CRANFIELD UNIVERSITY

IKER FERNANDEZ DE LARREA

INTERPI - A PYTHON OPEN-SOURCE PROJECT FOR LASER INTERFEROMETRY USING A RASPBERRY PI - PART 1: HARDWARE CONTROL

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
Computational and Software Techniques in Engineering
Digital Signal and Image Processing

MSc
Academic Year: 2015–2016

Supervisor: Dr Thomas Kissinger and Dr Thomas O. H. Charrett
August 2016

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND MANUFACTURING
Computational and Software Techniques in Engineering
Digital Signal and Image Processing

MSc

Academic Year: 2015–2016

IKER FERNANDEZ DE LARREA

InterPi - A Python open-source project for laser interferometry using a Raspberry Pi - Part 1: Hardware Control

Supervisor: Dr Thomas Kissinger and Dr Thomas O. H. Charrett
August 2016

This thesis is submitted in partial fulfilment of the requirements for the degree of MSc.

ABSTRACT

This MSc thesis describes the implementation of an open-source Signal Processing Unit, required in any laser interferometer, using low-cost and widely accessible hardware, more concretely, on a Raspberry Pi and on a compatible audio card. The primary concern of this project is to interface with the audio card in such a way that there is a predictable latency between modulation and demodulation, as this is a necessity for the signal processing approach.

This project forms part of an open-source initiative by Engineering Photonics at Cranfield University to widen the usage of laser interferometry, in this case by drastically reducing the prices of the Signal Processing Unit, to perform optical displacement measurements at nanometre levels or optical fibre sensor interrogation.

Keywords

Laser Interferometer; Open-Source; Raspberry Pi; Fixed Latency

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincere thanks to all the people that supported me during the development of this thesis.

In particular I would like to thank my supervisor Thomas Kissinger and the co-supervisor Tom Charrett for their effort, guidance, help and commitment; which have been of great importance for the completion of this thesis.

I would also like to thank to my family and friends for their help through the course of this thesis and for getting me motivated whenever I needed it.

TABLE OF CONTENTS

ABSTRACT iii
ACKNOWLEDGEMENTS iv
TABLE OF CONTENTS v
LIST OF FIGURES vii
LIST OF TABLES ix
LIST OF EQUATIONS x
LIST OF ABBREVIATIONS xi
1 INTRODUCTION 1
2 LITERATURE REVIEW 3
2.1 Applications 3
2.2 Platform 5
2.2.1 Raspberry Pi 6
2.3 Sound Cards Used in Data Acquisition 8
2.3.1 Cirrus Logic Audio Card 9
2.4 Latency in Audio Recording 10
2.5 Software 11
3 LATENCY REQUIREMENTS 13
3.1 Theoretical Considerations 13
3.2 Latency Measure 14
3.2.1 Method 14
3.2.2 Audio Input/Output Modes 17
3.3 Alternative approach 17
3.3.1 Results 18
3.4 Conclusions 27
4 SIGNAL PROCESSING UNIT 29
4.1 Theoretical Approach 29
4.1.1 Overview 29
4.1.2 Mathematical Development 31
4.1.3 Graphical Results 33
4.2 Implementation 36
4.2.1 Practical Considerations 36
4.2.2 Working Modes 42
4.2.3 Errors Occurring During Implementation 44
5 RESULTS 45

5.1 Dependent on the Range of the Channels 46
5.2 Evolution with time 48
5.3 Performance 51
5.3.1 Validity 51
5.3.2 Computational Cost 51
6 FUTURE WORK 53
6.1 Technical Improvements 53
6.2 Organization Improvements 54
7 CONCLUSIONS 56
REFERENCES 57
APPENDICES 61
Appendix A Hardware Requirements 61
Appendix B Software Set Up 63
Appendix C User Guide 64

LIST OF FIGURES

3.1 Latency Measure Configuration 15
3.2 Latency with Largest Values; $f_s = 8kHz$; 450 experimental repetitions; Single stream in Block mode; 10 bins 19
3.3 Short Term Latency; $f_s = 8kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins 22
3.4 Short Term Latency; $f_s = 192kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins 22
3.5 Short Term Latency; $f_s = 44.1kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins 23
3.6 Short Term Latency; $f_s = 48kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins 23
3.7 Long Term Latency; $f_s = 8kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins 25
3.8 Long Term Latency; $f_s = 192kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins 25
3.9 Long Term Latency; $f_s = 44.1kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins 26
3.10 Long Term Latency; $f_s = 48kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins 26
3.11 Latency for another carrier frequency, $f_s = 8kHz$; 450 experimental repetitions; Single stream in Callback mode 27

4.1 Flow diagram of the demodulation process 30
4.2 Generated signal that is sent to the laser for each modulation period 34
4.3 Received test and real signals 35
4.4 Window and carrier signals 35
4.5 Real part of the demodulated measured signal before LP 36
4.6 Low pass filter of the Cirrus Logic Audio Card 38
4.7 Working modes without fixed latency 39
4.8 Ten superimposed plots of the working modes with fixed latency 40
4.9 Working modes with fixed latency 40
4.10 Zoom of fixed latency 41

5.1 General set up for the measurements done 45
5.2 Results as a function of range for the manual mirror 46
5.3 Signal strength when hand is inserted 47
5.4 Results for different ranges with the movable mirror 47
5.5 Signal strength as a function of range for the electrically moving mirror 48
5.6 Evolution of the results with time 49
5.7 Evolution of the deviation with time 49

5.8 Mirror moving following a triangular signal 50
5.9 Sine signal with 1.85 µm peak-peak displacement.......................... 51

1 Developed Software.. 64

LIST OF TABLES

2.1 Raspberry Pi models 7

4.1 Low Pass Filter of the Cirrus Logic Audio Card - fs = 44.1kHz 38
4.2 Low Pass Filter of the Cirrus Logic Audio Card - fs = 192kHz 38

1 Hardware Cost 62

LIST OF EQUATIONS

(3-1) Outgoing Sine 15
(3-2) Latency in Seconds 16
(3-3) Latency in Samples 16
(3-4) IQ Decomposition........ 16
(3-5) Phase Calculation 16
(3-6) Chunk size........ 28
(4-1) Modulator Sine 31
(4-2) Received Signal........ 31
(4-3) Window Signal........ 32
(4-4) Carrier Signal 32
(4-5) Demodulated Signal 33
(4-6) Latency Dependence 33
(4-7) Bandwidth 37
(5-1) Deviation Calculation 48

LIST OF ABBREVIATIONS

SATM	School of Aerospace, Technology and Manufacturing
MSc	Master of Science
LIGO	Laser Interferometer Gravitational-Wave Observatory
LISA	Laser Interferometer Space Antenna
ESA	European Space Agency
NASA	National Aeronautics and Space Administration
GUI	Graphical User Interface
MB	Mega Byte
USB	Universal Serial Bus
GPIO	General Purpose Input/Output
RAM	Random Access Memory
HDMI	High-Definition Multimedia Interface
CPU	Central Processing Unit
PC	Personal Computer
AC	Alternating Current
DC	Direct Current
SPDIF	Sony/Philips Digital Interface Format
ALSA	Advanced Linux Sound Architecture
IO	Input Output
LP	Low-Pass Filter
BW	BandWidth

1 INTRODUCTION

This document presents the development of an open-source Signal Processing Unit implemented on a Raspberry Pi with a Cirrus Logic Audio Card, and constitutes a MSc degree thesis project of the course Computational and Software Techniques in Engineering with speciality of Digital Signal and Image Processing. This project forms part of an open-source initiative by Engineering Photonics at Cranfield University to widen the usage of laser interferometry.

The primary objective of this project is to control and demodulate laser interferometric measurements using only the previously mentioned Raspberry Pi and the Cirrus Logic sound card, with all developed software in the Python language. Therefore, the main objective of this project is to realize an efficient implementation of the proposed demodulation algorithms.

The impact of this project could be high, as together with other projects developed in the Engineering Photonics group, it could ease the use of laser interferometers in both academic and private environments, as it could reduce the price of the required signal processing units to a much more affordable level. Concretely, this project forms part of a bigger project called Inter-Pi, in which the control and visualization of the data obtained in this project will be possible by an external GUI based on a web-server approach.

The project that is described in this document is concerned both with the hardware interfacing and signal demodulation on the Raspberry Pi. More precisely, the primary issue to be solved is to interface with the Cirrus Logic sound card that drives and records the laser interferometric signals in such a way that there is a predictable latency between modulation and demodulation, as this is a necessity for the signal processing approach.

The structure of the presented report is the following. First of all, a literature review is presented, where the previous work related with the problem to be solved is analysed. Then, the primary concern of this project, that is to know if the output and the input of the audio data can be implemented in a way such that the delay between them is fixed, is analysed.

After that, the theory behind the signal processing that is applied to the optical signals is explained; followed by the implementation of this theory onto the hardware that is going to be used. Once the implementation has been successfully performed, some results are presented to prove the validity and quality of the designed signal processing unit. Finally, some future work that can be done to expand the scope of this project together with the conclusions are discussed.

Apart from that, three appendices are presented at the end of this document, in which the hardware requirements, the software set up and a user guide of the developed Signal Processing Unit are detailed.

2 LITERATURE REVIEW

A laser interferometer, also called an optical interferometer, is a precise instrument that can solve many measurement challenges that may arise within engineering, biomedical and applied science applications [1–3].

Historically, the concept of the optical interferometer was introduced at the end of the 19th century. Some classical interferometric configurations are the Michelson or the Mach-Zehnder interferometer, the Fizeau-Laurent surface interferometer, and the Fabry-Perot or variable-gap interferometer [4, 5]; which are still used in the optical industry.

The basic operational principle of an optical interferometer consists of dividing a beam of light into two beams travelling along different paths. In the detector, those beams will be added again, thus forming constructive or destructive interferences that produce local maxima and minima; hence creating a pattern formed by bright and dark areas. This information can then be used to determine precisely the parameters that are required; for example, distance.

Nevertheless, although the concept of laser interferometry was firstly introduced over one hundred years ago, it is a concept that is still under investigation in order to improve both the quality and the order of resolution of the obtained measurements. [6].

2.1 Applications

The use of interferometers have applications in a wide variety of real-world problems. For example, the technology of laser interferometer is actually being used to detect and prove the existence of gravitational waves that were predicted by Albert Einstein in two different projects, namely *LIGO* and *LISA* [7, 8].

On the one hand, the Laser Interferometer Gravitational-Wave Observatory (LIGO) project has as a main goal the detection and study of astrophysical gravitational waves and use data from them for research in physics and astronomy [7]. On the other, the Laser Interferometer Space Antenna (LISA), a joint space project of ESA and NASA, aims the detection and measurement of gravitational waves by laser interferometry [8].

Nevertheless, and as this project in particular is located within the Engineering Photonics group at Cranfield University, its utility will be focused in the measurements that are currently being performed in this group, such as applications in engineering; mainly, optical displacement measurements [9] at nanometre levels or optical fibre sensor interrogation [10, 11].

Generally speaking, the interferometers that are available in the actual market are expensive and often not affordable for many researchers. Considering all the possible fields of application that laser interferometers have nowadays and in order to solve this problem of accessibility, derived from the price factor, a cost-effective and open-source alternative for signal processing unit of the laser interferometer could have a great impact on the use of this technology. This open-source solution will provide an opportunity to expand the investigation of particular or public researches in this particular field, such as universities, massively.

Moreover, apart from offering a cheaper alternative to actual laser interferometer signal processing units, the proposed solution should continue providing accurate results considering the particular requirements of the interferometry together with being as user friendly as possible; which makes the decision of which technology to use of great importance.

As it will be explain later in Section 2.4, the most important requirement for this project to be successful is the necessity of having a predictable latency between modulation and demodulation, as this is a necessity for the signal processing approach [10].

2.2 Platform

As stated before, one of the most important parts of the project is to select what piece of hardware is chosen in order to implement the proposed solution; taking into account that the designed interferometer signal processing unit should be cost-effective and easy to use for the final user, as well as allow the implementation of an open-source solution.

Therefore, the two most used open-source hardware tools in academic and particular research nowadays, Raspberry Pi and Arduino, have been evaluated in order to select one of them. Although the working bandwidth of an FPGA implementation could be higher than the one of these two tools, it has been discarded because of its higher cost and the difficulties it presents for developing an open source implementation.

The main difference between Arduino and Raspberry Pi is that the first one is a micro-controller and the second one is a mini-computer. This difference makes the Arduino perfect for prototyping electronics projects, due to its ease of connection to external components and sensors. Although the Arduino can be programmed with small C applications, it presents limitations when more complex tasks are required to be performed, such as Ethernet networking [12].

On the other hand, the Raspberry Pi requires an operating system to be functional; but it is a very powerful platform that allows the performance of more difficult tasks due to the peripherals that are presented on it. These characteristics make this platform perfect for embedded systems and projects requiring more interactivity and processing power [12].

Furthermore, the Raspberry Pi has proven to be an efficient tool in order to implement the transmission of different data taken from it securely to a mobile phone [13] and to a cloud server [14]; which would allow an easier coupling to an independent project in which a GUI allows the end user the display of the different parameters as well as the choosing of the different work modes.

In terms of cost, both tools can be said to be cost-effective. The price of an Arduino board is on the range of 15-35$ plus the handling cost [15], while the Raspberry Pi can cost between 5$ and 35$ plus local taxes and handling fees [16]. Therefore, it can be said the price of both products is similar and enough to consider both alternatives as cost-effective.

The decision of which hardware use in order to perform the signal modulation and demodulation required for the implementation of the open-source laser interferometer needs to be based on the required technical performance; as both platforms have a similar cost as well as being easy to use for the final user, mainly because the functional software and the required specifications of the platform will be provided in the open-source project.

It would be desirable that the required algorithms could be easily implemented, as well as allowing an easy modification of the parameters involved in them. Therefore, the final decision was to use a Raspberry Pi platform , as this tool provides a bigger flexibility gained from the use of a ready available open-source operating system as well as an easier interaction with the user.

2.2.1 Raspberry Pi

Among all the Raspberry Pi available on the market, it was required to select which model was going to be used. Although this may look trivial, the selection of the model is of great importance, so the designed system could be compatible with earlier and even future versions of the Raspberry Pi.

The models that are available in the actual market (2016) [16] can be seen in Table 2.1.

Table 2.1: Raspberry Pi models

	1 A+	**1 B+**	**Zero**
Price($)	20	25	5
Spec.	256 MB RAM 1 USB port 40 GPiO pins No Ethernet	512 MB RAM 4 USB port 40 GPiO pins Ethernet	1Ghz single-core CPU 512MB RAM mini-HDMI USB On-The-Go
Extra	Original Raspberry Pi	Revision of the original	Released in 2016 Half size of 1 A+

Table 2.1: Raspberry Pi models (*Continued*)

	2 B	**3 B**
Price($)	35	35
Spec.	1 GB RAM 900MHz quad-core ARM Cortex-A7 CPU	1.2GHz 64-bit quad-core ARM Cortex-A53 CPU WiFi 802.11n Bluetooth 4.1
Extra	Released in 2015 Compatible with 1 A+/B+	Launched in 2016

The model that is most suitable for the design of the project is the Raspberry Pi Model 2B. The main reasons for choosing this model lie in the facts of being compatible with the previous models and the time that it has already been on the market, which provides much more technical support as it has been more widely used.

