tag{2.21}$$

The Equation (2.21) can also be defined in the frequency domain, giving origin to the following equation for the spectrum $Y(f)$:

$$Y(f) = X(f) \cdot H(f). \tag{2.22}$$

So, the input is amplified and phase-shifted at each frequency, independently of the other frequencies.

The main problem in measuring the inputs and outputs of real-life systems is the contamination of at least one of these signals by extraneous noise. So taking in consideration the diagram in Figure 2.9 and applying this noise to both input and output, the diagram in Figure 2.10 appears [15].

2.14 H_1 Estimator

With the objective to find a method for estimating the frequency response of the linear system, a simplification of the system represented in Figure 2.10 must be made. For the H_1 estimator it must be considered that there is no extraneous noise at the input, therefore $m(t) = 0$, and the block represented with the letter (a) in Figure 2.10 can be replaced simply by $x(t)$. Additionally, random noise at the input also has to be considered.

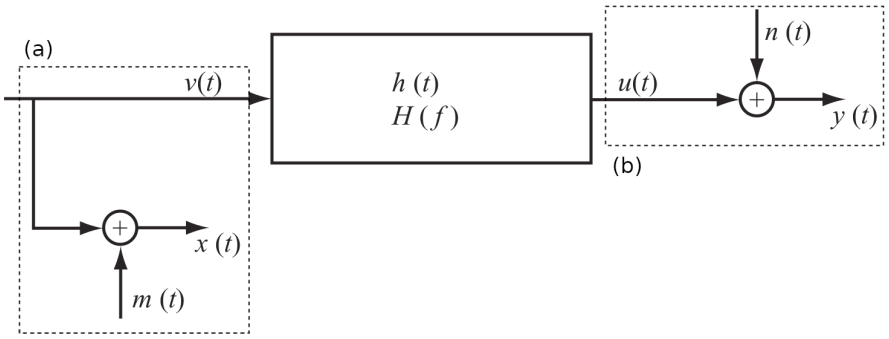


Figure 2.10: Linear system with noise at both the input and the output signals.

Similar to what was done in Equation (2.22), it is possible to define the spectrum of the output of the system under analysis, as follows:

$$Y(f) = X(f)H(f) + N(f). \tag{2.23}$$

If both sides of the Equation (2.23) are multiplied by the complex conjugate of $X(f)$ ($X^*(f)$), the following is obtained:

$$X^*(f)Y(f) = X^*(f)X(f)H(f) + X^*(f)N(f). \tag{2.24}$$

Taking the average of both sides of the Equation (2.24) separately, and scaling the results in a convenient way, the PSDs are obtained, as seen in section 2.8. From this, the cross-spectral and the auto-spectral densities are obtained

$$G_{yx}(f) = G_{xx}(f)H(f) + G_{nx}(f). \tag{2.25}$$

Since the cross-spectral density between the input and the noise (G_{nx}), approaches zero when the average is in place, thanks to the signals being uncorrelated, the H_1 estimator of $H(f)$ appears as

$$\hat{H}_1(f)^2 = \frac{\hat{G}_{yx}(f)}{\hat{G}_{xx}(f)}. \tag{2.26}$$

The estimator in Equation (2.26) is a least squares estimate of the system $H(f)$. The H_1 estimation will be biased if there is noise added to the input, since the model created is wrong [15].

2.15 H_2 Estimator

In this case, the H_2 estimator assumes that the noise is not in the output, as seen before, but at the input. Thus, this system is represented in Figure 2.10 but when $n(t) = 0$ and the block with the letter (b) is replaced simply by $y(t)$. This estimator could be used, for example, a shaker excitation test. More information about this techniques on a later chapter. Similar to what was done earlier for the H_1 estimator, first the system spectrum equation is obtained:

$$Y(f) = [X(f) - M(f)]H(f), \tag{2.27}$$

[2]The usage of the symbol ˆ (hat) is used to denote that it is an estimated function.

then both sides of Equation (2.27) are multiplied by the complex conjugate of the output, $(Y^*(f))$.

$$Y^*(f)Y(f) = H(f)[G_{xy}(f) - G_{my}(f)] \tag{2.28}$$

After this, taking the expected value, like in H_1, and scale it, the following is attained:

$$G_{yy}(f) = H(f)[G_{xy}(f) - G_{my}(f)]. \tag{2.29}$$

Analogous to the H_1, the cross-spectrum $G_{my}(f)$ will reach zero when the average is done, so the H_2 estimator is defined as

$$\hat{H}_2(f) = \frac{\hat{G}_{yy}(f)}{\hat{G}_{xy}(f)}. \tag{2.30}$$

Note that $G_{xy}(f) = G^*_{yx}(f)$, means that the phase of $G_{xy}(f)$ is equal to the phase of $G_{yx}(f)$, only with opposite sign. It is possible to state that the H_2 estimator can also be obtained by switching place between x and y and inverting the resulting H_1, as seen in the following equation

$$^2\hat{H}_{yx} = \frac{1}{^1\hat{H}_{xy}}{}^3. \tag{2.31}$$

Like the H_1 estimator, the H_2 estimator is also a least squares estimate of $H(f)$, and in the case that the model assumption is wrong, the estimate is biased [15].

2.16 H_v Estimator

The Hv estimator was developed to yield better results in the cases were contaminanting noises exists on both input and output, as described in Figure 2.10. This estimator assumes that the background noise sources are incoherent with the true input and output [15].

For single-input/single-output case, the Hv estimator is reduced to the geometric mean of the $H1$ and $H2$ estimators, and described by the following:

$$\hat{H}_v = \frac{G_{yx}}{|G_{yx}|} = \sqrt{\frac{G_{yy}}{G_{xx}}} = \sqrt{\hat{H}_1\hat{H}_2} \tag{2.32}$$

2.17 Coherence Function

The coherence function, $\gamma^2(f)$ can be defined as the ratio between $\hat{H}_1$ estimate and the $\hat{H}_2$ estimate,

$$\hat{\gamma}^2_{yx}(f) = \frac{\hat{H}_1(f)}{\hat{H}_2(f)} = \frac{\mid \hat{G}_{yx}(f) \mid^2}{\hat{G}_{xx}(f)\hat{G}_{yy}(f)}. \tag{2.33}$$

As seen in Section 2.14, in the event of existing input noise, $m(t)$, the magnitude of the estimate $\hat{H}_1$ will be less than or equal to the value of $H(f)$. In a similar way, the magnitude of the estimate $\hat{H}_2$ will always be greater than or equal to the true value

[3] With the little 1 refering to H_1 estimator and the 2 to the H_2 estimator

of $H(f)$. As long as the cross-spectrum terms are zero (through many averages), it is feasible to conclude that the true value of $\mid H(f) \mid$ fulfills the following

$$\mid \hat{H}_1(f) \mid \leq \mid H(f) \mid \leq \mid \hat{H}_2(f) \mid . \tag{2.34}$$

Since the coherence function in Equation (2.33) is represented as the squared function, then

$$0 \leq \gamma^2_{yx}(f) \leq 1. \tag{2.35}$$

So, if $\gamma^2_{yx}(f) = 1$ then $\hat{H}_1 = \hat{H}_2$, meaning that there is no extraneous noise and therefore the measure output, $y(t)$, derives purely from the measured input, $x(t)$.

The coherence function is used as a quality measure of an estimated frequency response, regardless of which estimator was used. As stated before, the coherence function drops below the value 1, if there is any noise on either input or output signals or even both [15].

2.18 IEPE/ICP

The accelerometers and the impact hammers used in vibration analysis are usually piezoelectric and follow the IEPE/ICP standard. So, in order to connect them to audio card for signal acquisition, an interface board is needed. This board will allow the devices to be powered according to the IEPE/ICP standard, and convert the signal voltage to be compatible with the audio card.

IEPE is a standard for piezoelectric sensors. These devices contain a built-in impedance converter, allowing them to convert the high impedance signal from the piezoelectric material into a voltage signal with a low impedance.

IEPE accelerometers are used in vibration analysis, with applications in areas such as aerospace, aviation and automotive industry. Usually these devices work with a constant current between 2 and 20 mA, being 4 mA a prevalent value. With the constant current supply to the IEPE sensor, result a positive bias voltage, generally between 8 and 12 volts at the output. These accelerometers allow measurements over a wide frequency ranges, typically from 1 Hz to 10 kHz, with some having frequency responses from very low frequency, $f \geq 0.001$ Hz.

The preeminent advantages over other types of sensors include low noise; wide dynamic, frequency and temperature range; high sensitivity; low output impedance and small form factor.

As stated before, these devices required a constant current and a voltage interval between 18 and 30 volts, in order to work properly. The basic circuit to power IEPE/ICP devices can be seen in Figure 2.11. The circuit is quite simple, consisting of a battery, a current regulator diode and a decoupling capacitor.

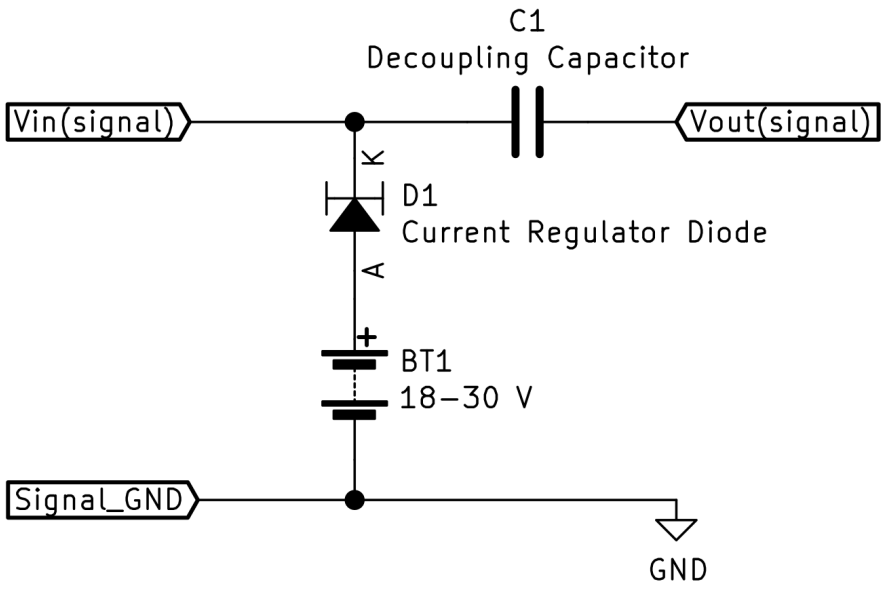


Figure 2.11: IEPE/ICP circuit for powering accelerometers and impact hammers.

Chapter 3

Spectrum Analyzer

Spectrum analyzers are used to analyse and display signal data where its spectral content is the most interesting information to observe. Their main function is to measure power over frequency, distortion and noise, by correlating the input and output signals. They are distinguished depending on the method used to obtain the spectrum of a signal. The main two types are swept-tune and FFT based analyzers. Swept-tune devices obtain the power over frequency by taking sucessive tunning scans, while the others use FFT techniques after acquiring the signal through an ADC.

An example of a commercial available spectrum analyzer, from Keysight, can be seen in Figure 3.1. These are devices with extreme capabilities but very expensive, having prices of around 5000 euros.

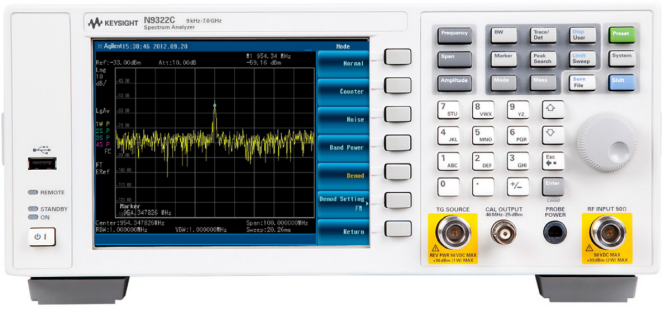


Figure 3.1: A spectrum analyzer from Keysight, model N9322C [5].

3.1 Types of Spectrum Analyzers

As already mention, there are two main types of spectrum analyzers, which are going to be described in the following sections.

3.1.1 Swept-tuned Spectrum Analyzer

The swept-tuned (superheterodyne) spectrum analyzer is the most used type of spectrum analyzer. This device is based on the superheterodyne principle, allowing it to operate with very high frequencies.

The fundamental disavantage of this device is its ineptitude to measure a signal phase (only measures amplitude) and to measure transient events, due to the sweeping taking some time. A block diagram of this type of analyzer is portrayed in Figure 3.2.

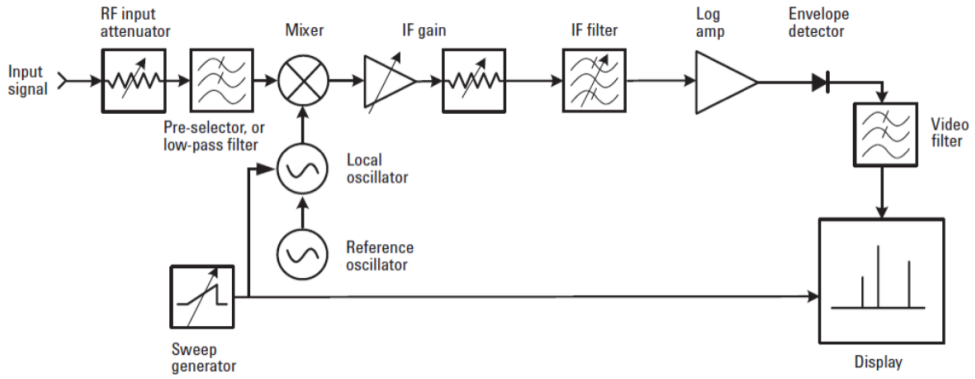


Figure 3.2: Block diagram of a swept-tuned analyzer [6].

3.1.2 FFT Spectrum Analyzer

A FFT spectrum analyzer samples the time domain signal and converts it afterwards to the frequency domain, using a DFT. After the acquisiton of the signal by an ADC, a DSP is used to compute the FFT algorithm. A simplified version of the block diagram of this type of device is represented in Figure 3.3.

