UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

MAESTRÍA EN INGENIERÍA ELÉCTRICA EN INSTRUMENTACIÓN

CONSTRUCCION DE UN SISTEMA TIPO FROG PORTATIL PARA LA CARACTERIZACION DE PULSOS OPTICOS DE FEMTOSEGUNDOS

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN INGENIERIA

INGENIERIA ELÉCTRICA - INSTRUMENTACION

P R E S E N T A :

GABRIEL KAPELLMANN ZAFRA

TUTOR:
JESUS GARDUÑO MEJIA
UNAM - CCADET

MÉXICO, D.F. NOVIEMBRE- 2012

JURADO ASIGNADO:

P r e s i d e n t e:	Dr. Roberto Ortega Martínez
S e c r e t a r i o:	Dr. Alejandro Farah Simón
V o c a l:	Dr. Jesús Garduño Mejía
1 er. S u p l e n t e:	Dr. Antonio A. Rodríguez Rosales
2 do. S u p l e n t e:	M.I. Rodrigo Regalado García

Lugar o lugares donde se realizó la tesis: CCADET, UNAM

TUTOR DE TESIS:

_____

Dr. Jesús Garduño Mejía

Dedicatoria

A mi familia por su apoyo incondicional durante toda mi vida.

Agradecimientos

A la Universidad Nacional Autónoma de México (UNAM), al Posgrado de Ingeniería Electrónica y el laboratorio de Óptica No Lineal del Centro de Ciencias Aplicadas y Desarrollo Tecnológico de la UNAM (CCADET) por darme la oportunidad de continuar con mi formación académica.

Se agradece el financiamiento en herramientas, equipo y al apoyo en la beca durante el ultimo periodo en el desarrollo del trabajo y escritura de la tesis por parte de la DGAPA-UNAM. PAPIIT proyectos: IN113809, IB101212, IACOD-I1100611, IN104112, BID-UNAM 98-B3-C-DGP-L0034-1077, CONACyT proyecto CB 131746.

A CONACyT por el apoyo económico de la beca de maestría.

Al departamento de electrónica por su apoyo en la fabricación de los múltiples prototipos de los circuitos impresos utilizados en el instrumento.

Al M. en C. Carlos J. Román Moreno por su apoyo en el mantenimiento y asesoría en el uso diario del Laser del laboratorio.

Al Ing. Jorge Alberto Arredondo Álvarez por su apoyo en la utilización de las maquinas de control numérico para la fabricación de los componentes del instrumento.

El apoyo del Tecnológico de Monterrey Campus Ciudad de México por las facilidades prestadas para el uso de su taller de manufactura.

Finalmente doy las gracias a mi comité tutoral, por su valioso tiempo dedicado en la lectura, revisión y comentarios de enriquecen esta tesis.

Abstract

El objetivo de este proyecto fue el de minimizar un instrumento SHG-FROG de tal manera que fuse amigable para el usuario. La construcción de dicho instrumento fue realizada por etapas, la electrónica, la mecánica y el software. Todas las etapas fueron previamente diseñadas y modeladas en poderosas paqueterías de software para después producirlas con los menos errores posibles. El objetivo principal fue de diseñar un instrumento capaz de simplificar la medición y caracterización de pulsos de laser ultracortos.

The objective of this project was to minimize a SHG-FROG instrument in such a way that it could be user friendly. The construction of an instrument like this was performed in different stages, the software, the electronics and the mechanics. All the stages were designed and modeled with powerful software's before its production, this to prevent possible design mistakes. The principal objective of this work is to design an instrument able to simplify the measurement and characterization of ultrashort laser pulses in the research laboratory.

Index

Introduction

Many scientific theses tend to focus on very specific subjects, thus becoming difficult and boring to read. The subject is too specific and may become difficult and even boring to read, as Rick Trebino argued[1]:

> *"Scientists have a well-deserved reputation for being dull people who write dull books for other dull scientists."*

Unfortunately, I know this work may look like one of those kind of works, but hopefully I will be able to "tell the story" in a friendly way. The main objective focuses over the direct interaction that physics has with the engineering in the design of a new instrument to measure ultra-short laser pulses.

Ultra-short pulses are relatively new light phenomena that human is starting to study and understand. Normally, to work with something physical a tool is needed, but ultra-short pulses are too short and this gets to be the big problem as there isn't anything shorter than them. The electronics speed is too slow in front of these laser pulses the same way that humans are slow to the electronics. The opto-electronic instrument described in this thesis is able to measure these pulses and show us (the slow ones) the characteristics of these phenomena with a "simple" technique that, curiously, has the acronym of a small cute amphibian: The FROG technique.

The Frequency-Resolved Optical-Gating (FROG) is a powerful combination between a complex algorithm and a simple array of optical and electronic components. In this work, the FROG technique is not being re-defined in any way, but the creation of a portable and user friendly instrument. Some years ago, a beautiful idea of simplicity for "the end user" has ruled almost all new products, the "Plug-&-Play". This idea mainly arrived with the invention of the USB port and has been evolving since those days. Our instrument is based on this idea of giving the researcher (that in this case is the end user) the possibility to focus on its research and stop losing time building his own instruments.

Step by step the design process will be shown and explained starting with the mechanics, followed by the electronics and finishing with the software. By the end of the thesis, some results of actual measurements are found, in addition, a comparison between different situations where some traces weren't good enough to supply reliable information with others that gave the expected results are shown. In the multiple appendixes it is possible to find all the information required to reproduce the instrument.

I hope that you will find this work interesting and easy going, just like any other manual for a new product.

[1] Fragment taken from Rick Trebino, "Frequency-Resolved Optical-Gating: The measurement of ultrashort laser pulses", KAP (Kluwer Academic Publishers), USA, page 1.

1. Ultra-Short Pulses

Ultra-short laser pulses are known to be the shortest events that man has been able to generate. These pulses are used in many applications where great precision and resolution is needed, some examples:

- *Biomedical applications*
- *Instantaneous chemical reactions*
- *Metal or semiconductors reactions*

An ultra-short pulse is composed of a large number of wavelengths, their main quality is that as it's really short in time, the spectrum has a greater range of colors (the bandwidth) and the instant power gets larger.

It is possible to define the electric field of an ultra-short laser pulse in time as[2]:

$$\mathrm{x}(t) = \tfrac{1}{2}\sqrt{I(t)}e^{i\left[\omega t - \phi(t)\right]} + c.c. \qquad \text{<1.1>}$$

Where $I(t)$ is the intensity of the pulse, $\phi(t)$ is the phase in time and "c.c." is the abbreviation of "complex-conjugate". It's possible to simplify this complex equation removing $e^{-i\mathrm{F}(t)}$ which is the frequency carrier, as it doesn't affect the purpose because it moves really fast in time and the measurement method doesn't need to be as fast as it. With this change, only the intensity and the phase of the pulse (in time) are required and for mathematical convenience also the $\frac{1}{2}$ and the c.c. may be ignored, as the imaginary part is equal but opposite in sign.

$$\mathrm{E}(t) = \sqrt{I(t)}e^{-i\phi(t)} \qquad \text{<1.2>}$$

In these equation $E(t)$ is the same as $\mathrm{x}(t)$ but only as the complex amplitude. The modulus of $E(t)$ is raised to the second power to obtain the intensity of the complete pulse as shown in equation <1.3>:

$$I(t) = \left|E(t)\right|^2 \qquad \text{<1.3>}$$

For example, an ultra-short pulse with intensity defined by $I(t) = 5e^{-\left(t^2/.05\right)}$, and with frequency $e^{-i\omega t}$ where ω is defined by $\lambda = 800\left[nm\right]$ and with temporal phase $e^{-i\phi(t)}$ defined by $\phi(t) = 0$ throws the equation:

[2] Fragment taken from Rick Trebino, "Frequency-Resolved Optical-Gating: The measurement of ultrashort laser pulses", KAP (Kluwer Academic Publishers), USA, page 21.

$$\mathrm{E}(t) = \sqrt{5e^{-\left(t^2/.05\right)}}\, e^{i\left(\frac{2\pi c}{750 x 10^{-9}}\right)t} e^{-i0} \qquad \text{<1.4>}$$

As ω is the frequency carrier and $I(t)$ and $\phi(t)$ are the time-dependent intensity and phase of the pulse the mathematical expression is simplified. The pulse equation then looks like equation <1.5>, in Figure 1 the electric field, amplitude and intensity are plotted.

$$\mathrm{E}(t) = \sqrt{5e^{-\left(t^2/.05\right)}}\, e^{-i0} \qquad \text{<1.5>}$$

