LAAS-CNRS

INP TOULOUSE ENSEEIHT

Département de Formation Génie Electrique et Automatique

MASTER INTERNATIONAL
« GENIE ELECTRIQUE ET SYSTEMES ELECTRIQUES »

INTERNATIONAL MASTER
« ELECTRICAL ENGINEERING SYSTEMS »

- RAPPORT DE STAGE -

Titre:

Conception d'un démonstrateur électronique versatile pour la mesure de déplacements par réinjection optique dans une diode laser, avec contrôle du faisceau émis.

Nom de l'Etudiant:

VILCHIS MEDINA José Luis

Nom du Responsable:

PERCHOUX Julien / LUNA ARRIAGA Antonio

Année Universitaire 2013-2014

This work is dedicated to my parents, brothers
for their love, unconditional support
and encouragement.

Abstract

Self-Mixing (**SM**) interferometry in a laser diode (**LD**) is a very powerful tool in measurement science (e.g. Medical, Biomedical, Astronomy application, etc.). The SM interferometer is robust and low cost interferometer with extreme simplicity in alignment and setup.

In this work, SM interferometer theory is analyzed as a basis to understand the behavior of fringes detection algorithm to estimate the displacement of an object, once known SM theory, and using simulation from the original code in MATLAB, the interest is the transformation of code, from MATLAB to C, the results are shown using a MATLAB application named Coder, and also a second investigation is conducted by another graphic MATLAB application, Simulink Coder, both to code conversion, for this three different methods are developed to carry out the implementation to be implemented in a Digital Signal Processor (DSP).

The results obtained by applying these methods to different experimental data sets input are presented.

In the last part of this work, owing to my profile is oriented to control systems, is proposed the control of mirrors to scan a specific area. In order to add the embedded system developed and estimate the displacement of an area, not a static point, is presented.

Acknowledgements

The realization of this work was only possible due to the several people's collaboration, to which desire to express my gratefulness.

I would like to thank from a special way to Professor Thierry BOSH, director of the Optoelectronics for Embedded Systems (LAAS-OSE), having been a privilege to belong this work-team, and also my Professor Maria DAVID, director of the Master Degree MIGESE, for the unconditional support her.

To professors Julien PERCHOUX and Antonio LUNA, my supervisors, I am grateful for the trust deposited in my work and for the motivation demonstrated along this arduous course.

The all my colleagues and researchers from LAAS-OSE, thank for the encouragement, advices and suggestions of the work, thank for the friendship that always demonstrated along these months of realization of the work.

I would like express my appreciation to my friends, ***Joaquin Pulido Aguilar***, ***Kamil Mrozewski*** and ***Jodin Gurvan*** for the mode that my enthusiasm and encouragement.

Finally, I would like to thank so much to my parents, my brothers and God, because their love gave me forces to make this work.

Table of Contents

1. Introduction

❖ ***Location***

The internship was held at Laboratory for Analysis and Architecture of Systems (**LAAS**) in the Optoelectronics for Embedded Systems (**OSE**) research group installation, having as leader Prof. Thierry BOSCH, and including 8 professors, 1 post-doctoral student, 8 doctoral student and 2 trainees.

The main objective in LAAS-OSE group is investigating sensorics through the physical limits and performances of real-time remote embedded optoelectronic sensors (accuracy, signal to noise ratio, bandwidth, robustness in harsh environments, long term measurements, etc.) from solid targets to fluidics.

The research is paving the way to functional diversification of sensing devices by migrating from the system board-level into a particular package-level (SiP) or chip-level (SoC) system solution, according to the concept More than Moore (analog and mixed signal design technologies for sensors, new methods for co-design of SIP and biotechnology). The laboratory is essentially investigating new interferometric-based measuring systems and integrated electronics for optoelectronic systems.

❖ ***Previous knowledge***

An *interferometer* is an instrument used to measure waves through interference patterns. So, *interferometry* is the process by which two waves are combined so they can be studied for differences in their patterns. The fields of study where interferometry is used are astronomy, physics, optics, and oceanography. **[1]**

In conventional interferometers (e.g. Michelson), the different arrangements of lenses, beam splitters, and mirrors aim to provide a reference and a measurement beam traveling different optical paths. By gathering the mixed wave fronts with a receptor, a measurement can then be performed based on the obtained interference fringes. **[6]**

Optical feedback interferometry phenomenon can be simply understood as the interaction between an emitted laser beam and a small portion of backscattered light from a pointed target that re-enters the laser's cavity. It is manifested as a different kind of interference fringes produced by the active modulation in amplitude and frequency of the standing beam, therefore its common name of self-mixing interferometry. By exploiting this phenomenon, similar measurements to those with conventional interferometers can be expected. **[3]**

The self-mixing interferometers on its most common configuration are intended to use a laser diode as the light source, a built-in photodiode within the laser package, a focusing lens and the electronics circuitry suitable to exploit the interferometric signal recovered from the active cavity. Notable features of this sensing scheme are the reduced number of external optical components, the compactness, as well as its self-aligned and integrated nature.**[2]**

As technology advances, nowadays the cumbersome arrangement of conventional interferometers can be avoided with expensive fabrication process involving the difficult alignment of optical components. Conversely, basic self-mixing configuration is a cost-effective solution for laser interferometry instrumentation. **[4][5]**

❖ *Objective*

The objective of this work is based on the algorithm created by doctoral student, Antonio LUNA ARRIAGA (LAAS-OSE, 2014), which focuses on self-mixing displacement measurement to reconstruct the movement of a remote solid target, lies in transform the original algorithm created and verified on MATLAB program to C language, to have a code with ability to be versatile, flexible and incorporate into an embedded system, and the most important point: working in real-time, to be used on medical applications, this last topic has been the field of research these last years, at LAAS-OSE. About real-time, compute algorithm is critical to obtain this notion, for this application the best option for this important requirement was to use a Digital Signal Processor (**DSP**).

A last part of this work, have purpose for a control of mirrors to scan a target area, and at the same time estimate the displacements from this area, using the work done previously.

❖ *Tools of employment*

At present, exist some code converters, however the same MATLAB program includes an application to transform code, simple to use, named CODER. The ability to auto-generate C code, for a large subset of image processing and mathematical functions, which could then use in other environments, such as embedded systems or as a component in other software.

Another application used from MATLAB program, Simulink CODER, can automatically generate C source code for real-time implementation of systems. As the efficiency and flexibility of the code improves, this is becoming more widely adopted for production systems, in addition to being a popular tool for embedded system design work because of its flexibility and capacity for quick iteration.

Both applications are used to perform code generation in an automatic form, with the ultimate goal of autonomy, it's mean, to have the ability to generate any algorithm and load the code into the device. In order to have the most possible compact embedded system for the measurement of displacement of the object point, and the ability of the system response in real time, for example biomedical applications, and the last part was dedicated to study mirrors laser control to scan a determined area, measuring the displacement of the object surface and including embedded systems previously descripted.

Firstly, an introduction of interferometry and theory of SM are addressed to start this work and put the starting point for our development.

1.1. Theory

1.1.1. Interferometry

The above **Figure 1** shows a light source emitting waves of light. These waves are continually expanding out from the light source in all directions, just like ripples in a pond. The crests of the waves are represented by lines in the **Figure 1**. The wavelength of the light is the distance between the peaks. A particular wavelength of light corresponds to a particular color, from red (longest) to blue (shortest).

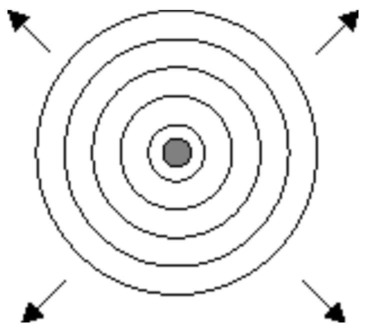


Figure 1. Source emitting waves of light.

As described in the context, an interferometer is an instrument that uses the interference patterns formed by waves (usually light, radio, or sound waves) to measure certain characteristics of the waves themselves or of materials that reflect, refract, or transmit the waves. Interferometers can also be used to make precise measurements of distance or in this case, displacement.

Now, Optical interferometers can be used as spectrometers for determining wavelengths of light and for studying fine details in the lines of a spectrum. Optical interferometers are also used in measuring lengths of objects in terms of wavelengths of light, providing great precision, and in checking the surfaces of lenses and mirrors for imperfections.

The introduction of laser diodes into Self-mixing interferometry can be seen as the boost element to spread this technique. This might be explained by the *low-cost* quality conferred to the instrument.

Self-mixing interferometry, or optical feedback interferometry is an interferometric measurement technique based on the optical mixing inside the laser cavity with the field back reflected by a remote object. This phenomenon is a very promising method. No external photo detector is required because de monitor photodiode of the laser diode package is able to pick up the signal by itself. Therefore, the device is simple, compact and easy to build and adjust.

In light of those achievements, it is appropriate to point out self-mixing phenomenon as an active research domain with high potential to incursion on industry and embedded applications.

According to the context, next part present SM theory phenomenon, more information detailed, see **[2]**

1.2. Self-Mixing

1.2.1. Theory

The self-mixing effect in a laser occurs when some part of light emitted by the laser re-enters the laser cavity. It can be modeled as two mirrors for the Fabry-Perot structure of the laser and the external mirror for the object reflecting the laser back. **[8]**

The three-mirror model can be used to explain self-mixing in lasers other than diode lasers as well however in this thesis we will focus on diode lasers. The schematic diagram of the self-mixing effect in a semiconductor laser is presented.

As shown in **Figure 2**, the mirror facets, M_1 and M_2 , are positioned at $z = 0$ and $z = L$ with an amplitude reflection coefficient, r_1 , r_{2S} , respectively. The external target (M_{ext}) is positioned at $z = L + L_{ext}$ with an amplitude reflection coefficient, r_{2ext1} .

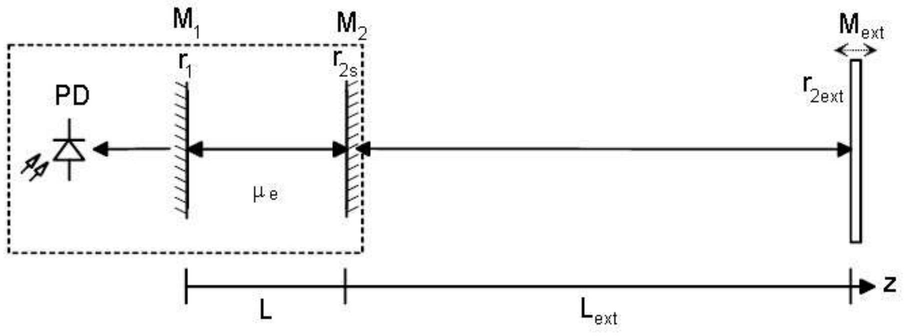


Figure 2. Diagram of self-mixing.

We replace the mirror at $z = L$ and the mirror at $z = L + L_{ext}$ by an effective mirror at $z = L$. The reflection coefficient, r_c , for the new compound cavity is given by $r_c(n) = r_{2s} + (1 - R_{2s})r_{2ext}e^{-ipn_c t_{ext}}$ where n_c is the optical frequency of the compound cavity, $t_{ext} = 2L_{ext}/c$ is the round trip delay time, and c is the speed of light. **[7]**

The term $(1 - R_{2s})$ is the fraction of light that passes through the laser diode facet at $z = L$ and is consequently reflected by the external reflector with amplitude reflection coefficient of r_{2ext}. The exponential term add $2pn_c t_{ext}$ (i.e. $w_c t_{ext}$) of phase change for one round trip.

The following section explains the analysis of laser diodes with optical feedback using Lang and Kobayashi equations.

Let us again consider Figure 2, where the laser diode is modeled by mirrors M_1 and M_2. A fraction of the optical beam from the laser diode is backscattered into the laser diode cavity by the external target. The photodiode at the rear facet of the laser diode monitors the power, P, of the laser diode by means of photogenerated current, $I = sP$, where s is spectral responsivity.**[9]**

According to the Lang and Kobayashi's equations, the dynamics of a laser diode in the presence of feedback can be written as:

$$\frac{d}{dt}E_0(t) = \frac{1}{2}\left[G_N(N(t) - N_0) - \frac{1}{\tau_P}\right]E_0(t) + \frac{k}{\tau_L}E_0(t-\tau)\cos\left(\omega_0\tau + \phi(t) - \phi(t-\tau)\right)$$

$$\frac{d}{dt}\phi(t) = \frac{1}{2}\alpha G_N[N(t) - N_T] - \frac{k}{\tau_L}\frac{E_0(t-\tau)}{E_0(t)}\sin\left(\omega_0\tau + \phi(t) - \phi(t-\tau)\right)$$

$$\frac{d}{dt}N(t) = R_P - \frac{N(t)}{\tau_s}G_N(N(t) - N_0){E_0}^2(t)$$

The instantaneous optical frequency is given by:

$$\omega(t) = \omega_0 + [\frac{d}{dt}\phi(t)]$$

The contribution $\frac{d}{dt}\phi(t)$ represents a frequency deviation.