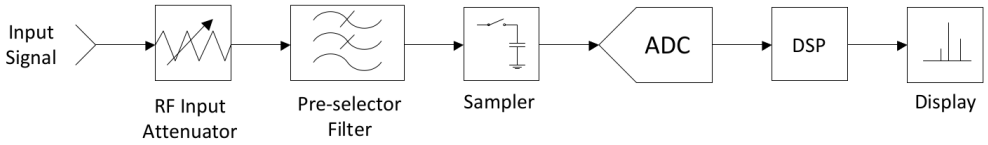


Figure 3.3: Simplified block diagram of a FFT analyzer [7].

The Fourier spectrum analyzer has two main advantages. First, in a FFT analyzer the signal is primarily acquired and then analyzed, allowing it to be captured in a small amount of time. Second, and as opposed to the Swept-tuned analyzer, it allows the capture of the signal phase.

As compared to the Swept-tuned analyzer, the main disadvantage of the FFT analyzer is the maximum frequency limit, which is imposed by the ADC. Even with today's high performance ADCs, the sampling rate is not sufficient to high frequency signals. Altough this can be considered as a disadvantage, to the purpose of vibration analysis, there is no need to digitize high frequency signals. So, to the objective of this project, the FFT spectrum analyzer is the one to follow [7].

3.2 Spectrum Analyzer Fundamentals

To find the spectrum of the signal, first the signal must be acquired in the time domain, and after that, the FFT algorithm is executed. As seen in Figure 3.3, the input signal first is passed through an attenuator, which provides multiple measurement ranges. Then the signal passes through a low-pass filter, removing the undesirable high-frequency content. Then, by using a combination of the sampler circuit and an ADC, the waveform is sampled and converted to digital form. Afterwards, the microprocessor (or any other digital circuitry) receives the sampled waveform and, with the use of the FFT algorithm, the spectrum is computed. Finally the results are show in the display [8].

3.2.1 Sampled Waveform

In a sampled system, the sampled waveform is produced by multipling the time domain waveform by the sample function. The sampling function, as seen in Figure 3.4 (b), is defined as a series of impulse functions, each spaced at $T = 1/f_s$, being f_s the sample rate of the system. The following equation describes the sampling function:

$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT). \tag{3.1}$$

By the time the sampling function is multipled by the original waveform, a new series of impulse functions appears, with each weighted according to the original waveform. This

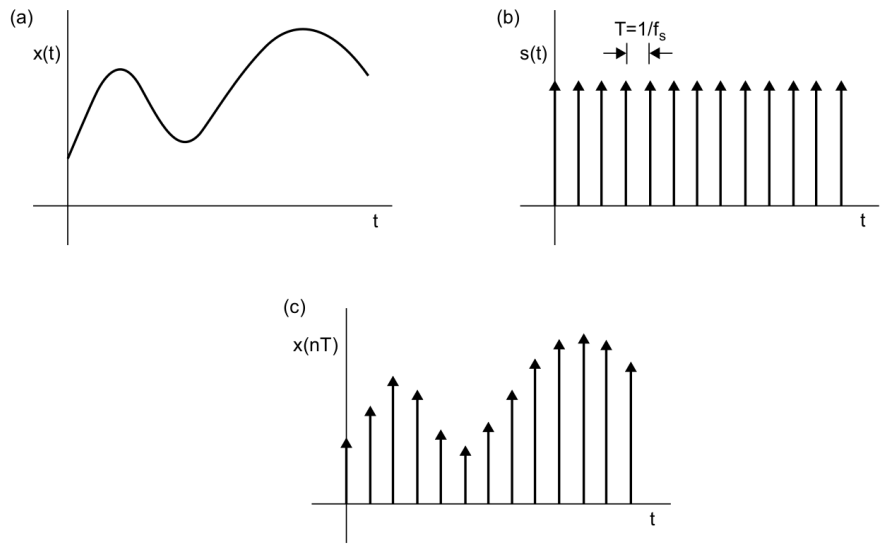


Figure 3.4: A time domain waveform (a). The sampling function (b). The sampled waveform (c) [8].

impulse functions are defined by:

$$x(nT) = \sum_{n=-\infty}^{\infty} x(t)\delta(t - nT), \tag{3.2}$$

and it is visually presented in Figure 3.4 (c). If the sample rate is sufficiently for the acquisition in hand, the sampled and digitized version will retain the shape and information of the analog waveform at the input [8].

3.2.2 Sampling Theorem

In order to produce a digital signal that faithfully represents the analog waveform, this one has to be sampled enough times, to mantain the characteristics of the original waveform. The sampling theorem establishes that a signal must be sampled at rate greater than twice the highest frequency contained in the signal. Therefore, the minimum acceptable sample rate, also known as the *Nyquist rate* is defined as

$$f_s > 2f_{max}. \tag{3.3}$$

Something that occurs in sampled systems is the repetition of the baseband signal, at multiples of the sample frequency. This can cause symmetries around the frequency $f_s/2$, called the *folding frequency*, also known as the *Nyquist frequency*, f_f.

A consequence of the sampling theorem not being fulfilled is the occurrence of the *aliasing* phenomenon. This can be seen in Figure 3.5, where the top signal was sampled correctly, and the bottom one violated the *Nyquist theorem*. This phenomenon makes the digital signal different from the original continuous one, by containing undesirable frequency components mixed in together [8].

To avoid this problem, there are a couple of conditions that must be met:

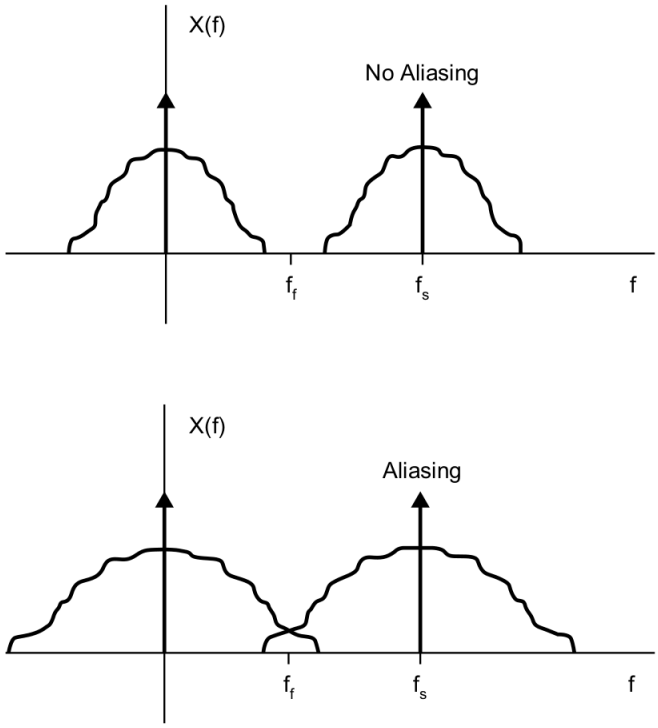


Figure 3.5: Example of *aliasing* in a signal.

- Limiting the band of the input signal, attenuating frequencies above f_{max}. This can be often accomplished by utilizing a low pass filter.
- As said before, the signal must be sampled at a rate that satisfies Equation (3.3).

3.2.3 Frequency Span

Since the FFT is intrinsically a baseband transform, it means that its frequency range always starts at 0 Hz and goes to the maximum frequency $f_s/2$. This can be a limitation while measuring small frequency band samples. For example, with a $f_s = 512$ kHz, the FFT range would be from 0 Hz to 256 kHz ($f_s/2$). With $N = 1024$ samples, the frequency resolution would be given by $f_s/N = 500$ Hz. Thus, spectral lines closer than 500 Hz could not be resolved.

This problem can be solved by increasing the number of samples, N, in the time record, which will increase the number of bins in the FFT output and consequently increase the computation time for the FFT algorithm.

Another solution is reducing the f_s, which will improve the frequency resolution at the cost of the upper-frequency limit of the FFT, giving the user control over the frequency resolution and the frequency range. With this method, the cutoff frequency of the anti-alias filter must be adjusted, in order to prevent aliasing. For this situation a *decimating digital filter* provides a better result than the analog anti-alias filter, decreasing simultaneous the bandwidth and the sample rate.

There is still the need to use the analog anti-alias filter, in order to protect the digital

filter from aliasing [8].

3.3 FFT Properties

Taking into account the concepts described in Section 2.5 from Chapter 2, the FFT algorithm is record oriented, meaning that at the input there is a signal, with N samples long in the time domain, and at the output there is another signal with N samples long, but now in the frequency domain. Usually N is limited to a power of 2, simplyfing the calculation of the FFT. A common record length for an FFT based spectrum analyzer is 1024 sample points. The FFT produces a frequency spectrum that is symmetrical around the folding frequency. For this reason, the first half of the output is redundant with the second, therefore unnecessary. So, the effective lenght of the output signal is $N/2+1$, this being complex points (real + j imaginary) they encompass both magnitude and phase information.

In practice, the output extends from 0 Hz to f_f, as seen in Subsection 3.2.3 , despite to the fact that no all points are displayed as a result of the anti-alias filter, that starts to roll off before f_f. A standard configuration of 1024 samples in the time domain, produces 513 unique complex points in the frequency domain, where only 401 are displayed.

Commonly refered as *bins*, the frequency domain points are ordinarily numbered from 0 to $N/2$. For example, in the case of $N = 1024$, 0 to 512. The bins are identical to the outputs in the bank-of-filters analyzer. The first bin, Bin 0 also known as the *DC bin*, contains the DC (Direct Current) level existing in the input signal. These bins are equally spaced in frequency, with the frequency step, f_{step} being inversely related to the time record length.

$$f_{step} = 1/length\,of\,time \tag{3.4}$$

The length of time is calculated from the sample rate, f_s, and the number of sample points, N, since

$$f_{step} = f_s/N \tag{3.5}$$

ergo the frequency related with each bin is delivered by

$$f_n = nf_s/N \tag{3.6}$$

where n represents the bin number.

The last bin contains the maximum frequency of the FFT output, which is $f_s/2$, thus the frequency range of an FFT is 0 Hz to $f_s/2$. Taking for example the sine wave represented in Figure 3.6 a, its a one cycle sine wave that fits into one time record, and its output of the FFT will show up in bin 1, as seen in Figure 3.6 b. For example, if the frequency of the wave was doubled, the output would appear in bin 2 [8].

3.4 Leakage

In order to approximate the Fourier transform, which integrates over all time, the FFT is computed on a finite length time record. Even though it operates in a finite length it can replicate this record over all time, as observed in Figure 3.7. When the time record represents the actual waveform correctly, the result of the FFT will approximate the Fourier integral quite well, as seen in Figure 3.7 b [8].

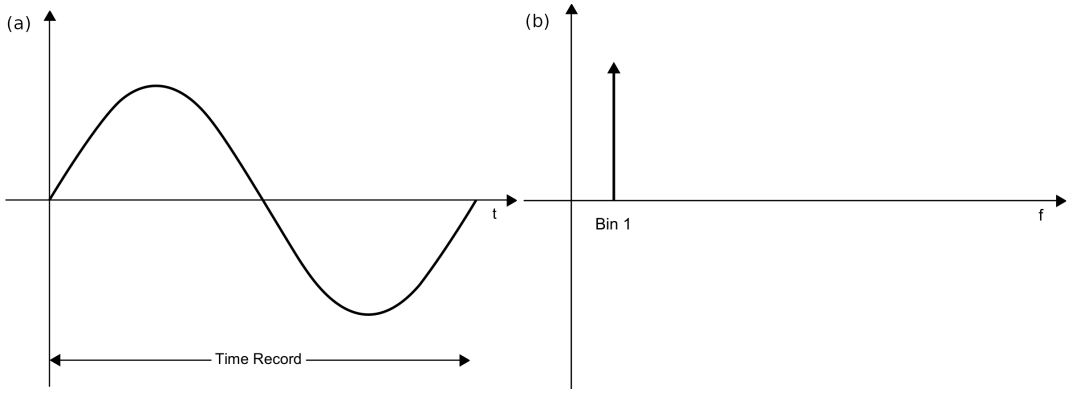
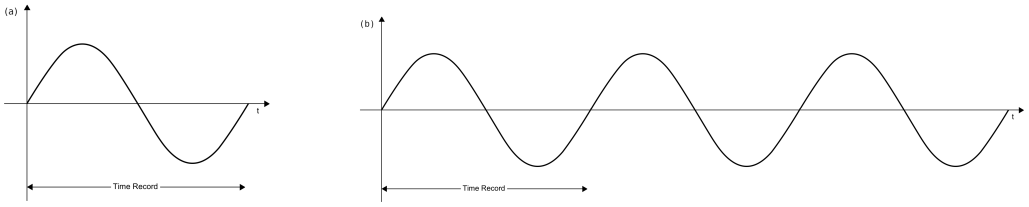


Figure 3.6: (a) Representation of a sine wave that fills precisely one time record. (b) Representation of the spectral line in the first bin of the FFT output.



(a) A signal that fits one time record. (b) Replication of the previous signal.

Figure 3.7: Representation of finite length time record replicating over time.

However, a signal will start to exhibit discontinuities at the boundaries of the observation interval when this periodic extension is done with a signal that does not commensurate with its natural period. This phenomenon is usually refered as leakage (or spectral leakage), and is the outcome of processing finite-duration records. This effect is conveniently demonstrated in Figure 3.8 [9]. This effect is common in the frequency domain, therefore the spectral line spreads out over a wide frequency range, instead of appearing as a thin and slim line [8].

The prevalent solution to this problem is to eliminate the transients when the time record is replicated, and this is done by forcing the waveform to zero at the ends of the record [8]. This can be achieved by applying windows to the time record.

3.5 Windowing

As said before, windows, when applied to the data acquired, reduce the spectral leakage associated with finite time records, by smoothing out the discontinuities. Windows are weighting functions that are multiplied by the time record, reducing the discontinuity at the boundary of the periodic extension. The windowing process affects many features of a spectrum analyzer such as resolution, dynamic range, and ease of implementation [9].