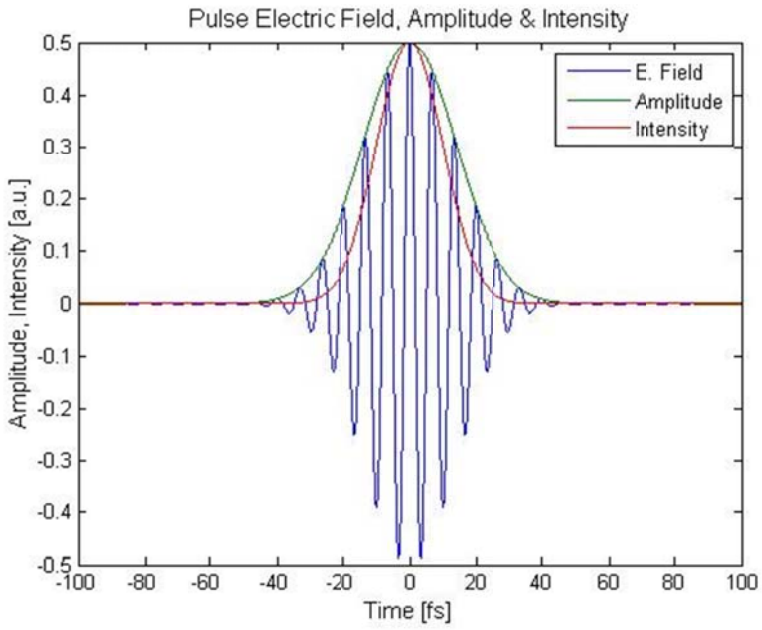


Figure 1: Electric field, Amplitude and Intensity of an ultrashort pulse.

The temporal phase is calculated using equation <1.6>:

$$\phi(t) = -\arctan\left\{\frac{\mathrm{Im}\left[E(t)\right]}{\mathrm{Re}\left[E(t)\right]}\right\} \qquad \text{<1.6>}$$

Were, as mentioned before, the phase of the pulse in Figure 1 is constant, $\phi(t) = 0$.

All this definitions have been made over a pulse defined in time, but to fully understand it, the frequency domain is needed. The *Fourier Transform* (FT) is used to transport the pulse to its frequency domain:

$$\tilde{\mathrm{E}}(\omega) = \int_{-\infty}^{\infty} E(t) e^{-i\omega t} dt \qquad \text{<1.7>}$$

Where $\tilde{E}(\omega)$ is the *Fourier Transform* of the pulse, this leads to the spectrum of the pulse:

$$\tilde{E}(\omega) = \sqrt{S(\omega)}e^{-i\varphi(\omega)} \qquad \text{<1.8>}$$

Equation <1.8> describes the pulse spectrum centered at the origin, it still needs to be shifted to ω_0 of the actual pulse. The only modification that equation <1.8> needs is the subtraction of ω_0:

$$\tilde{E}(\omega - \omega_0) = \sqrt{S(\omega - \omega_0)}e^{-i\varphi(\omega - \omega_0)} \qquad \text{<1.9>}$$

Returning to the example, the FT is applied to the equation <1.5> to find the spectrum.

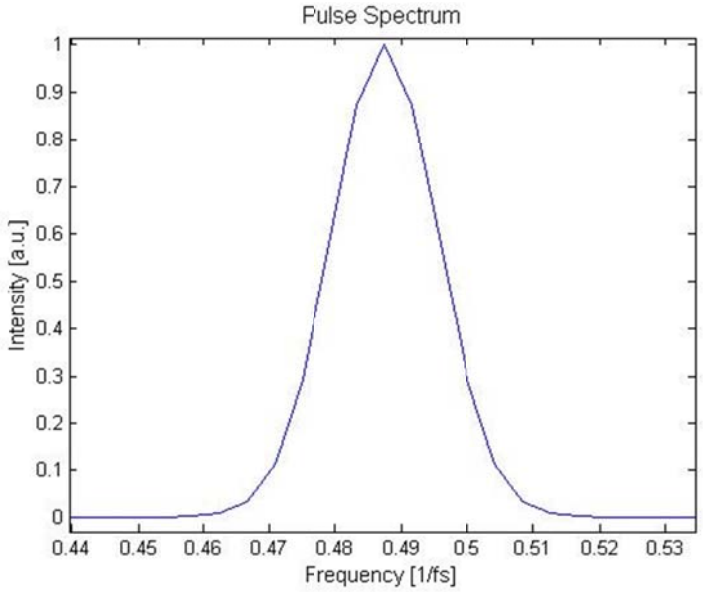


Figure 2: Spectrum of the ultrashort pulse of the example.

The calculation of the spectrum's phase is performed in the same way as in time but with the function of the electric field in the frequency domain:

$$\varphi(t) = -\arctan\left\{\frac{\mathrm{Im}\left[\tilde{E}(t)\right]}{\mathrm{Re}\left[\tilde{E}(t)\right]}\right\} \qquad \text{<1.10>}$$

In this case, there isn't any spectrums phase as shown in Figure 3.

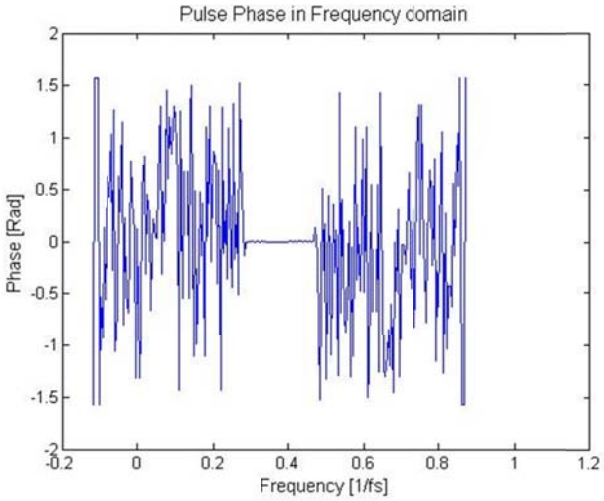


Figure 3: Zero phase of the spectrum.

The spectrums phase plotted in Figure 3 shows a zero phase in the non-zero spectrum zone and a lot of noise where the spectrum is zero. Let's add a spectral phase to the pulse, as seen in equation <1.9> this phase is given by $e^{-i\varphi(\omega)}$ where:

$$\varphi(\omega)=\varphi_0+(\omega-\omega_0)\varphi_1+(\omega-\omega_0)^2\frac{\varphi_2}{2}+(\omega-\omega_0)^3\frac{\varphi_3}{6}\ldots \qquad <1.11>$$

This kind of phase could be considered if a pulse is transmitted through some lens or medium, for this example the new phase parameters will be given by:

$$\varphi_0=0;\varphi_1=447;\varphi_2=63;\varphi_3=60 \qquad <1.12>$$

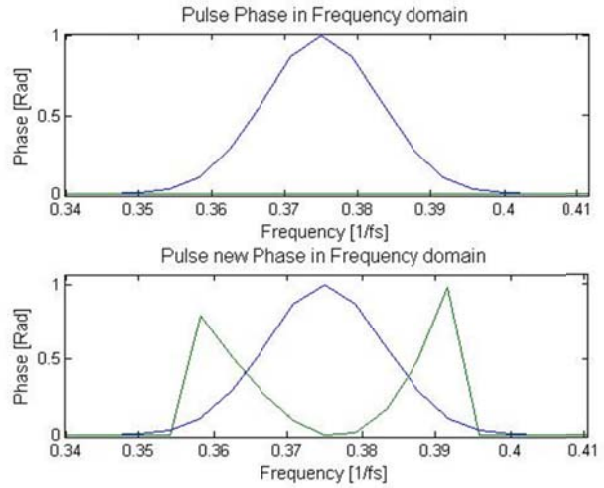


Figure 4: Comparison between the zero phase (up) vs. the new phase (down) and the spectrum.

The new expression for the pulse spectrum is:

$$\tilde{E}(\omega) = \sqrt{S(\omega)}e^{i\left(\varphi_0 + (\omega-\omega_0)\varphi_1 + (\omega-\omega_0)\frac{\varphi_2}{2} + (\omega-\omega_0)\frac{\varphi_3}{3}\right)} \quad \text{<1.13>}$$

With the new phase the spectrum will not suffer any changes since any new frequency components have been added, but the pulse in time will be affected. To see the effects of the new spectral phase over the pulse in time the ”Inverse FT” is applied over the spectrum, a comparison of the initial pulse intensity and amplitude vs. the new ones is shown in Figure 5.