Furthermore, the compatibility with the sound card that will be selected is another argument for choosing this Raspberry Pi model. Although the selected sound card model will be introduced later, it is known that the chosen sound-card is supposed to work with the Raspberry Pi versions 1 and 2, while it has not yet been tested yet for the version 3 and zero.

2.3 Sound Cards Used in Data Acquisition

One of the most important part of the scope of this project is the data acquisition process and its conversion from analog to digital once the several beams are received in the photo detector. As the data that the photo detector provides is analog it needs to be digitized in order to apply the different signal processing techniques that the laser interferometer measurement requires.

As the received data lies on the same spectral range as the audio (from 20 Hz to 4 kHz in the case of the voice and until approximately 40 kHz in case of music), treating the received data as audio to convert it from analog to digital could be a good solution. Nevertheless the native Raspberry Pi audio recording and playing capacity, i.e. audio input and output ports, is not enough when more quality or more channels are required [17]. Therefore, if improved audio quality is required an extra sound card needs to be attached.

Historically, the implementation of instrumentation based on a PC sound card has been successfully demonstrated and has allowed the development of several tools. An audio waveform generator, a two-channel digital oscilloscope and a spectrum analyser can be implemented using a PC and its sound card, as reported in [18]. As sound cards can transmit and receive digital (audio) files in two different channels, as well as being used to generate and display waveforms with the help of some piece of software; they are considered an interesting tool due to their price and availability.

In [19] the development of potentiometric sensors for acquiring an AC signal which corresponds to a scanning-DC type data acquisition system has been reported, also with the use of a PC sound card and providing experimental results which show a good promise for this instrument. The speaker port of the sound card has been used here to deliver an AC signal that is converted later with another hardware equipment into DC, while the microphone input has been used to collect the AC signal corresponding to the DC voltage steps. The selection of the sampling frequency is configurable depending on the frequency of the input signal in order to meet Nyquist theorem criteria.

Another interesting project of a data acquisition and signal processing system is reported in [20], used in capillary electrophoresis. Although this may look not of big interest for the scope of this project, the demonstration that the results obtained with this system are of the same quality or even better comparing to the ones obtained by conventional data acquisition boards, makes it interesting as it proves that data acquisition systems with a sound card can be of great use.

2.3.1 Cirrus Logic Audio Card

As stated before, and although the Raspberry Pi could be used to process audio in a effective way [21], it has some disadvantages that make it advantageous to add an extra piece of hardware to perform the audio input and output [17].

Namely, this limitations of the inbuilt Raspberry Pi audio system are the following [22]:

1. Lack of flexibility in terms of multiple types of audio input and output sources
2. There are no ways to capture audio just using a Raspberry Pi, as high-quality recording is needed. Therefore, the use of an audio card turns out to be necessary.
3. Audio output is limited to two paths:
   (a) Analogue, via its onboard 3.5 mm stereo output jack, which gives an audio quality universally recognised as being no more than acceptable
   (b) Digital, via its onboard HDMI output, which provides the potential for high quality rendering of audio but needs an extra piece of hardware to extract the audio for the whole signal

Due to its assured compatibility with the Raspberry Pi [22], the Cirrus Logic Audio Card has been selected. This product, designed by element14 and Cirrus Logic in partnership, is a replacement for the Wolfson Audio Card which was not compatible with 40-pin versions of Raspberry Pi. The main features of this particular sound card are the following [22]:

1. Compatible with Raspberry Pi B+ and A+ onwards, as it has 40-pin extended GPiO, and no P5 connector
2. Analogue line-level output and input, as well as digital stereo audio input and output (SPDIF)
3. High quality headphone output, with microphone facility
4. Back power protection, which allows the Raspberry Pi to be powered from the Cirrus Logic Audio card

2.4 Latency in Audio Recording

As stated before, the primary concern of the scope of this project for being successful is to have a predictable latency between modulation and demodulation, as this is a necessity for the signal processing approach [10].

In that paper a novel signal processing technique that allows highly linear interferometric phase measurements in a simple and self-referencing set-up was proposed, and it will be used to implement the open-source interferometer in a Raspberry Pi. This new technique uses as a source a continuous-wave laser diode with a sinusoidal optical frequency modulation; while a smooth window function is used in the demodulation to suppress unwanted signal components.

With the signal processing method proposed, if a fixed latency between the input and the output is achieved, signals from several interferometers with unequal optical path differences can be multiplexed and measured. Nevertheless, previous audio projects related with audio recording performed with a Raspberry Pi that have a concern with latency are not focused on obtaining a fixed latency, but on obtaining the lowest latency possible [17, 23].

That is why the problem of obtaining a fixed latency will be addressed in the remainder of this thesis. As this particular issue is crucial, an emergency plan has also been considered and described in Sec. 3.3, to mitigate the project risk that may arise from the latency issue.

2.5 Software

The last decision that needs to be taken is the language in which the interferometer software will be programmed. As Python is the recommended programming language used with the Raspberry Pi and has proven to be enough to handle with complex algorithms [24], it is the language that is going to be selected for the implementation of the demodulation algorithms. Furthermore, this will facilitate the incorporation with the existing Python communication framework for data and parameter transfer within the Engineering Photonics group.

Another advantage of using Python is that it will be suitable for forming part of the open source project that the developed interferometer's signal processing unit is aimed to be; as Python is a cross-platform language. Nevertheless, if a C based algorithm is necessary for performance reasons it will be used although the first aim is not to do it. However, this would not be a problem as Python is capable of easily interfacing with code written in other languages.

Once the main programming language is selected, some decisions that may affect the development of the project need to be taken :

1. Which Python version is going to be used, either version 2.X or version 3.X (respectively, versions 2.7 and 3.5 in the year 2016). Although Python 3 is the newer version, to assure the compatibility of the developed software with other software within the Engineering Photonics group as well as with some external libraries, **Python 2.7** is the selected version.

2. Use the Python external libraries which need to be installed, mainly pyaudio, or use bash commands with software frameworks ALSA, which is a

part of the Linux kernel. Even though the **installation of external libraries** may lead to a higher complexity of the interferometer's signal processing unit software configuration and setting up, it is the option that has been selected. The main reason for this is due to the larger control that these library offer in the recording configuration parameters, as well as in the design and development of the signal processing algorithms.

3 LATENCY REQUIREMENTS

As stated earlier on Sec. 2.4, having a fixed latency between the input and the output is of great importance for this project to be successful. In other words, if the latency between the input and the output of the used audio card would not be constant, the proposed project would not be realisable. Therefore, measuring the latency between the input and the output of the audio card before the development of the signal processing unit is crucial, as well as taking into account that an alternative approach for the data transmission and reception could be necessary. Nevertheless, it has to be remembered the fact that the absolute value of the latency is not important as long as it is stable, that is, the signal processing can be set to any delay for proper working as long as it is stable.

3.1 Theoretical Considerations

Even though knowing the latency between the input and the output of the audio card is the most important fact to be solved in this chapter, there are some factors that influence the quality of the measurements and hence they need to be taken into account. These are related with the use of an audio card as the data acquisition system, and the particularities that this particular hardware has.

The first parameter that needs to be taken into account is the value of the sampling frequency used to acquire data with the audio card. The Cirrus Audio card, the one used in this project, can handle sampling frequency values from 8 kHz to 192 kHz, having as the default sampling frequency 44.1 kHz [22]. It is desired to have the value of the sampling frequency as large as possible, as long as the audio card is capable of dealing with this sampling rate and, more important, being able to continue working in real time.

Another parameter that may influence the obtained results is the frequency of the modulated signal, that will be used both to measure the latency and in the signal processing part (as explained in this chapter and in Ch. 4). Whilst it may be trivial what value of frequency is taken, as this has no influence in the calculations of the latency, care needs to be taken with the emphasis filters that may be activated on the audio card while working with small fundamental carrier frequencies.

Other parameter that has a high influence on the measurement is the chunk size. The chunk size characterizes how many samples are read at a time by the audio stream, as well as delimiting how much data is stored before writing it. Ideally, the chunk size should be equal to the division of the sampling frequency and the fundamental carrier frequency, so that a complete sine-wave appears for every chunk. Nevertheless, it also has to be taken into account that if the chunk size is increased, it will result in more delay due to the time it needs to read; but decreasing the size of the chunk could arise in overflow problems as writing may be faster than reading.

Finally, the volume settings of the audio card together with the amplitude of the output signal are needed to be considered in order to avoid the clipping of the input signal. The best way to check it is to save every signal and play them back in an oscilloscope, as the saturation of the input signal can easily be spotted.

3.2 Latency Measure

3.2.1 Method

In order to solve the problem of knowing if the latency is fixed, an experimental method must be used. When both the input and output latency of the audio card obtained with a random audio stream are measured with the tools provided in the Python package *pyaudio*, the value of 0.128 ms is obtained. Therefore, it can be said that apparently this latency value will be fixed. However, it should be proved analytically that this value is always fixed and independent of the value of the carrier frequency as well as the sampling rate and the number of audio channels.

With this purpose, a simple experiment has been done; which has the following main steps:

1. Create a sine of a fixed carrier frequency and sampling rate, of a length long enough to measure a number of periods that allows to conclude that the latency is fixed at all the time.

$$x = A \cdot sin(2\pi f t) \tag{3-1}$$

   with $A =$ amplitude of the signal; $f =$ carrier frequency; and $t =$ array of time instants with a number of samples equal to f_s per second.

2. Send the sine from the output port of the audio card, and directly connect this port with the input port with an male-male jack wire. Finally, record the values of the input sine and save them onto a file. The hardware configuration can be seen on Fig. 3.1.

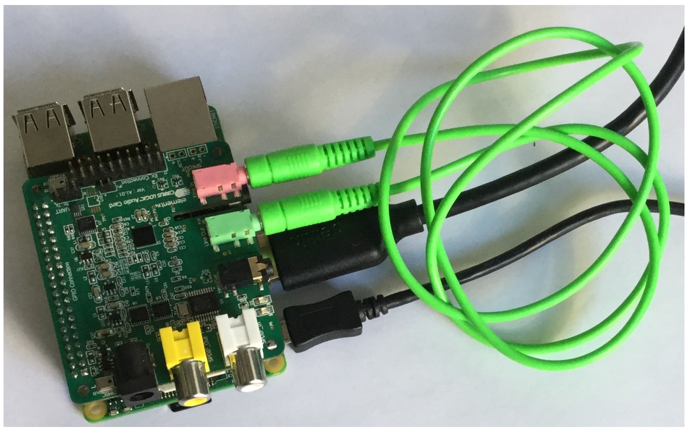


Figure 3.1: Latency Measure Configuration

3. Measure the latency of the input and the output signal, or more concretely, the phase of both signals. Once the phase shift of the signal is known, together with the carrier frequency, the latency can be computed as if it is desired to be obtained in seconds:

$$L(seconds) = \frac{phase(^\circ)}{360^\circ} \cdot period(seconds) \tag{3-2}$$

It can also be computed as if it is desired to transform the previous result into samples:

$$L(samples) = L(seconds) \cdot Fs \tag{3-3}$$

In order to measure the phase of the signal an IQ decomposition is performed. Firstly, the signal is decomposed in the components I and Q (real and imaginary) by the following formulas:

$$\begin{aligned} I &= x \cdot cos(2\pi ft) \\ Q &= x \cdot sin(2\pi ft) \end{aligned} \tag{3-4}$$

with $x =$ input signal.

Then, all the samples of I and Q are summed independently, to obtain two different values of I_{summed} and Q_{summed}. With those two values, an imaginary number can be formed as:

$$e^{I_{sum} + j \cdot Q_{sum}}$$

which phase is equal to:

$$phase(^\circ) = \arctan(\frac{Q_{sum}}{I_{sum}}) \tag{3-5}$$

3.2.2 Audio Input/Output Modes

As mentioned in Sec. 2.5, the software that has been selected for IO purposes is the python library *pyaudio*. This library offers two different ways of reading and writing data, namely, the blocking and the callback mode. In the blocking mode, the data is written and read by blocks until all the given/requested frames have been played/recorded. Alternatively, in the callback mode, the data is written whenever it is available and it is immediately processed [25].

Furthermore, it is also required to select if one or two audio streams are going to be used; that is, if the input and output are controlled by one or two different instances. Although this may look trivial, it is an important decision as it determines if the audio is handled in a full-duplex mode (with one stream) or not. For that purpose, all the alternatives have been analysed, in order to know if the obtained latency with each method is fixed or not. There are six available audio input/output modes, concretely:

1. Using one stream:
   (a) In Blocking Mode
   (b) In Callback Mode
2. Using two streams:
   (a) Both in Blocking Mode
   (b) Read in Blocking Mode and Write in Callback Mode
   (c) Read in Callback Mode and Write in Blocking Mode
   (d) Both in Callback Mode

3.3 Alternative approach

As the good performance of the implemented Signal Processing Unit is the main objective of this project, an alternative plan is needed to be taken into account in case of the latency between the input and the output of the signal is not fixed. As it will be explained on Ch. 4, if this would happen the whole signal processing unit would be useless, so a solution was needed to be considered.

This contingency plan consists on simply connecting with a wire one input and one output of the audio card, so that the instantaneous latency could be tracked in one of the audio channels; while the other channel is used for transmission and reception purposes. Once the value of the latency is tracked, it can be corrected and therefore the use of the proposed algorithms be possible again.

Nonetheless, this method has some disadvantages. The main disadvantage is that it forces the user to use another specific cable, which would need to be soldered specially with one channel looping back for tracking the latency while the other channel would go to the laser and come back from the photo detector; which would increase the price of the developed hardware as well as the complexity of the setting up process. Moreover, it could increase the computational complexity of the used algorithms.

3.3.1 Results

To check if the latency is fixed two different validation stages have to be performed. Basically, it has to be checked that the short term stability is fixed, as well as the long term stability. The short term stability, is the obtained latency when the transmission and reception is started; while the long term stability corresponds to the latency within a single stream once it is started. Nevertheless, some of the input and output modes could be discarded at first sight, as the values of the measured latency vary among a wide range of values; so that this validation process is not computed for all the possible methods.

After checking at first sight the latency values obtained with every method, the only one that seemed to give a fixed latency was when the recording and playback are made using just one stream in the callback mode. The rest of the methods have been discarded either because the recorded number of samples varied among different measurements, which makes the obtained signal inadequate; or because the latency of subsequent tests varied on unacceptable levels.

For example, for a prove for a single stream using the blocking mode, it can be seen in figure 3.2 that the latency for this kind of measurement is clearly not fixed, as its value in samples has a standard deviation of almost 9 samples, together with a range of possible values from 140 samples latency to 215 samples.