The multiplication of window functions occurs both in time and frequency domain signals. The windowing process can be represented by the following equation:

$$x'(t) = w(t)x(t), \tag{3.7}$$

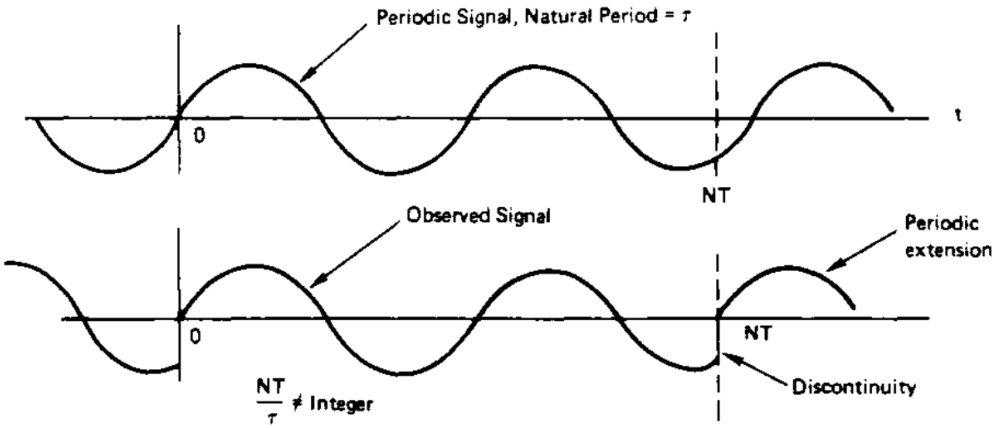


Figure 3.8: Periodic extension of a sine wave, not periodic in observation interval [9].

where $x(t)$ is a time domain signal, $w(t)$ a window function, and $x'(t)$ the result of windowing. An identical operation can be computed in the frequency domain, by the application of a Fourier transform to Equation (3.7) [1], thus resulting in the following convolution:

$$X'(\omega) = W(\omega) \otimes X(\omega). \tag{3.8}$$

Windows are commonly used in the following applications:

- Spectral analysis
  - Reduction of leakage
  - Reduction of effect of speed fluctuation on rotating machines
  - Smoothing of spectral estimates
  - Overlapped processing
- Impulse testing
  - Improvement of SNR
- Digital filter design
  - Windowed-Sinc filters

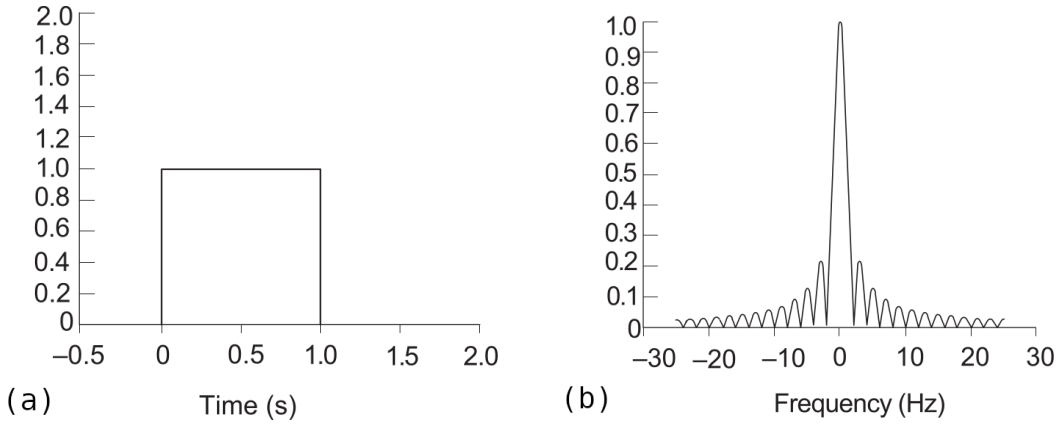
There are a horde of windows that can be used to control spectral leakage, but only a few are used in vibration analysis. In the following subsections these windows will be described, in order to understand their use.

3.5.1 Rectangular Window

The rectangular window is probably the simplest window here described and it is the result of the direct truncation of the signal. It is defined as:

$$w_R(n) = \begin{cases} 1, & 0 \leq n \leq N-1 \\ 0, & \text{otherwise.} \end{cases} \tag{3.9}$$

So, $x'(t) = x(t)$ and the signal analyzed is not modified. The window is characterized by a main lobe and decreasing sidelobes as seen in Figure 3.9b, with zeros at multiples of $1/N\Delta_t$, being N the number of samples and Δ_t the sampling interval [1,9].



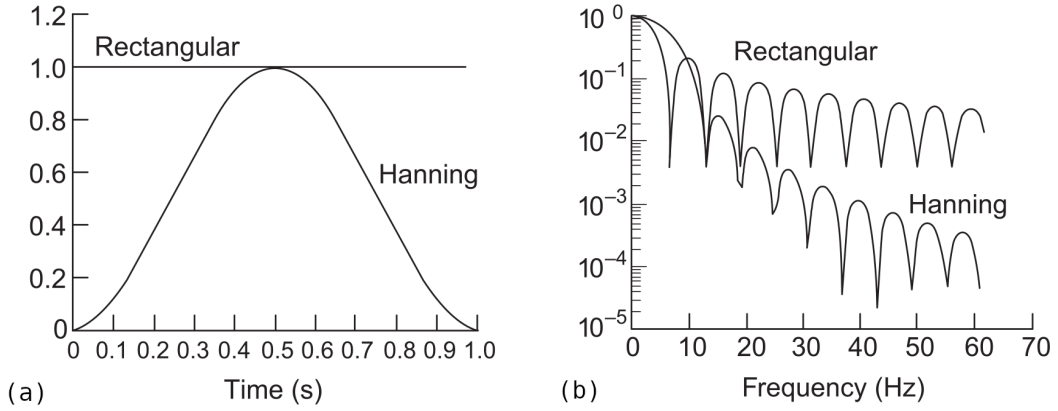
(a) Time domain. (b) Magnitude in frequency domain.

Figure 3.9: Rectangular window in the two different domains [1].

3.5.2 Hanning Window

The Hanning window, also known as Hann window, is probably one of the most prevalent windows in spectral analysis. This type of window provides a gradual attenuation at both ends of the time record, as seen in Figure 3.10a. In the frequency domain, the sidelobe maximum is undoubtedly diminished and the sidelob attenuation has a higher slope, as seen in Figure 3.10b.

$$w_H(n) = \begin{cases} 0.5\left[1 - cos\left(2\pi\frac{n}{N}\right)\right], & 0 \leq n = N-1 \\ 0, \quad \text{otherwise.} \end{cases} \tag{3.10}$$



(a) Time domain windows. (b) Magnitude in frequency domain.

Figure 3.10: Comparison between rectangular and Hanning windows [1].

3.5.3 Hamming Window

Similar to the Hanning window, the Hamming window is another window used in leakage control, that belongs to the $cos^{\alpha}(X)$ family. This window can be considered an altered version of the Hanning window. The Hamming window is described by the following equation:

$$w_{HM}(n) = \begin{cases} 0.54 + 0.46cos\left(2\pi\frac{n}{N}\right), & 0 \leq n = N-1 \\ 0, \quad \text{otherwise.} \end{cases} \tag{3.11}$$

3.5.4 Flat-top Window

The flat-top window is mainly used to reduce the fluctuation of the line spectra within the main lobe. This fluctuation occurs on rotating machinery, since frequency depends on the geometry and kinematics of the rotating elements, and in practice this speed is not constant. Another use of this window is for equipment calibration purposes, using harmonic signals [1].

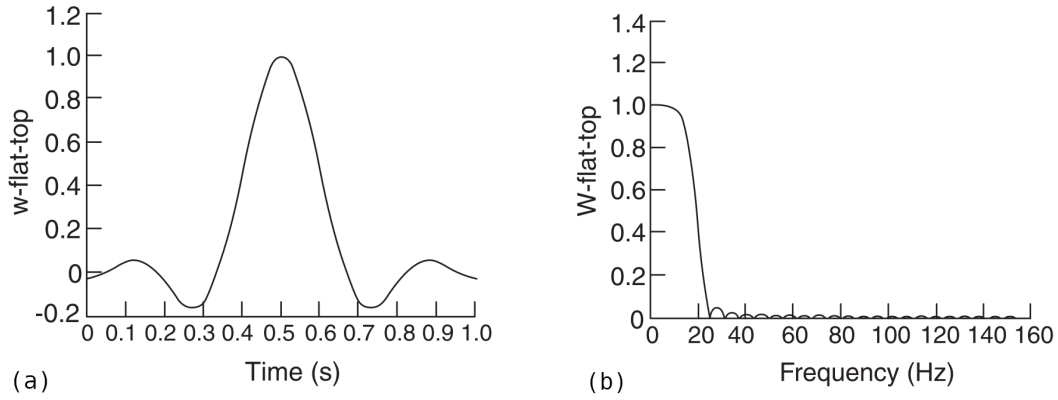
The flat-top window is defined by the following equation:

$$w_{FT}(n) = a_0 + 2\left[2a_1cos\left(2\pi\frac{n}{n}\right) + a_2cos\left(4\pi\frac{n}{n}\right) + a_3cos\left(6\pi\frac{n}{n}\right)\right] \tag{3.12}$$

where the parameters a_0, a_1, a_2 and a_3 are represented in Table 3.1.

Table 3.1: Parameters of the flat-top window equation [1].

$a_0 = 0.9945$	$a_1 = 0.95573$
$a_2 = 0.53929$	$a_3 = 0.09158$



(a) Time domain.

(b) Magnitude in frequency domain.

Figure 3.11: Flat-top window in the two different domains [1].

The information contained in Table 3.2 allows for a better comparison between the different windows used for leakage control.

3.5.5 Windows for Impulse Testing

In order to improve the SNR windowing transientes are used. The idea behind it is to weight the signal according to the certainty associated with it, thus giving less weight to

Table 3.2: Main characteristics of commonly used windows [1].

Window	Highest sidelobe level (dB)	Flow of sidelobe (dBoct)	Scallop loss (dB)
Rectangular	-13	06	3.92
Hanning	-32	-18	1.42
Hamming	-43	✗	1.78
Flat-top	-95	✗	0.01

the signal points with less magnitude, assuming a stationary additive random noise to the signal.

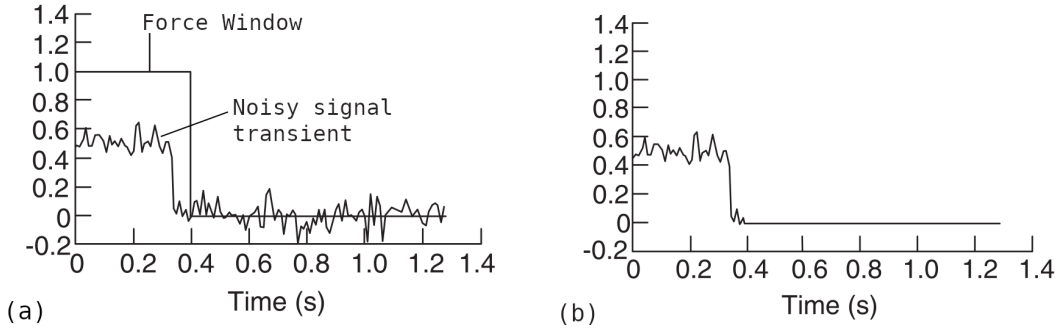
Force Window

The force window is a rectangular window, of duration spanning at least the force duration. It is represented by the following:

$$w_F(n) = \begin{cases} 1, & 0 \leq n \leq N_T - 1 \\ 0, & \text{otherwise,} \end{cases} \tag{3.13}$$

where N_T is a number slightly larger than the number of samples spanning the signal.

This type of window is used in impulse testing, for example of a structure. The structure suffers a short impact, exciting it, with a decaying response of longer duration that the impact. So, taking the impact signal and applying the force window, both seen in Figure 3.12a, a window transient is obtained as in Figure 3.12b. This way, the information of the impact is kept, but the noise in the rest of the signal is reduced, improving the SNR.



(a) Location of window and signal.

(b) Window transient.

Figure 3.12: Force window, identification and convolution with the signal [1].

Exponential Window

As stated before, taking in the assumption of stationary continuous noise, an exponentially weight decrease is recommended for exponentially decaying oscillations. With that in

mind, the equation for the exponential window is given by:

$$w_{exp}(n) = exp(-an) \qquad 0 \leq n < N \tag{3.14}$$

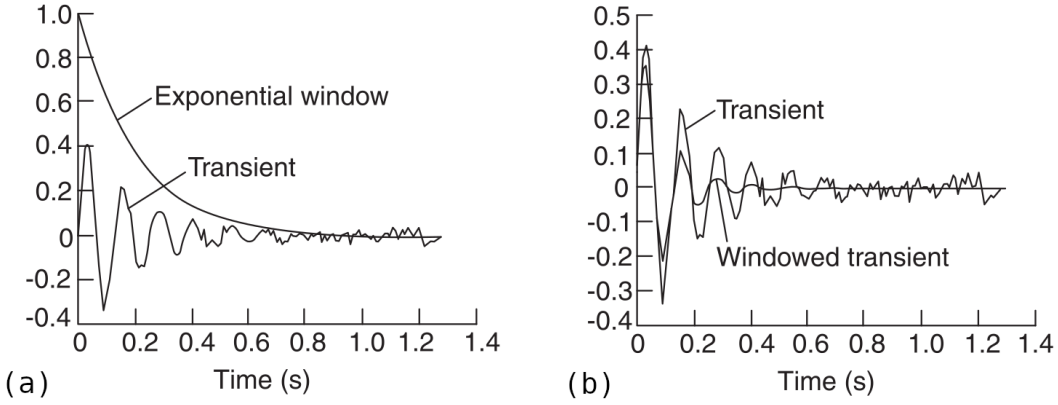
where the parameter a is chosen in order to have a decay of zero for $n = N$. Additionally, the noise variance is roughly reduced by a factor of 3, and the SNR improved by 1.7. Due to the fact that the effective decay is faster with windowing that without (the physical situation), any damping factores obtained from the decaying oscillations must be correted, since its value has been increased by the windowing phenomenon. Bear in mind that the decaying oscillations are a combination of components of the following form:

$$x(t) = A_{exp}(\zeta\omega_n t)sin\left[\omega_n(1-\zeta^2)^{1/2}t\right] \tag{3.15}$$

where ζ is the damping ration and ω_n the natural frequency. Therefore the equivalent damping is given by:

$$\zeta_{eff} = \zeta + \frac{a}{\omega_n} \tag{3.16}$$

with a/ω_n being the factor used for the damping correction [1].