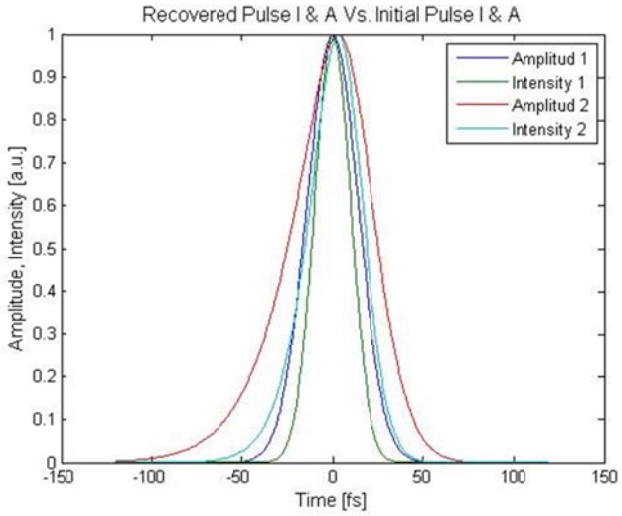


Figure 5: Pulse comparison between the initial state and the phase affected state.

2. Frequency-Resolved Optical-Gating (FROG) Technic

- **Time and frequency domain**

> *"In order to measure an event in time, you must use a shorter one. But then, to measure the shorter event, you must use even a shorter one. And so on. So, now, how do you measure the shortest event ever created?"*[3]

To measure an ultrashort pulse in its time domain an autocorrelation is required, in the frequency domain a spectrometer does the work, but to recover the complete information of an ultrashort pulse a hybrid technique is required that works in the *time-frequency* domain. This hybrid domain isn't as strange as it may sound, a musical score would be a perfect example but with different sounds (frequencies) in time (seconds or some milliseconds).

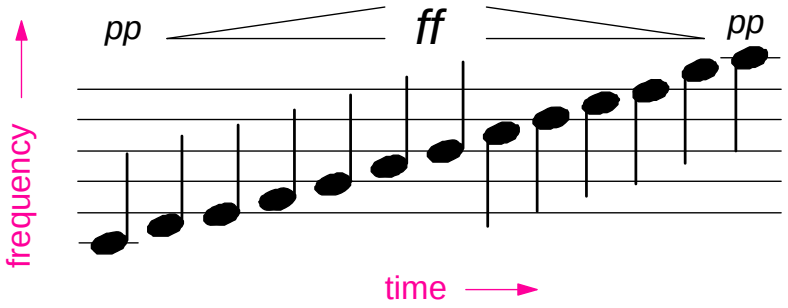


Figure 6: Time and frequency plotted in a music score.

This is a kind of a spectrogram, it's a set of "gated chunks" of $E(t)$ as the position of the gate varies in time:

$$\sum_g^E(\omega,\tau) \equiv \left| \int_{-\infty}^{\infty} E(t)\, g(t-\tau)\, e^{-i\omega t} dt \right|^2 \qquad \text{<2.1>}$$

[3] Fragment taken from Rick Trebino, "Frequency-Resolved Optical-Gating: The measurement of ultrashort laser pulses", KAP (Kluwer Academic Publishers), USA, page 1.

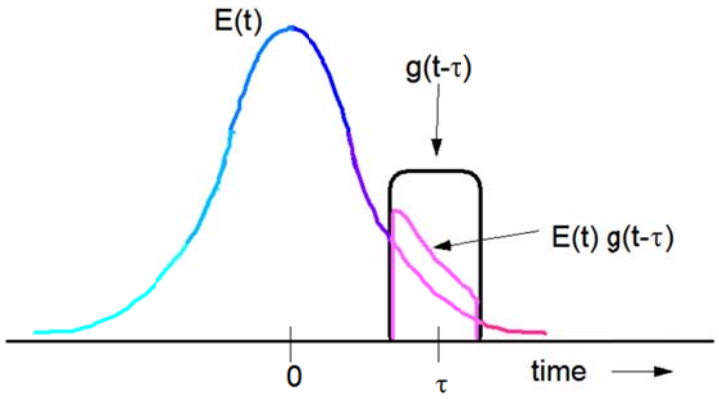


Figure 7: Gate vs. Signal relation result.

Basically, the function of FROG is to measure a spectrogram of an ultrashort pulse. With this information it is possible to completely determine the structure of $E(t)$ except for some ambiguities, such as the absolute phase.

- **SHG-FROG Basics**

Unfortunately, a spectrogram needs a shorter gate (in comparison to the original pulse) to measure the pulse. But if the ultrashort pulse is one of the smallest events, the gate is a kind of event that "doesn't exist". The FROG technic uses the autocorrelation so that the same ultrashort pulse works as gate over himself, and then resolve the gated piece of the pulse. By gating the ultrashort pulse with itself the spectrogram is going to get the information of an unknown pulse with the help of… an unknown gate. Because of this, it's not possible to use typical spectrogram inversion algorithms.

The FROG is an autocorrelation measurement, but instead of measuring energy vs. delay (which is a common autocorrelation) it focuses on the signal spectrum vs. delay. For example, the SHG-FROG equation can be derived from equation <2.1> where the gate is changed to the same pulse but delayed:

$$I_{FROG}^{SHG}(\omega,\tau)=\left|\int_{-\infty}^{\infty}E(t)E(t-\tau)e^{-i\omega t}dt\right|^2 \qquad <2.2>$$

So the delayed pulse is working as the gate of the same pulse with no delay.

- **SHG-FROG Properties, disadvantages and advantages**

The SHG-FROG is practically a standard SHG-based autocorrelator. The main advantage of the SHG-FROG is sensitivity as it involves only a second order nonlinearity. Because of this, this type of

FROG is commonly used to measure unamplified pulses that may be as weak as 1 pJ[4], slightly less sensitive than an autocorrelator.

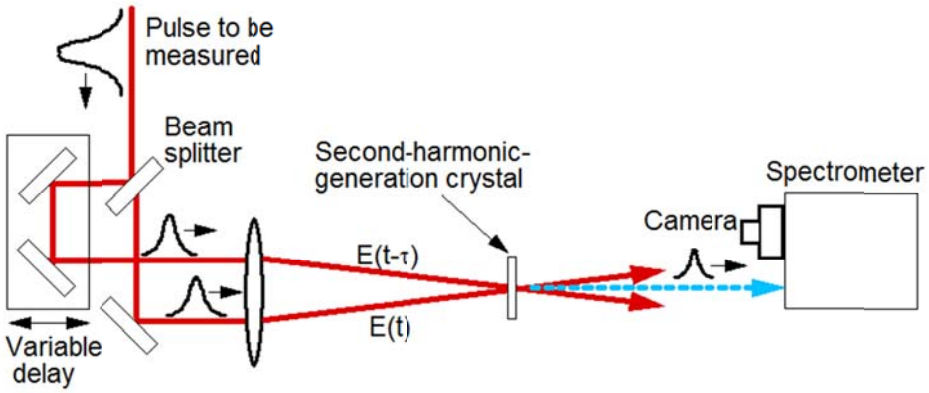


Figure 8: Simple SHG-FROG architecture.

There are many nice features of FROG, it's a very accurate technic for measuring pulse intensity and phase in time and frequency. It can measure pulses from some femto-seconds to a couple of pico-seconds and can work from the mid IR to the UV, and systematic errors may often be removed by preprocessing the measured trace and by taking in account three mayor constraints:

- *A maximum in intensity at zero delay.*
- *Zero intensity for large delays.*
- *Symmetry with respect to delay.*

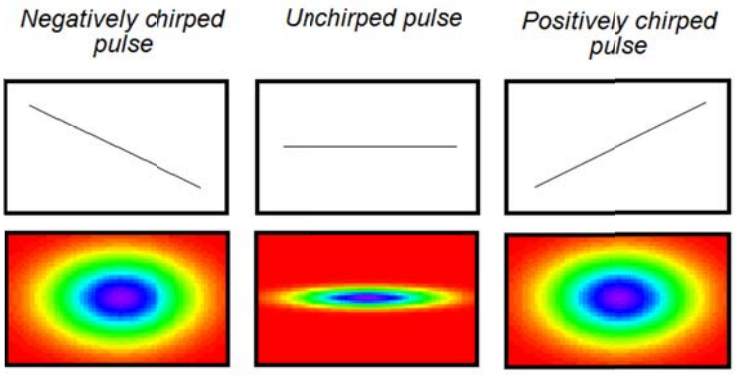


Figure 9: SHG-FROG example traces.

Unfortunately, the direction of time is ambiguous. The computation of marginals in delay and frequency help in a deterministic way to reveal the existence of systematic errors by comparing them independently with the spectrum or the autocorrelation. Also this technic has a good S/N ratio because the signal beam has a different color (half of the wavelength) so the scattered light is easily filtered.