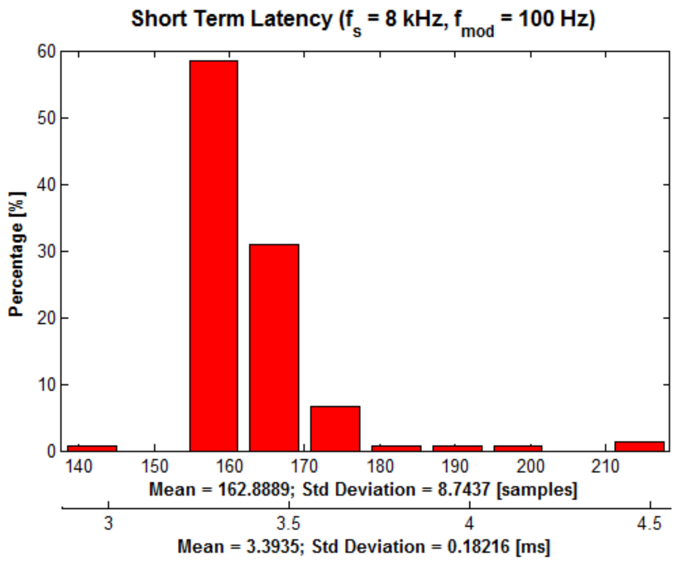


Figure 3.2: Latency with Largest Values; $f_s = 8kHz$; 450 experimental repetitions; Single stream in Block mode; 10 bins

Once is it clear that the only method that could be suitable is the one using a single stream in the callback mode, the short and the long term stability of the latency are required to be measured. For the purpose of assuring this latency stability, some measurements have been done; with the following parameters varying:

1. Four different sampling frequencies have been tested: 44.1 kHz (that is the default one for this audio card), 8 kHz (the lowest sampling frequency permitted), 192 kHz (the biggest allowed), and 48 kHz (another frequency in the middle different for the default one).

2. Do the transmission and the reception of the data with one or two channels, i.e. with the left and/or right audio channels.

3.3.1.1 Short Term Latency Stability

In order to confirm that the latency is fixed, the first verification that needs to be done is that it never changes whenever a new measurement is started. Therefore, the procedure to follow in each experiment consists on simultaneously start sending and recording data, and after one second has passed (in order to avoid non desired effects derived from the electronic components' set up) ten periods of both the input and the output signals are saved. Then, the latency of this 10 periods is measured as stated earlier, repeating this process 450 times for each set-up of sampling frequency and channels, so the statistical validity is confirmed.

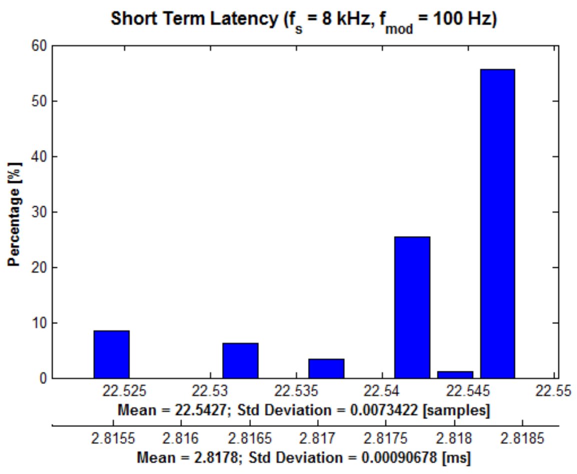
On the one hand, in Fig. 3.3 and 3.4, it can be seen that for the smallest sampling frequency available in the sound card and for the biggest sampling rate possible, the latency is fixed independently if one or two audio channels are used. Although it may look like the latency is not fixed if the variation in ms is analysed, when this variation is converted to samples it can be seen that it corresponds to less than one sample, i.e. stable latency.

On the other hand, in Fig. 3.5 and 3.6, corresponding to the default sampling rate of the audio card and to a close frequency, the stability of the latency between the input and the output is not so clear for both one or two audio channels. The main problem is derived from the fact that the latency in samples has a standard deviation of one sample, which makes it not fixed and could affect to the accuracy of the results; but as this instability is not really big, it can be considered as a fixed latency.

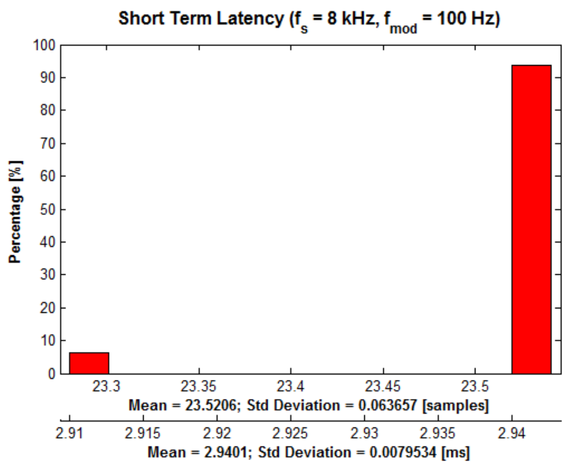
It has to be noted that as with the phase extraction technique the latency is obtained in seconds, due to the fact that the length in time of a period is known, and then this value is converted to samples; sub-sample resolution is achieved on the results. Therefore, in with this particular technique sub-sample resolution could be obtained, which would not be possible for a digital stream.

Another important point to notice is that when the latency value is negative, as in

Fig. 3.4, the real latency value is equal to that negative value plus the number of samples present in one signal period. This is due to the fact that the latency value needs to be positive, as the input signal is recorded after the output signal is played. For example, in the case of Fig. 3.4, as one period has $\frac{192ksamples/s}{100Hz} = 1920$, the total latency would be $1920 - 26 = 1894$ samples.

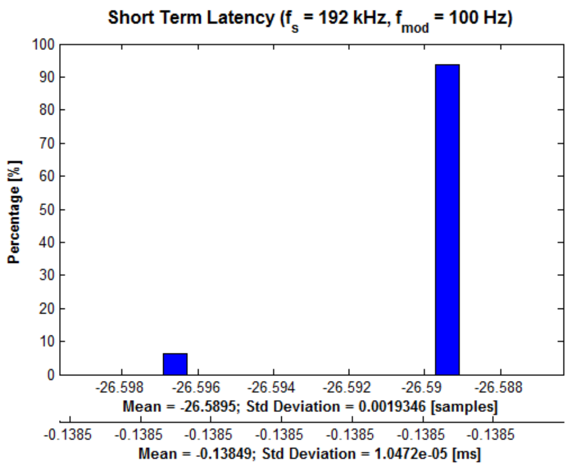


(a) Channels = 1

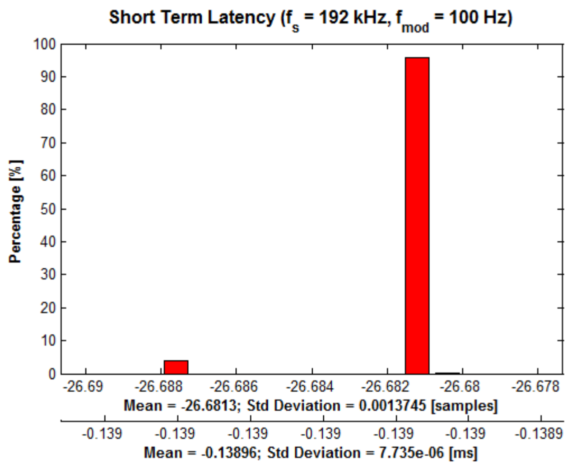


(b) Channels = 2

Figure 3.3: Short Term Latency; $f_s = 8kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins

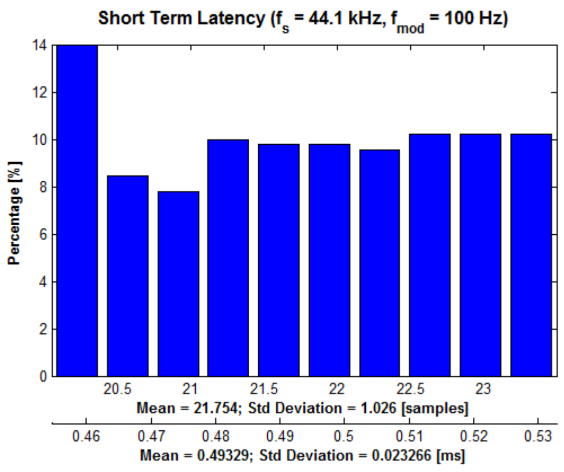


(a) Channels = 1

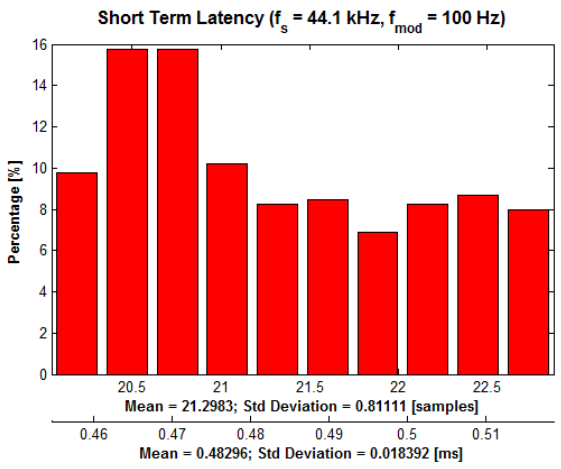


(b) Channels = 2

Figure 3.4: Short Term Latency; $f_s = 192kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins

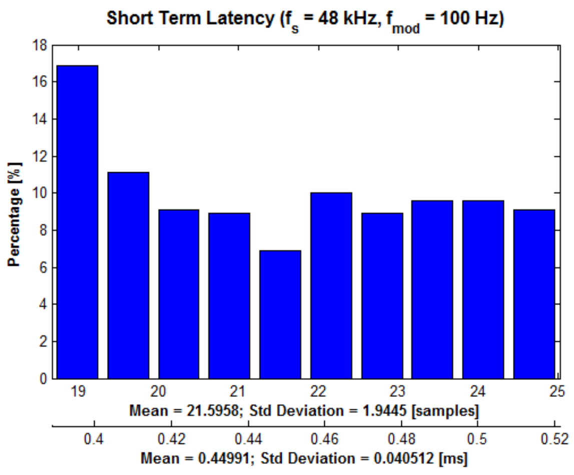


(a) Channels = 1

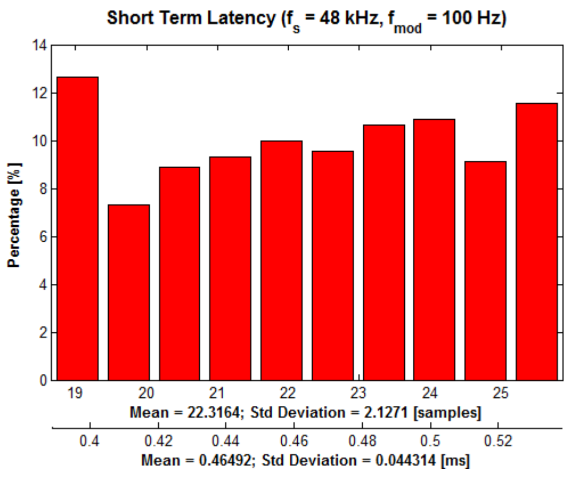


(b) Channels = 2

Figure 3.5: Short Term Latency; $f_s = 44.1kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins



(a) Channels = 1



(b) Channels = 2

Figure 3.6: Short Term Latency; $f_s = 48kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins

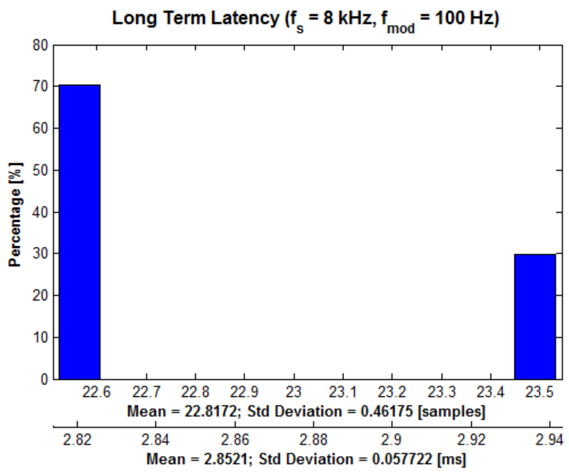
3.3.1.2 Long Term Latency Stability

The second test that is required to be performed is to confirm that the latency is stable after a long period of time running the same stream, without restarting; if the Raspberry Pi is overloaded randomly.

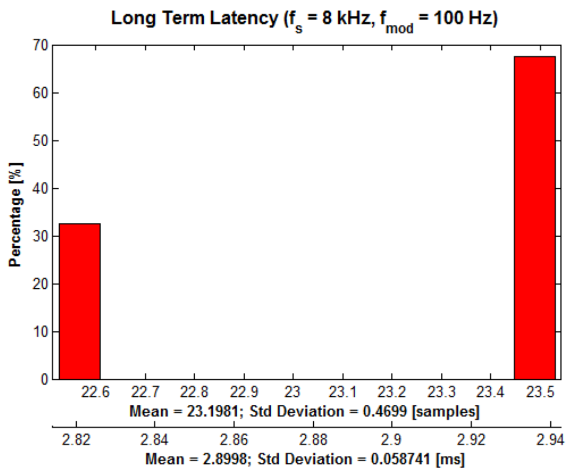
For this purpose the data is going to be sent and recorded simultaneously during 10 minutes time-slots, while other random task are performed in the Raspberry Pi (such as Internet surfing or use of another installed programs). In those 10 minutes time, ten periods of both the input and the output signals are measured every 2 minutes, and as in Sec. 3.3.1.1 the process is repeated till 450 latency values are obtained for each set-up of sampling frequency and channels.

In this case, for all the range of analysed sampling frequencies it can be said that the latency is fixed; as for each case the deviation in the latency value is smaller than half a sample which in practice makes it fixed.

It has to be mentioned that in this section the reasons why sub-sample accuracy can be obtained, as well as the transformation of a negative latency value into a positive one, are also applicable; as in Sec. 3.3.1.1.