(a) Noisy transient and exponential window.

(b) Original and windowed transient.

Figure 3.13: Exponential window [1].

Taking the noisy oscillating decaying signal and applying the exponential window, both seen in Figure 3.13a, the windowed transient seen in Figure 3.13b is obtained. Also in Figure 3.13b it is possible to observe the discrepancy between the decays, hence the need to use the correction factor.

Chapter 4

Implementation

After analysing the capabilities and limitations of a spectrum analyzer based on the FFT, in the previous chapter, and taking in consideration the objectives of this project, a solution was proposed and implemented. In the following chapter the implementation of the solution created will be described. The basic functional diagram can be observed in Figure 4.1.

As seen in the Figure 4.1, the solution has two main parts, the hardware components and the graphical interface. The hardware components are divided into two groups, an audio card, which is responsible for the signal acquisition, and the Raspberry Pi, responsible for computing all the functions implemented to be used in vibration analysis.

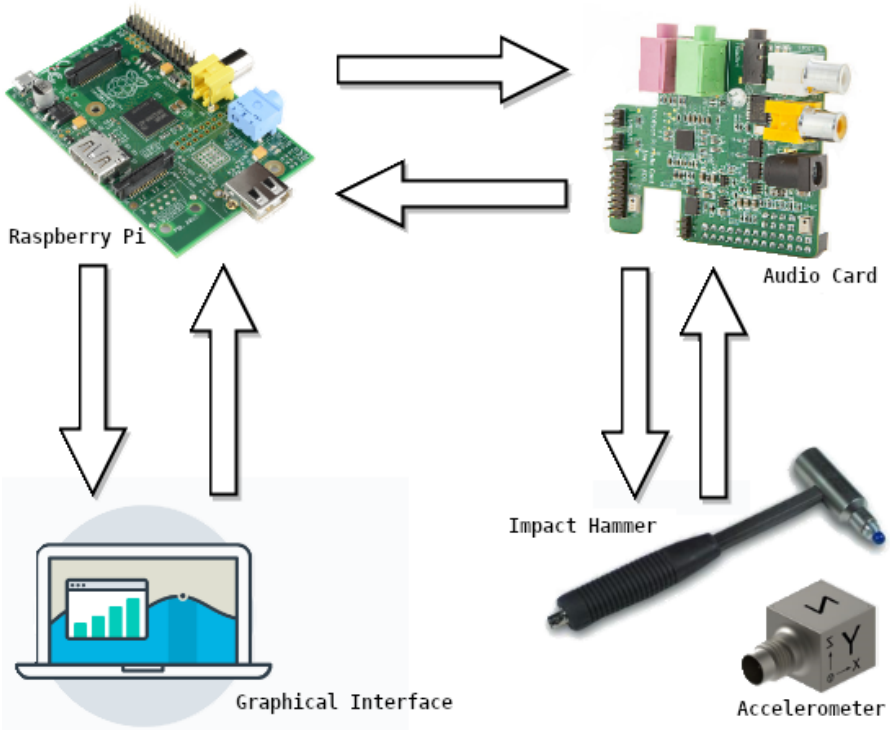


Figure 4.1: Functional diagram of the solution created.

4.1 Excitation Techniques

With the development of modal theory, it is clearly defined that for extracting a modal model, measured frequency responce functions are needed. Even though there are a considerable number of alternatives for exciting a structure in oder to measure response characteristics, in order to obtain a set of calibrated measures, a response from an apllied known input force is mandatory. Therefore the type of excitation is somewhat limited, for obtaining frequency response functions. With that in mind, there are generally two main categories of apllied forcing functions used for experimental modal analysis, shaker and impact excitation. While there are many others, these are the ones that provide a value to the input force [10].

In the following subsections, these methods are going to be explained in more detail.

4.1.1 Shaker Excitation

In regards to this method, there are several techniques used for achieving an experimental modal model using force shakers. In this method there are two main types of force inputs, random and deterministic. Both types can be used to determine the character of the system, like the generation of the FRF and an appraisal of the linearity of the system under assessment [10].

4.1.2 Impact Excitation

Impact testing has become one of the most popular methods for acquiring FRF in experimental modal testing. The portability and simplicity of equipment used make this a valuable test technique, not only for experimental modal tests but also for troubleshooting applications. With regards to the input force and output response of the system, there are a few issues that need to be addressed concerning this technique.

For a single degree of freedom system, a common impact response measurement is described in Figure 4.2. It is noticeable that while the input force is completely observable within one sample interval T of the analyzer, the same can not be said for the response of the system, since this one continues to decay exponentially beyond the recorded sample block. This will create distortion in the signal, hence the need to use antialising filters and the application of windows, eliminating unwanted frequencies and leakage.

There are many topics to discuss relative to impact testing, such as hammer tip selection, use of force and exponential windows and double impact. The topic relative to the use of windows was discussed in Subsection 3.5.5. And the topics related to the impact hammer constitution are not included in the scope of this project, so they will not be looked further into [10].

4.2 Hardware

The initial proposal consisted in using a *Raspberry Pi 3 B+* and an audio aquisition HAT (Hardware Attached on Top), in this case the *Cirrus Logic Audio Card*. However, due to the HAT being out of stock, another audio card had to be chosen, and the *Wolfson Audio Card* was the nearly ideal substitute, having almost the same characteristics.

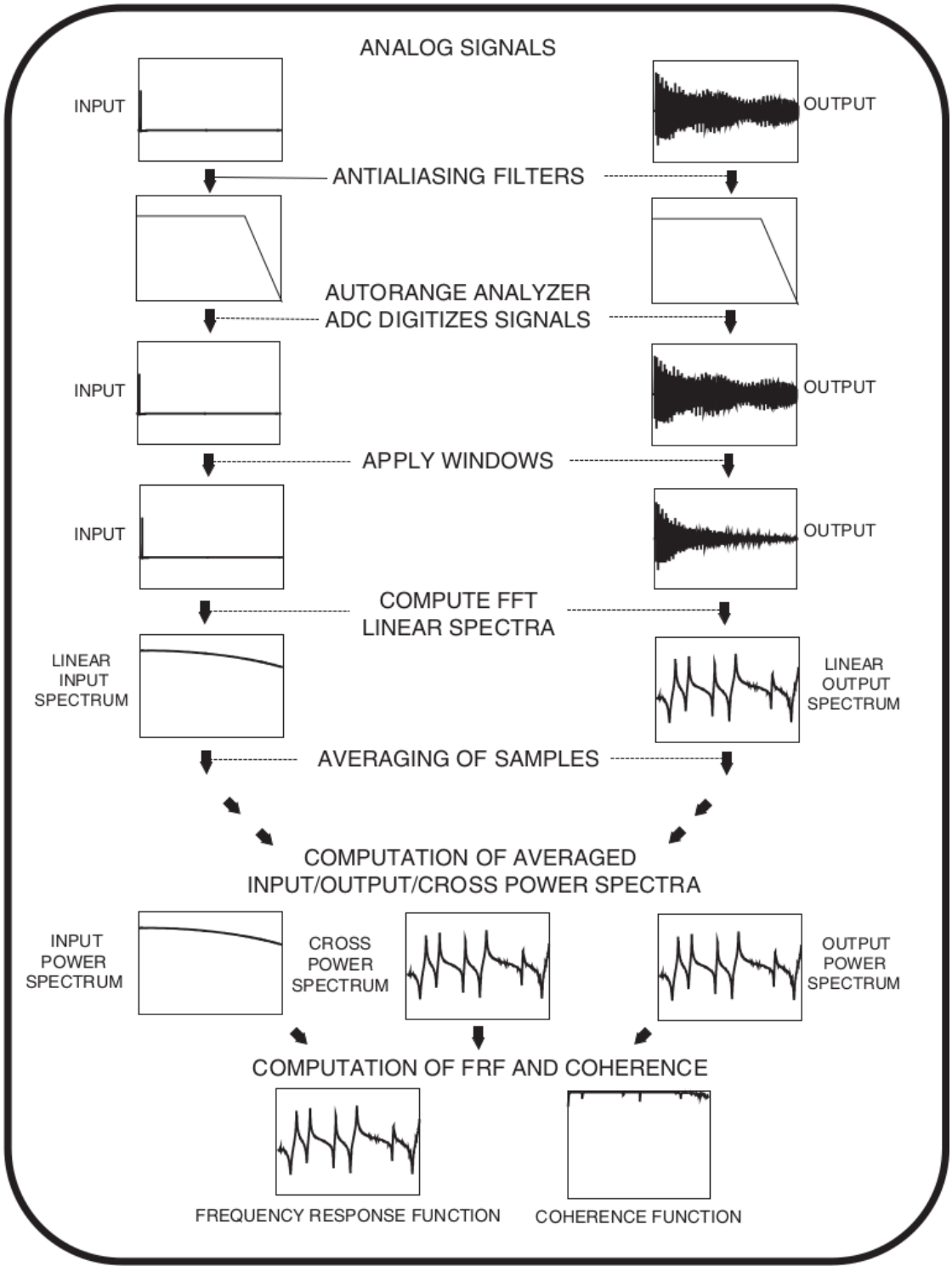


Figure 4.2: Measurement process for impact testing [10].

4.2.1 Development Platform

In spite of the fact that essentially no performance was lost in the audio card replacement, the same cannot be assumed for the *Raspberry Pi*. As a result of the replacement, the new card is no longer compatible with the model *3 B+* of the *Raspberry Pi Foundation*, due to the fact that the *Wolfson Audio Card* is only suitable for older models, that feature the 8-pin audio GPIO connector (known as P5), which is marked with the yellow rectangle in Figure 4.3, alongside with other 26 GPIO pins, instead of the new 40-pin available in the *Raspberry Pi 3 B+*.

Beside the issue with the hardware compatibility, there were also problems with the audio card kernel drivers. To solve this problem it was necessary to patch and re-compile the kernel software running on the Raspberry Pi, in order to provide support for the audio card.

Figure 4.3: GPIO pins of the *Raspberry Pi B*.

With this change in hardware some limitations where introduced. Table 4.1 allows the comparison of both models, and it is conceivable to affirm that with a fourth of the core count, and with half the Central Processing Unit (CPU) speed and Random-Access Memory (RAM), the *Raspberry Pi B* might have a subpar performance than the newer model.

Table 4.1: Raspberry Pi's CPU and RAM information.

Name	Processor	No. OF Cores	CPU Speed	RAM
Raspberry Pi B	BCM2835	1	700 MHz	512 MB
Raspberry Pi 3 B+	BCM2837B0	4	1.4 GHz	1 GB

In the general Input/Output (I/O) (Input/Output) ports the deficiency is not tremendous, only missing onboard *Wi-Fi* and two Universal Serial Bus (USB) ports, as seen in Table 4.2.

Table 4.2: Raspberry Pi's general information.

Name	USB Ports	Wi-Fi	Bluetooth	GPIO	Operating Voltage
Raspberry Pi B	2x USB 2.0	✗	✗	26 pins	5 V
Raspberry Pi 3 B+	4x USB 2.0	✓	✓	40 pins	5 V

Despite its limitations in comparison to the newer model, the *Raspberry Pi B* is still a great development platform, with great utilities. This board has an High-Definition Multimedia Interface (HDMI) output, allowing the use of an external monitor, and with the USB interface, the use of mouse and keyboard facilitating the user interface. Since the *Raspberry Pi* operates in the open source ecosystem it runs Linux (in a variety of distributions), being the main supported one Raspbian.

4.2.2 Audio Card

The *Wolfson* audio card offers the *Raspberry Pi* nearly the same utility and flexibility as a computer soundcard, enabling stereo capture and playback. The I/O layout of the board can be observed in Figure 4.4.

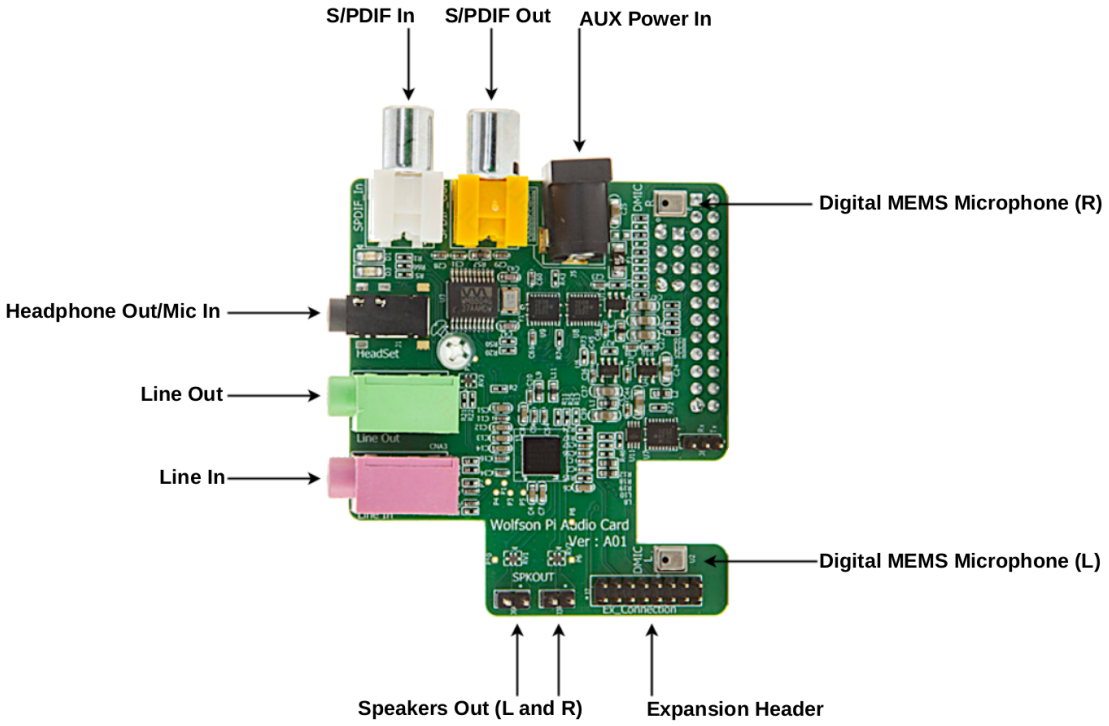


Figure 4.4: Connections layout of *Wolfson* audio card.