[4] Fragment taken from Rick Trebino, "Frequency-Resolved Optical-Gating: The measurement of ultrashort laser pulses", KAP (Kluwer Academic Publishers), USA, page 127.

- **What SHG-FROG doesn't measure**

Through the entire chapter it has been argued that FROG can recover phase, intensity in time and frequency, but there are some "trivial ambiguities" that it is not able to measure. As the pulse uses itself to do the gaiting there is no absolute time reference for measuring the pulse arrival time. It removes the direction of time, so it is not possible to determine whether the recovered pulse is inverted or not.

3. Mechanical Design

The mechanics of the SHG-FROG instrument were first designed in CAD software (Pro-Engineer Wildfire 5.0), thus before any part was built or manufactured, the simulation of the design and construction of all the components would provide preliminary results and prevent a big amount of errors like erroneous dimensions or wrong positioning of mechanical components, defects and/or mistaken locations on general geometry.

- **Module definition**

The mechanical design was divided in four stages:

- *The interferometer*: A basic Michelson interferometer in a non collinear arrangement.
- *The SHG crystal*: A simple module where the pulse is focused over the SHG crystal and then collimated.
- The spectrometer: Composed by a diffraction grating, a lens and a webcam CCD.
- The main base: One same piece where all components are placed.

This division of the system was defined to make the adjustment, alignment and manufacturing processes as simple as possible. The manufacturing of most of these pieces was made in a HAAS VF-2 CNC machine with three axis of liberty. The software that was used to translate from CAD/CAM 3D geometries to G code (In .NC format) for the CNC machine was Mastercam. All schematics and CNC G codes can be found at annex C.

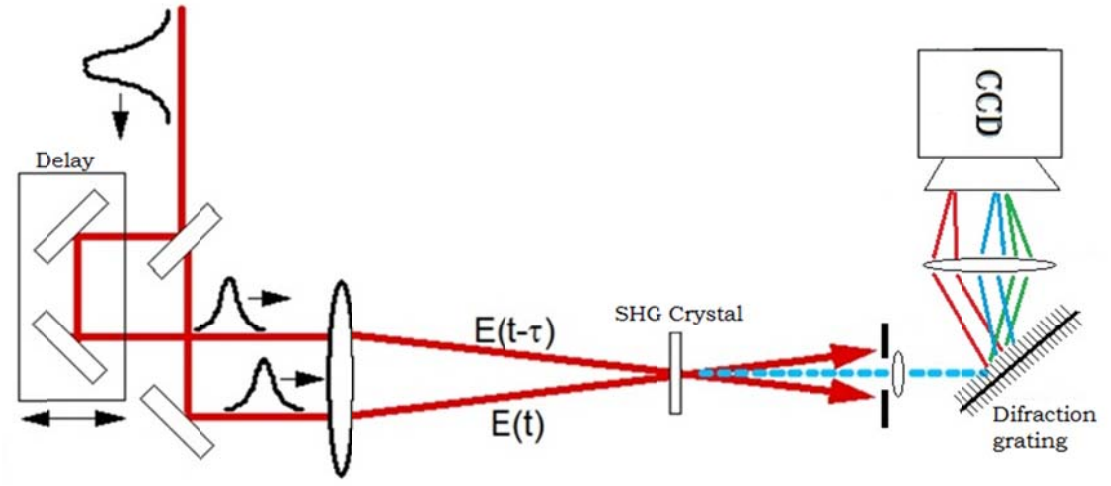
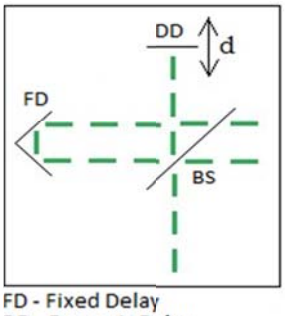


Figure 10: The implemented SHG-FROG architecture.

- **The Michelson Interferometer Delay Line**

This is the first section of the first stage, like any other Michelson interferometer the ultra-short pulse is divided in two replicas with a 50/50 ratio. One of these replicas is delayed in time while the other is spatially separated in a non collinear array.



FD - Fixed Delay
DD - Dynamic Delay
BS - Beam Splitter
Collinear

This section is mainly built with three mayor components, the "interferometer", the dynamic delay device and the fixed delay. The fixed delay is supported by a translation stage (Comar) controlled by a micrometer with $2\mu m$ resolution, attached to this a base for the two mirror mounts (Thorlabs KMSS-1) was specially designed. Each mount is at 90° of inclination in respect to the other, and in 45° in respect to the horizontal plane of the base:

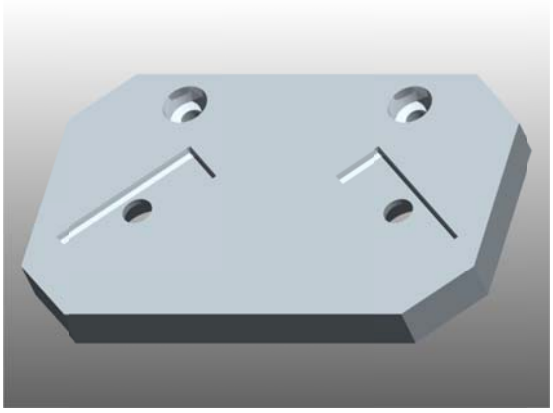


Figure 12: Fixed delay mirror mounts base.

With the mirror mount screws it is possible to align the pulse beam in the vertical and horizontal directions so that both replicas are placed in the same horizontal plane. The delay distance can be adjusted with the translation stage so that at the middle point of the dynamic delay both distances are matched. The complete fixed delay module looks like:

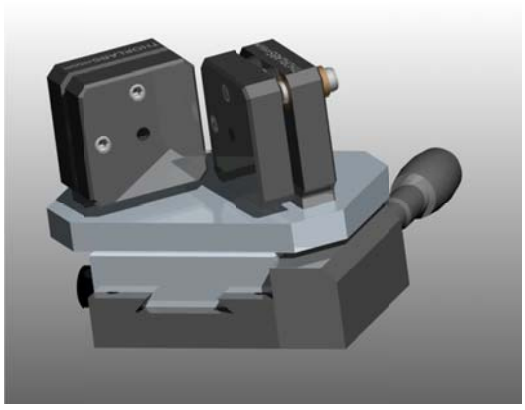
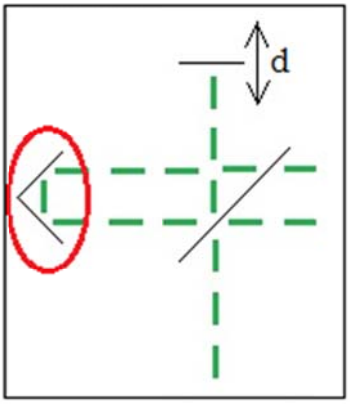



Figure 13: The fixed delay 3D representation and its place in the interferometer.

The second section of the first stage is the dynamic delay; the required range of the distance d depends on the time duration of the ultra-short pulse[5] to be measured. To generate this delay a *Webcams Focus Device* (WFD) was chosen because of its small size and the significant low power it requires to work. The detailed explanation of the control of this device will be explained on the next chapter.

Figure 14: The WFD.

The WFD then is attached to a special support that at the same time is attached to another Thorlabs KMSS-1 support. With the KMSS-1 screws the WFD has the possibility to be adjusted in the horizontal and vertical directions.

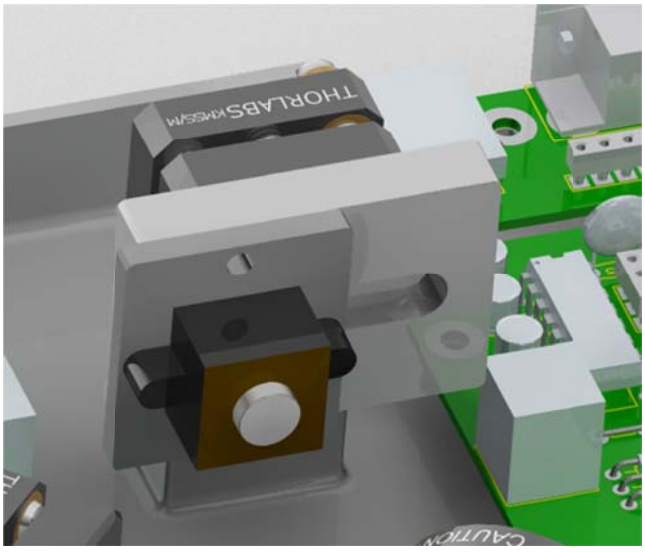
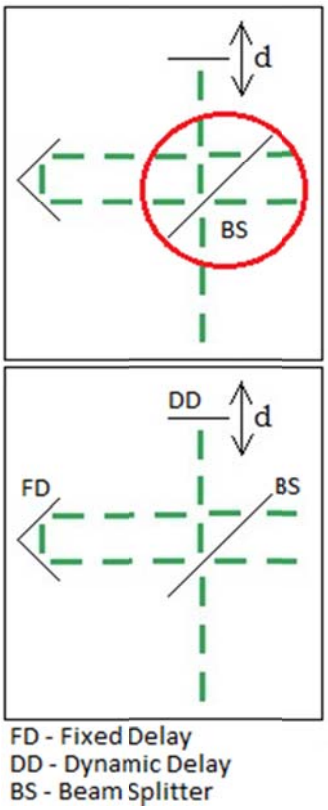


Figure 15: The 3D representation of the WFD.