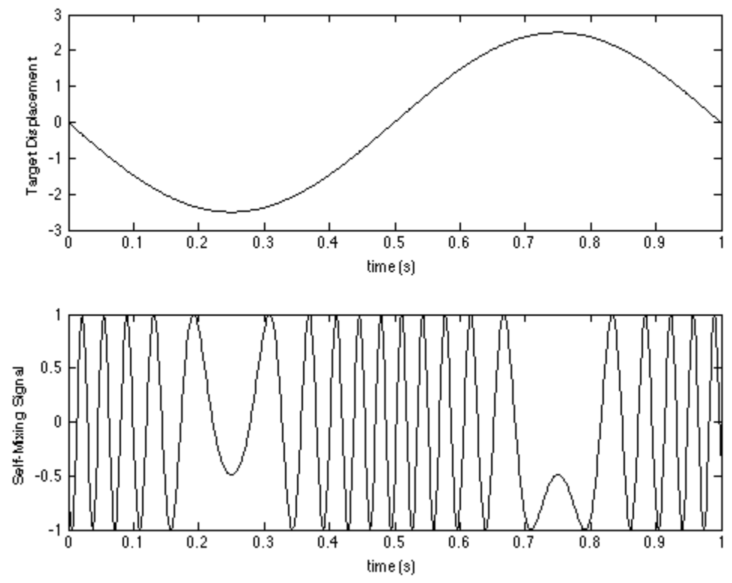
One of the parameters most important SM is described in the next part.

1.2.2. C - Parameter

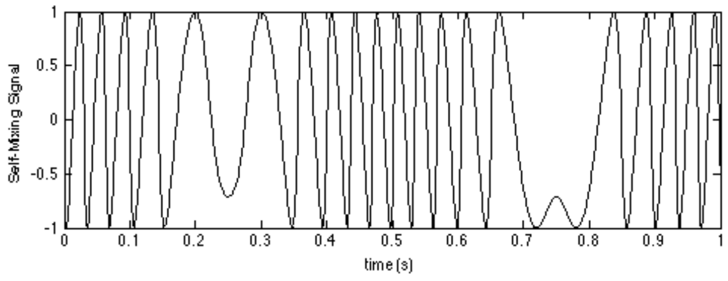
Base on this behavior, feedback parameter C, can be defined as:

$$C = \zeta\frac{\tau_{ext}}{\tau_l}\sqrt{(1+\alpha^2)}$$

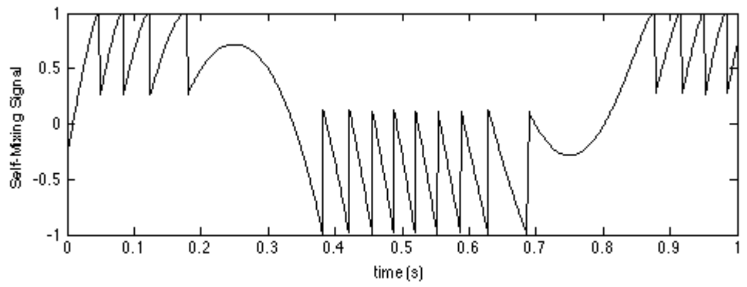
Adimensional parameter *C* is of great importance in SM metrology, it depends simultaneously on the time of flight through the external cavity τ_{ext} and therefore on the distance to the target. It relies also on the target to LD coupling parameter ζ, and therefore on the quantity of light retro-diffused by the target. **[2]**



a)



b)



c)

Figure 3. a) Up, moving target curve and down, SM signal output with parameter C=0.001 **b)** SM signal output with parameter C=0.5 **c)** SM signal output with parameter C=5.

The graphs above show the different curves that can be obtained according to the parameter C. With this value of the parameter C, it can identify in 4 categories, in the **Figure 3** the values and waveform are observed.

1.2.3. Measuring Displacement by SM

Having explained the basics concepts of interferometry, self-mixing signal and parameter C, next, the algorithm presented has the main purpose detects the fringes of the self-mixing signal input, which is important information for the process of reconstruction of the signal displacement, following in the figure show the purpose. **[2]**

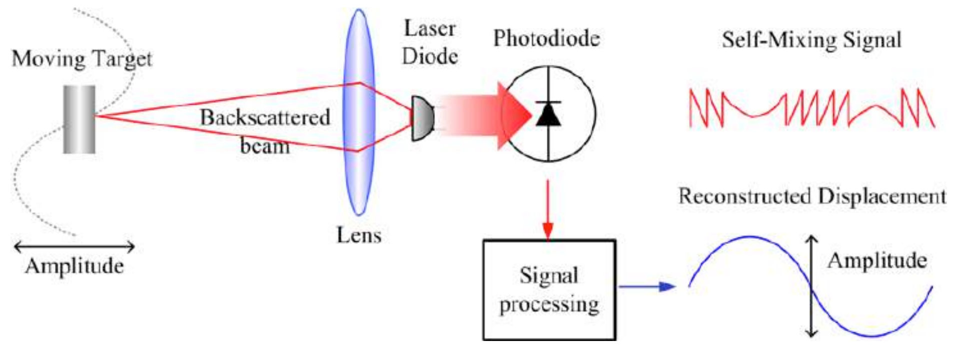


Figure 4. Measurement of displacement by Self-Mixing phenomenon

Considering the basic SM setup of **Figure 4**, when the target is moving on the optical axis of the falling beam, the length of the external cavity is varying and thus a SM signal can be recovered. A proper exploitation of the signal shall lead to retrieve the target's displacement.

The information on the direction of the target movement is implicit on each of the interferometric fringes. For the sake of illustration, **Figure 3.b** is a signal on Matlab software, obtained with a target performing a sinusoidal movement, **Figure 3.a**

As described the objective in the context, the important part is the detection of fringes to subsequently predict the movement of an object, thanks to the phenomenon of SM. Below is the structure of the system and the way to attack the problem.

1.3. Fringe Detection Algorithm

1.3.1. Introduction

Reconstruction process allows describe displacement of the objet, one time captured a SM signal, and using mathematical methods, the purpose is to provide a train of peaks corresponding to a temporal detection of SM phase transitions, all this compute it in a DSP device. Next, to understand the most important part of the algorithm, Hilbert transform, a description of this transform is presented.

1.3.2. Hilbert Transform

Hilbert transform finds a companion function $y(t)$ for a real function $x(t)$ so that $z(t) = x(t) + i * y(t)$ can be analytically extended from the real line $t \epsilon \mathbb{R}$ to upper half of the complex plane.

In the field of signal processing, Hilbert transform can be computed in a few steps:

- First, calculate the Fourier transform of the given signal $x(t)$.
- Second, reject the negative frequencies.
- Finally, calculate the inverse Fourier transform, and the result will be a complex-valued signal where the real and the imaginary parts form a Hilbert-transform pair.

When x(t) is narrow-banded, $|z(t)|$ can be regarded as a slow-varying envelope of $x(t)$ while the phase derivative $\frac{\partial}{\partial t}\tan^{-1}(\frac{y}{x})$ is an instantaneous frequency.

Thus, Hilbert transform can be interpreted as a way to represent a narrow-band signal in terms of amplitude and frequency modulation.

Once known Hilbert transform theory, fringe detection algorithm is next presented.

1.3.3. Algorithm

Fringe detection algorithm can be seem like a general block, **Figure 5**, called Hilbert method, having a structure of seven individual blocks, which are described for more details.

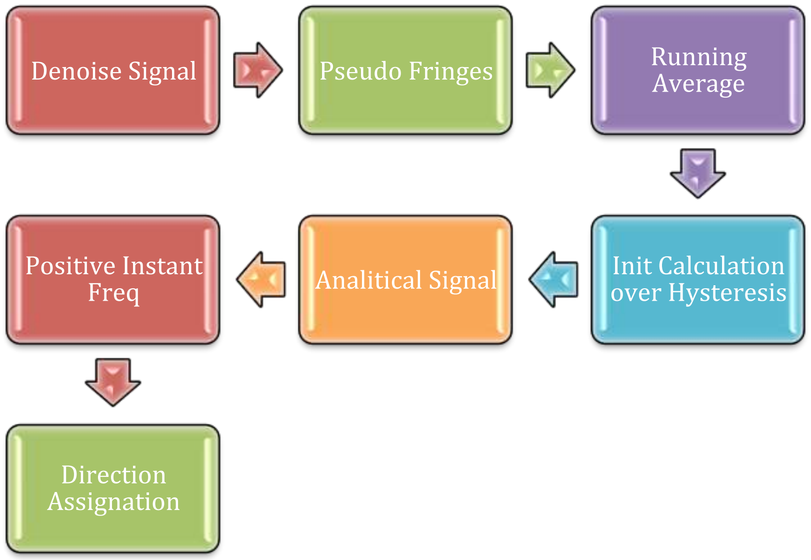


Figure 5. Algorithm diagram blocks

1.3.4. Blocks description

This part have as purpose describe in a general way all the blocks, for more details of each function, consult **[2]**.

- **Denoise Signal**

This function has 2 purpose, one of them is calculate Hilbert transform to estimate raw phases and then calculate noise percentage, and the second one, apply a digital filter (Moving average filter) to input vector depending on noise percentage criteria.

- **Pseudo Fringes**

Function to derivate input vector, then positive and negatives points based on derivate to estimate and generate a logical vector.

- **Running Average**

Function to estimate average according a frame calculated in Denoise Signal block.

- **Init Calculation over Hysteresis**

Function to subtraction denoise signal output block and running average signal output.

- **Analytical Signal**

Function to calculated Hilbert transform, angle phase and derivate.

- **Positive Instant Freq**

Function to remove the next transition found if a positive frequency peak is found this positive frequency peak means a biased Hilbert transformation.

- **Direction Assignation**

There are 2 methods to apply in this block: one is based on duty cycle algorithm and the other with Transition generating method.

The functions presented before are explained with more details in the final program file made it in Code Composer Studio.

Now, having knowledge of the algorithm and structure of each of the blocks, described earlier, in the next part of the text describes the tools for code conversion.
In the first investigation, Code Composer Studio, MATLAB Coder and Simulink Coder are presented as a solution to the problem described in the context.

2. Required Tools

2.1. Software

2.1.1. Code Composer Studio

Texas Instrument has a variety of development tools available that enable quick movement through the digital signal processor (DSP) based application design process from concept, to code/ build, through debug analysis, tuning, and on to testing. Many of the tools are part of TI's real-time eXpressDSP software and development tool strategy, which is very helpful in quickly getting started as well as saving valuable time in the design process. TI's real-time eXpressDSP Software and Development Tool strategy includes three components that allow developers to use the full potential of TMS320 DSPs:

• Powerful DSP-integrated development tools in Code Composer Studio (CCS)

• eXpressDSP Software, including:

- ✓ Scalable, real-time software foundation: DSP/BIOS kernel
- ✓ Standards for application interoperability and reuse: TMS320 DSP Algorithm Standard
- ✓ Design-ready code that is common to many applications to get you started quickly on DSP design:
  - ▪ eXpressDSP Reference Frameworks

• A growing base of TI DSP-based products from TI's DSP Third Party Network, including eXpressDSP, compliant products that can be easily integrated into systems.

2.1.1.1. Validation

During the validation process of the code generated by MATLAB, took place several problems which some could solve with ease.
Unfortunately, other issues such as the incompatibility of syntax could not be resolved.
The problem detected was about compiler, limited standard syntax, C-89/C-90 ANSI.

The possibility to use MATLAB Coder to transform the original code it takes to the board development compatibility is next presented.

2.1.2. MATLAB Coder

MATLAB Coder generates standalone C and C++ code from MATLAB code. The generated source code is portable and readable. MATLAB Coder supports a subset of core MATLAB language features, including program control constructs, functions, and matrix operations.