The card has two main components, the WM5102 and the WM8804, both from *Wolfson Microelectronics* now owned by *Cirrus Logic*. The interactions of this components with the *Raspberry Pi* are described in the diagram of the audio architecture of the card in Figure 4.5. This audio card is based on *Wolfson* WM5102 audio hub codec [1]. This is a highly-integrated, low-power audio system, usually used in smartphones, tablets and other compact audio devices. The WM5102 provides an impressive combination of fixed-function signal processing blocks, paired with a programmable DSP. With its all-digital audio mixing and sample rate converters it allows for wide flexibility. The fixed-function signal processing blocks also include filters, Equalization (EQ) and dynamics

[1] A device or a software for enconding or decoding a digital data stream or signal.

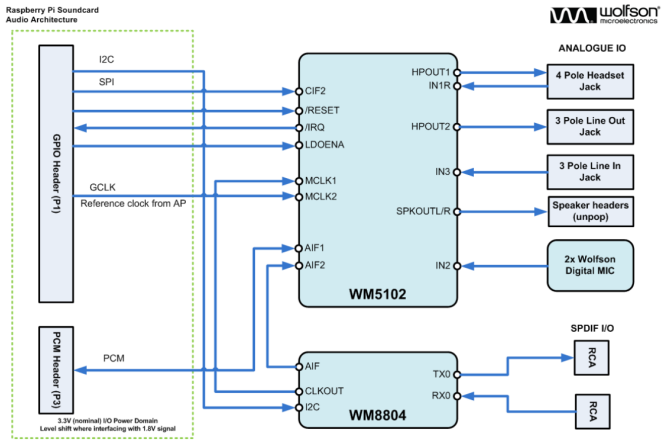


Figure 4.5: Diagram of the audio architecture of the *Wolfson* audio card.

processors.The last ones provide a way of altering the dynamic range of an audio source, making it easy to manipulate the audio signal. Some other features of this component are:

- Integrated voice processor DSP
- Multi-channel asynchronous sample rate conversion
- Sample rates up to 192kHz
- 24-bit resolution

The *Wolfson* audio card also has a WM8804 digital interface transceiver with Phase-Lock Loop (PLL) [2] (Phase-Lock Loop), which is a high performance Sony/Philips Digital Interface (S/PDIF) transceiver with support for 1 received channel and 1 transmitted channel. Its main functions are:

- S/PDIF compliant
- Advanced jitter attenuating PLL
- Auto frequency detection
- Programmable audio data interface:
  - I^2S
  - from 16 up to 24 bit word length

As seen in Figure 4.5 both components communicate with the *Raspberry Pi* through I^2C. The S/PDIF I/O is maintained by the WM8804 using I^2S, allowing for audio formats of 24 bit word length and with sample rates up to 192kHz.

[2] A feedback system with a phase detector that drives a Voltage-Controlled Oscillator (VCO) in a feedback loop, making the oscillator frequency track that of an applied reference frequency.

From all the different types of I/O seen in Figure 4.4 the ones that were considered for the implementation were the Line In/Out. Since the Line Input and Output use a 3 pole connector, they manage to acquire two different signals simultaneous. The main parameters for the Line Input and Output are described in tables 4.3 and 4.4.

Table 4.3: Main characteristics of the *Wolfson* Audio Card Line Input.

Line Input			
Parameter	**Description**	**Typical Value**	**Units**
Connector	Electrical input via 3 pole 3.5mm Socket	✗	✗
Sample Rates	32, 44.1, 48, 88.2, 96, 176.4, 192	✗	kHz
R_{in}	Input Impedance	16	kOhms
F_{cut}	Cutoff Frequency	13	Hz
V_{noise}	Equivalent input noise level	20	uV_{rms}
THD	Total Harmonic Distortion @ 0.9 V_{rms} input	0.035	%

Table 4.4: Main characteristics of the *Wolfson* Audio Card Line Output.

Line Output			
Parameter	**Description**	**Typical Value**	**Units**
Connector	Electrical output via 3 pole 3.5mm Socket	✗	✗
R_{out}	Output Impedance	16	Ohms
C_{load}	Max capacitive load	2	nF
V_{out}	Full scale output	1	V_{rms}
V_{noise}	Noise Floor	4.5	uV_{rms}
THD	Total Harmonic Distortion	0.005	%

As said before the Line I/O uses a 3 pole audio jack connector. With this connector it is possible to acquire different signals in the left and right channel. As shown in Figure 4.6, the audio jack, also known as Tip Ring Sleeve (TRS), has a tip used for the left channel, the middle ring used for the right channel and the sleeve used for the ground connection.

In order to acquire the signals from the impact test, the connections to the audio jack were made as described in Table 4.5. The impact hammer signal is delivered by the left pin and the accelerometer signal by the right pin.

4.2.3 IEPE/ICP interface

Since the sensors used for impact excitation, being them accelerometers and impact hammers, are mainly IEPE/ICP devices there is a need to power them accordingly to the respective norm, as seen in Section 2.18 from Chapter 2.

So, a circuit was developed with two goals in mind, the first was powering the IEPE/ICP sensors, and the second was lowering the signal voltage, matching the 0.9 V_{rms} of the audio card. To achieve the first goal, the circuit from Figure 2.11 was used as a starting point. To accomplish the second goal, a voltage divider was used, like the one detailed in Figure 4.7. Considering the schematic and using Ohm's Law, the output

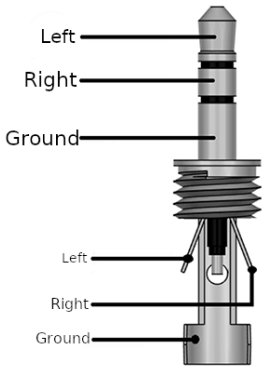


Figure 4.6: Diagram of a 3 pole audio jack.

Table 4.5: Pin Configuration of a 3 pole audio jack.

Pin No.	Pin Name	Description
1	Left	Impact Hammer
2	Right	Accelerometer
3	Ground	Ground

voltage is given by:

$$V_{out} = \frac{R_2}{R_1 + R_2} * V_{in} \tag{4.1}$$

Taking into consideration the 0.9 V_{vrms} of the audio card, the resistors R_1 and R_2, were selected. Also, it was added an optocoupler, isolating the two circuits (the IEPE/ICP

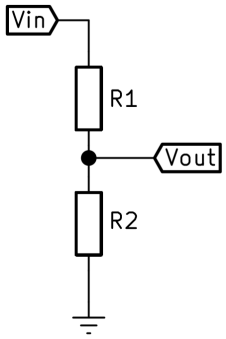


Figure 4.7: Simple resistive voltage divider.

power and acquisition), protecting both circuits from possible malfunctions. Thus, an optocoupler was chosen, in this case a IL300 from *Vishay*. Furthermore, following the manufacturer recommendation, two operational amplifiers were added, one at the input and other at the output of the optocoupler, as depicted in Figure 4.8.

Lastly, an anti-aliasing filter is needed at the input of an ADC, in this case the audio card Line In, in order to remove frequencies above half the sampling rate. To do so, an analog filter wizard was used, in this case the one from *Analog Devices*, allowing the creation of a 10th order Butterworth filter, with a 10kHz passband and a 16kHz stopband.

4.3 Data Acquisition

In order to achieve the main objective of the solution, acquiring the signals of the impact test, an algorithm was created. The primary function of this algorithm is to acquire the

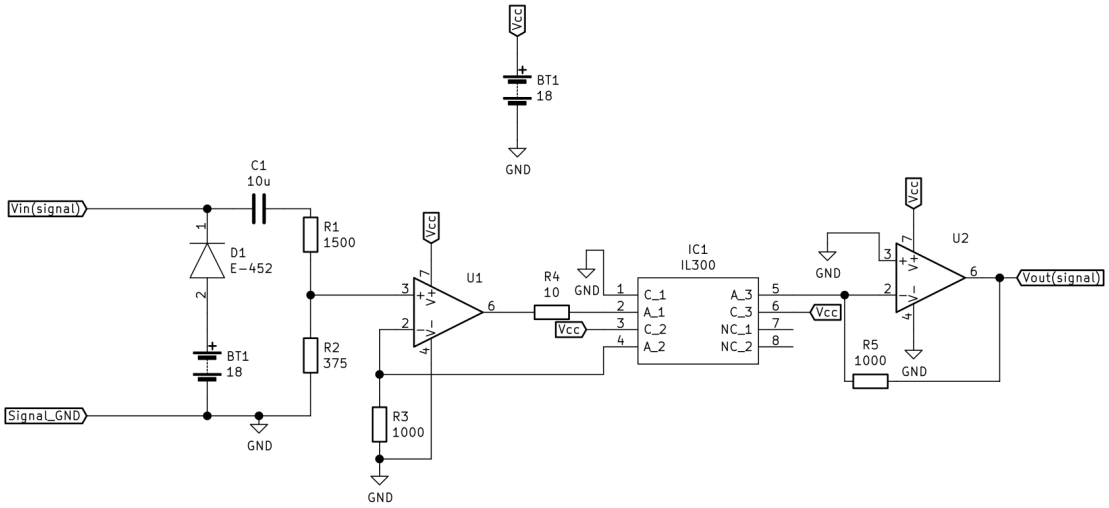


Figure 4.8: Developed circuit for single channel IEPE/ICP power and signal conditioning.

signal data, display it to the user in order to be accepted or rejected, and finally save it to a file. This process as a start and two possible ends.

It starts by defining the Data Acquisition (DAQ) settings. In this step the user has to specify the following parameters:

- trigger options
  - delay
  - threshold
- file name
- frequency range
- number of channels
- number of nodes
- number of samples.

After the user defines the previous settings, communication between the *Raspberry Pi* and the audio card is initiated. If the audio card is properly setted up, the *Raspberry Pi* receives a response, in the case of no response the user must redefined the DAQ settings.

Following the response, acquisition is initiated. After the signal is acquired the sample is plotted, allowing the user to assess the sample and decide if it will be kept or eliminated. This cycle is repeated a number of times equal to the number of samples defined by the user.

Finally, after acquiring the last sample, the DAQ stops and all the accepted samples are saved in the respective file. To help understanding the DAQ algorithm, it is described in Figure 4.9.

In the case of issues with communication between the *Raspberry Pi* and the audio card, the user must shutdown the *Raspberry Pi*. After the shutdown, the user must

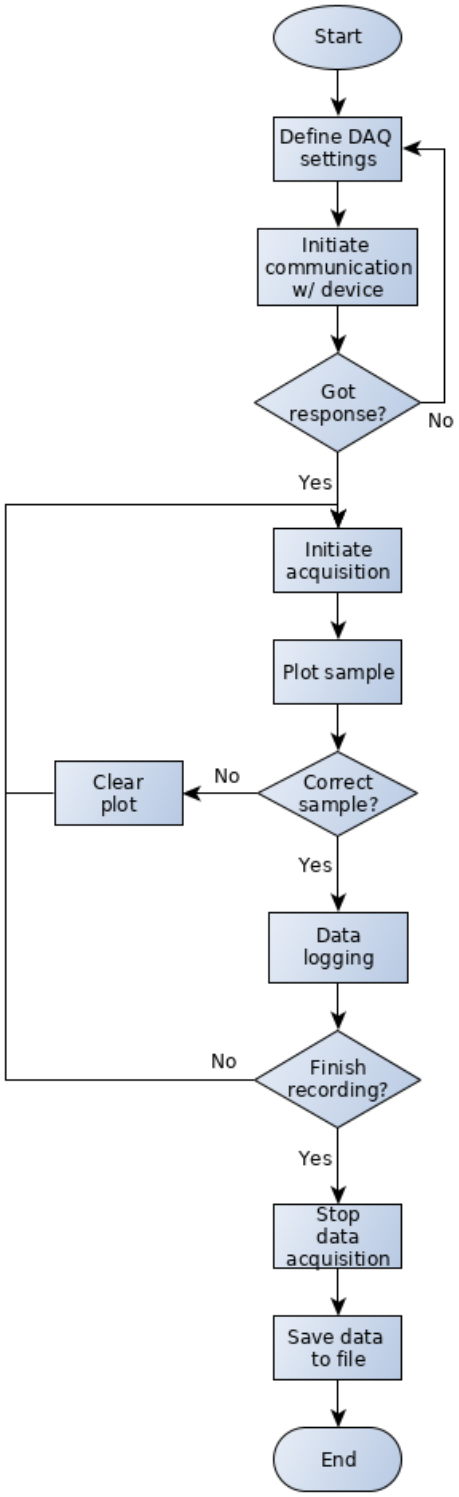


Figure 4.9: Flowchart of the data acquisition.

reconnect the audio card making sure that the P5 connector is making perfect contact between the audio card and the *Raspberry Pi*.

4.4 Software

The software solution was developed in order to achieve the initial purposed objectives.

The main functions of the program are the following:

- Signal acquisition and generation
- Windowing
- FFT
- Spectral functions
  - PSD
  - Auto-power and Cross-power spectrum
- FRF
  - H_1, H_2 and H_v estimators
  - Coherence function

It was also implemented spectrogram, correlation, cross-correlation and autocorrelation functions.

The graphical interface component of the software was developed using Python and the PyQt binding tool, for the Qt framework. The graphical windows were developed using a custom class and the PyQtGraph module.

All the functions defined in the objectives were implemented using the NumPy and the SciPy libraries for the Python programming language.