[5] For further explanation of the dynamic-delay control refer to chapter 4.

The complete device is attached to the main base of the FROG instrument, the main "fixed" distance is controlled by the translation stage and the dynamic delay by the WFD. The WFD is synchronized with the webcam capturing frequency so that for each shift of the distance *d* only an image (or a defined number of images) is taken by the webcam. The interaction webcam-WFD will be discussed further in the software chapter.

Finally, the third section of the first stage is the Beam Splitter (BS) (Thorlabs BP145B1) which is in charge of dividing the pulse in two equal pulses of 50% of the original power. The BS needs to be located at the same distance between the fixed delay and the dynamic delay so that each replica travels the same distance of the optical path. The support of the BS (Thorlabs KM100BP) is fixed over the main base in a 45° angle in respect to the incident beam. Just as the KMSS-1 supports, the KM100BP also has two adjusting screws that give the ability to calibrate the inclination of the beam splitter so that the vertical and horizontal planes of the pulses are the same. As shown in Figure 16 the Michelson interferometer array is non-collinear, the spatial separation is achieved by the two fixed mirrors (FD) located at a 90° angle in respect to each other.



FD - Fixed Delay
DD - Dynamic Delay
BS - Beam Splitter

The complete Michelson interferometer stage has divided the pulse in two 50/50 replicas and, depending on the position of the WFD, one of the replicas delayed in time with respect to the other. Up to now, one of the replicas has been delayed a small portion of time (depending on the WFD position) in respect to the other.

ray.

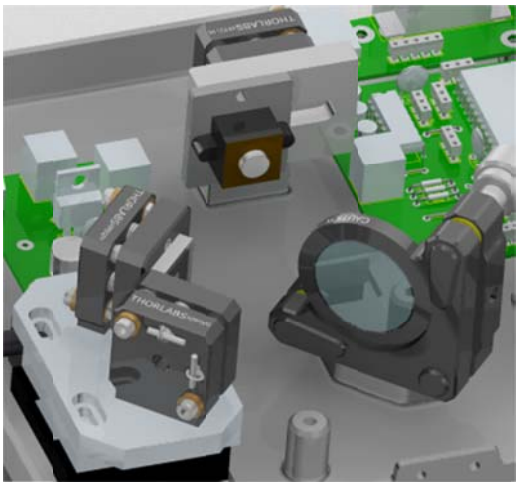


Figure 17: Non-Collineal Michelson Interferometer Delay Line

- **The SHG (Second Harmonic Generation) Stage**

At the second stage the SHG (Second Harmonic Generation) crystal (BBO, type I) and two focusing lens are found. The first lens is used for focusing both replicas, the other for collimating the SFG (Sum Frequency Generation) generated in the crystal. This part needs of a lot of precision since the SHG crystal is really thin (200 microns) and the condition to generate the SFG pulse is that the focused spot of both replicas match in time and space inside the SHG crystal at the phase matching angle. When the SHF pulse is generated it will be diverging and then collimation will be needed, for this the second lens was placed.

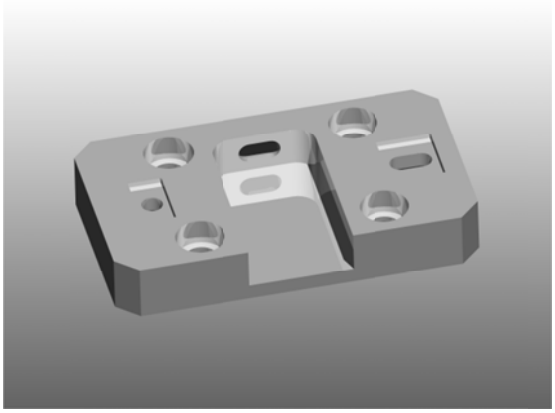
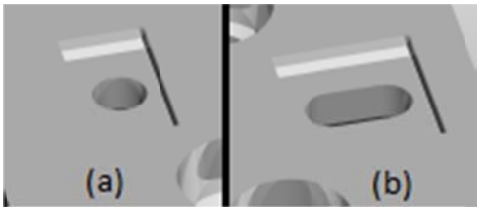


Figure 18: Focusing lens support base with an adjustable canal for the SHG crystal.

In Figure 18, the four clear holes are for attaching the complete module to the main base. At the left, the location for the first lens is fixed but the position for the second lens (the one performing the collimation of the SFG pulse) may be adjusted. The comparison between both support slots can be seen in Figure 19.



The SHG crystal support is not rocket science, as it only needs little space to move through the SHG main base. This support (presented in Figure 20) lies in a rectangle of the SHG main base that is $2mm$ wider than the SHC support, this lets the support to be adjusted so that the crystal is coincident to the focal spot position of the beams. The collimating lens is attached to a rectangular slot (Figure 19, letter b) so it can also be adjusted to obtain a successful collimation.

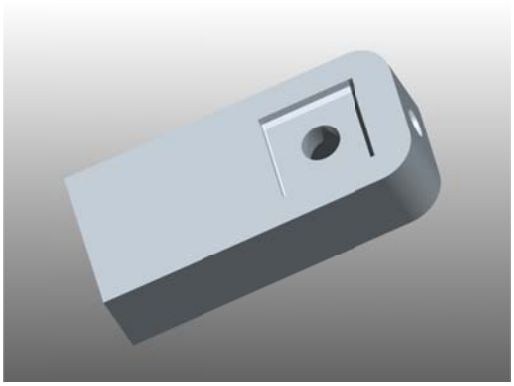


Figure 20: SHG crystal main base.

Each one of the lenses is mounted on a Thorlabs FMP1-M support and the SHC is mounted on a Thorlabs RSP05 support. At this point, both replicas have been focused over the SHG crystal and a third SFG pulse has been obtained and collimated. The complete setup for the SHG stage is presented in Figure 21.

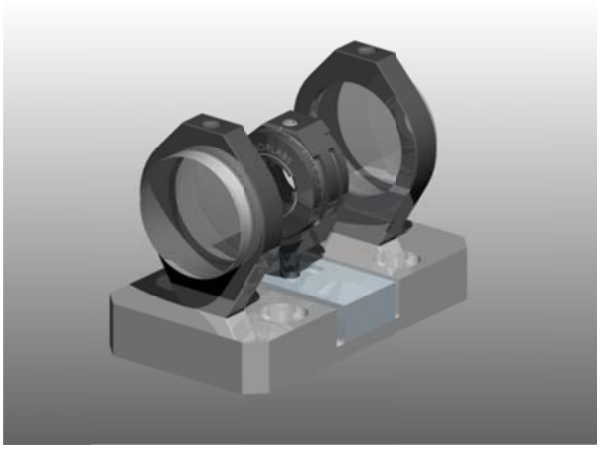


Figure 21: Complete SHG stage.

- **The Spectrometer**

The third stage is the spectrometer. The SFG pulse that was generated in the last stage is separated in its spectral components by a diffraction grating (Newport, 1200 lines/mm, 750 nm blaze angle). A special support (presented in Figure 22) with rotation capability of almost 45° was designed to support the grating so that the obtained line spectrum could be re-directed to a specific location where the webcam is located.

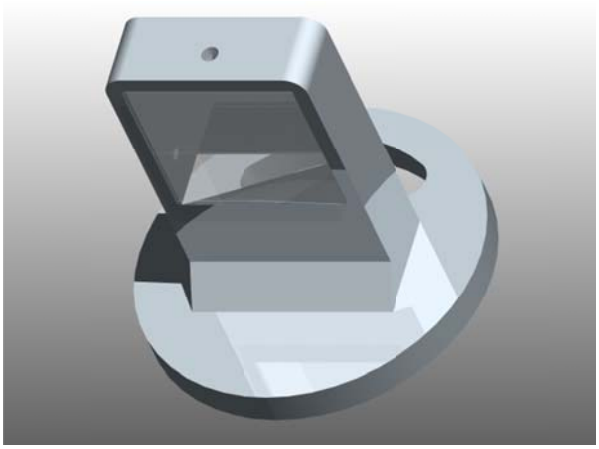


Figure 22: Diffraction grating support.