Prerequisites for Code Generation from MATLAB to generate C/C++ from MATLAB algorithms, you must install the following software:

• MATLAB® Coder™ product

• C/C++ compiler

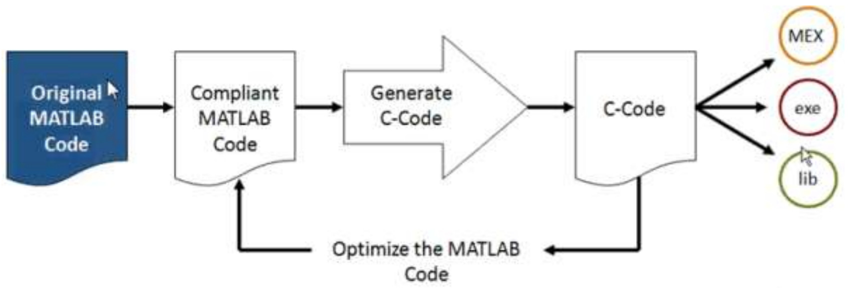


Figure 6. MATLAB Coder diagram

Translating MATLAB algorithms to C code involves specifying implementation requirements. The good point is that MATLAB Coder produces a report that identifies any errors, which must be fixed to make your MATLAB algorithm compliant for code generation, **Figure 6**.

✓ C or C++ source code enables to:

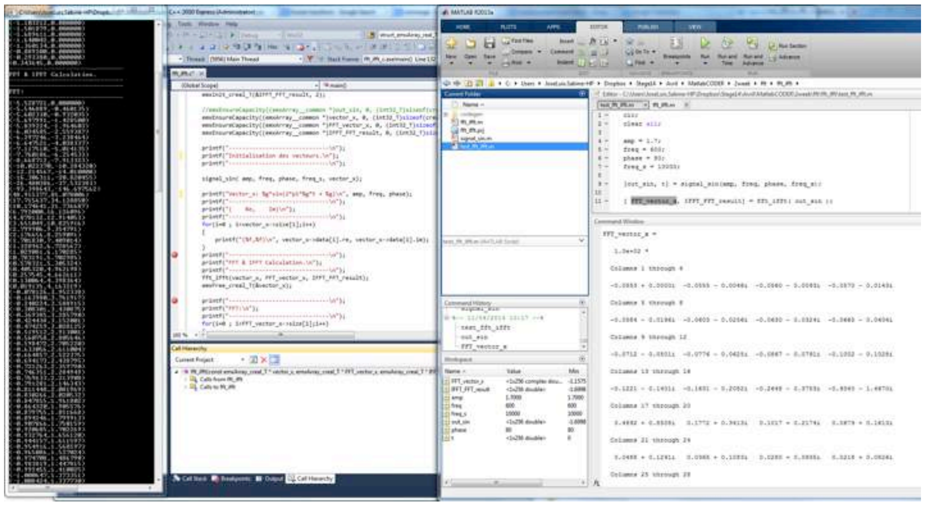
▪ Prototype on PCs

▪ Create a library

▪ Implement as embedded code

2.1.2.1. Validation

Verification of the code generated by MATLAB Coder was made with the Visual C++ express 2010 program, **Figure 7** and **Figure 8**, which is a free development environment to develop applications, in this part, the initialization of the input vectors, calculation and outputs, they had to be declared, because MATLAB Coder develops only the function to use.
In parallel, the results obtained were compared with MATLAB obtaining the same results.



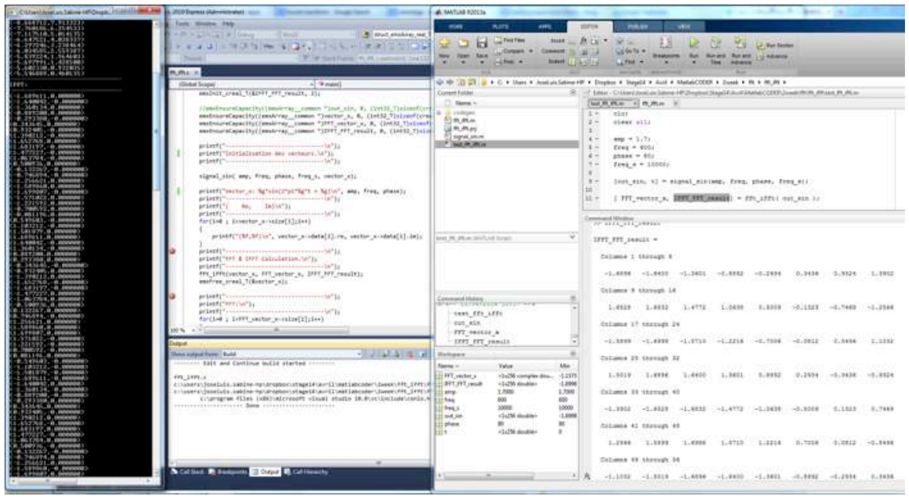
```
clc;
clear all;

amp = 1.7;
freq = 600;
phase = 80;
freq_s = 10000;

[out_sin, t] = signal_sin(amp, freq, phase, freq_s);

[ FFT_vector_x, IFFT_FFT_result] = fft_ifft( out_sin );
```

Figure 7. Code verification (FFT function) on Visual C++ Express 2010



```
clc;
clear all;

amp = 1.7;
freq = 600;
phase = 80;
freq_s = 10000;

[out_sin, t] = signal_sin(amp, freq, phase, freq_s);

[ FFT_vector_x, IFFT_FFT_result] = fft_ifft( out_sin );
```

Figure 8. Code verification (IFFT function) on Visual C++ Express 2010

Code final generated had complex function structures, doubles pointers that was no easy to have a clear notion to understand what really happen with all data operation. Spend time to get an idea more solid about operation results, in this part of work.

At this point, code conversion was successfully performed, in keeping with the context, the description of the device to be used for the validation of the transformation of the code is presented.

Due to the possibility of generating the transformation of code on graphic form, investigated and carried out another method more complex by previous configurations, is now described.

2.1.3. Simulink Coder

The MATLAB Function block for simulation and code generation lets you add MATLAB algorithms written in the MATLAB subset for integrating MATLAB code into Simulink models. Simulink Coder lets you generate code from these Simulink models that contain MATLAB code.

Embedded Coder generates code for supported embedded processors, on-target rapid prototyping boards, and microprocessors used in mass production. It extends MATLAB Coder and Simulink Coder by providing configuration options and advanced optimizations for fine-grain control of the generated code's functions, files, and data. Embedded Coder improves code efficiency and facilitates integration with legacy code, data types, and calibration parameters used in production.
Code generated with Embedded Coder can be exported into third-party development environments, enabling you to automate the creation of an executable to deploy on your embedded system. The generated code can also be executed on processors to verify behavioral performance and gather resource utilization metrics through processor-in-the-loop (PIL) and profiling techniques.

Once correctly installed CCS and having appropriate licenses of Simulink Coder, can establish communication from Simulink graphical environment to the device DSP, creating a work environment simpler which allows an algorithm easier to understand the design, because the programming is through blocks.

2.1.3.1. Validation

Simulink validation was not possible due principally to the problem with MATLAB license, DSP modules configuration were not available in the MATLAB used. Making the communication impossible between this application and with the DSP, for the validation of this part.

Problems generated and listed in the previous paragraphs of MATLAB Coder and Simulink Coder, led us to the conclusion for the transformation of code manually.
Taking as reference the functions and math operation of the original code in MATLAB.

The description of the DSP used for the validation and implementation of the code, once transformed, as it is mentioned in the context, next is presented.

2.2. Hardware

2.2.1. Digital Signal Processor (DSP)

Looking around, we find ourselves to be surrounded by various types of embedded systems. Be it a digital camera or a mobile phone or a washing machine, all of them has some kind of processor functioning inside it. Associated with each processor is the embedded software. If hardware forms the body of an embedded system, embedded processor acts as the brain, and embedded software forms its soul. It is the embedded software which primarily governs the functioning of embedded systems.
During infancy years of microprocessor based systems, programs were developed using assemblers and fused into the EPROMs. There used to be no mechanism to find what the program was doing. LEDs, switches, etc. were used to check correct execution of the program. Some 'very fortunate' developers had In-circuit Simulators (ICEs), but they were too costly and were not quite reliable as well.

In brief, DSPs are processors or microcomputers whose hardware, software, and instruction sets are optimized for high-speed numeric processing applications an essential for processing digital data representing analog signals in real time. What a DSP does is straightforward. When acting as a digital filter, for example, the DSP receives digital values based on samples of a signal, calculates the results of a filter function operating on these values, and provides digital values that represent the filter output; it can also provide system control signals based on properties of these values. The DSP's high-speed arithmetic and logical hardware is programmed to rapidly execute algorithms modelling the filter transformation.

The combination of design elements arithmetic operators, memory handling, instruction set, parallelism, data addressing that provide this ability forms the key difference between DSPs and other kinds of processors, **Figure 9**.

Understanding the relationship between real-time signals and DSP calculation speed provides some background on just how special this combination is.

The real-time signal comes to the DSP as a train of individual samples from an analog-to-digital converter (ADC). The DSP must complete all the calculations and operations required for processing each sample (usually updating a process involving many previous samples) before the next sample arrives. When it comes to digital signal processing, the preferred name is Digital Signal Processor (DSP).

The DSP used for the application and validation of algorithms once converted to C language is the development board eZdsp with TMS320F28335, **Figure10**.

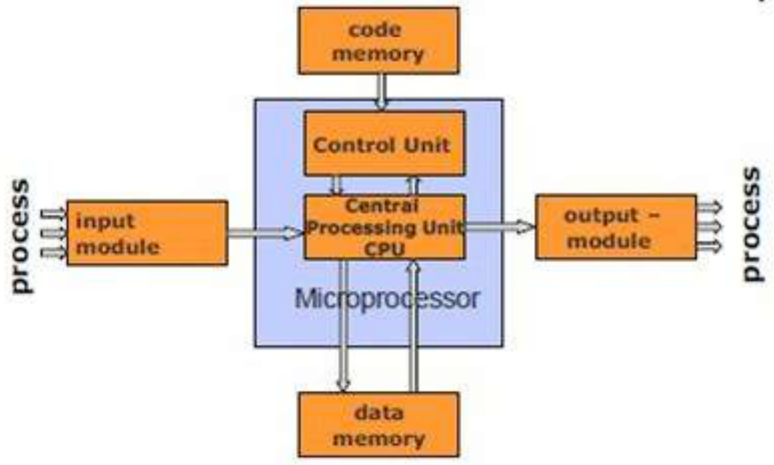


Figure 9. F28335 microprocessor block diagram

A Digital Signal Processor is a specific device that is designed around the typical mathematical operations to manipulate digital data that are measured by signal sensors. The objective is to process the data as quickly as possible to be able to generate an output stream of new data in real time, some application can be observed in the next figure.

Figure 10. DSP Applications

The utilization of the DSP, it was suggested by the doctoral student, owed a previous study, by the compatibility by MATLAB, speed calculation, floating point operation, etc. In the following section describe the characteristics of the DSP to using.

2.2.2. eZdsp TMS320F28335

The eZdsp F28335 is a stand-alone card--allowing developers to evaluate the TMS320F28335 digital signal controller (DSC) to determine if it meets their application requirements. Furthermore, the module is an excellent platform to develop and run software for the TMS320F28335 processor.

The eZdsp F28335 is shipped with a TMS320F28335 DSC. The eZdsp F28335 allows full speed verification of F28335 code. Several expansion connectors are provided for any necessary evaluation circuitry not provided on the as shipped configuration.

To simplify code development and shorten debugging time, a C2000 Code Composer Studio driver is provided. In addition, an onboard JTAG connector provides interface to emulators, with assembly language and 'C' high level language debug.

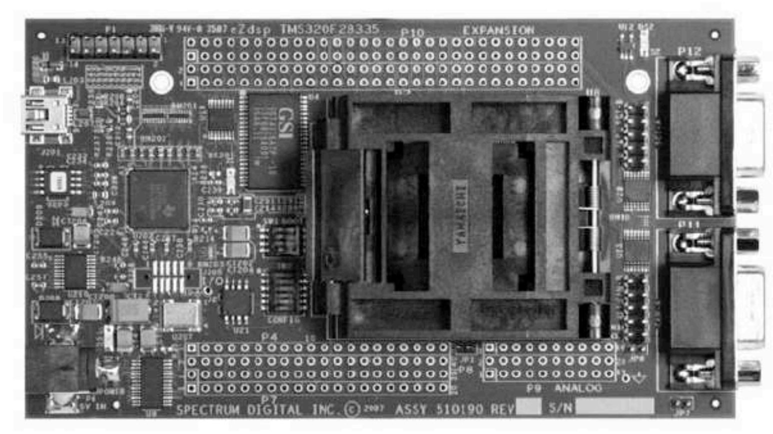


Figure 11. eZdsp F28335

The eZdsp F28335 has the following main features:

❖ ***Hardware Features***

- 150 Mhz. operating speed
- On chip 32-bit floating point unit
- On chip 12 bit Analog to Digital (A/D) converter with 16 input channels
- 30 MHz input clock
- Multiple Expansion Connectors (analog, I/O)
- On board embedded USB JTAG Controller

❖ ***Software Features***

- TI F28xx Code Composer Studio Integrated Development Environment, Version 3.3
- Texas Instruments' Flash APIs to support the F28335
- Texas Instruments' F28335 header files and example software

The intention of this prototype focus on the need to realize biomedical measurements in embedded systems, first time done at LAAS-OSE, for easy transportation and in a future to commercialization of a model capable to compete with other important manufacturers of the world.