The program is divided into 5 main parts. The home, wave/signal generator, measure, analysis and FRF pages. The home page is the initial page when the program starts and has an abstract of the main functions, as seen in Figure 4.10

In the wave generator options page, the user can create sine, square, triangular and sawtooth waves and white noise signals. In order to generate the signal, the user has to define a few characteristics of the signal, such as frequency, amplitude, number of samples, sampling rate and which type of signal is going to be generated, as represented in Figure 4.11. There is also the option to save the signal and using it as an output.

The measure page, depicted in Figure 4.12, is used for signal acquisition. In this page the user has to define the trigger and the measure options. As represented in Figure 4.13a, while using the trigger options, the user has to choose between a positive or a negative slope, delay and threshold value so that the trigger function works correctly. In the measure options, as seen in Section 4.3, the user has to choose the sampling rate suitable for the acquisition and the number of channels, nodes and samples, as described in Figure 4.13.

In the data acquisition page, the user has the option to preview the acquired data before saving it. Besides the buttons to either keep or delete the acquired signal, there

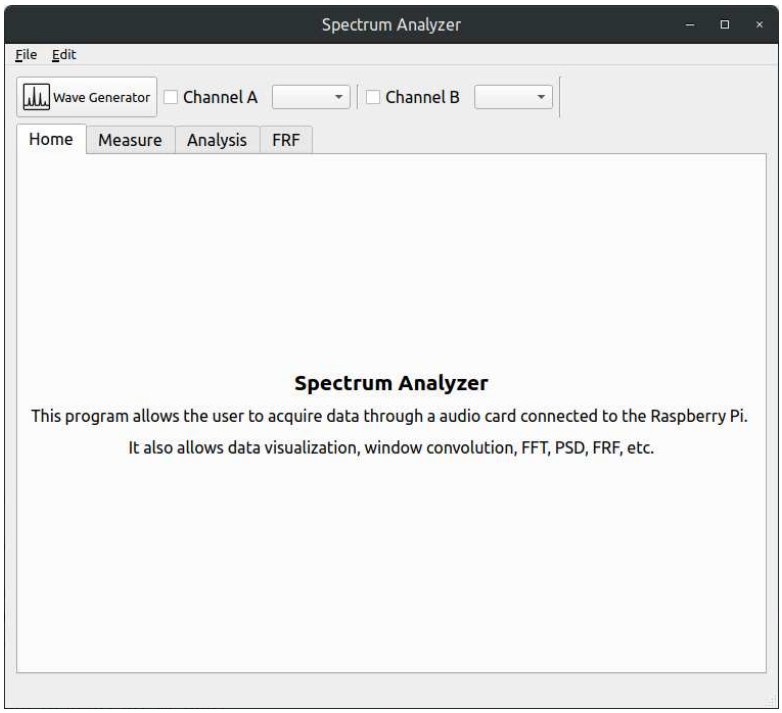


Figure 4.10: Home page of the software solution developed.

is also a counter allowing the user to know how many measures are left to do, as seen in Figure 4.14.

For the analysis window, the main functions used in signal/vibration analysis were implemented, each with their respective button, such as FFT, PSD, spectrogram and data plotting. A multiplot tool was also implemented, allowing the user to display multiple plots side-by-side, as seen in Figure 4.15a. Finally, there is the option to apply windows to the signals, like rectangle, flat-top, Hanning, Hamming and exponential, as depicted in Figure 4.15b.

For the final main part, there is the FRF page, as seen in Figure 4.16. This page is used for the FRF analysis. As refered in the beginning, the functions implemented were the H_1, H_2 and H_v estimators and the coherence function.

Since the user has the option to acquire data of impact tests on multiple nodes, it was also implemented a solution to display the FRF functions on the different nodes saved in the test file. In order to select the data for each node, the user just has to click the respective node.

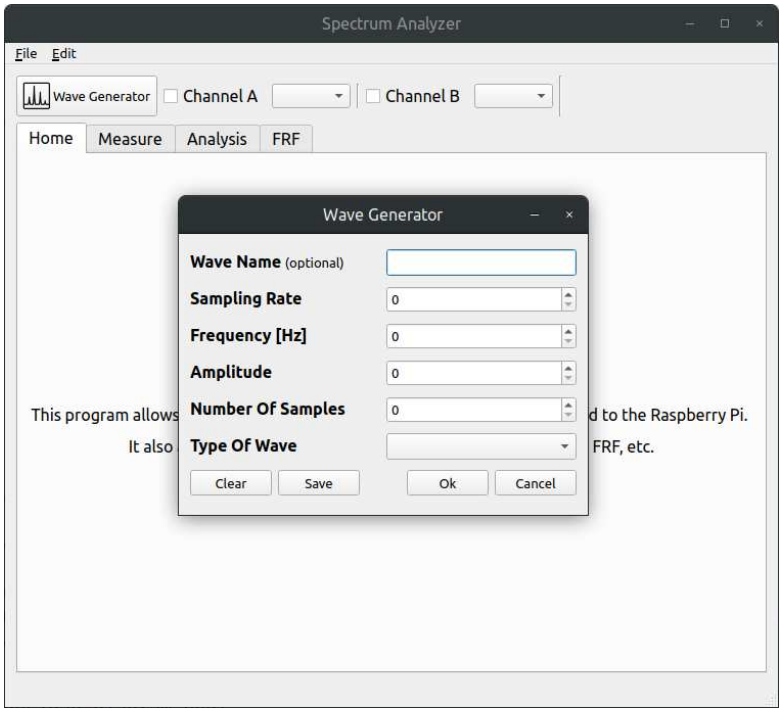


Figure 4.11: Wave Generator options.

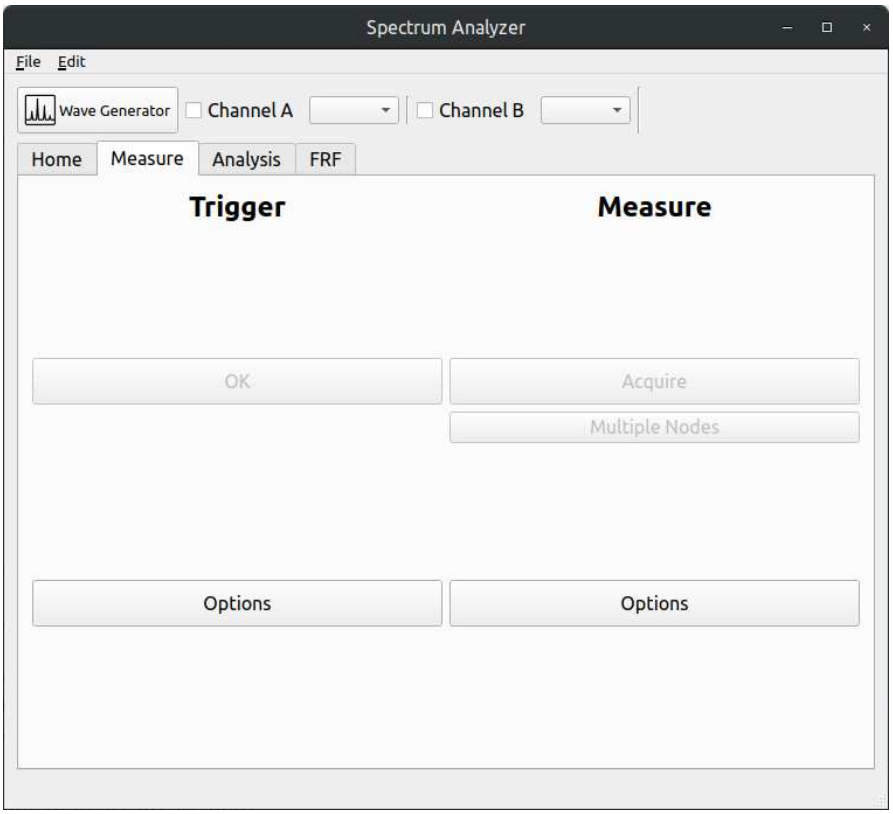
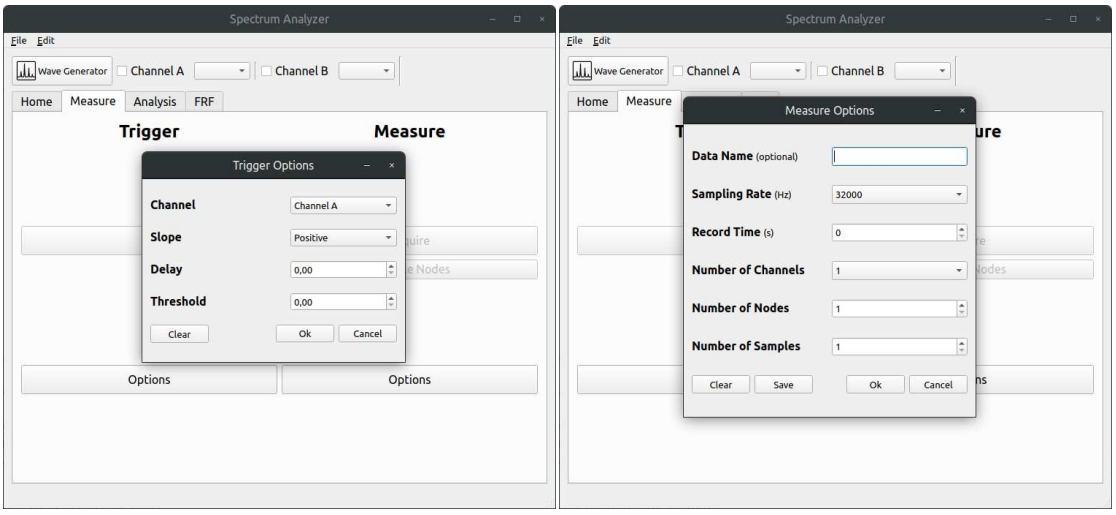


Figure 4.12: Measure page, used for signal acquisition.



(a) Trigger options. (b) Measure options.

Figure 4.13: Trigger and acquisition options used for signal measuring.

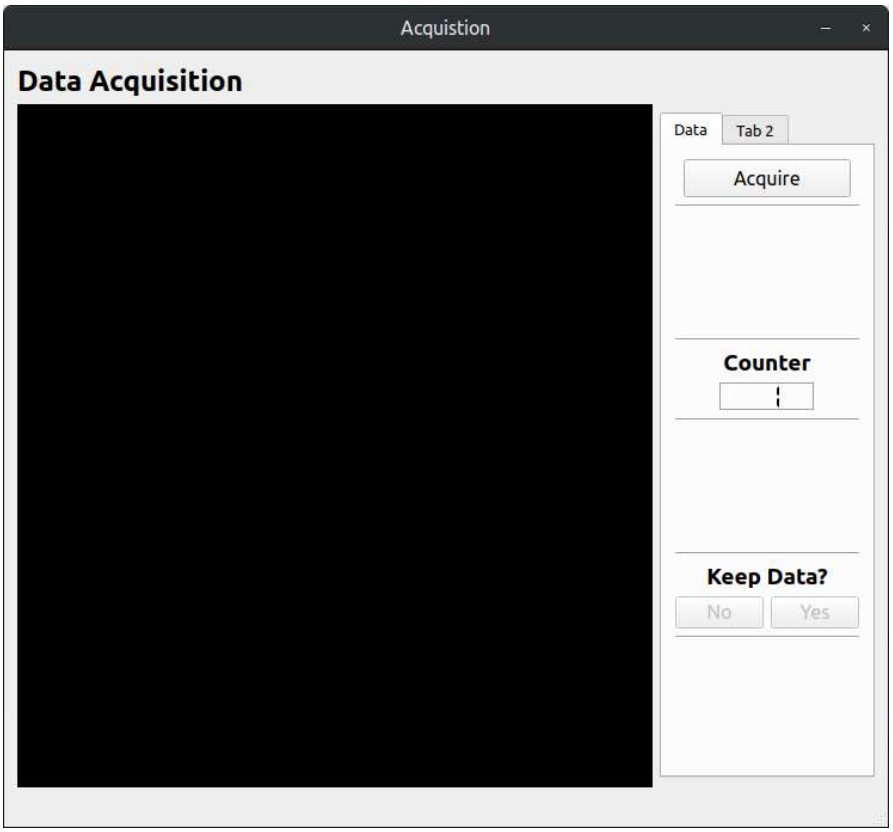
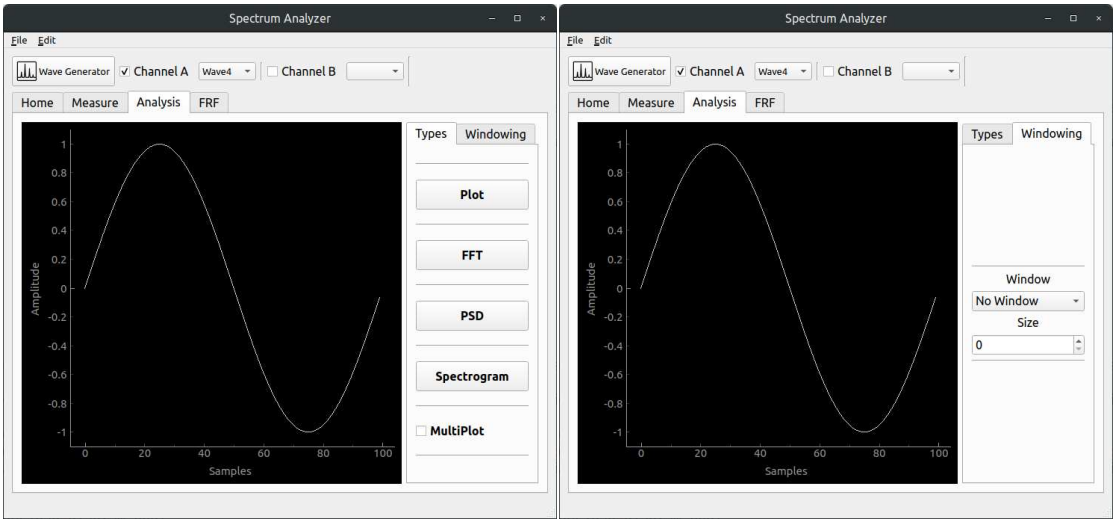


Figure 4.14: Data Acquisition page.