The pulse spectrum is obtained with a Fourier array, for this another lens is added and located at the focal distance from the grating. At the other side of the lens, at the Fourier plane, the CCD of the webcam is located to capture the line spectrum. This focusing is performed with a Thorlabs lens AC254 that is attached to the webcam base on a Thorlabs FMP1-M support at the focal distance from the grating and the CCD. Finally, the whole instrument is complete, the next steps for processing the SFG spectrum are done by software. The algorithm will be explained in detail in chapter 5.

So the third stage looks like:

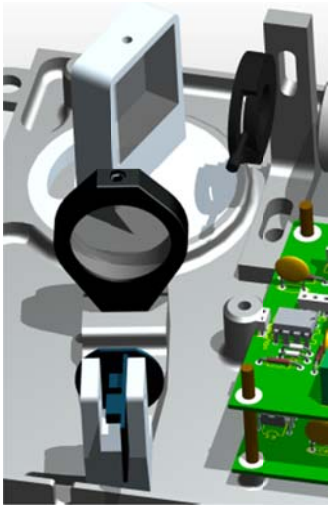


Figure 23: The Spectrometer.

- **The Main Base**

The last stage (ironically) is the main base; all the stages and the electric system in charge of controlling the instrument and communicating with the computer are located here. The locations of all the components were carefully chosen so that the configuration wasn't affected by small vibrations or when the instrument is moved. The electronic components, and specially the power supply, were located near the walls of the base for more efficient heat dissipation.

Four supports were added at the external corners, this was thought to give a formal support to the instrument in case that any attachment to an optic table or any other platform was desired. It also can be seen (Figure 24) that the main supports of the optic components were designed with adjusting capabilities, so that the instrument could work with different lasers and pulse lengths. Additionally five supports were added for the upper casing where some minor electronic controls (for human interface) are placed, such as some buttons and LED indicators. Four of this casing supports were placed in each corner, the fifth support was located near the center of the instrument because it will be of plastic and a middle support helps the extra components to be held without the deformation of the casing letting it to thinner.

This is the most important piece of all, the principal objective of the instrument design demanded that the base could be small, contain all the optic and electronic components, give alignment options and be safe enough to protect the complete instrument. The base was designed, manufactured in just one piece; this is an advantage for calibration purposes and for transportation means, as the internal calibration of the instrument remains the same when the instrument is moved from one place to another. The machining process of this particular piece was quite difficult; the time spent in the CNC machine for the complete process to finish was of around 40 hours and 12 different cutters were needed.

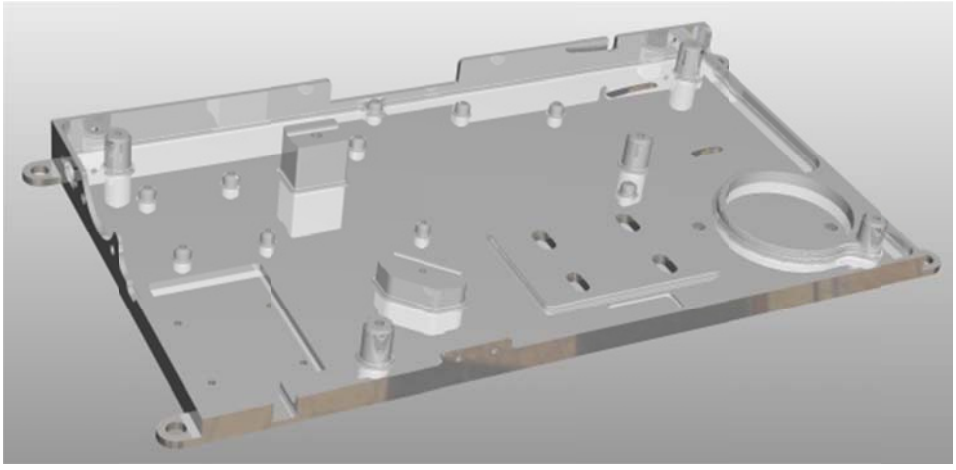


Figure 24: The main SHG-FROG instrument base.

And the complete 3D CAD simulation of the SHG-FROG instrument:

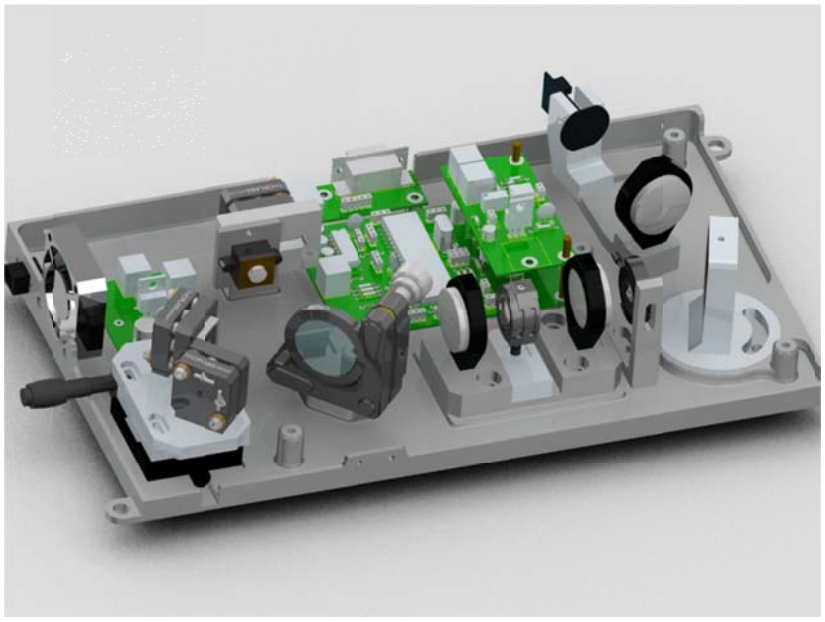


Figure 25: The SHG-FROG Instrument.

Adding the upper case:

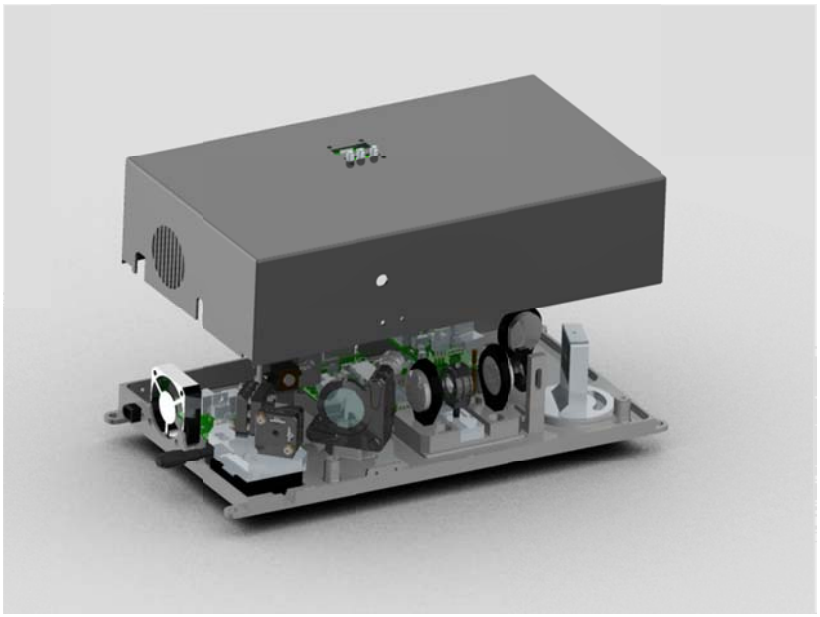


Figure 26: The SHG-FROG Instrument with upper case.

The complete instrument weighs just 2 kilograms, and most of this is because of the aluminum pieces, its dimensions are 24 x 15 x 8 centimeters and one it's aligned it just needs minor adjustments for every time it is used. These details make this instrument user friendly and portable.

4. Electronic Design

The electric system is divided in three mayor segments: The Power System, The Main Control System and The Delay Control System, there is an extra segment used for temperature control especially over the SHG Crystal. All the circuits were designed in segments so that any correction or expansion may be possible without the need of complete redesign, this also helps for expansion purposes as each segment may work with others segments as long as the communication protocols are respected.