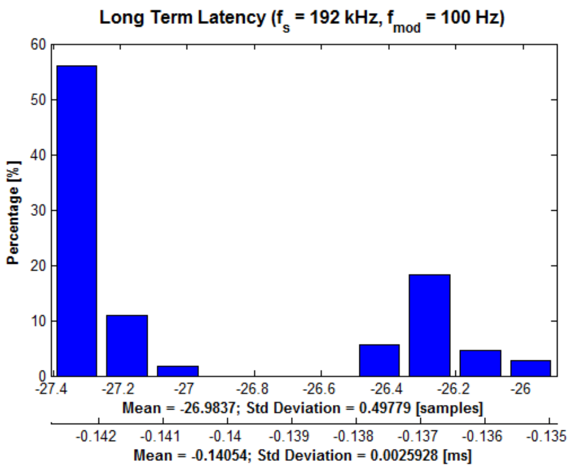


(a) Channels = 1

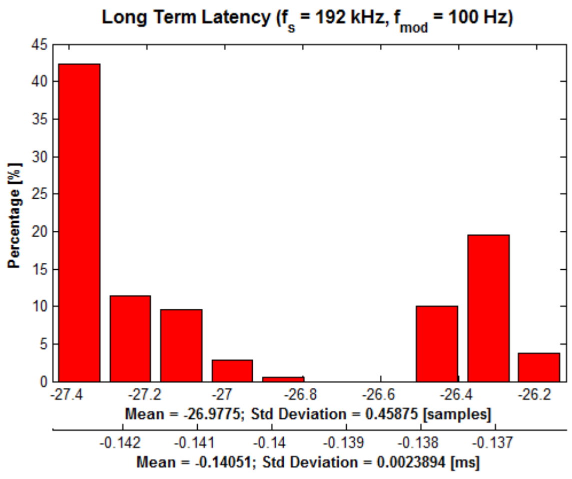


(b) Channels = 2

Figure 3.7: Long Term Latency; $f_s = 8kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins

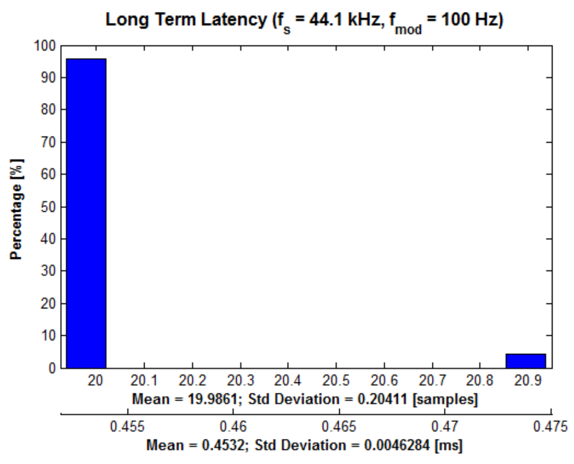


(a) Channels = 1

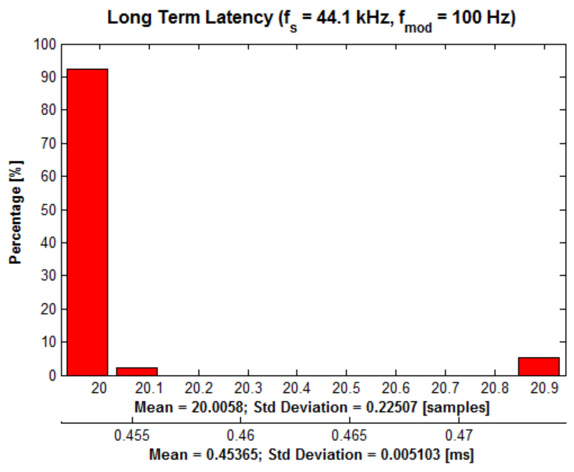


(b) Channels = 2

Figure 3.8: Long Term Latency; $f_s = 192kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins

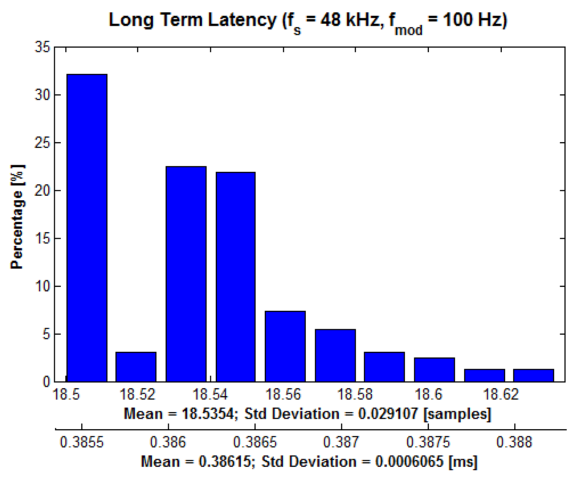


(a) Channels = 1

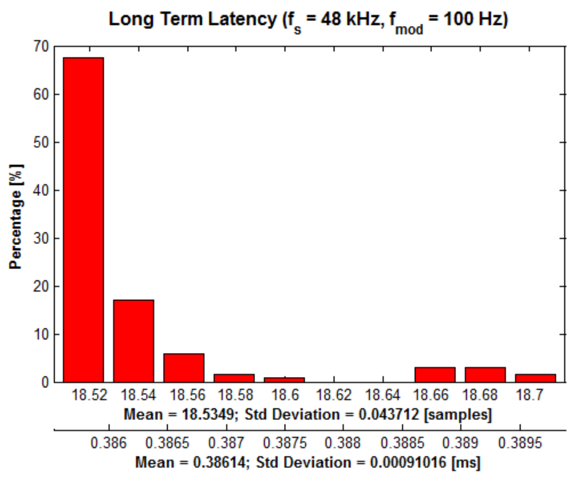


(b) Channels = 2

Figure 3.9: Long Term Latency; $f_s = 44.1kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins



(a) Channels = 1



(b) Channels = 2

Figure 3.10: Long Term Latency; $f_s = 48kHz$; 450 experimental repetitions; Single stream in Callback mode; 10 bins

3.3.1.3 Stability for other carrier frequency

Finally, it needs to be tested that the latency stability does not change if the frequency of the carrier signal changes. With that purpose, the value of the sine has been changed to 500 Hz and a prove with the sampling frequency of 8 kHz and two audio channel has been done, and as it can be seen on Fig. 3.11 the latency is fixed; which therefore proves that the frequency of the carrier signal does not influence on the latency that is obtained.

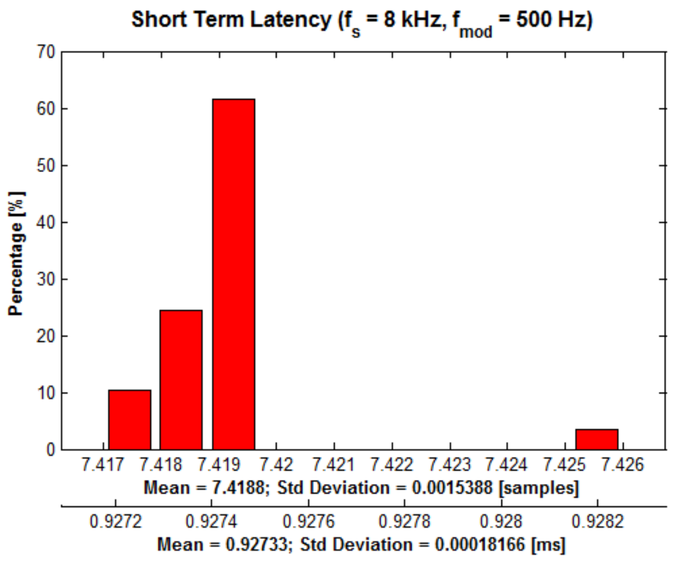


Figure 3.11: Latency for another carrier frequency, $f_s = 8kHz$; 450 experimental repetitions; Single stream in Callback mode

3.4 Conclusions

After the latency of the hardware formed by the Raspberry Pi and the Cirrus Audio Card has been analysed, the following can be concluded.

1. The value of the sampling frequency used to acquire data with the audio card is not a problem, as long as is one of the values accepted by the audio card

2. As the audio card deals with audio, it is prepared to deal with any frequency from 20 Hz to 20 kHz. Therefore, if the frequency of the modulated signal is

between this range, the result would be independent of it.

3. The chunk size should take a value equal to a multiple of the sampling frequency divided by the carrier frequency signal, so that a exact number of periods enters in each chunk, but it has to be sufficient to handle the overflows that may arise in the reading and writing buffers. Therefore, and taking into account that the algorithms prospossed in [6] do the processing for one signal period at a time, it has been decided that the CHUNK size will be equal to:

$$CHUNK = \frac{f_s}{f_m} \tag{3-6}$$

so that the signal processing can be applied every time that a chunk is read, as well as avoiding possible issues with the overflow of the buffers for all the sampling frequencies available at the audio card.

4. Although it has not a direct influence in the measured latency, the volume of the audio card is the last parameter to control and it has been indirectly tested when different data has been outputted. This parameter determines the amplitude of the output signals, and if it is not adequate data clipping could occur. Nevertheless, using the default settings and amplitude values of the sine smaller than 1.0 is enough for the settings used.

Therefore, it can be said that this particular audio card is suitable for the realisation of the Signal Processing Unit for a laser interferometer system; due to the fact that it can record and play audio with a constant latency between the input and the output.

4 SIGNAL PROCESSING UNIT

In order to develop an appropriate Signal Processing Unit that could be used for a laser interferometer, the approach proposed in [6] has been used. It has to be mentioned that although several working modes have been proposed in this thesis, only some of them have been implemented. Moreover, it has to be underlined that although the approach that has been taken was meant to be the most appropriate for the objectives of this project, that is, to build an open-source Signal Processing Unit for laser interferometers as cheap as possible, some modifications had to be done due to the different adversities faced in the development of this Signal Processing Unit.

4.1 Theoretical Approach

As mentioned before, the Signal Processing Unit that is going to be developed is completely based on the algorithms proposed in [6]. In this PhD thesis a novel technique for measuring optical interferometries is proposed, which clearly improves the performance of previous methods used for this purpose as well as been easier to implement when it is compared with other techniques. Therefore, in this section just a brief explanation of the proposed algorithms will be used; for further details see [6].

4.1.1 Overview

In Fig. 4.1 an overview of the procedure followed for completing the demodulation of each signal can be seen. Each step of the process, together with the mathematical formulas that characterize each one as well as the physical signals expected will be explained in the next sections.

The measuring process starts with sending a modulator signal to the laser, which will then travel as an optical signal through space where the desired interference is going to be measured and will finally strike the photo-detector where the light will be converted into an electrical signal again.

In order to measure the interference, the received signal will be then multiplied by a window function followed by a complex carrier signal different for each amplitude channel that is wanted to be measured and then followed by a low pass filtering. Finally, the obtained signal is decimated to reduce the data rate, with one measurement per modulation period to obtain one stream of amplitude data and one stream of interferometric phase data per channel.

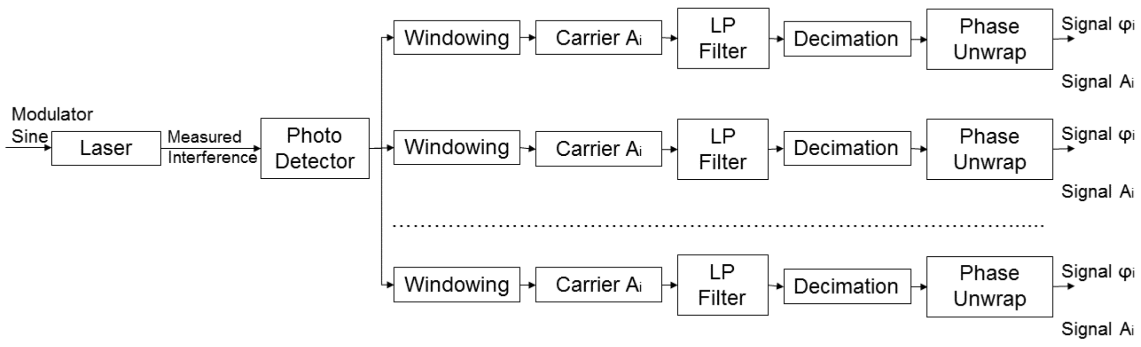


Figure 4.1: Flow diagram of the demodulation process

With this processed data, four different types of results will be obtained, concretely:

1. Signal Strength (A_d) depending on the Range (A_i)
2. Interferometric Phase (P_d)depending on the Range (A_i)
3. Evolution of the Signal Strength (A_d) with time t
4. Evolution of the Interferometric Phase (P_d) with time t

The most interesting results for performing interferometric measurements are the first data stream (the signal strength depending on the range) which is used to find the range of a signal; and the last data stream (the evolution of the interferometric phase with time) which is the main measurement result and corresponds, for example, to a distance travelled.

4.1.2 Mathematical Development

The flow diagram shown in Fig. 4.1 characterizes the Signal Processing Unit that has been developed in [6]; however, each of the steps need to be modelled by a series of equations so that the system characterization is done completely. The equation that models the first step that needs to be done, that is, to output continuously from the audio card a sine which will modulate the optical signal of the laser, is the following one:

$$U_{sent} = A \cdot \sin(2 \cdot \pi \cdot \mathbf{f_{opt}} \cdot t) \tag{4-1}$$

Once this signal has travelled through the area where the interferometry is desired to be measured and strikes the photo detector, it is recorded by the designed hardware. It has to be mentioned that in order to have a proper signal, at least the size of each chunk, i.e. the number of samples read at a time, has to be sufficient so that the received signal shape does not differ from the one expected. Taking into account what is said in Sec. 3.4 about the chunk size value, for modulation frequencies bigger than 20 Hz the sampling frequency of 8 kHz is not adequate, while for rates bigger than 44.1 kHz it is desirable to not use modulator sines of more than 100 Hz. This received signal can be modelled following the next equation:

$$U_{pd}(t') = 0 + 1 \cdot \cos[\theta_1(t') + \varphi_1] \tag{4-2}$$

with

$$\theta_1(t') = A_k \cdot \sin(2 \cdot \pi \cdot f_{opt} \cdot (t - 0.5\tau_k)$$

Once the received signal samples are read, for each chunk the following procedure is followed. The first step that needs to be done is to multiply the data by a window function, which follows in Eq. 4-3. In the particular case of the Signal Processing Unit implemented in this project, and as just one period of the modulator signal is read at a time, n is chosen to take the values of $-1, 0$ and 1.

$$W(t', \sigma, \tau_d) = \sum_{n=-\infty}^{\infty} \left\{ exp\left[-\frac{1}{2} \left(\frac{(t' - \tau_d) - nT_m}{T_m \sigma} \right)^2 \right] + \exp\left[-\frac{1}{2} \left(\frac{(t' - \tau_d) - (n + 0.5)T_m}{T_m \sigma} \right)^2 \right] \right\} \tag{4-3}$$

Once the received signal is multiplied by the window function, which is always the same; it is multiplied by a number of carrier signals, each one having a different amplitude which corresponds to a different range. For each carrier a resulting signal is obtained, which will be then used to obtain a value of the signal strength and interferometric phase. .

$$C(t', A_d, \tau_d) = \exp\left[j \cdot A_d \sin[2\pi f_m (t' - \tau_d)] \right] \tag{4-4}$$

The last step is to obtain a complex number of which amplitude and phase, after it is unwrapped, will be used to obtain the desired results. For doing so, a low-pass filtering (LP) of each of the obtained signals is performed; followed by a decimation process, in which a reduction of the data rate is implemented to obtain one measurement per modulation period. Finally, the phase is unwrapped to ensures that all the multiples of 2π are included and therefore a phase response which can be converted to true space delay is obtained phase unwrapping [26]

$$U_{dem} = LP\{W \cdot C \cdot U_{pd}\} \tag{4-5}$$

For this particular project, one of the easiest Low Pass-Filter has been implemented, i.e. an average filter. The bandwidth of this filter is approximately equal to half of the modulation frequency. As it will be explain later in Sec. 6, the improvement of this Low Pass Filter is one of the actions that needs to be done.

It has to be mentioned that all the equations that model the proposed Signal Processing unit are dependent of t' and not of t. This two variables are related by the following equation:

$$t' = t - \tau_{sp} \tag{4-6}$$

As said before, latency does not influence on the quality of the measurement as long as is fixed. The importance of the latency being fixed derives from the fact that for the proper operation of the proposed technique any signal processing delay τ_{sp} that occurs after modulation needs to be compensated, as all the equations depend on it. Without loss of generality, τ_{sp} can be set to include all analogue and digital delays occurring between the laser modulation step and the beginning of the signal processing [6]; and the measured latency values (see Ch. 3) are used for this compensation.

4.1.3 Graphical Results

To understand the work-flow better in Fig. 4.1 the signals that are obtained after each part of the processing will be shown in this section. It has to be taken into account that the part of this diagram including optical signal, i.e. laser transmission and the photo-detector, are obviously not available.

The first signal that is shown is Fig. 4.2, which corresponds to the sine that will modulate the laser transmitter. This signal is characterized by two main parameters. First of all its modulation frequency will limit the amount of samples that are processed; while secondly its amplitude together with the volume settings of the audio card will determine if the received signal does not suffer from clipping, a situation that is desired to be avoided.