3. Developed Prototype

3.1. Code implementation into DSP

The manual conversion of code was the last option for carry out, this because the main idea described in the context, it was to have the ability to generate code automatically with the DSP previously described.
However, the problems of syntax and software licenses ended up choosing this manual method.

As in previous fragment is mentioned the manual code conversion, transformation between MATLAB Code and C code was slow, because each result obtained was verify with the original code on MATLAB, each function, each operation. Next part below shows the final result of the transformed code.

In the **Figure 12**, shows thee carried out methods, which only one was realized with total success.

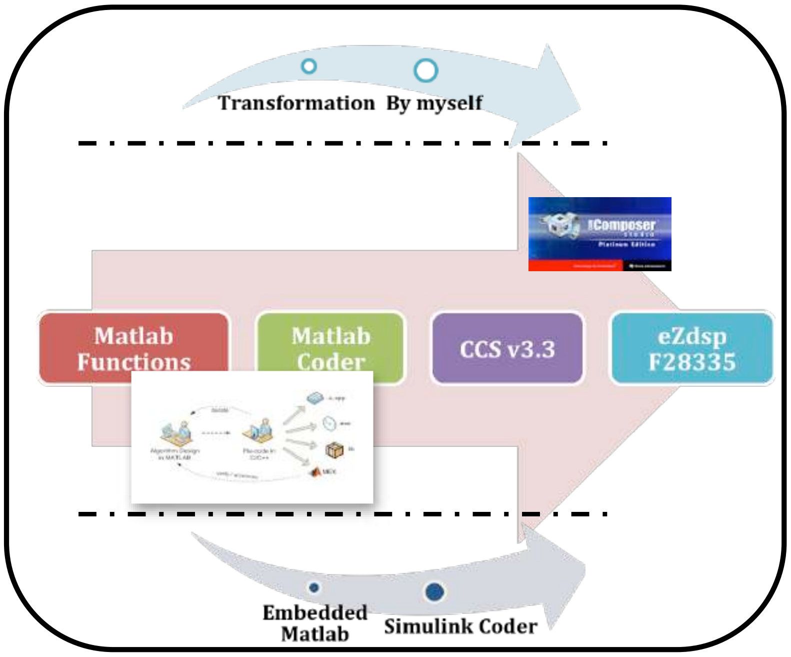
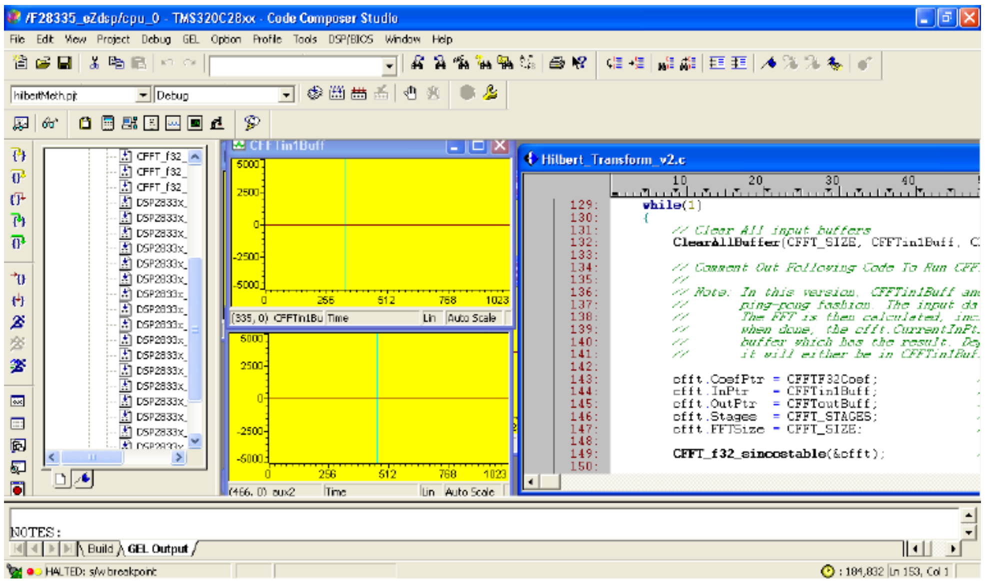


Figure 12. Methods used

The code transformed manually and validated on CCS, starts by cleaning all the vectors used in the calculations, having a time of 1.23 µsec, **Figure 13**


/F28335_eZdsp/cpu_0 - TMS320C28xx - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
hilbertMeth.pjt
Debug
CFFTin1Buff
Hilbert_Transform_v2.c
NOTES:
Build GEL Output
HALTED: s/w breakpoint
: 184,832 Ln 153, Col 1

```
while(1)
{
    // Clear All input buffers
    ClearAllBuffer(CFFT_SIZE, CFFTin1Buff, C

    // Comment Out Following Code To Run CFF
    //
    // Note: In this version, CFFTin1Buff an
    //       ping-pong fashion. The input da
    //       The FFT is then calculated, inc
    //       when done, the cfft.CurrentInPt
    //       buffer which has the result. De
    //       it will either be in CFFTin1Buf

    cfft.CoefPtr = CFFTF32Coef;
    cfft.InPtr   = CFFTin1Buff;
    cfft.OutPtr  = CFFToutBuff;
    cfft.Stages  = CFFT_STAGES;
    cfft.FFTSize = CFFT_SIZE;

    CFFT_f32_sincostable(&cfft);
```

Figure 13. Clear buffer function with 184832 CPU cycle.

In MATLAB the SM signal is represented in the figure, with C-parameter equal to 5, having a good representation of the real SM signal (**Figure 14**) to compare the results of the calculation next done. Capture of real SM signal in figure is presented, with 3129820 CPU cycle and 20.86 µsec execution time, using a vector size of 2048 with floating point representation.

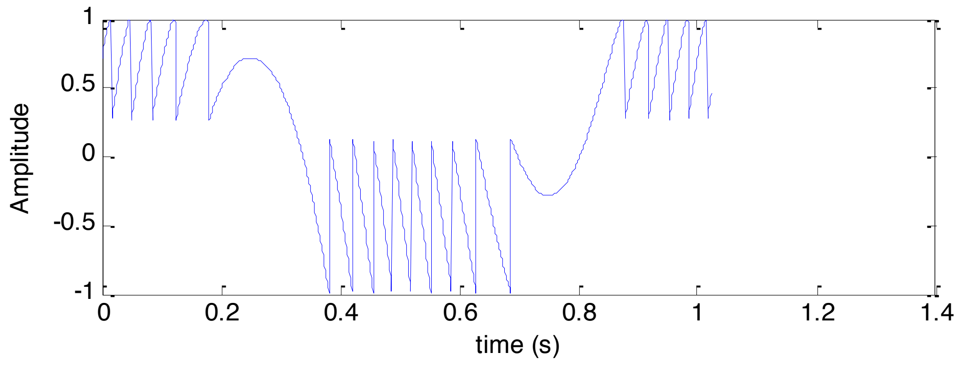
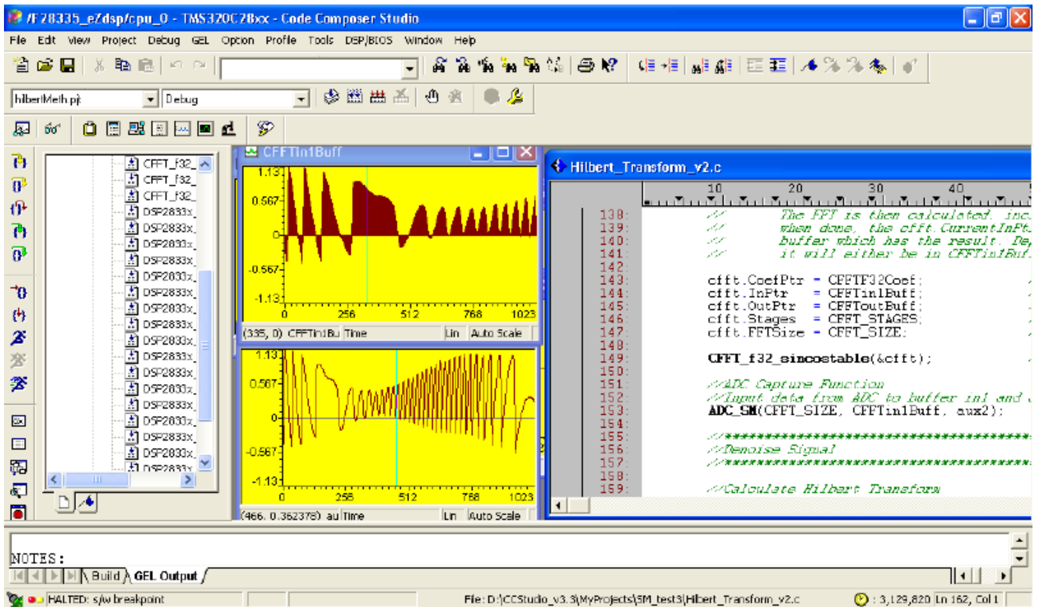


Figure 14. MATLAB SM signal input


/F28335_eZdsp/cpu_0 - TMS320C28xx - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
hilbertMeth.pjt
Debug
CFFTin1Buff
Hilbert_Transform_v2.c
NOTES:
Build
GEL Output
HALTED: s/w breakpoint

```
    //        The FFT is then calculated, inc
    //        when done, the cfft.CurrentInPt
    //        buffer which has the result. De
    //        it will either be in CFFTin1Buf

    cfft.CoefPtr = CFFTF32Coef;
    cfft.InPtr   = CFFTin1Buff;
    cfft.OutPtr  = CFFToutBuff;
    cfft.Stages  = CFFT_STAGES;
    cfft.FFTSize = CFFT_SIZE;

    CFFT_f32_sincostable(&cfft);

    //ADC Capture Function
    //Input data from ADC to buffer in1 and 
    ADC_SM(CFFT_SIZE, CFFTin1Buff, aux2);

    //****************************************
    //Denoise Signal
    //****************************************

    //Calculate Hilbert Transform
```

File: D:\CCStudio_v3.3\MyProjects\SM_test3\Hilbert_Transform_v2.c : 3,129,820 Ln 162, Col 1

Figure 15. ADC capture, on top graph, 2048 samples were sampling having a complex representation to the next calculation, on bottom graph, 1024 samples were saved in a vector having real representation

3.2. Hilbert transform example

In this part of the algorithm it applies the Hilbert transform, angle function and difference function (discrete derivate function analogy, **Figure 16**), then with the results of these operation detects the phase of the SM input signal to calculate the noise percentage and to eliminate it, also the Hilbert transform is implemented in order to generate a unique analytic representation in the complex plane of the data and as a practical implementation to track frequency variations over the time. **[15][16]**

Having 1263614 CPU cycle with an execution time of 8.42 µsec, in practice, **Figure 17**.