(a) Implemented functions for signal analysis. (b) Windowing functions.

Figure 4.15: Analysis page.

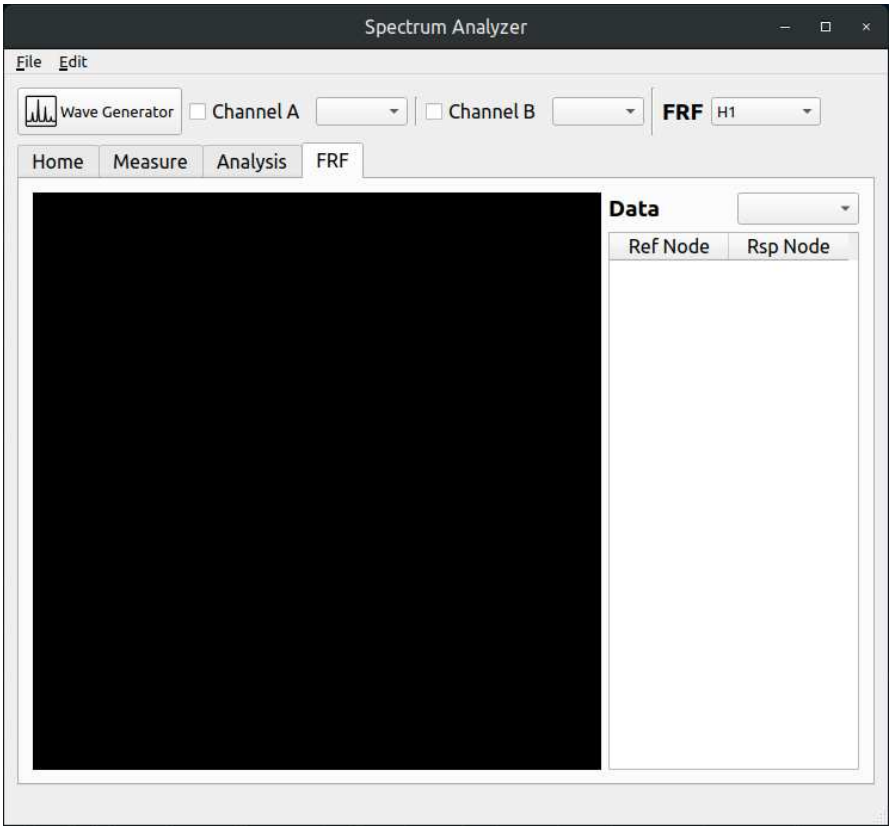


Figure 4.16: FRF page.

Chapter 5

Tests & Results

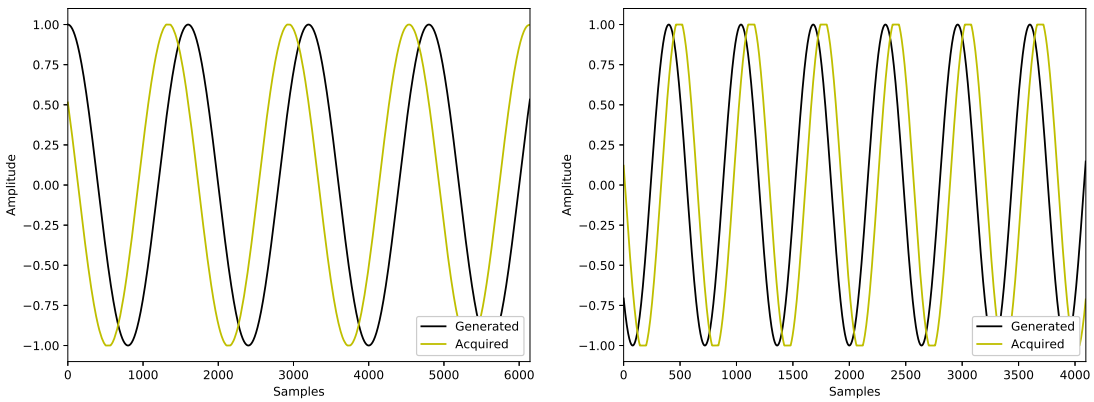
In this chapter the results gathered from the tests performed on the developed solution will be presented.

In order to get a better understanding of the audio card's capabilities, multiple sine waves were generated, from 5 Hz up to 10 kHz, and later acquired by the device. From Figure 5.1 to 5.3 some of the acquired signals are displayed with their respective generated signal. Besides the plotting, the FFT algorithm was used to obtain the frequency of the signal, and with that the relative errors were obtained.

With the true value being x and the measured value x_0, the relative error is defined by the following equation:

$$\delta x = \frac{\Delta x}{x} = \frac{x_0 - x}{x} = \frac{x_0}{x} - 1 \tag{5.1}$$

where Δx is the absolute error. In order to obtain the percentage error, the relative error is multipied by 100%.

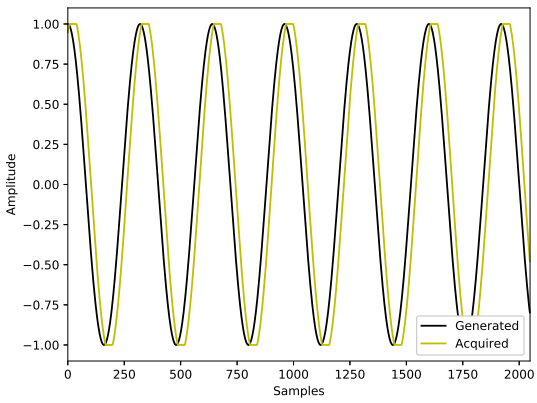


(a) Sine wave of 20 Hz.

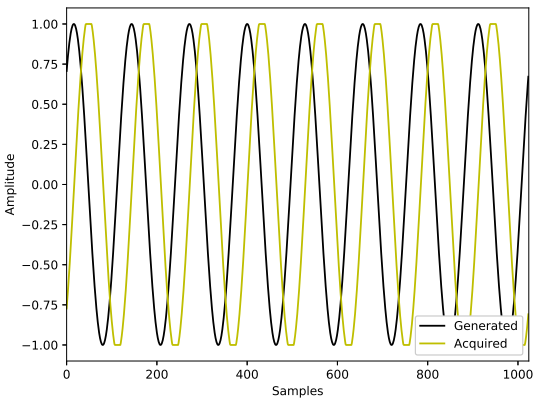
(b) Sine wave of 50 Hz.

Figure 5.1: Signals of 20 Hz and 50 Hz, with a 32 kHz sampling rate.

The information about the frequency range of the audio card, available in the datasheet, mentioned that tests were performed starting at 20 Hz. With that in mind, and in order to obtain a more concrete knowledge about the limits of the audio card, the percentage error was calculated for frequencies below 20 Hz.

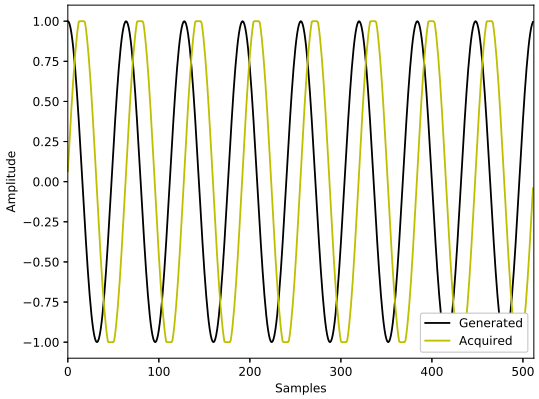


(a) Sine wave of 100 Hz.

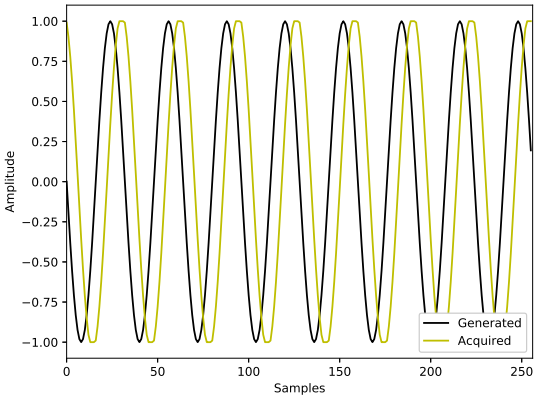


(b) Sine wave of 250 Hz.

Figure 5.2: Signals of 100 Hz and 250 Hz, with a 32 kHz sampling rate.



(a) Sine wave of 500 Hz.



(b) Sine wave of 1000 Hz.

Figure 5.3: Signals of 500 Hz and 1000 Hz, with a 32 kHz sampling rate.

The results obtained in the range of 5 up to 20 Hz are displayed in Table 5.1. It is possible to observe that below 20 Hz, overall the error assumes values well above the acceptable ones, reaching -10.7% at 5 Hz. Therefore the audio card should not be used for frequencies lower than 20 Hz, since the information obtained from the signal may be misleading.

After gaining insight of the lower limit of audio card, sine waves from 20 Hz up to 10 kHz were generated to understand the evolution of the error across a wide range of frequencies. All the values obtained from the FFT algorithm are represented in Table 5.2. By analysing these values some initial conclusions can be gathered. The maximum percentage error obtained is 0.446%, the minimum is -0.298%, and the average percentage error is 0.0389%.

After knowing the error limits of the audio card, the percentage error was plotted against its respective frequency and the plot in Figure 5.4 was created. From this graphic it is possible to notice that, while the frequency increases, the absolute value decreases, tending to zero. Therefore the audio card can be used for frequencies between 20 Hz

Table 5.1: Percentage errors for frequencies between 5 and 20 Hz.

x	x_0	Percentage Error
5.000	4.464	-10.714
10.000	10.045	0.446
15.000	14.509	-3.274
20.000	20.089	0.446

and 10 kHz. Frequencies higher than 10 kHz were not tested, since this frequency range is sufficient for vibration analysis. Although, with a sampling rate up to 192 kHz and taking into account the sampling theorem, the audio card should be capable of acquiring signals up to 75 kHz.

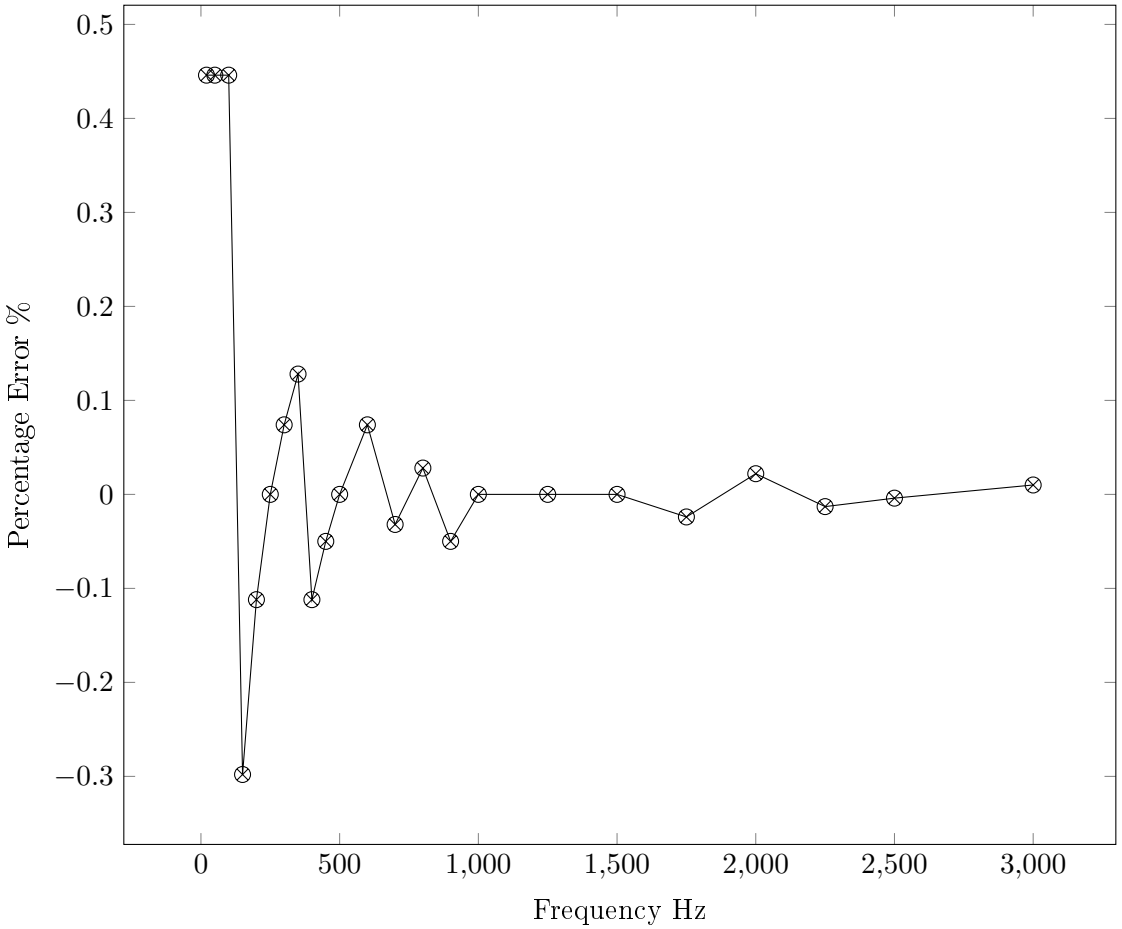


Figure 5.4: Percentage Error plotted against frequency.