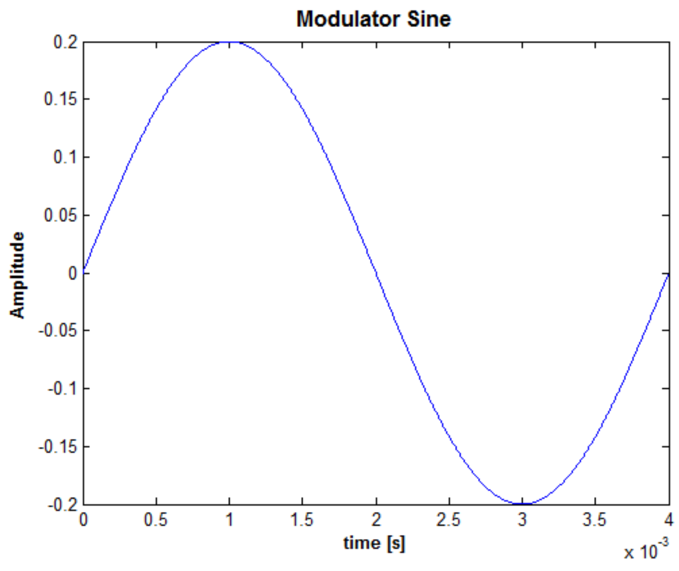
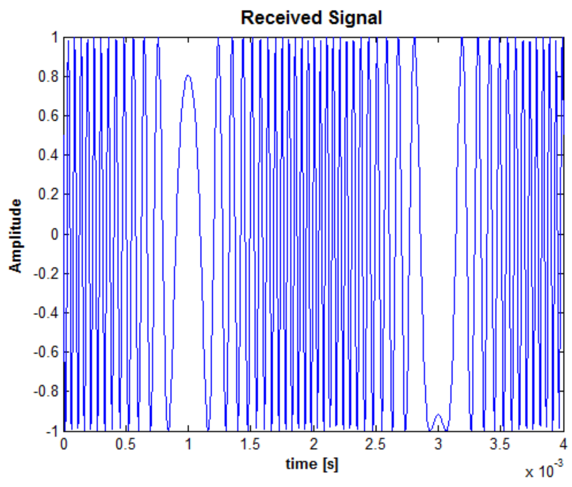
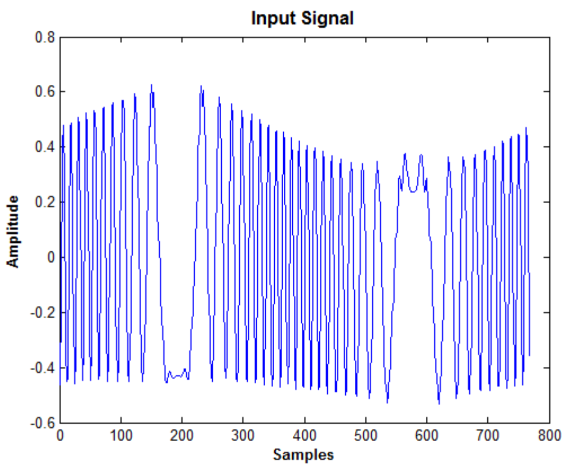


Figure 4.2: Generated signal that is sent to the laser for each modulation period

After the optical signal travels through the area where the interference area is happening and it strikes in the photo-detector, the shape of it can be seen on Fig. 4.3. As it can be appreciated in this image, the received and the test(expected) signals are similar, taking into account that the differences that they have are due to intensity modulation of the laser, although this is not a problem, as discussed in [6]. This proves that the optical transmission and reception part has worked as expected.



(a) Theoretically expected signal

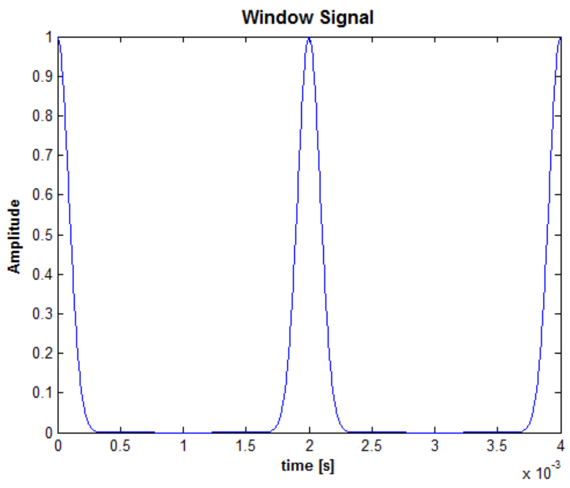


(b) Actually measured signal

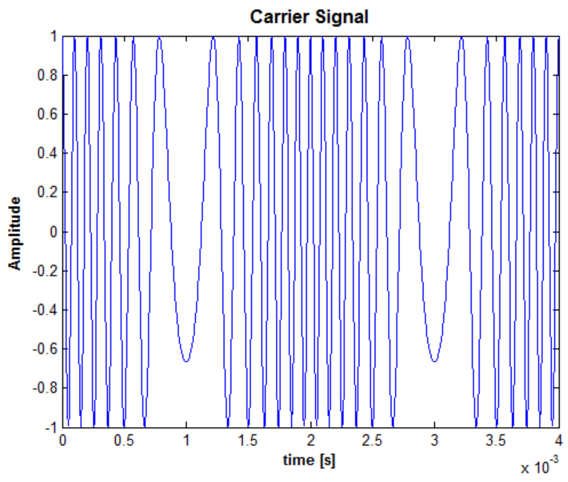
Figure 4.3: Received test and real signals

It has to be mentioned that the received signal is expressed in samples while the expected one is represented in the time domain, but as explained before, one axis can be easily converted to other.

Finally, in Fig. 4.4 and 4.5 the window and carrier functions that are used to do the signal demodulation can be seen, together with the resultant signal after this process is completed.



(a) Window signal



(b) Real part of the carrier signal $A_d = 40$

Figure 4.4: Window and carrier signals

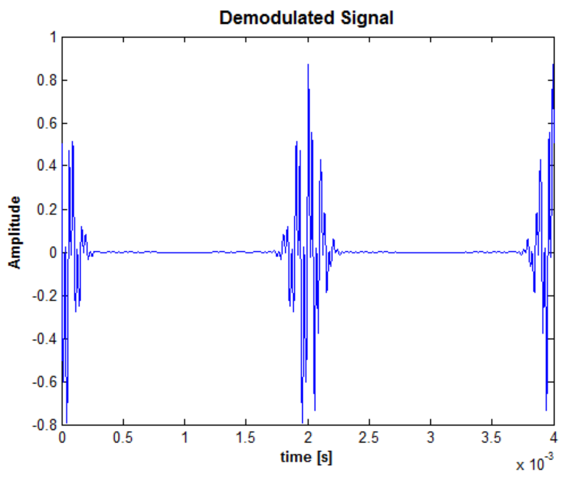


Figure 4.5: Real part of the demodulated measured signal before LP

4.2 Implementation

Once the procedure is clear in order to realise the proposed Signal Processing Unit, its implementation on the selected hardware needs to be done. In this section, some practical considerations that need to be taken into account will be explained and then the different working modes that have been implemented as well as the errors that have occurred during the implementation process will be explained.

4.2.1 Practical Considerations

In order to correctly implement the Signal Processing Unit proposed within this thesis and with the purpose of establishing some parameters that are dependent of the Cirrus Logic Audio Card, the available bandwidth and the absolute latency value need to be investigated.

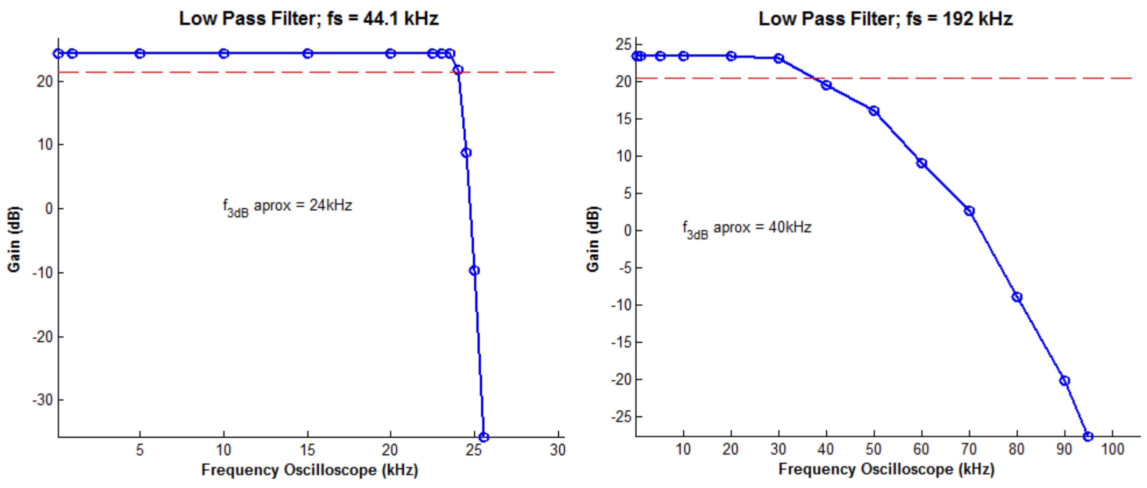
4.2.1.1 Audio Card's LP Filter

Normally, any commercial audio card present on the market has implemented a LP filter, and as in the documentation of the audio card used in this project [22] this information is not available; it is necessary to characterize the audio card in this sense. It is important to not mix this filter that is inherent of the audio card and it is used to control the amount of noise that is recorded by it with the LP filter that has been implemented for the demodulation process and that has half the cut-off frequency of the modulation frequency.

The reason why it is interesting to know how much bandwidth is it available is because it is the factor that limits the possible range of the signal sources. As can be seen on Eq. 4-7 the available bandwidth BW is equal to the amplitude of the channel that is being measured A_d multiplied by the frequency of the modulator sine f_m. Therefore, when f_m increases it is possible to have signal sources placed at larger ranges, as the number of periods of the modulator sine for the same amount of time increases.

$$BW \approx A_d \cdot f_m \tag{4-7}$$

In order to measure the available bandwidth of the audio card, the LP filter corresponding to two different sampling frequencies has been analysed. The procedure that has been followed is to manually input several signals of the same amplitude and of different frequencies into the audio card input with the help of a signal generator and and simultaneously check the output voltage with an oscilloscope. Later, the obtained values have been converted to dB and the transfer function of each LP filter has been plotted as seen on Fig. 4.6. The data that has been used to create these transfer functions is available at Tab. 4.1 and 4.2.



(a) $f_{3dB} \approx 24kHz$ (b) $f_{3dB} \approx 40kHz$

Figure 4.6: Low pass filter of the Cirrus Logic Audio Card

Table 4.1: Low Pass Filter of the Cirrus Logic Audio Card - fs = 44.1kHz

f_{sig_gen} (kHz)	0,1	1	5	10	15	20	22,5	23	23,5	24	24,5	25	25,5	26	26,5
$f_{oscill.}$ (kHz)	0,0978	0,969	4,9	9,96	15,2	19,53	21,98	22,47	23	23,25	23,98	24,5	25	25.5	26
Input (mV)	134	134	134	134	134	134	134	134	134	134	134	134	134	134	134
Output (mV)	453	453	453	453	453	453	453	453	453	397	209	83	22,5	0	0

Table 4.2: Low Pass Filter of the Cirrus Logic Audio Card - fs = 192kHz

f_{sig_gen} (kHz)	10	20	30	40	50	60	70	80	90	95	100
$f_{oscill.}$ (kHz)	9,8	19,3	29,3	39,1	49	59,2	68,5	78,7	87,3	94	98
Input (mV)	134	134	134	134	134	134	134	134	134	134	134
Output (mV)	431	434	425	356	300	211	153	85,6	48,8	33,8	0

As it can bee seen on Fig. 4.6 both LP filters have different cut-off frequencies, so if more information is wanted to be sent for the same channel, the sampling frequency of 192 kHz should be chosen. Furthermore, as for this application the noise of the received signal is not an issue, the cut-off band of the filter is not required to be necessarily sharp and therefore both filters are suitable for the developed system.

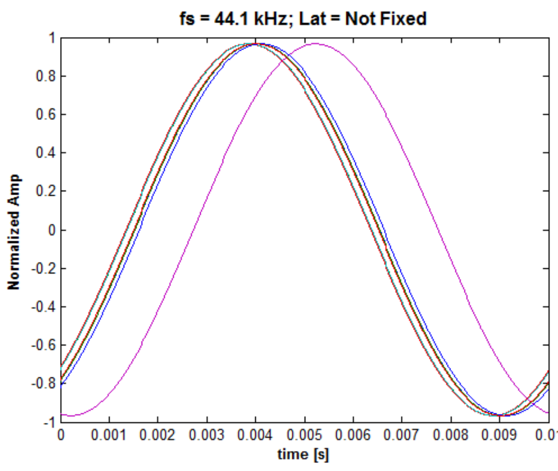
Apart from that, and as mentioned before it is interesting to have as many samples per chunk as possible, because when the received signal (Fig. 4.5) is constructed with not enough samples, the received signal shape varies from its actual shape. Taking into account that the chunk size is equal to the sampling frequency divided by the frequency of the modulator sine as seen on Eq. 3-6, it is important to have the largest possible LP filter bandwidth.

Consequently, and taking all these considerations into account, the sampling frequency that has been chosen as the default one for this system is **192 kHz**.

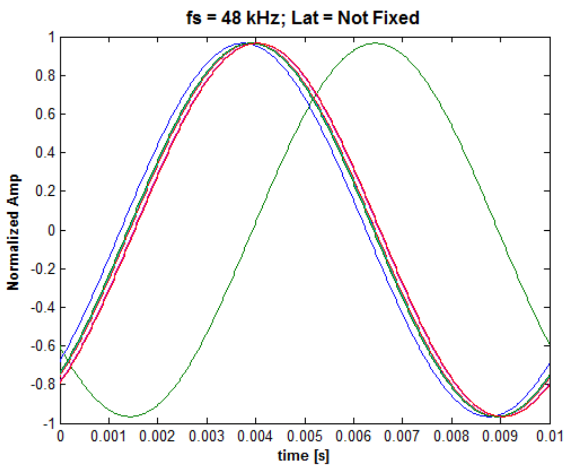
4.2.1.2 Latency Considerations

The second practical consideration that needs to be taken is the value of the absolute latency. In order to check so, the values obtained in Sec. 3.3.1.1 will be used as a guide. The procedure that will be followed is to send 10 periods of a sine signal and them record them so that all the periods start at a 0 amplitude value. In other words, start to read the data samples after *latency* number of samples have passed so that the signal is fixed.

On the one hand, it can be seen that for methods without fixed latency, the recorded sines are not superimposed; which shows that, as expected, each pair of sent-received signal suffers a different delay.