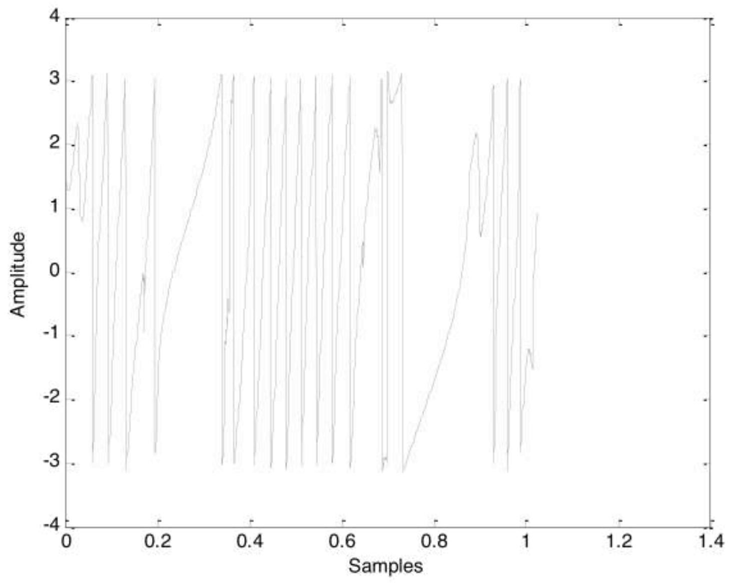
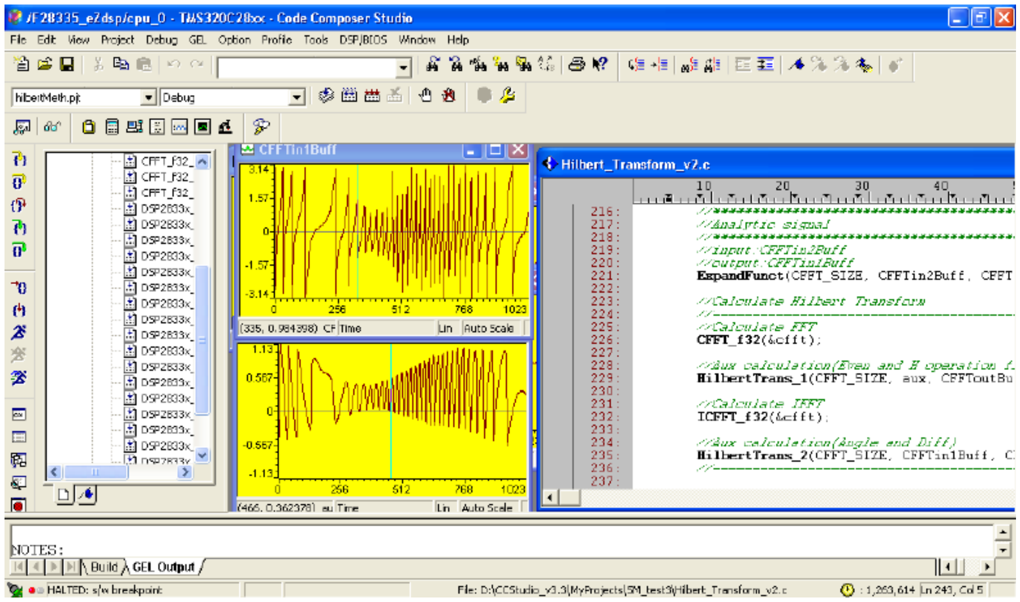


Figure 16. MATLAB Hilbert transform result


/F28335_eZdsp/cpu_0 - TMS320C28xx - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
hilbertMeth.pjt
Debug
CFFTin1Buff
Hilbert_Transform_v2.c
NOTES:
Build
GEL Output
HALTED: s/w breakpoint
File: D:\CCStudio_v3.3\MyProjects\SM_test3\Hilbert_Transform_v2.c
: 1,263,614
Ln 243, Col 5

```
//*****************************************
//Analytic signal
//*****************************************
//input:CFFTin2Buff
//output:CFFTin1Buff
ExpandFunct(CFFT_SIZE, CFFTin2Buff, CFFT

//Calculate Hilbert Transform
//-----------------------------------------
//Calculate FFT
CFFT_f32(&cfft);

//Aux calculation(Even and H operation f
HilbertTrans_1(CFFT_SIZE, aux, CFFToutBu

//Calculate IFFT
ICFFT_f32(&cfft);

//Aux calculation(Angle and Diff)
HilbertTrans_2(CFFT_SIZE, CFFTin1Buff, C
//-----------------------------------------
```

Figure 17. On top, Hilbert transform result of a real SM signal (On bottom), both vectors having a size of 1024 samples.

In MATLAB, assignation of the direction is based on duty cycle algorithm (**Figure 18**), having as input Hilbert transform operation result, before explained, and SM signal filtered. **[2]**
In practice (figure), the function was computed, having a vector input size of 1024 samples and the same size for the result, this function had 495941 CPU cycle and an execution time of 3.30 µsec, **Figure 19**.

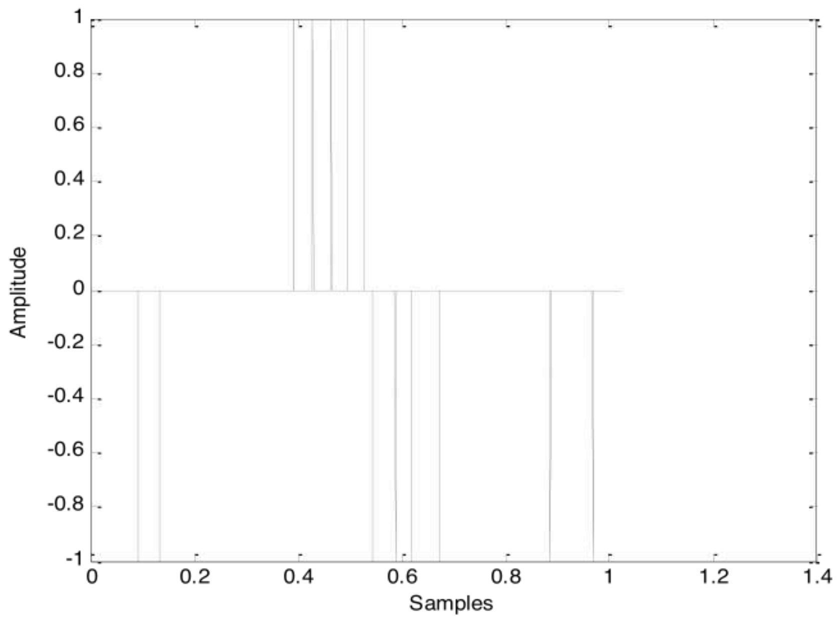
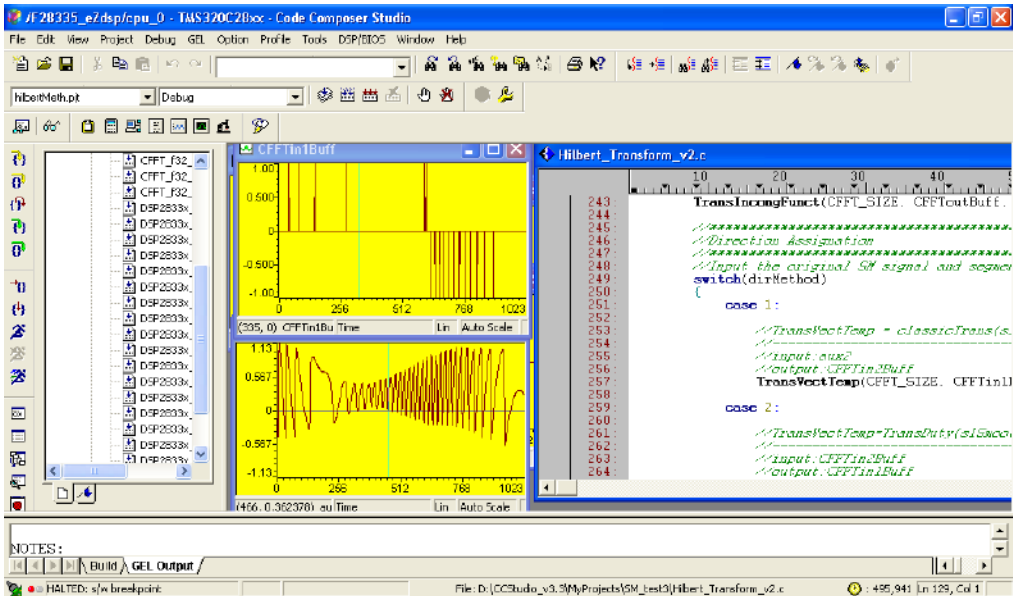


Figure 18. MATLAB direction assignation function result


/F28335_eZdsp/cpu_0 - TMS320C28xx - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
hilbertMeth.pjt
Debug
CFFTin1Buff
Hilbert_Transform_v2.c
(335, 0) CFFTin1Bu Time Lin Auto Scale
(466, 0.362378) au Time Lin Auto Scale
NOTES:
Build GEL Output
HALTED: s/w breakpoint
File: D:\CCStudio_v3.3\MyProjects\SM_test3\Hilbert_Transform_v2.c
495,941 Ln 129, Col 1

```
        TransIncongFunct(CFFT_SIZE, CFFToutBuff,

        //****************************************
        //Direction Assignation
        //****************************************
        //Input the original SM signal and segme
        switch(dirMethod)
        {
            case 1:

                //TransVectTemp = classicTrans(s
                //--------------------------------
                //input:aux2
                //output:CFFTin2Buff
                TransVectTemp(CFFT_SIZE, CFFTin1

            case 2:

                //TransVectTemp=TransDuty(s1Sacc
                //--------------------------------
                //input:CFFTin2Buff
                //output:CFFTin1Buff
```

Figure 19. On bottom, real SM signal, on top, direction assignation function result, both vectors having a size of 1024 samples.

The final function to detect the fringes in MATLAB (**Figure 20**) has as vector input the direction assignation result, previously described, and a tolerance constant, this two information to detect the fringes on a SM signal input.
Implementation on CCS (**Figure 21**) demonstrates the algorithm result, with a execution time of 47.23 μsec without initialization part, and 7085612 CPU cycle. Input vector of this function have a size of 1024 samples, vector result have the same size.

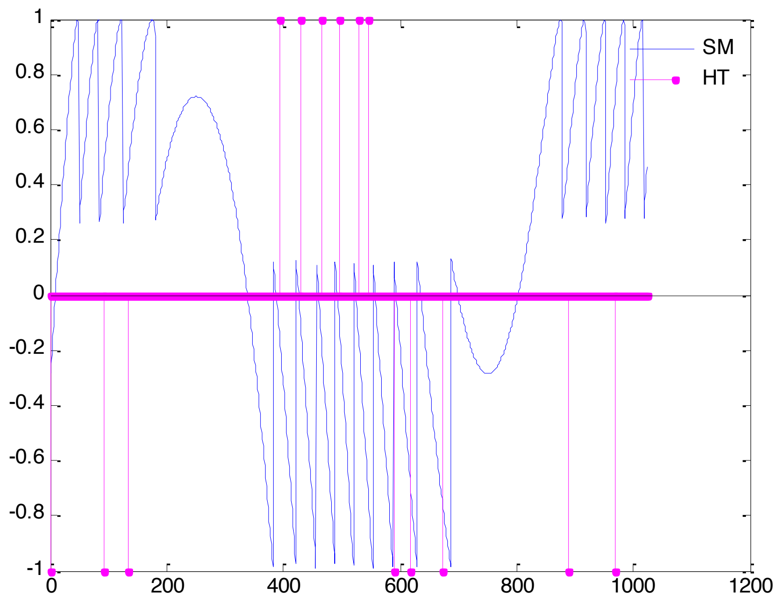
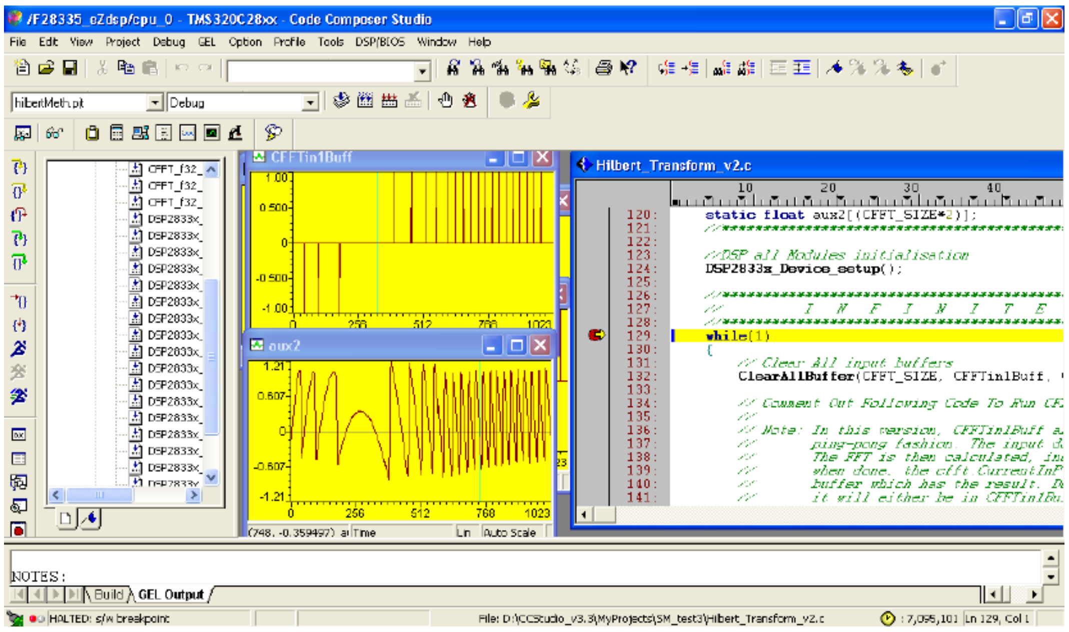