5.1 Impact test

After the initial tests, that allowed for a deeper understanding of the audio card's capabilities, it was necessary to understand how the audio card behaved in vibration

Table 5.2: Percentage errors for frequencies between 20 and 10000 Hz.

x	x_0	Percentage Error
20.000	20.089	0.446
50.000	50.223	0.446
100.000	100.446	0.446
150.000	149.554	-0.298
200.000	199.777	-0.112
250.000	250.000	0.000
300.000	300.223	0.074
350.000	350.446	0.128
400.000	399.554	-0.112
450.000	449.777	-0.050
500.000	500.000	0.000
600.000	600.446	0.074
700.000	699.777	-0.032
800.000	800.223	0.028
900.000	899.554	-0.050
1000.000	1000.000	0.000
1250.000	1250.000	0.000
1500.000	1500.000	0.000
1750.000	1749.573	-0.024
2000.000	2000.435	0.022
2250.000	2249.707	-0.013
2500.000	2499.902	-0.004
3000.000	3000.293	0.010
5000.000	4999.805	-0.004
10000.000	9999.660	-0.003

analysis, specifically in an impact test. Therefore a commonly impact test was performed. The test consisted in the excitation of a steel tube, with a square section of 50x50x3 mm, using an impact hammer and an accelerometer.

The accelerometer used was a BK4508 from *Brüel & Kjær*. This is a piezoeletric accelerometer designed for modal and structural analysis, with a frequency range of 0.1-8000 Hz and a constant current of 4 mA.

The impact hammer used was a 9722A from *Kistler*, suitable for modal analysis, with a mass of 100 g.

As mentioned in Subsection 4.2.3, from the previous chapter, the initial implementation included an IEPE/ICP power interface with optical coupling. Unfortunately, due to the fact that the necessary components were not available in a timely manner, it was crucial to find an alternative. To do so, a DeltaTron power supply, depicted in Figure 5.5, was used.

Since this power supply only has one input and one output, it was not possible to power the accelerometer and the impact hammer at the same time. To solve this problem two test cases were developed.

The first one, depicted in Figure 5.6, consisted in powering the accelerometer with the

Figure 5.5: DeltaTron power supply, model WB 1372.

DeltaTron and using a BNC (Bayonet Neill Concelman) tee connector to split the signal to the audio card and spectrum analyzer, while the analyzer powered and acquired the signal from the impact hammer. The second test case, as seen in Figure 5.7, is similar to the first one, but with the accelerometer and the impact hammer swapped, thus the spectrum analyzer powered the accelerometer and the IEPE/ICP interface the impact hammer.

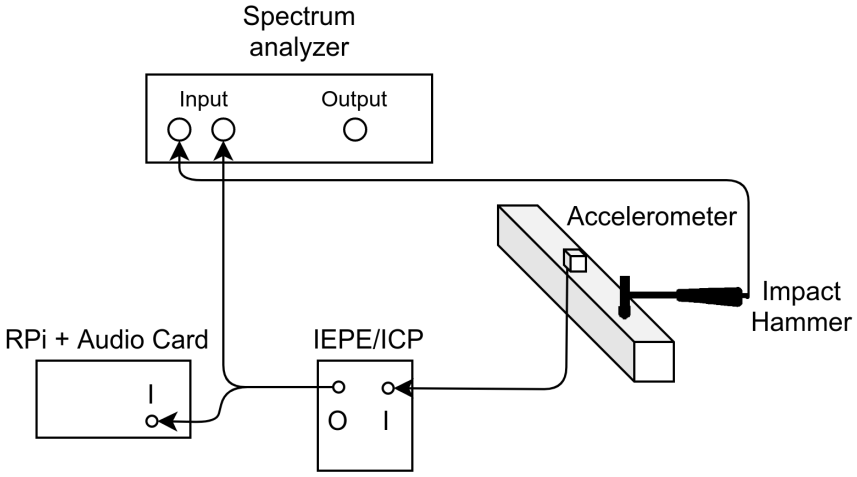


Figure 5.6: First test case.

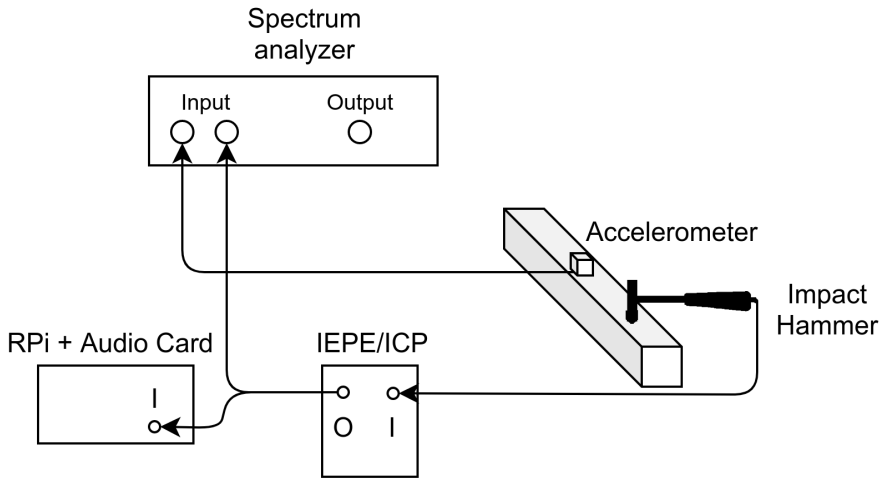


Figure 5.7: Second test case.

The objective of these tests was to attain the FRF of the system, using the acquired signals from the spectrum analyzer and the audio card and comparing them. Since the analyzer has two inputs it is possible to acquired both signals and calculate the FRF of the system. However, taking in consideration the restraints already refered, the audio card is only capable of acquiring one signal. Therefore, in the first test case it uses additionally the signal of the impact hammer acquired by the analyzer, and in the second one the signal of the accelerometer.

The results obtained from the first test case are displayed in Figure 5.8. Regarding the figure, the top two graphs are related to the signal acquired by the spectrum analyzer, and the other two are related to the signal acquired by the audio card. At the same time, the graphs on the left represent the acquired signal and on the right the respective PSD of both signals. Analyzing the accelerometer signal acquired by the analyzer, the amplitude is around ± 4 volts, on the other hand, the amplitude of the signal acquired by the audio card is ± 1 volt, leading to loss of information. This is due to the fact that the audio card is limited to 0.9 V_{rms}.

It was not possible to obtain a viable FRF of the system, using the audio card. The potential causes for this are the overvoltage in the signal, leaving the card with a truncated signal, losing information and the differences in the sampling rates, since the spectrum analyzer had a 5120Hz and the audio card a 32kHz.

Nevertheless, by analyzing the graphs on the right, it is possible to verify that the information obtained regarding the frequency is identical to one another, having the same peaks in the correct frequency.

In the second test case, since the amplitude of the impact hammer was well bellow the maximum voltage of the audio card, the signal was acquired without loss of information, allowing for the FRF of the system to be obtained. Both FRF functions are depicted in Figure 5.9, and it is possible to observe that the signals overlap each other in multiple ocasions. Even when the magnitude levels are quite different, the corresponding frequency for both FRF functions is the same, allowing for a satisfactory analysis, as depicted in Figure 5.9.

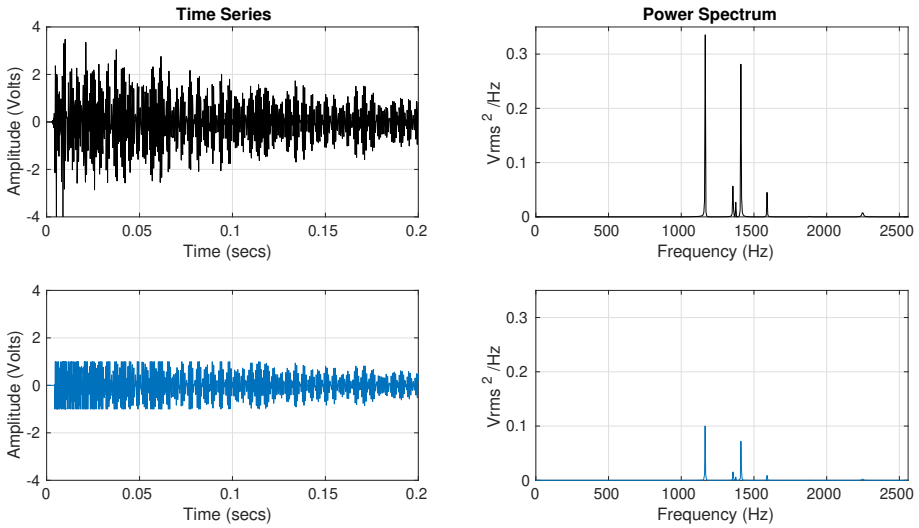


Figure 5.8: Results from the first test case.

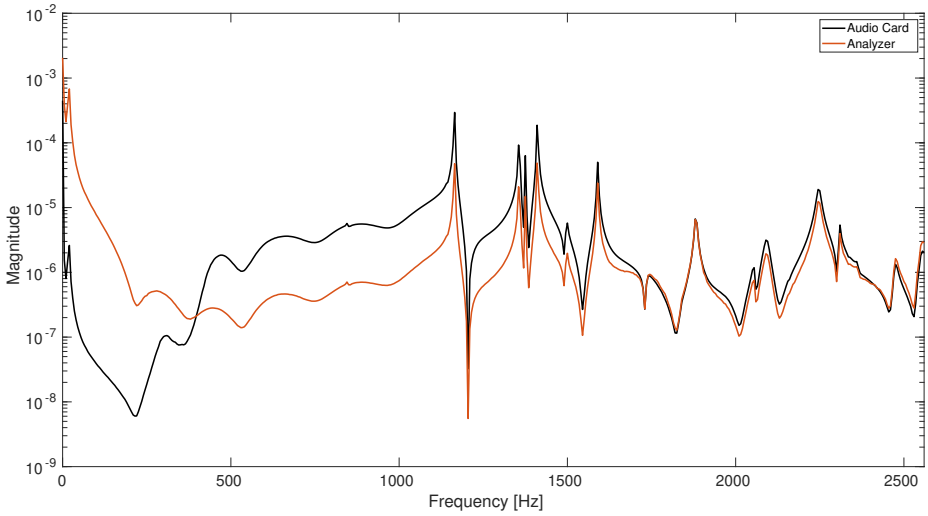


Figure 5.9: Frequency-response function of a single-input/single-output hammer excitation.

Intentionally blank page.

Chapter 6

Conclusions

6.1 Final Conclusion

The primary objective of this project was to create a spectrum analyzer, based on the Raspberry Pi development board and an audio card. Before starting the hardware and software implementation, it was necessary to understand the basic principles of vibration analysis and digital signal processing, as well as what is a spectrum analyzer and how it works, and of course the theory subjacent to it.

After gathering the required principles, the software solution was implemented, taking into considereation the different solutions already available. Afterwards, the software was tested in the development board, before using the audio card. Lastly, the audio card was introduced, a few tests were performed, in order to understand the capabilities of the solution created, but also possible problems.

To do so, two tests were performed. The first one consisted in using a function generator to create sine functions with frequencies starting at 5 Hz up to 10 kHz, allowing to understand the limits of the audio card, being these the lower limit of 20 Hz and the maximum input voltage. The second test, consisted in acquiring the FRF of a system (steel square tube) with the audio card and a spectrum analyzer, in order to compare the results. This was divided into two parts, one for the accelerometer and other for the impact hammer signals. Due to limitations the FRF from the accelerometer could not be obtained, while the one from the impact hammer could and with promising results.

Although the audio card did not behaved as expected in the impact excitation, it allowed to confirm the capabilities of the Raspberry Pi. A few details could have contributed to the failed test with the accelerometer, such as loss of information due to overvoltage and perhaps the difference between the sampling rates of the audio card and the spectrum analyzer. Due to timing limits, it was not conceivable to experiment with other test cases in order to better understand the existing problem.

6.2 Future Work

After analyzing the developed solution, as well as the obtained results, some improvements were thought of in order to enhance the solution created.

The first suggestion for future work is to switch the current Raspberry Pi for an early version, allowing for better performance and usability. Another suggestion is to improve user interface, allowing for a better experience and faster analysis. A third suggestion

is to implement a headless configuration, granting the user with the option to use the solution to only acquiring the signals, without the use of external peripherals, like a monitor, a keyboard and a mouse.

The last suggestion for improvement is to replace the audio card with a developed solution, which may include a microcontroller to receive commands from the Raspberry Pi and passing them to the ADC's and the DAC's. Also, incorporate the IEPE/ICP power needed for the accelerometers and impact hammers.

References

[1] S. G. B. David J. Ewins, Singiresu S. Rao, *Encyclopedia of vibration: three volumes set*. Engineering, Academic Press, 1st ed., 2001.

[2] W. Thomson, *Theory of Vibration with Applications*. CRC Press, 4 ed., 1996.

[3] S. Smith, *Digital Signal Processing: A Practical Guide for Engineers and Scientists*. Elsevier, 2013.

[4] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Pearson, 3 ed., 2010.

[5] "Keysight technologies website." `https://www.keysight.com/zz/en/home.html`. [Online; accessed 19-July-2019].

[6] "Microwave journal website." `https://www.microwavejournal.com/articles/print/27120-part-3-overcoming-rfmicrowave-interference-challenges-in-the-field-using-rtsa`. [Online; accessed 19-July-2019].

[7] J. Carreira, *Hardware implementation of a spectrum analyzer based in SDR*. 2010. [Available; accessed 21-July-2019].

[8] R. A. Witte, *Spectrum and Network Measurements*. Institution of Engineering and Technology, 2014.

[9] F. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proceedings of the IEEE*, vol. 66, pp. 51 – 83, 02 1978.

[10] P. Avitabile, *Modal testing : a practitioner's guide*. John Wiley and Sons Ltd : The Society for Experimental Mechanics, first edition ed., 2018.

[11] J. B. J. Fourier, *Mémoire sur la propagation de la chaleur dans les corps solides*, vol. 2 of *Cambridge Library Collection - Mathematics*, pp. 213 – 222. Cambridge University Press, 2013.

[12] F. E. H. George B. Arfken, Hans J. Weber, *Mathematical Methods for Physicists: A Comprehensive Guide*. Academic Press, 7 ed., 2013.

[13] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[14] D. Thorby, *Structural Dynamics and Vibration in Practice: An Engineering Handbook*. Butterworth-Heinemann, 2008.

[15] A. Brandt, *Noise and Vibration Analysis Signal Analysis and.* John Wiley & Sons, 2011.