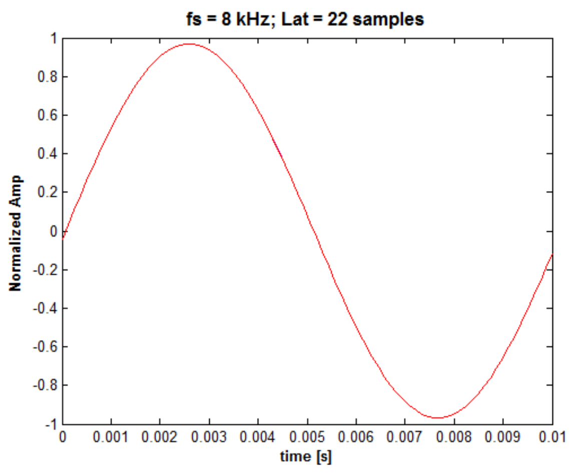
(a) $fs = 44.1kHz$



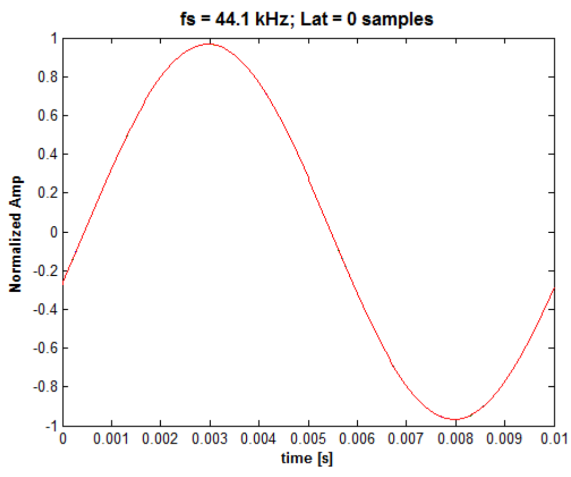
(b) $fs = 48kHz$

Figure 4.7: Working modes without fixed latency

On the other hand, for working modes that offer a fixed latency value it can be seen that all the received signal periods are superimposed, which means that the delay between output and input is always the same.

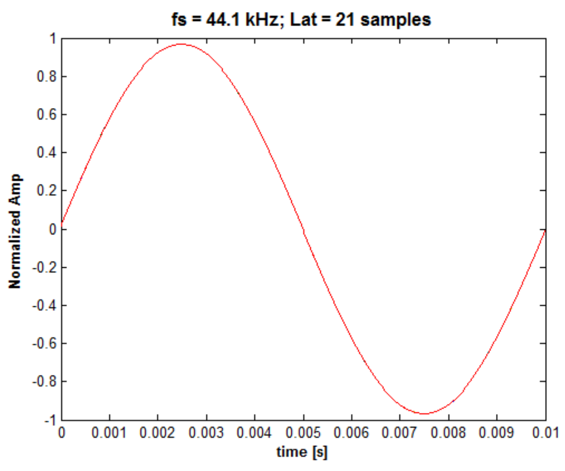


(a) $fs = 8kHz$

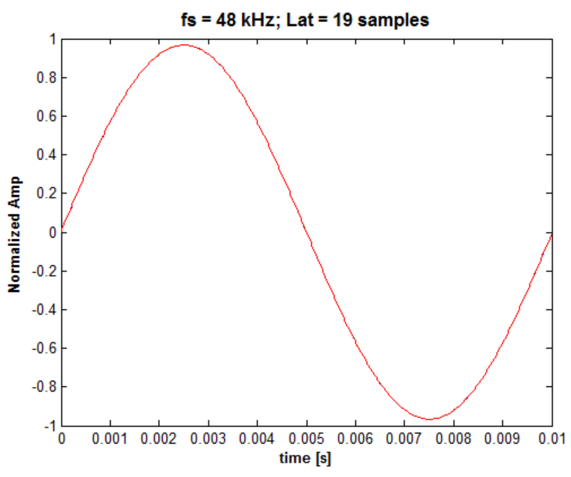


(b) $fs = 44.1kHz$

Figure 4.8: Ten superimposed plots of the working modes with fixed latency



(a) $fs = 44.1kHz$

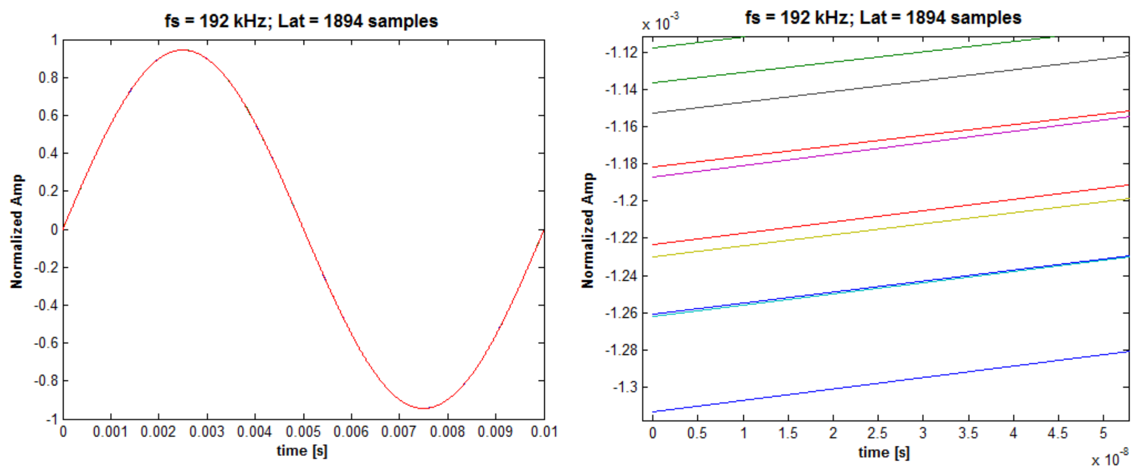


(b) $fs = 48kHz$

Figure 4.9: Working modes with fixed latency

In Fig. 4.8b it can be seen that although the latency is fixed, it is not equal to zero; so the received signal has to be shifted so that each period starts with the closer amplitude to zero. The value that each latency period needs to be shifted has been obtained before in Sec. 3.3.1.1. It has to be mentioned that if the obtained value is negative, it means that the latency needs to be shifted the number of samples equal to the data points presented in one chunk minus this absolute latency value.

Although it have been said that the latency is fixed, this is true for absolute samples shifts. Nevertheless, this real sample shift, which is the magnitude that the audio card works with; is not an integer, although the number of samples can only be integer. Therefore, when this number of samples is taken to be exact, the shift is not exact any more, making each received sine not to be centred in zero values any more, but onto a close value.



(a) $fs = 192kHz$ (b) $fs = 192kHz$ Zoom

Figure 4.10: Zoom of fixed latency

Finally, in needs to be added to these latency values the delay corresponding to the optical path and together this is equal to the signal processing delay τ_{sp} in Eq. 4-6. This value is always the same for the different sampling frequencies used, but this should be added in samples so that it differs depending on the sampling frequency used.

The delay added due to the laser modulation and the photo-detecting process is equal to 3.3802 ms, and for example if this value is passed to the equivalent samples for a sampling rate of 192 kHz, that is the sampling frequency that is going to be used as said before, it is equal to 649 samples.

$$L(samples) = L(seconds) * f_s \tag{4-8}$$

$$L(samples) = 3.3802ms \cdot 192kHz = 649samples$$

4.2.2 Working Modes

Finally, two different working modes for the Signal Processing Unit have been implemented. Although each of the modes has a different purpose and will output a type of results, both of them follow the general flow diagram presented in Fig. 4.1.

Furthermore, and taking advantage of the fact that the Raspberry Pi has four different cores, a multiprocessing scheme has been implemented in order to perform the different calculations and improve the performance of the designed hardware in terms of computational time. Therefore, the flow diagram has been divided into two main processes:

1. One process is entirely used for IO purposes, so that no data is lost during the processing time.
2. The other process is used to compute the signal processing algorithms, so that the total elapsed time decreases.

Apart from that, the sine that will modulate the laser will be selected by a *wav* file which length should be at least one period of the modulated signal and always having a length multiple of this value. The main reason for that is that when all the samples of the file are played, if more data is required the first sample will be outputted again.

Moreover, the configuration of the general parameters for the demodulation such as the latency value and or the output sine's modulation frequency an amplitude can only be using fixed parameters in the source code.

4.2.2.1 Live Mode

The first working mode is the live mode, which has as main purpose to locate the range of the channels where interferometry is happening. Therefore, the results that will be computed in this mode are the demodulated signal strength and interferometric phase dependent of the range. This measurement mode is more oriented to help the user understanding at which range the signal sources are situated to select the correct range channels for the capture mode, see below.

The measurement procedure that is applied in this working mode is to make two measures per second for four different channels and then plot the obtained results in pseudo-real time. After approximately 50 seconds have passed, the user could choose between continue measuring, configure the measurement for another channel amplitude values or exit.

4.2.2.2 Capture Mode

The second working mode is the capture mode, which has as main objective to obtain a complete interferometry measure, i.e. to compute the evolution of the signal strength and interferometric phase corresponding to a channel over time. The procedure that is followed for this mode is to record 3 seconds of the interferometric signal without interruptions, together with the half a second of measurement that is discarded for avoiding problems related with the electronic setting up of the audio hardware. These signals are subsequently demodulated and the results are plotted.

Eight different range channels are computed every time that this working mode is used, and the selection of these range channels can only be done once after the 3 seconds are recorded. After that, the user can select which of the channel(s) wants to plot; finally saving all the results onto a file before exiting the programme.

4.2.3 Errors Occurring During Implementation

During the implementation process, several errors have been encountered that affect the performance of the implemented Signal Processing Unit, mainly:

1. The biggest problem that has been faced during the implementation process derives from the use of the ALSA library. Originally, the Capture Mode was intended to record at least 20 seconds of the input signal and then perform the pertinent computations. Nevertheless, when recordings that last more than 5 seconds where done, the ALSA library started suffering from several errors derived from the reading buffer restarting process, although the used stream is never restarted. Therefore, it has been taken an alternative option and measurements of just 4 seconds are done, so that this error disappears and the implemented process is computed without a problem. However, with this alternative the measurements that can be taken are not really long and hence this should be improved in the future.

2. Other problem that has been noticed is that sometimes, when the Signal Processing Unit is executed more than one time, some errors appear due to ALSA not finding available audio systems. This error occurs randomly and happens due to a bug of the ALSA library which is still not solved. Nevertheless, as this errors randomly appears there is not much that can be done until the ALSA library is updated.

3. The last problem that needed to be solved is that the live mode does not have the expected time performance because it cannot process and plot data at the same time without stopping playing and recording data. Nevertheless, as in the final version of the open source-project this Raspberry Pi will only be in charge of outputting the results and they will be plotted on an external Web-Based GUI, this will no longer be a problem.

5 RESULTS

In this section the obtained results will be explained, as well as a performance comparison of them in terms of validity and computational cost. As stated before, there are four different types of results, namely the demodulated signal strength or interferometric phase depending on the range of the different channels and the evolution of the demodulated signal strength or interferometric phase within time.

The general set up for performing the measurements to obtain these results can be seen on Fig. 5.1. In this graph, the electrical signals are expressed in black arrow, while the optical signal are represented with blue arrows. The procedure followed to measure the distance can be resumed in the following steps:

1. The developed hardware sends a modulator signal of a certain frequency to the laser.
2. The laser transmits the modulated optical signal, that strikes on a movable mirror and returns to the photo-detector.
3. The photo-detector converts the optical signal into a electrical signal that is transmitted to the Signal Processing Unit.
4. The computations required to obtain the distance travelled by the optical signal are performed in the Signal Processing Unit.

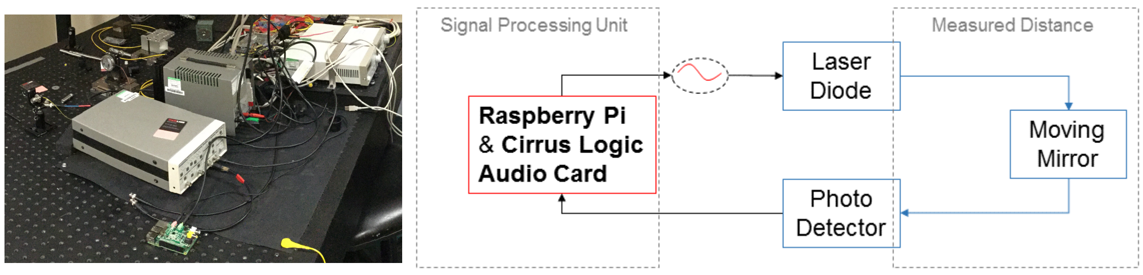


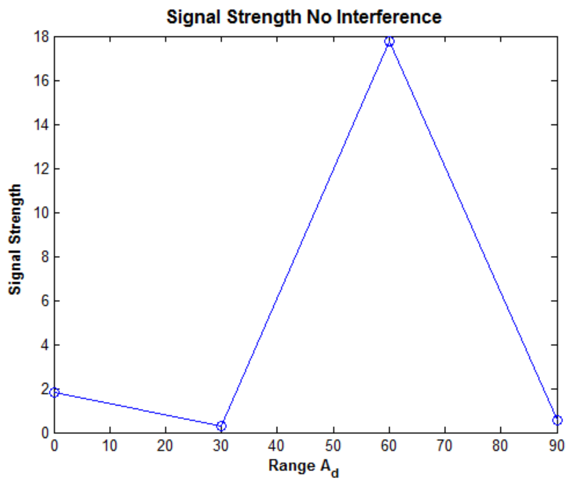
Figure 5.1: General set up for the measurements done

In order to perform the different measurement and analyse the performance of the developed hardware two different movable mirrors have been used:

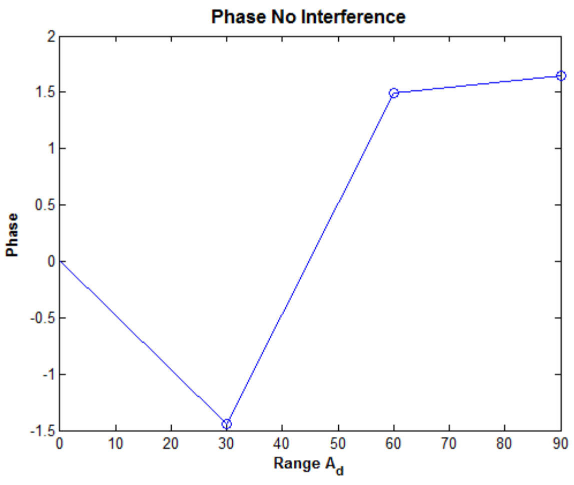
- The first configuration uses a mirror that can be moved manually with a precision of approximately 10 µm.
- The second configuration has a mirror which that can be moved following an electrical signal that is introduced to it, with a resolution of 0.1 µm.

5.1 Dependent on the Range of the Channels

When all the processing is done, the first type of obtained results are the signal strength and interferometric phase depending on the range of the different channels (also called carriers); which are used to check in which channel the interferometry is happening. In the case of the measurements made for the mirror that is moved manually, the results are the ones shown in Fig. 5.2.



(a) Signal strength as a function of range for the manual mirror



(b) Interferometric phase as a function of range for the manual mirror

Figure 5.2: Results as a function of range for the manual mirror

In this case of results the graph that gives the useful information is the one that shows the amplitude. As it can be seen on Fig. 5.2a, the channel that is interesting to measure is around an amplitude value of 60 *rad*. If in the middle of the live mode measurement an obstacle is put between the laser and the photo-detector (for example a hand), it can be seen that the amplitude of the received signal vanishes for that channel Fig. 5.3. Therefore, it can be concluded that physically plausible measurements are taken.