Figure 20. MATLAB fringes detection function result


/F28335_eZdsp/cpu_0 - TMS320C28xx - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
hibertMeth.pjt
Debug
CFFTin1Buff
aux2
Hilbert_Transform_v2.c
NOTES:
Build
GEL Output

```
static float aux2[(CFFT_SIZE*2)];
//***********************************************
//DSP all Modules initialisation
DSP2833x_Device_setup();

//***********************************************
//          I   N   F   I   N   I   T   E
//***********************************************
while(1)
{
    // Clear All input buffers
    ClearAllBuffer(CFFT_SIZE, CFFTin1Buff, 

    // Comment Out Following Code To Run CF
    //
    // Note: In this version, CFFTin1Buff a
    //       ping-pong fashion. The input d
    //       The FFT is then calculated, in
    //       when done. the cfft.CurrentInP
    //       buffer which has the result. D
    //       it will either be in CFFTin1Bu
```

```
(748, -0.359497) a Time    Ln    Auto Scale
HALTED: s/w breakpoint    File: D:\CCStudio_v3.3\MyProjects\SM_test3\Hilbert_Transform_v2.c    : 7,095,101  Ln 129, Col 1
```

Figure 21. On top, fringes detection function computed result, on bottom, real SM signal captured, both vectors on graph having a size of 1024 samples.

In the **Figure 21**, Code Composer Studio program in which is developing the conversion of code manually.
This conversion method was used like an alternative, because the problems explained before refused to do it in the right way, like in context describe.

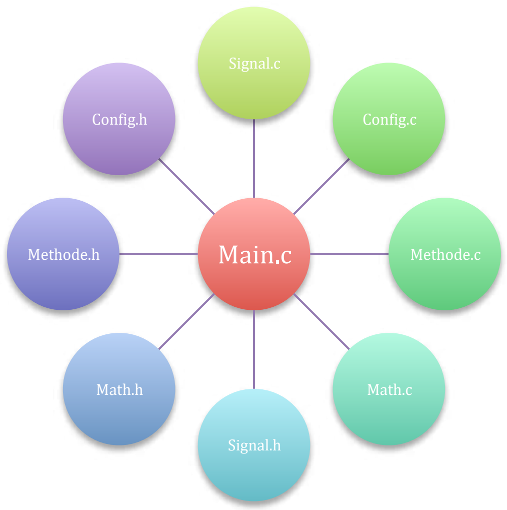


Figure 22. Functions created manually

3.3. Modules files description

In the **Figure 22**, presents the diagram used for the comprehension of the code done manually. The description of each block is shown below:

- **Main.c** file contain principal program where all function are called.
- **Signal.c** has SM signal functions to use.
- **Signal.h** has declaration of the SM functions.
- **Config.c** has all modules Setup functions.
- **Config.h** has declaration of the Setup functions.
- **Methode.c** contain the Hilbert method functions, and fringe detection function (figure 5).
- **Methode.h** contain declaration of the functions of the Hilbert method, and fringe detection functions (figure 5).
- **Math.c** has mathematical functions used for the Hilbert method.
- **Math.h** has declaration of the mathematical functions used for the Hilbert method.

Having done all the functions of the algorithm, took place the implementation for validation using the MATLAB results that next described.

3.4. Experiment

In the implementation of the system, we used a low frequency generator to reproduce the signal SM (piece of metal data target) with 1 Vpp at 10 Hz, to be subsequently injected to the DSP as input, **Figure 23**.

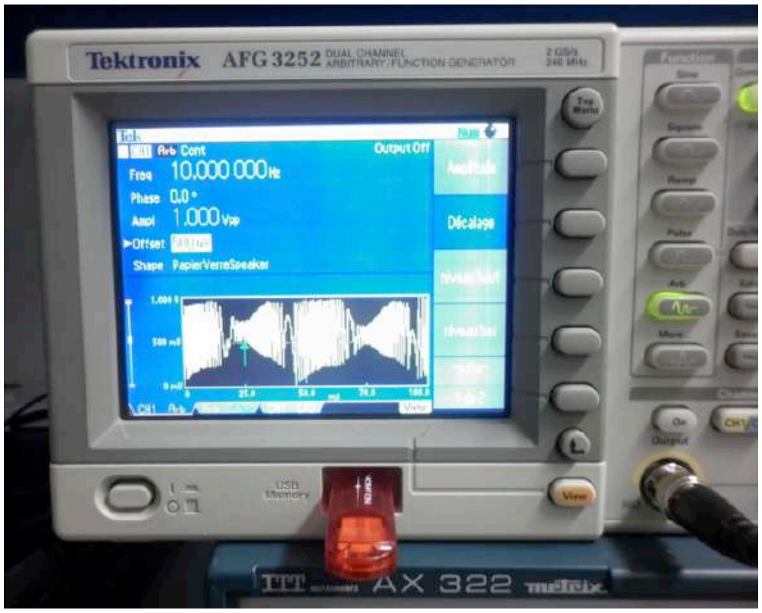


Figure 23. SM input signal

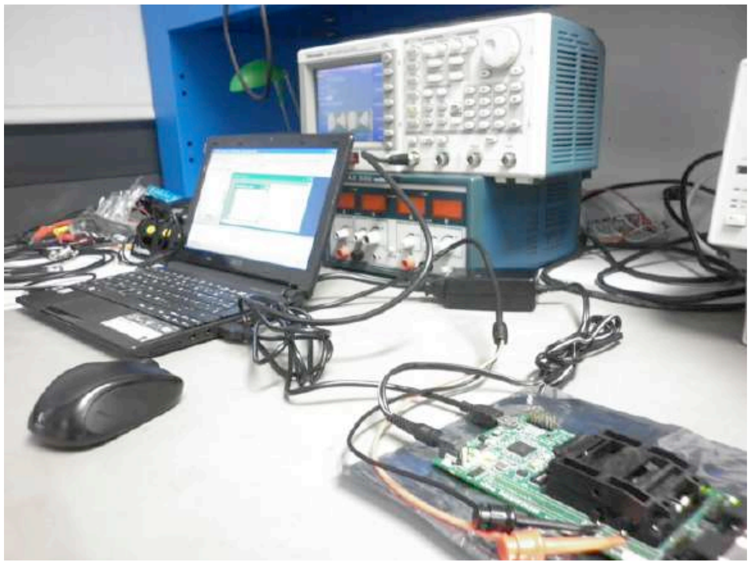


Figure 24. Complete system

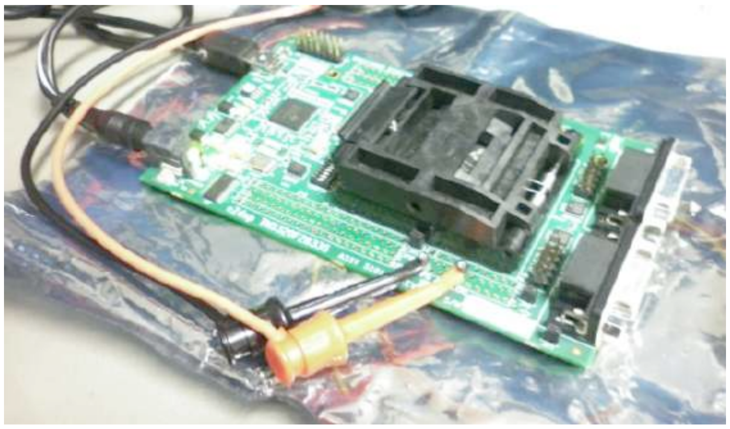


Figure 25. DSP and SM input signal

All parts of the system can see in the **Figure 24**, with input connection from low frequency generator, development board DSP (**Figure 25**) and computer to display the results of interest.

The part most critical of the objective described in the context is the notion of real time, so then a graph shows the feature of distribution of work performed by the DSP. **Figure26**

To have real time notion, signal process technique was studied, which is simple to compute, called triple buffer processing. **[14][17]**
Using the triple buffer process, the first one is just to capture the input when information it's come, while the second buffer process all the operation that buffer one pass him and the last one is to present the result computed from the second buffer. This buffer sequences are working all the time having the real time notion when the systems is running. **[10]**

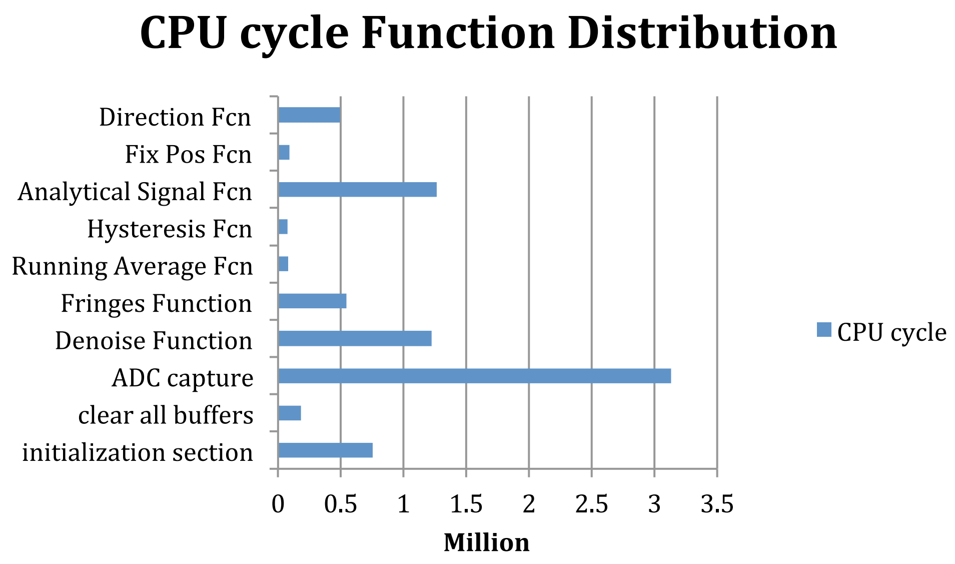


Figure 26. CPU cycle Graph

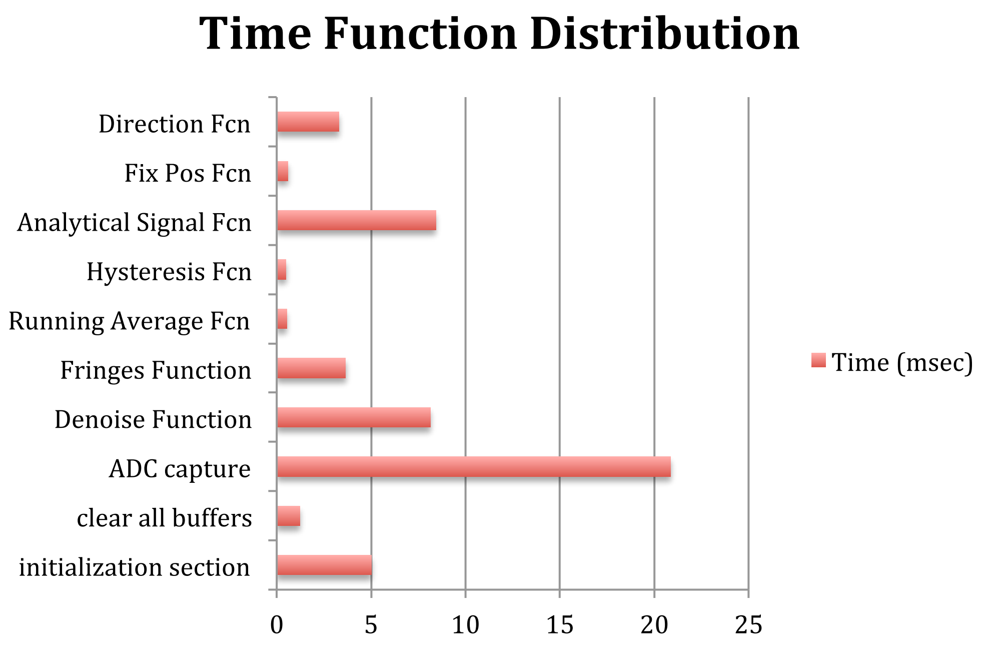


Figure 27. Time distribution diagram

Distributions of the time and clock cycles, in figure 20 and figure 21, respectively. Show that the function responsible for capturing data from the ADC of the DSP, is the largest use for system executions, this is because at all times is receiving data from the low frequency generator, as input to the global system, with a time 20.86 msec. **Figure27**

Another point about this time is because ADC vector in memory to be fill, has a size of 2048 elements declared as floating notation, knowing that DSP device configuration works at 150 Mhz, which mean that each instruction takes 6.66 nsec to be execute, the ADC capture time presented before, in figure 21, can be estimate with this information. **[12]**

The critical point of this work is the real time notion, which depends on each application, to our work the purpose of real time, described in the context, was that all the algorithm calculations were carried out in a way that did not have dead-time between calculation output and input, or minimize this intermediate time. **[13]**

	CPU cycle	**Time (msec)**
Algorithm time	7085612	47.23741
Total time	7839855	52.26570

Table 1. Average time and CPU cycle representation

In practice, good results were obtained (**Table 1**), having computing input data on a continuous notion.