- **Power Supply System**

This is a simple regulator, the instrument may be powered with 6 to 12 volts, but as the digital circuits and mainly the microcontroller works with 5 volts the regulation of the voltage is needed. The internal regulator has the capability to supply up to 1 continuous ampere of current. The WFD consumes a maximum of 130 mA and the heating resistor takes the energy directly from the Vpp source (Vpp is used for abbreviation of the source voltage). The full demand of current from the complete circuit is about 200 mA in full operation mode, this gives the option of adding new segments that may work together with the same power source. Anyway, a cooling fan was added just for precautions, this fan also works with the Vpp voltage so that the regulator is free of this charge.

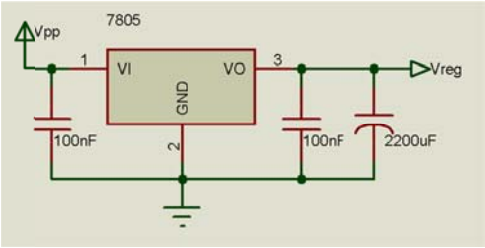


Figure 27: Supply voltage circuit.

The power regulation is performed by a simple LM7805 regulator; because the power demands are low, this commonly used component achieves the stability required for the electronic circuits. Also, because of the low power consumption the heat generation is minimum and the power fluctuations in the main power source are prevented.

- **Main Control System**

The main control system is basically the microcontroller with the communication electronic components. The microcontroller is in charge of synchronizing the communication with the computer and the instrument components; the delay position with the webcam and the temperature control around the SHG crystal. The microcontroller is a PIC18F4550, it is a Microchip PIC18 family device.

There were two important reasons for choosing this device:

- *Multiple pins for possible further expansion without the need of changing the software.*
- *The capability of the USB 2.0 port included in the microcontroller's hardware.*
- *The capability of connecting to a computer by serial port (RS232) or by USB port (USB 2.0) with the same device.*

The PIC18F4550 processor internally works at 48 MHz nevertheless it is clocked with a 20 MHz crystal oscillator. The possibility of working with faster processing execution is achieved by an internal high precision PLL (Ironically Phase Lock Loop) from the PIC18 family. By internal specification of Microchip devices this oscillator frequency (48 MHz) is required for the USB port to work at full speed. At this frequency the machine cycle is of 93 ns average so that all of the actions are performed at really short times in comparison with the webcam capture speed. Because of this, the top speed of the complete instrument is just limited for the webcam image capturing speed.

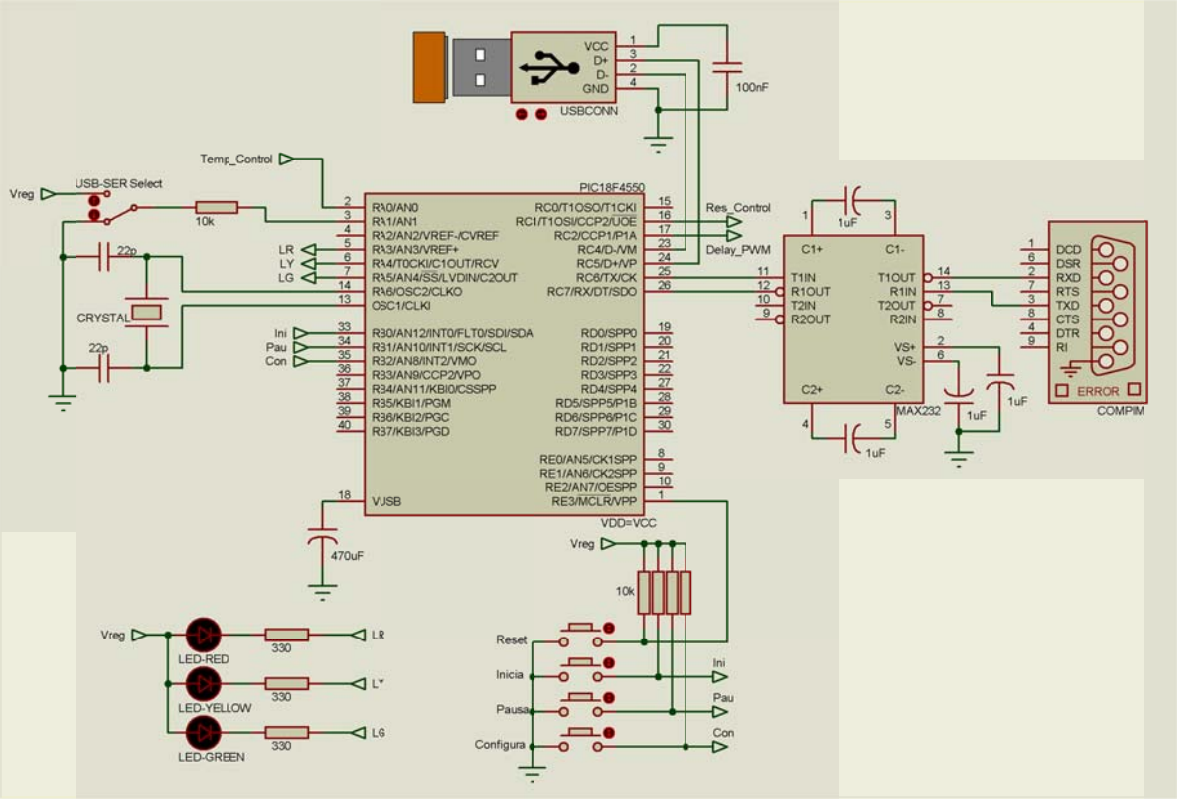


Figure 28: Main PIC microcontroller circuit with the PC interface.

As it was mentioned before, the instrument counts with two communication ports: The Serial port and the USB port. For "state" indications it has four LED indicators: the Power On LED (Red, Not show in the diagram), the Busy LED (also Red), the On Pause state LED (Yellow) and the Capturing LED (Green). Finally, for human interface it has four buttons: The "Reset" button, for initializing the complete instrument if needed, the "Start" button, the "Pause" button and the "Calibration"

button. The “Calibration” button was implemented in case that the user decided to change the PIC18F4550 configurations, such as the center of the moving delay, the control temperature, etc.

- **Delay Control System**

This may be the most complicated system in the whole device, for this reason it will be explained in two sub-divisions: The H-bridge - that is in charge of giving enough current to the WFD - and a low pass filter - that is in charge of eliminating the high frequency’s and giving a stable voltage to the WFD.

The WFD (Webcam Focusing Device) as its name says is an electro-magnet that depending on the current that travels through it generates an electromagnetic field that pushes a group of permanent Neodymium magnets forward, generating a displacement in one axis.

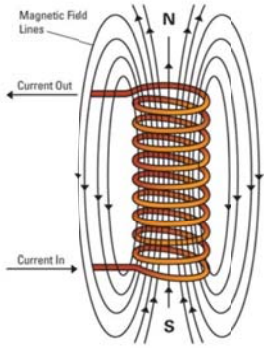


Figure 29: An electro-magnet field simulation.

The electric model of the WFD is a resistor in series with a coil. To calculate the resistance value, different voltages were supplied and the current traveling through the coil was measured:

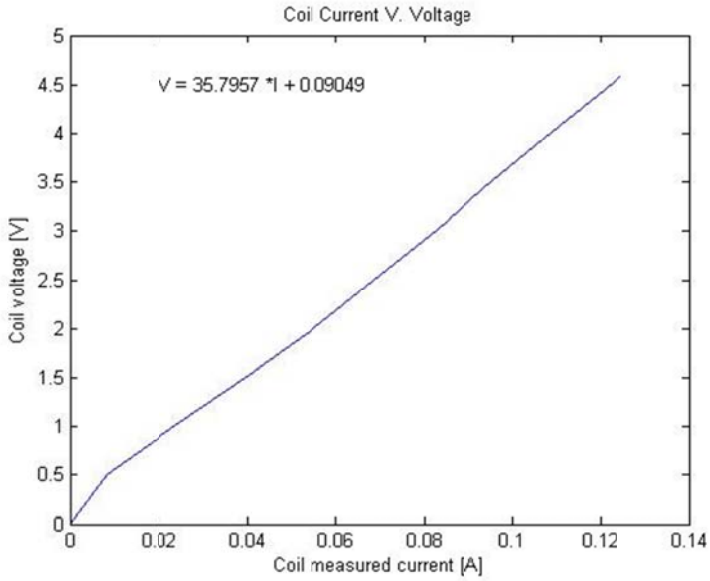


Figure 30: Relation between the coil current and the voltage applied to the WFD.