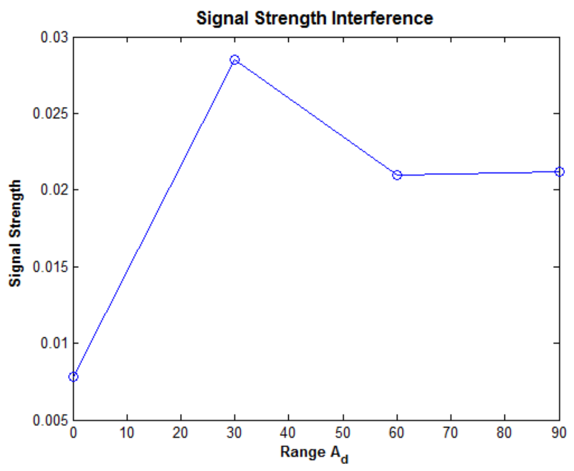
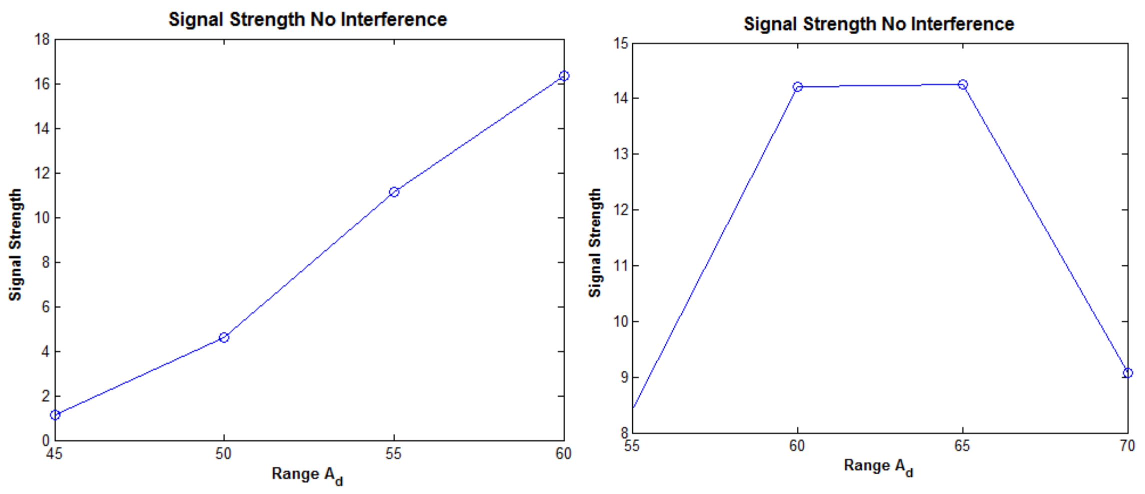


Figure 5.3: Signal strength when hand is inserted

Nevertheless, as in the live mode only four range channels are used for the computations, the range value of 60 *rad* is only approximate. When the range channels are selected with a closer spacing around the preliminary value of 60 *rad*, it can be seen in Fig. 5.4 that the actual peak value is between 60 and 65 rad.



(a) Signal strength - range $A_d = 45 - 60$ (b) Signal strength - range $A_d = 55 - 70$

Figure 5.4: Results for different ranges with the movable mirror

In the case of the second configuration, that is when the electrically movable mirror is used, the obtained amplitude is smaller but is still sufficient for the developed hardware to measure it. In this case, as the laser has not been changed, the interference channel is obviously the same.

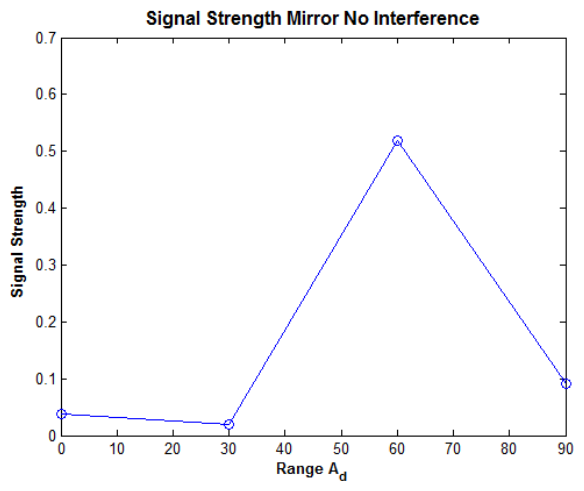


Figure 5.5: Signal strength as a function of range for the electrically moving mirror

5.2 Evolution with time

The other type of results that can be obtained are the evolution of the demodulated signal strength and interferometric phase within the time. These results are the most important ones, as they are the actual interferometric measurements, specially the phase evolution dependant on time. When this phase change is measured, the displacement of an object with respect to the interferometer head can be expressed as [6], using the laser wavelength of 1551 nm:

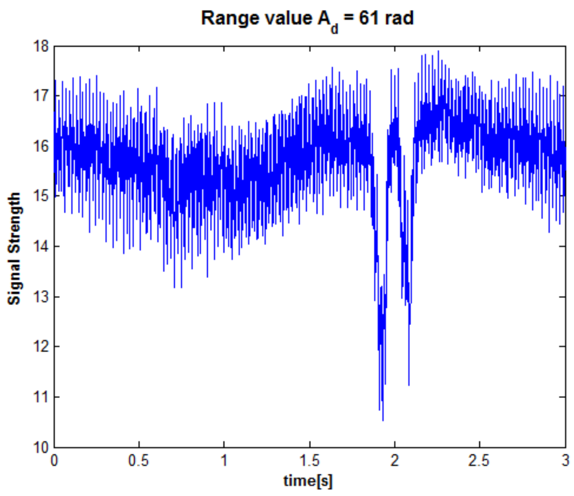
$$Displacement(m) = 0.1234e^{-6} \cdot Variation(phase[rad]) \tag{5-1}$$

with the constant value that appears on Eq. 5-1 being dependant of the wavelength of the used laser.

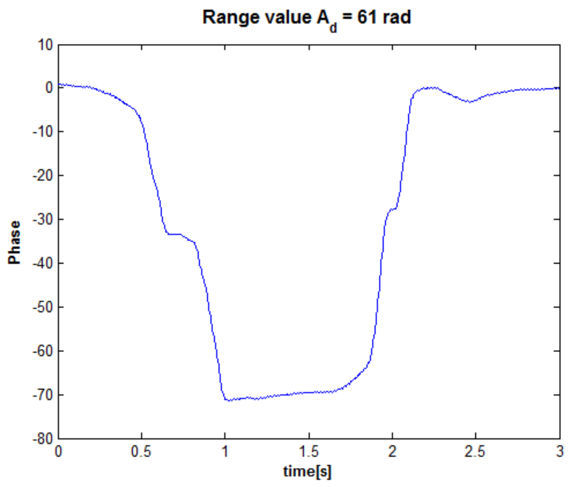
For the manual mirror configuration, the procedure that has been done is to move it approximately 10 µm in one direction and then move it back to original position. As it can be seen in Fig. 5.6b, the phase went 70 *rad* down and the it went back again. Taking into account the Eq. 5-1, in this case the deviation would be.

$$Displacement = 0.1234e^{-6} \cdot (0-70) = -8.34\mu m$$

It has to be mentioned that the value of $10\,\mu\text{m}$ is approximate, due to the fact that is has been done manually and cannot be done completely accurate.



(a) Evolution of the demodulated signal strength with time



(b) Evolution of the Interferometric phase with time

Figure 5.6: Evolution of the results with time

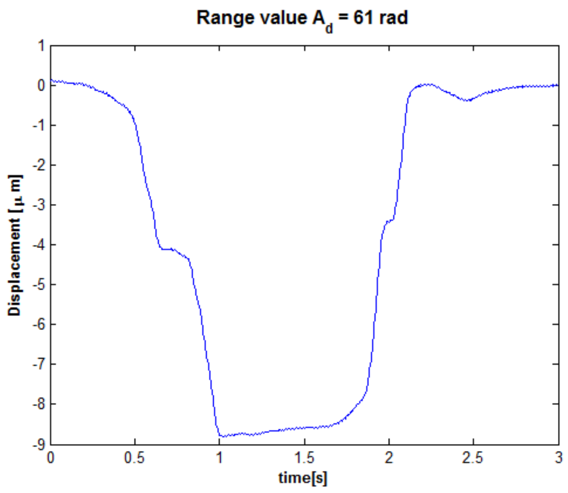
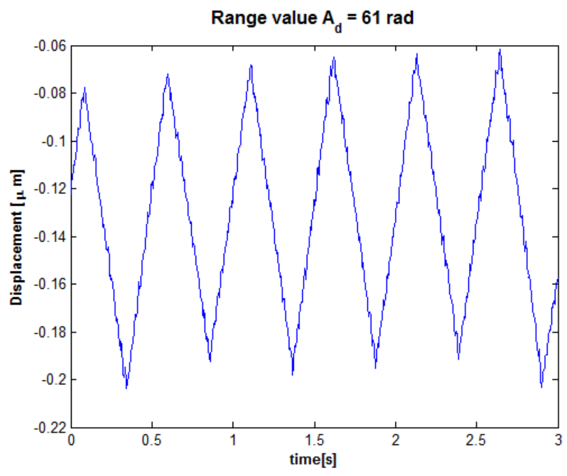


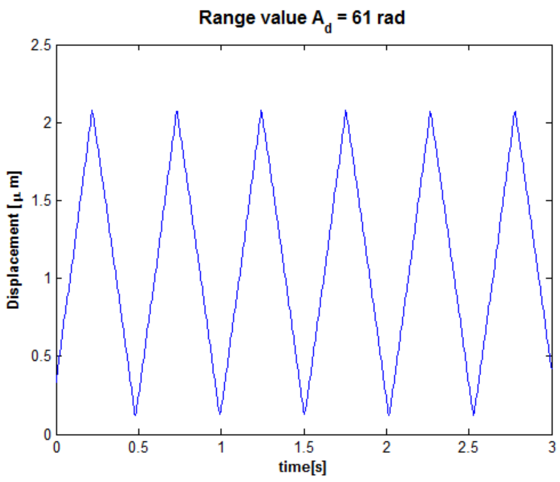
Figure 5.7: Evolution of the deviation with time

The last experiment that has been done using a Piezo-mirror that moves following an electrical input signal. This signal is generated with a signal generator, and it has been selected to have 2 Hz fundamental frequency. The first signal that has been tried is one with a triangular shape, and two different values of amplitude. For the first one (Fig. 5.8a) a noise of 50 Hz can be seen on the demodulated phase; anyway, the movement following a triangular signal of 2 Hz can be perfectly spotted.

For the second amplitude, which is the maximum admitted by the (Fig. 5.8a) it is clearly seen that this noise is attenuated as the amplitude is bigger. For both cases, the measured displacement have been plotted in µm following the Eq. 5.2.



(a) Triangular signal with $0.12\,\mu\mathrm{m}$ peak-peak displacement, where 50 Hz noise is still visible



(b) Triangular signal with $1.85\,\mu\mathrm{m}$ peak-peak displacement

Figure 5.8: Mirror moving following a triangular signal

In both cased the signals look physically plausible and correspond the waveforms selected at the function generator, therefore it can be considered that the implemented hardware works in a proper way.

Finally, the experiment is repeated for the same amplitude and frequency configuration as before, but in this case with the Piezo-mirror being modulated with a sine signal.

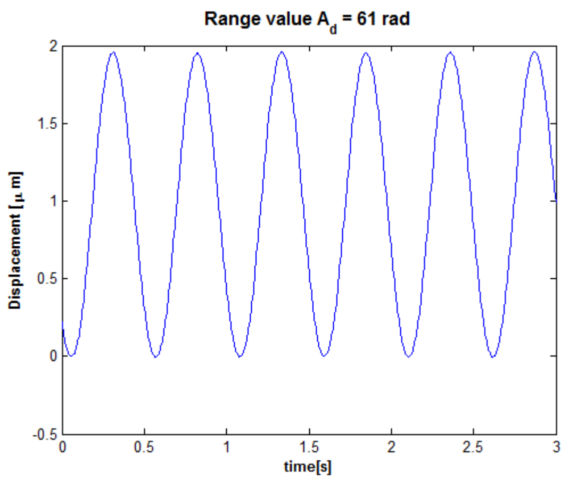


Figure 5.9: Sine signal with 1.85 μm peak-peak displacement

5.3 Performance

Once the process of obtaining the results has been demonstrated successfully, their validity and computational cost needs to be estimated.

5.3.1 Validity

The most important factor that needs to be analysed is to assure if the obtained results are valid or not, as this will stablish if the developed Signal Processing Unit can be used or not. As it has been proved with the configuration of the moving mirror, the obtained results appear physically plausible and are therefore considered to be valid.

Nevertheless, it has to be mentioned that the developed hardware can only measure movements with frequencies smaller than the modulation frequency of the laser, in this case 250 Hz. That is, if the displacement that is required to be measured happens faster than the period of the modulator signal it will be impossible to measure it.

5.3.2 Computational Cost

The second factor that needs to be considered is the elapsed time or computational cost of each of the working modes of the developed Signal Processing

Unit. Although this factor does not influence directly on the obtained results, it is important to know how much computing time each method needs before any result is obtained.

For the Live mode, it takes around 5 seconds for each update involving from when the signal is sent till the results are plotted. As this process is repeated 10 times before another values of channel amplitude can be used, it can be said that a measurement takes around 50 seconds to be computed.

On the other hand, for the Capture mode it takes 4 seconds the overall IO process plus around 45 seconds to compute all the results. As the results are just computed once, the computational cost can be considered to be around 50 seconds also.

6 FUTURE WORK

This project has proven the fact that an acceptable Signal Processing Unit for laser interferometers could be built with the use of an economic mini-computer as the Raspberry-Pi and an audio card for playing a sine modulation and recording the photo detector signal that is going to be used for the measurements. Nevertheless, some improvements could be done to the developed project; mainly divided in technical and organization tasks.

6.1 Technical Improvements

All the potential improvements that help to increase the performance of the implemented system in terms of efficiency, elapsed time, computational cost or accuracy are included among these, namely:

1. Improve the low-pass filtering corresponding to the last stage of the signal processing. The LP filter implemented, an average, is one of the worst LP filters in terms of cut-off frequency and ripple; therefore the use of a better low-pass filter in which the gain, the cut-off frequency and the ripple can be controlled by the user would enhance the performance of the implemented algorithms.
2. Investigate in detail the ALSA library and keep it updated to the latest version so that the errors occurring when long measurements are done can be avoided. In this way, the Capture Mode can be improved and longer events can be measured, which obviously will increase the functionality of this working mode.
3. Reduce the computational cost of both of the working modes, i.e. the live and capture modes. The biggest improvement that can be done is to start with the multiprocessing once the recording of data has started and not wait till the end of the recording is done. Nevertheless, this procedure has not

been yet implemented because of a bug of the ALSA library, so it will be only possible to implement once this bug is solved.

4. Once this bug has been solved, another improvement than can be implemented is to not stop the stream in the live mode every time the data corresponding to one period is recorded; as half a second is wasted every time the stream starts to avoid the undesired effects of setting up and hence all the processing time increases.

5. Related with the computational cost and processing time of each mode another improvement would be to prioritize the Audio IO process so that it is less likely that data is lost; in other words, that the system never stops recording data until all the amount of data is recorded.

6. Analyse the influence of different parameters in the performance of the system in terms of the time required to obtain results, such as:

   - The number of interferometric channels used for the computations
   - The amount of periods (chunks) measured before a result is displayed

6.2 Organization Improvements

This second group of improvements are more related with the fact of facilitating the publication of the open-source project as well as making it more user friendly.

1. The first improvement that can be done is to couple the project with others done in the Engineering Photonics Group in Cranfield University, specially with the Web-Based GUI that is the Part II of this project; so that the user interaction is easier and the visualization of results is more fluent. With this purpose, two different approaches can be taken.