For the last part in this investigation, and like my specialization must be applied in this work, control mirrors laser was proposed to scan a surface target to obtain information, which can be analyzed.

3.5. Scanning surface

As described in the context, LAAS- OSE investigations are aimed at bio-medical applications, therefore, the interest of scanning surfaces in two dimensions, not on a specific object, in this last part of the work. The scan method used in LAAS-OSE, is a system, which has two motor, orthogonal positions one each other, with a mirrors in the end of each axis (**Figure 28**), and the purpose is find a method to scan an target area with optimal points to get the real information, because during the process when target area is scanned, noise and fail information (Doppler effect) is captured.

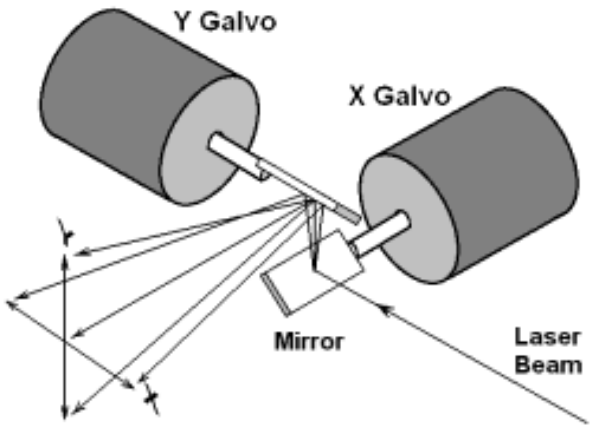
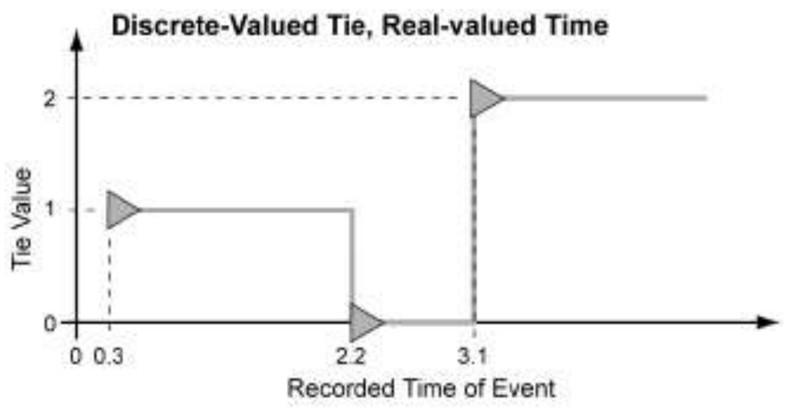


Figure 28. Surface Scaning method

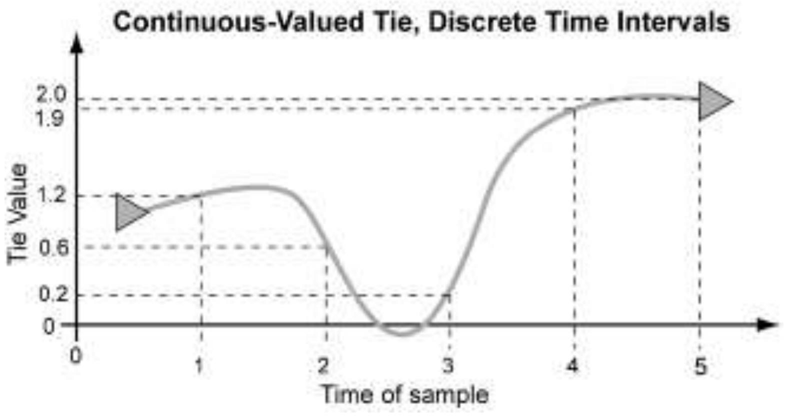
3.5.1. Mirrors Theory

Following with the context's interest, laser scanning is a method in biomedical application, next part, two methods are proposed to capture information, in others words, scan technique will be presented.

The first method proposed to scan the area, is digitally, this means, for a period it captures information from laser, then is turned off to move the laser beam and turn it on to capture new data again, in successive manner, **Figure 29.a**; For the second method, intends to produce, the measurements on a continuous form, having all the time measurement data, **Figure 29.b**



a)



b)

Figure 29. Discrete (a) and continues (b) capture laser form proposed

The advantage more important to scan a target area and capture the really good information with this method proposed, is to reduce measure time in biomedical application, normally it take about 2 hours, which is a slow process for the patients. Because traditional methods scan, has capture problems like instability position to measure the good point, also the time to stability is higher.
First step proposed is to inject two parametrical functions at each motor, to represent XY axis, with Lissajous curve equations, this because is simple to compute and generate on DSP.

$$x(t) = Asin(at + \delta)$$
$$y(t) = Bsin(bt)$$

Where;

A and B represents the amplitude of each axis.
a and b are the frequency of each function.
δ represents the phase.

Incrementing a and/or b different response can be reproducing to scan a surface, like next **Figure 30** show.

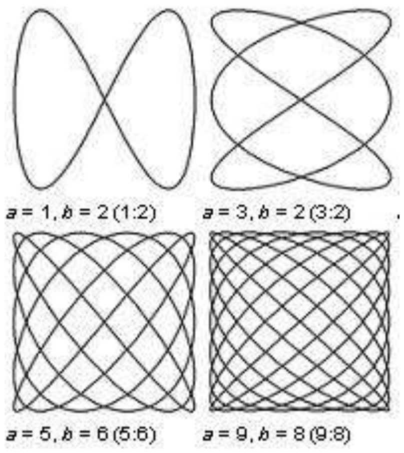


Figure 30. Lissajous curve response, a and b frequency variations.

On time scanned the target area, a possible forms of data representation is shown in the **Figure 31** and **Figure 32**, once taken the data to a known mathematical space, methods to find the maximum or minimum points would be implemented like Powell method, for example (**Figure 33**), or the other way more sophisticate, estimate a function which minimize or maximize, it's depend of the sense, an optimal function, applying optimal control theory, with this method could find the point of interest, real information.

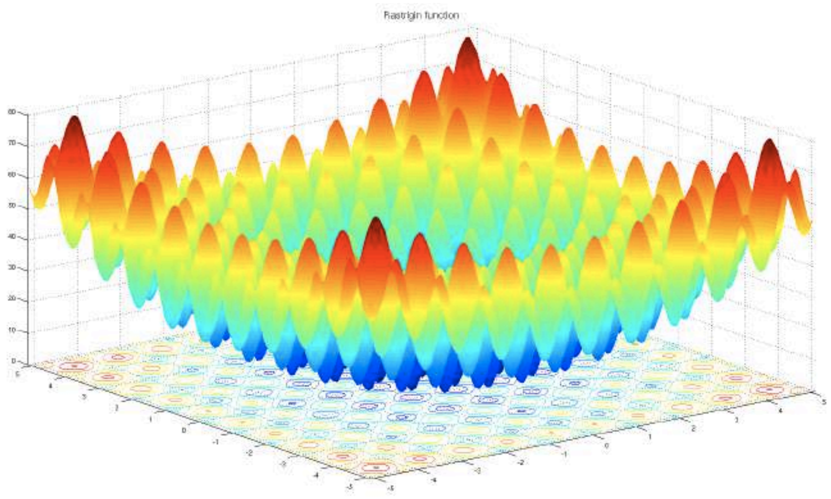


Figure 31. 3D representation

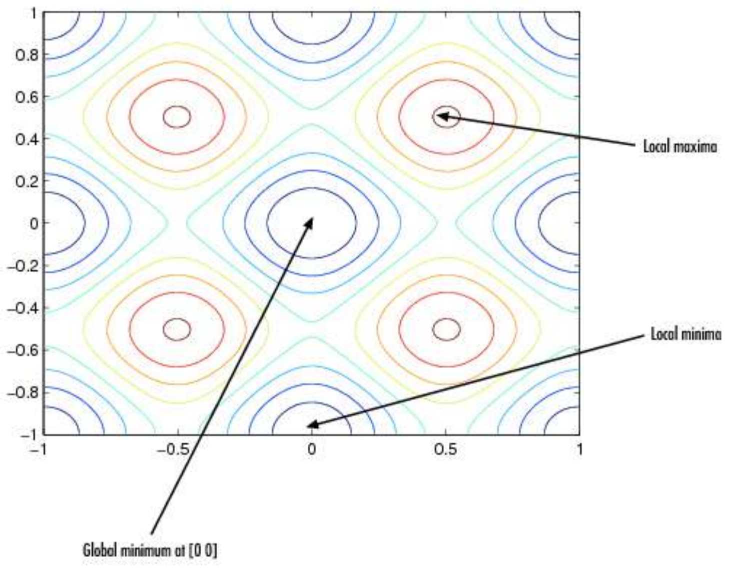


Figure 32. 2D representation

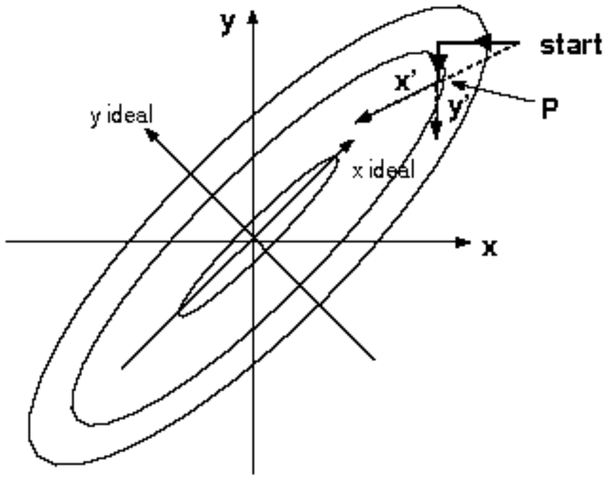


Figure 33. Powell method

Method presented before, guaranty obtain the good information one time scanned the target area, because Lissajous curve equations is simple to compute and just with 2 variation of parameters can have different response, making flexible to cover the target area. These curve equation proposed must be injected to the motors to scan decreasing measure time around few minutes, one time do it that and making use of optimal control theory to maximization/minimization the good points is proposed, another method could have the same results is the implementation of a method to find the good points, called Powell method, one time scanned the target area, like was explained at the begging.

In recent years at LAAS-OSE, they study particles solid and fluids for the medical environment, using the self-mixing, having important results and also competing with laboratories recognized in the world. The transformation, implementation and validation of the code in this work developed during these 6 months, it is unique compared with others research center around the world.
Today embedded systems sensors are developing with sophisticated techniques like, MEMS, the tendency at LAAS-OSE going for that investigation field.

4. Conclusion

In this work met the importance of the SM phenomenon, which allows to make measurements, as the distance of an object, without necessity of many elements, and others parameters, also met different concepts and tools to converter code from MATLAB to C, and notion of real-time implemented on an embedded system board, like validation of all system.

For Code transformation two application of MATLAB were used (Coder and Simulink Coder), unfortunately in the process to conversion was discovered problems that made it impossible to carry out the objective described from the beginning, with MATLAB Coder code conversion syntax wasn't incompatible with Code Composer Studio software use, and the other side, Simulink Coder had problem at the beginning of investigation, because configuration modules to communicate to the device (DSP) were not available for our MATLAB version used during this investigation.