   - Output the results to the network, so the Web-Based GUI can access to them directly, as well as allowing the configuration of the measurement via an Internet connexion. This option is the most suitable one for the purposes of making the project more user friendly, as well as safer for the final user as the system could be set up without any human-laser interaction; but it has a more difficult implementation.

- Save the obtained data in a file and the transfer manually to the platform where the results will be plotted. This approach will be easier and faster to implement, but less convenient for a portable and user friendly system.

2. Facilitate the installation of all the required software on the Raspberry Pi by creating an image of the required files; so that the final user can just download it and after an easy installation on a SD-card start making measurements.

3. Complete the documentation of the developed software, as well as improve the installation and using guides of the system so that the open-source project can be publish in a short period; and more importantly, used and modified by multiple users.

7 CONCLUSIONS

In this thesis the development of an open-source Signal Processing Unit has been achieved. With the developed software, together with a Raspberry Pi and a Cirrus-Logic audio card, a Signal Processing Unit that can be used in a similar way as the commercial ones can be implemented.

The main characteristics of the developed system are:

- **Scalability**: The designed system could be used alone and save the results into a file or plot them, as well as a part of a bigger system that handles with the results plotting automatically.
- **Portability**: Although this system has been designed to work on a Raspberry Pi 2B+ together with the Cirrus Logic Audio Card, the implemented software is capable of working with any audio card compatible with this model of Raspberry Pi; or even with other compatible Raspberry Pi models or with any hardware that can handle with python files as well as input and output of audio data files.
- **Cost efficiency**: The total cost of the proposed signal unit is around £75, which is very cost-effective when compared to commercial Signal Processing Units.
- **Time efficiency**: The required time for outputting results both for the live mode and the capture mode is around 50 seconds.

REFERENCES

1. Tatam RP. Applied optics to engineering photonics: A retrospective. Photonic Sensors. 2012; 1(4): 295–322. Available at: DOI:10.1007/s13320-011-0041-4

2. Feng J, Wu S, Li C, Su D, Yu M. Effect of self-vibration on accuracy of free-fall absolute gravity measurement with laser interferometer. 2015; 9446: 94460E. Available at: DOI:10.1117/12.2084784

3. Yan H, Duan H-Z, Li L-T, Liang Y-R, Luo J, Yeh H-C. A dual-heterodyne laser interferometer for simultaneous measurement of linear and angular displacements. Review of Scientific Instruments. 2015; 86(12). Available at: DOI:10.1063/1.4936771

4. Saha SK. Modern optical astronomy: Technology and impact of interferometry. Reviews of Modern Physics. 2002; 74(2): 551–600. Available at: DOI:10.1103/RevModPhys.74.551

5. Lee BH, Kim YH, Park KS, Eom JB, Kim MJ, Rho BS, et al. Interferometric fiber optic sensors. Sensors. 2012; 12(3): 2467–2486. Available at: DOI:10.3390/s120302467

6. Kissinger T. Range-Resolved Optical Interferometric Signal Processing - PhD Thesis. Cranfield University; 2015.

7. Abramovici A, Althouse WE, Drever RWP, Gursel Y, Kawamura S, Raab FJ, et al. LIGO: The Laser Interferometer Gravitational-Wave Observatory. Science. 1992; 256(5055): 325–333. Available at: DOI:10.1126/science.256.5055.325

8. Danzmann K, Rdiger A. LISA technology concept , status , prospects. Institute of Physics Publishing. 2003; 1.

9. Kissinger T, Charrett TOH, James SW, Adams A, Twin A, Tatam RP. Simultaneous laser vibrometry on multiple surfaces with a single beam system using range-resolved interferometry. Proc. SPIE 9525, Optical Measurement Systems for Industrial Inspection IX, 952520 (June 22, 2015). 2015; 9525: 952520. Available at: DOI:10.1117/12.2183281

10. Kissinger T, Charrett TOH, Tatam RP. Range-resolved interferometric signal processing using sinusoidal optical frequency modulation. Optics Express. 2015; 23(7): 9415. Available at: DOI:10.1364/OE.23.009415

11. Kissinger T, Correia R, Charrett TOH, James SW, Tatam RP. Range-resolved signal processing for fibre segment interferometry applied to dynamic long-gauge length strain sensing. Proceedings of SPIE - The International Society for Optical Engineering. 2015; 9634: 4–7. Available at: DOI:10.1117/12.2194347

12. Bell CA. Beginning sensor networks with Arduino and Raspberry Pi. [electronic resource]. Technology in action. [New York]?: Apress?; New York?: Distributed to the Book trade worldwide by Springer Science+Business Media New York, 2013.; Available at: https://search.ebscohost.com/login.aspx?direct=true&db=cat00164a&AN=cran.708861&sit live

13. Banerjee S, Sethia D, Mittal T, Arora U, Chauhan A. Secure sensor node with Raspberry Pi. IMPACT 2013 - Proceedings of the International Conference on Multimedia Signal Processing and Communication Technologies. 2013; 26–30. Available at: DOI:10.1109/MSPCT.2013.6782081

14. Nguyen H-Q, Loan TTK, Mao BD, Huh E-N. Low cost real-time system monitoring using Raspberry Pi. Ubiquitous and Future Networks (ICUFN), 2015 Seventh International Conference on. 2015; 857–859. Available at: DOI:10.1109/ICUFN.2015.7182665

15. Arduino. Available at: https://www.arduino.cc/

16. Raspberry PI. Available at: https://www.raspberrypi.org/

17. Meier F. Master Thesis Low-Latency Audio over IP on Embedded Systems. 2013; (April).

18. Hagler MO, Mehrl D. A PC with sound card as an audio waveform generator, a two-channel$\backslash$ndigital oscilloscope and a spectrum analyzer. IEEE Transactions on Education. 2001; 44(2): 75086. Available at: DOI:10.1109/13.925849

19. Chandra S, Ismail ABM. PC sound card-based simple instrument for the potentiometric sensors. Sensors and Actuators, A: Physical. 2009; 154(1): 65–68. Available at: DOI:10.1016/j.sna.2008.11.002

20. Mandaji M, Buckup T, Rech R, Correia RRB, Kist TL. Performance of a sound card as data acquisition system and a lock-in emulated by software in capillary electrophoresis. Talanta. 2007; 71(5): 1998–2002. Available at: DOI:10.1016/j.talanta.2006.09.007

21. Reuter J. Case Study: Building an Out Of The Box Raspberry Pi Modular Synthesizer. Linux Audio Conference (LAC-2014). 2014;

22. Cirrus Logic Audio Card User Documentation. 2015;

23. Dudas R, VandenBussche C, Baras A, Ali SZ, Olson MT. Inexpensive telecytology solutions that use the Raspberry Pi and the iPhone. Journal of the American Society of Cytopathology. Elsevier Ltd; 2014; 3(1): 49–55. Available at: DOI:10.1016/j.jasc.2013.09.005

24. Sahani M, Mohanty MN. Realization of Different Algorithms Using Raspberry Pi for Real-Time Image Processing Application. Advances in Intelligent Systems and Computing. 2015. pp. 99–104. Available at: DOI:10.1007/978-81-322-2009-1

25. PyAudio Docs. Available at: https://people.csail.mit.edu/hubert/pyaudio/docs/

26. Itoh K. Analysis of the phase unwrapping algorithm. Appl. Opt. OSA; July 1982; 21(14): 2470. Available at: DOI:10.1364/AO.21.002470

27. Noobs Setup. Available at: https://www.raspberrypi.org/help/noobs-setup/

28. Cirrus Logic Audio Card Set Up. Available at: (https://www.element14.com/community/thread/42202/l/cirrus-logic-audio-card-working-on-the-raspberry-pi-2)

APPENDICES

Appendix A Hardware Requirements

In order to implement the Signal Processing Unit that has been developed in this thesis, just two different hardware pieces are required, alongside with the developed code. Namely, a **Raspberry Pi Model 2B+** [16] and a **Cirrus Audio Card** [22]. Furthermore, as the memory capacity of the Raspberry Pi is not enough, an external microSD card of at least 8 GB is required.

It has to be mentioned that the compatibility with older and future models of the Raspberry Pi has not been tested. Nevertheless, the compatibility with the previous models should be assured, if the memory requirements are fulfilled.

On the other hand, the software compatibility is assured with other available audio cards in the market, however some of them could suffer from another compatibility issues like the Wolfson Audio Card which requires a special connector to connect the Raspberry Pi to it (a P5 connector).

A.1 Extra Hardware

Nevertheless, with the Signal Processing Unit only, a functional laser interferometer can not be build. Therefore, more equipment is required if a functional interferometer needs to be built. Concretely, the required hardware is the following:

- A **mini-USB wire** to provide with electrical power to the Raspberry Pi.
- One **optical laser** to transmit the signal that is going to be used for measuring.
- One **photo-detector** to receive the signal transmitted by the laser.
- Two **input-input jack wires** to connect the optical sensor to the Raspberry Pi.

A.2 Hardware Cost

Table 1: Hardware Cost

Product	**Price**
Raspberry Pi 2B+	£ 30
Cirrus Audio Card	£ 30
8 GB microSD card	£ 5
miniUSB wire	£ 2
Input-Input Jack wire (x2)	£ 1.5

N.B.: The price of the optical laser and the photo-detector is not included in this table as it varies a lot from one supplier to another and also depending the specifications of each hardware piece.

Appendix B Software Set Up

Apart from acquiring the required hardware that has been mentioned in Appendix A, some extra hardware is required for setting up the software that has been developed in this thesis, i.e., a HDMi wire, an USB keyboard, an USB mouse, a monitor with a HDMi input, an Ethernet wire and a microSD card reader. Once all this equipment is available for the user, the steps that are needed to be followed in order to install the signal processing unit into the hardware pack of the Raspberry PI and the Cirrus Audio Card are the following:

1. Install the operating system on the SD Card [27]
   - First of all format the SD card
   - Secondly, copy onto the SD card the operating system install manager NOOBS
   - Finally, select Raspbian as the operating system and install it
2. Install image onto the installed kernel to be able to use the audio card [28]
   - Test if the image has been correctly installed by executing the provided scripts on ...
3. Connect the Raspberry to the internet by simply connecting the LAN wire and execute the following commands on the command window as a super user (using the keyword sudo before every command).
   - apt-get install pyaudio
   - apt-get install matplotlib
   - apt-get update
   - apt-get upgrade
4. Copy the folder *Developed_Software* in a location where the access and execute the programmes as specified in Ap. C

Appendix C User Guide

In Fig. 1 how the software that has been developed is organized. Each of the squares corresponds to a folder, while the files that are presented on them end with a *.py*.

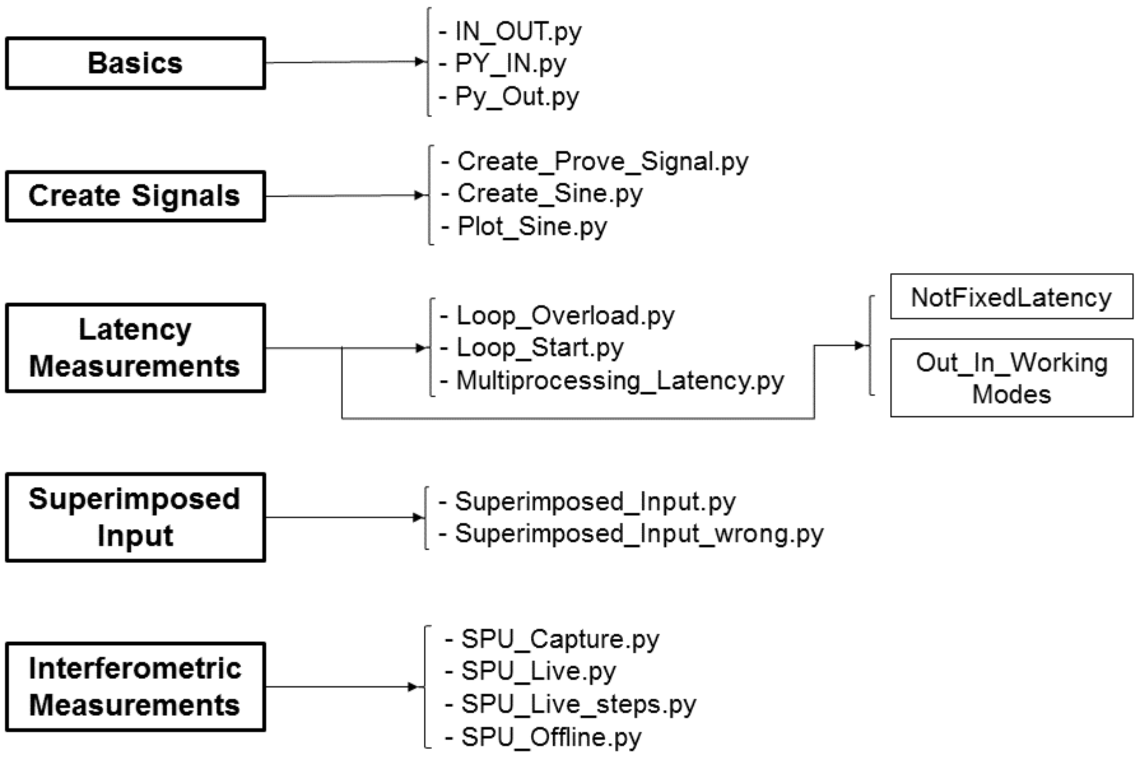


Figure 1: Developed Software

The developed software has been divided in 5 different folders, and their content and the tasks that each file can do will be detailed hereafter. It has to be mentioned that all the configuration parameters that each file requires are located at the beginning of each of them, with a detailed information about the values that these can take; so that this process is easier for the final user.

The first folder, *Basics*, has three files in order to test that the basic operations of the audio card can be done. That is, these three files allow to input audio and saved it onto a file, output audio data from a file and record and playback the recorded audio simultaneously.

In the second folder, *Create Signals*, the files are located the files that allow the creation of the different .wav files required for the development of the Signal Processing Unit. On the one hand there is the file that allows to create the sine modulation signal that is going to be outputted with the desired parameters. On the other hand there is a file that permits the creation of a signal similar to the one expected after the photo-detector, which can be used for testing purposes of the Signal Processing Unit if the output and the input are connected directly. Finally, there is a file that allows to plot the created sines just for testing purposes.

Thirdly there is the folder with the software used for making the latency measurement between the input and the output, *Latency Measurements*. In this folder two sub-folders are located; *Out_In_WorkingModes* where different programmes to perform the recording and playback of data on the Raspberry Pi and the Cirrus Logic audio card each one with a different working mode are located; and *NotFixedLatency* where the files with the working modes that give an unfixed latency together a measurement of it are located. Finally, in the main folder there are three files to measure the latency of the working mode that is known to have a fixed latency. This can measured just when a stream is open, after two minutes have passed with the Raspberry Pi overloaded and with a multiprocessing approach respectively.

The fourth folder, *Superimposed Input*, has two files in order to be able to see ten periods of an input signal one on top of the other, and therefore be able to test firstly if a working mode gives a fixed latency and later on the sample deviation that this particular working mode introduces.

Finally, there is the most important folder ***Interferometric Measurements***, where the Signal Processing Unit is implemented. The files located in this folder make use of the .wav files that have been created with the software located in the *Basics* folder to output a modulator sine, and they perform different measurements. The off-line version of the Signal Processing Unit just processes a period of the expected signal and it is implemented for testing purposes of the algorithm. Apart from that, there are two programmes for the Capture and Live mode respectively, which are the main implementation Software; and another programme which shows the demodulation process implemented, for the live mode, step by step.