Due to this problem, before mentioned, was decided to convert MATLAB code to C, manually, spending a lot of time to understand the algorithm, each digital signal function, each mathematical operation, having favorable results in the implementation (DSP), execution time and response of calculations (FFT, IFFT, Angle, etc) was compered with MATLAB original results, obtained similar results one each other.

The last part of this work, was the control mirrors investigation to scan a target area, using only two motors, two mirrors and a laser source, the principal purpose was develop a possible way to create a parametrical functions to inject to the motors (XY axis), and with this method can controller the area to be scanned, after to scan the area, optimal method to find the really goods point of information are proposed, to minimize measurement time .

5. Project Management

In this final part, distribution of time and hours are presented showing a Gantt chart of the internship project (**Figure 36**) at LAAS-OSE. Also, 12 important activities were developed during 27 period weeks including vacations period, **Figure34** and **Figure35**.

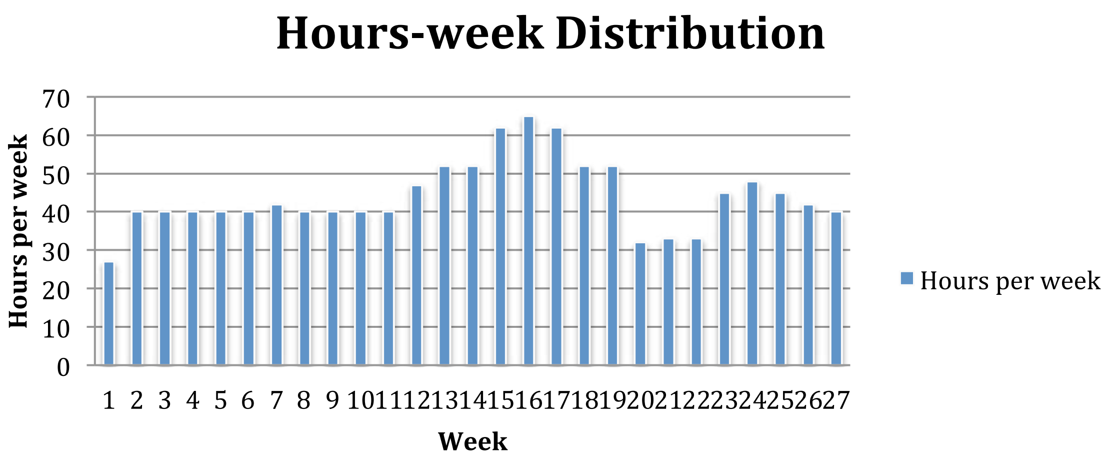


Figure 34. Hours vs. Week Graph

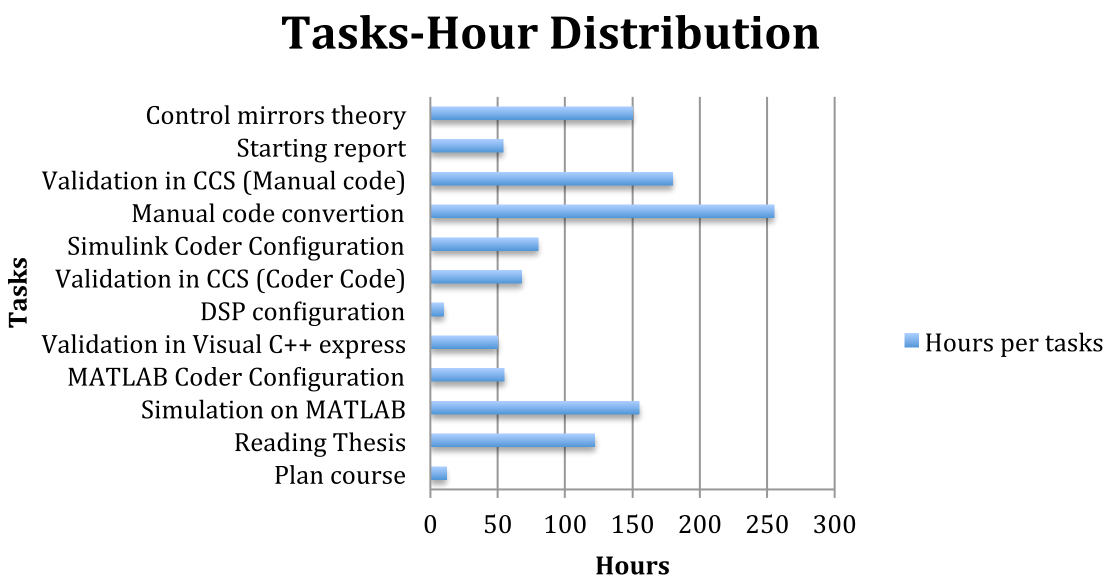
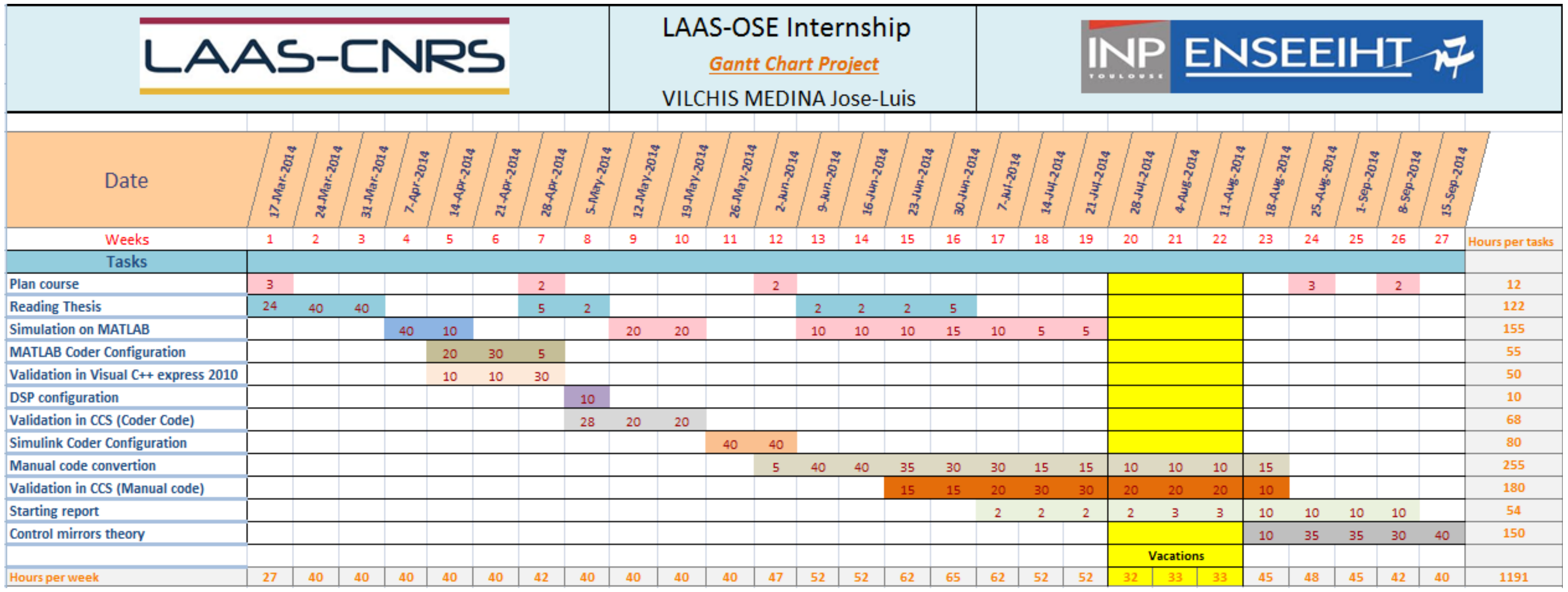


Figure 35. Tasks vs. Hour Graph



LAAS-OSE Internship

Gantt Chart Project

VILCHIS MEDINA Jose-Luis

Date	17-Mar-2014	24-Mar-2014	31-Mar-2014	7-Apr-2014	14-Apr-2014	21-Apr-2014	28-Apr-2014	5-May-2014	12-May-2014	19-May-2014	26-May-2014	2-Jun-2014	9-Jun-2014	16-Jun-2014	23-Jun-2014	30-Jun-2014	7-Jul-2014	14-Jul-2014	21-Jul-2014	28-Jul-2014	4-Aug-2014	11-Aug-2014	18-Aug-2014	25-Aug-2014	1-Sep-2014	8-Sep-2014	15-Sep-2014	
Weeks	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	Hours per tasks
Tasks																												
Plan course	3						2					2												3		2		12
Reading Thesis	24	40	40				5	2					2	2	2	5												122
Simulation on MATLAB				40	10				20	20			10	10	10	15	10	5	5									155
MATLAB Coder Configuration					20	30	5																					55
Validation in Visual C++ express 2010					10	10	30																					50
DSP configuration								10																				10
Validation in CCS (Coder Code)								28	20	20																		68
Simulink Coder Configuration											40	40																80
Manual code convertion												5	40	40	35	30	30	15	15	10	10	10	15					255
Validation in CCS (Manual code)															15	15	20	30	30	20	20	20	10					180
Starting report																	2	2	2	2	3	3	10	10	10	10		54
Control mirrors theory																							10	35	35	30	40	150
																					Vacations							
Hours per week	27	40	40	40	40	40	42	40	40	40	40	47	52	52	62	65	62	52	52	32	33	33	45	48	45	42	40	1191

Figure 36. Gantt Chart project

Bibliography

[1] P. Hariharan, Basics of Interferometry. Academic Press, 1991, p. 45. (p. 1.)

[2] Antonio LUNA ARRIAGA, "Characterization & Development of Self-mixing Interferometry Algorithms for Embedded Displacement Sensors Design", **Ph.D. Thesis INPT,** 2014.

[3] D. Clunie and H. Rock, "The laser feedback interferometer," Journal of ScientificInstruments, vol. 41, pp. 489–492, 1964. (p. 1.)

[4] I.P. KAMINOV, An introduction to electro-optic devices, Academic Press, New York, (1974)

[5] A. YARIV, Introduction to optical electronics, Holt, Rinehart and Winston, New York, (1976)

[6] T. Bosch, C. Bès, L. Scalise, and G. Plantier, "Optical feedback interferometry," in Encyclopedia of Sensors. American Scientific Publishers, 2006, vol. 7, pp. 107 – 126.(pp. 2 and 17.)

[7] T. Bosch, N. Servagent, and R. Chellali, "A scanning range finder using the self-mixing effect inside a laser diode for 3-D vision," in Proc. IEEE-I2MTC, 1996, pp. 226–231. (p. 2.)

[8] S. Donati, "Developing self-mixing interferometry for instrumentation and measurements," Laser Photonics, vol. 6, no. 3, pp. 393 – 417, 2012. (pp. vi, 23, 30 and 33.)

[9] G. Plantier, C. Bes, and T. Bosch, "Behavioral model of a self-mixing laser diode sensor," IEEE J. Quantum Electron., vol. 41, no. 9, pp. 1157–1167, sep. 2005. (pp. 18 and 20.)

[10] A. Magnani, A. Pesatori, and M. Norgia, "Self-mixing vibrometer with real-time digital signal elaboration," Applied Optics, vol. 51, no. 21, pp. 5318 – 5325, 2012. (pp. 25 and 33.)

[11] L. Campagnolo, M. Nikolic, J. Perchoux, L. Yah Leng, K. Bertling, K. Loubiere, L. Prat, A. D. Rakic, and T. Bosch, "Flow profile measurement in micro-channel us ing the optical feedback interferometry sensing technique," Microfluidics and Nanofluidics, vol. 14, pp.1613–4982, Jul. 2012. (p. 30.)

[12] R. Oshana, DSP software development techniques for embedded and real-time sys tems. Newnes, 2006. (p. 43.)

[13] J. G. Proakis and D. G. Manolakis, Digital Signal Processing - Principles, Algorithms and Applications. Pearson Prentice Hall, 2007. (p. 54.)

[14] A. Oppenheim and R. Schafer, Discrete-time signal processing. Prentice-Hall, 1989. (p. 55.)

[15] B. Boashash, "Estimating and interpreting the instantaneous frequency of a signal-part 1: Fundamentals," Proceedings - IEEE, vol. 80, pp. 520–538, 1992. (p. 99.)

[16] R. N. Bracewell, The Fourier transform and its applications. McGraw-Hill, 1978.(p. 100.)

[17] D. Markovic and R. W. Brodersen, DSP Architecture Design Essentials, A. P. Chandrakasan, Ed. Springer, 2012. (p. 148.)