The slope defines the WFD resistance, which in this case is $m \approx 35.8\Omega = R_B$, so with Ohm's law it is possible to estimate the current in the coil. There is a relation of the magnitude of current vs. position of the delay, but because the WFD is basically an inductor with resistance the regulation of the current is done by controlling the voltage, this is done with a H-bridge L293D device. This integrated circuit is capable of providing maximum 600 mA per channel separating the TTL signal from the PIC microcontroller. The PIC18F4550 generates a 47 kHz PWM signal that is sent to an input channel of the bridge, the parameter that the PIC microcontroller changes is the DC (Duty cycle) of the PWM signal. Changing the DC generates an average voltage over a system that integrates the signal, which is the reason for using a low pass filter between the H-bridge and the WFD.

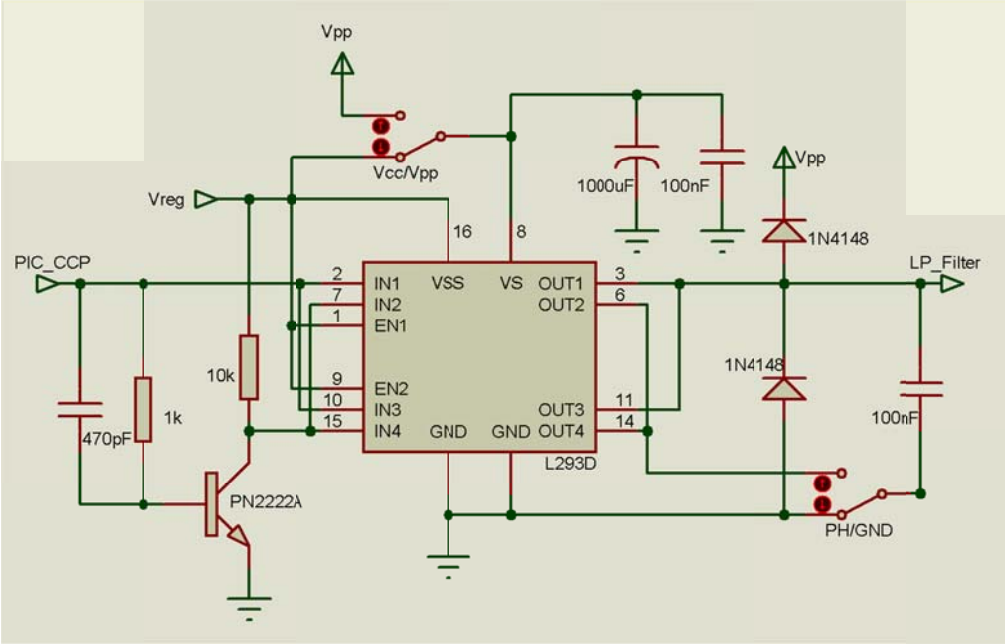


Figure 31: H-Bridge circuit for the interfacing of the WFD.

The DC is defined as the "On" percentage of the complete period of the PWM signal:

$$D = \frac{t_{on}}{T} \qquad \text{<4.1>}$$

Where "ton" is the "On" time of the signal period "T", so a PWM signal would be defined as:

$$V_{PWM} = V_{avg} = V_{on}(D) + V_{off}(1-D) \qquad \text{<4.2>}$$

Because this PWM signal has $V_{on} = 5V$ and $V_{off} = 0V$ the average voltage and the effective voltage are the same:

$$V_{avg} = \frac{1}{t_f - t_i}\int_{t_i}^{t_f} v(t)\,dt = \frac{1}{T}\int_{0}^{T} 5(D) + 0(1-D)\,d\tau = 5D[V]$$

$$V_{eff} = \sqrt{\frac{1}{t_f - t_i}\int_{t_i}^{t_f} v^2(\tau)\,d\tau} = \sqrt{\frac{1}{T}\int_{0}^{T} \left(5(D) + 0(1-D)\right)^2 d\tau} = \sqrt{\frac{(5D)^2 T}{T}} = 5D[V]$$

If the PIC microcontroller varies the DC, it is possible to obtain different average voltages, these depends on how many DC divisions the PIC is capable to make. The CCP (Capture/Compare/PWM) modules of the PIC are able to produce a 10-bit resolution PWM output signals, this means that the total period of the PWM can be divided in $2^{10} = 1024$ sections. If the maximum average voltage is achieved when the DC is 100% and the minimum average voltage at DC 0%, the minimum voltage change is calculated by equation<4.3>.

$$V_{\min} = \frac{5}{1024} = .0048828125 \approx .0049V \qquad \text{<4.3>}$$

To obtain the average voltage from the modulated PWM signal another device between the H-Bridge and the coil is required. The frequency of the PWM is 47 kHz so a low-pass filter is designed to cut the high frequency and leave the integrated voltage go out of it.

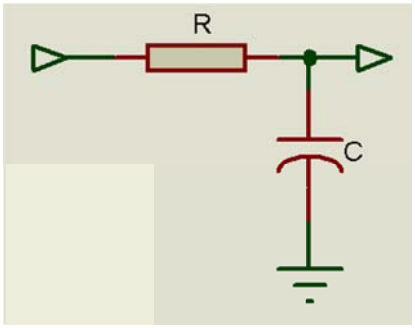


Figure 32: Simple generic low-pass filter.

Here many details come into consideration:

- *The maximum current needed to drive the WFD.*
- *The cut-off frequency of the filter.*
- *The maximum displacement needed from the WFD.*
- *The variation of the maximum current and the cut-off frequency.*

The first factor of the filter design was that it needed to be adjustable, so that the maximum current in the filter resistor could limit the maximum displacement amplitude that the WFD is capable to do. At 5V the WFD by itself demands 139.7 mA, but the filter needs a resistor to work. To enhance the WFD maximum performance some conditions must be taken into consideration for the selection of the filter resistor:

- *The resistor needs to be small enough to allow the maximum displacement of the delay line.*
- *The resistor needs to tolerate the dissipation power without suffering damage.*

To meet these conditions a 10Ω resistor was chosen so that the current could be high enough and the dissipated power doesn't represent a problem:

$$R_{Serie} = \frac{5}{10+35.8} = 109.2[mA]$$

$$R_{10_p} = (109.2)^2 \cdot 10 = 119.2[mW]$$

To give the possibility of resistor adjustment with fine calibration factor but without big increment the next filter was proposed:

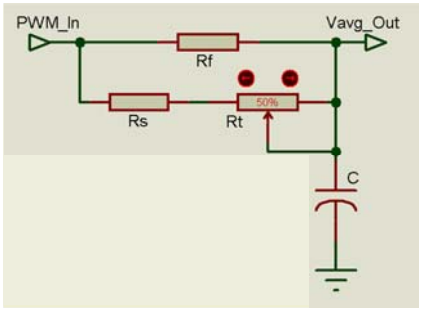


Figure 33: Established designed for the low-pass filter.

The maximum resistance can't be larger than one of both values because it is a parallel array, so it doesn't matter how large RT may be, the total parallel resistance won't exceed Rf. Since Rf restricts the maximum current in the circuit, the parallel relation between Rs and Rf was established as $R_f = 10R_s$.

The equivalent parallel resistance[6] can be calculated as:

$$R_p = \frac{(R_s + R_T)R_f}{R_s + R_T + R_f} \qquad <4.4>$$

$$R_p = \frac{(10 + R_T)100}{110 + R_T} = \frac{100R_T + 1000}{R_T + 110}$$

The R_T potentiometer defines the actual resistance of the filter, because the R_p resulting resistance can't be larger than R_f value, the potentiometer may be of $1K\Omega$ and this will give fine adjusting capabilities, so a Trimpot potentiometer was chosen.

Now, the capacitor will define the cut-off frequency of the filter. The PWM frequency is set at 47 kHz so a low pass filter will integrate the signal and the average voltage will be the resultant signal.

[6] For demonstration of how this equation was generated go to Annex A.

The chosen capacitor value was of $10uF$, which working together with the variable parallel resistor array will define the variable cut-off frequency of the filter:

$$f_c = \frac{1}{RC} \qquad <4.5>$$

$$f_c = \frac{1}{R_p C} = \frac{1}{\frac{100R_T + 1000}{R_T + 110} C} = \frac{R_T + 110}{(100R_T + 1000)C}$$

$R_T[\Omega]$	$R_p[\Omega]$	$R_{Tot}[\Omega]$	$I_{\max}[A]$	$f_c[Hz]$
0	9.3	44.3	0.113	1704
100	52.4	87.4	0.057	303
200	67.8	102.8	0.049	235
300	75.6	110.6	0.045	210
400	80.4	115.4	0.043	198
500	83.6	118.6	0.042	190
600	85.9	120.9	0.041	185
700	87.7	122.7	0.041	182
800	89.0	124.0	0.040	179
900	90.1	125.1	0.040	177
1000	91.0	126.0	0.040	175

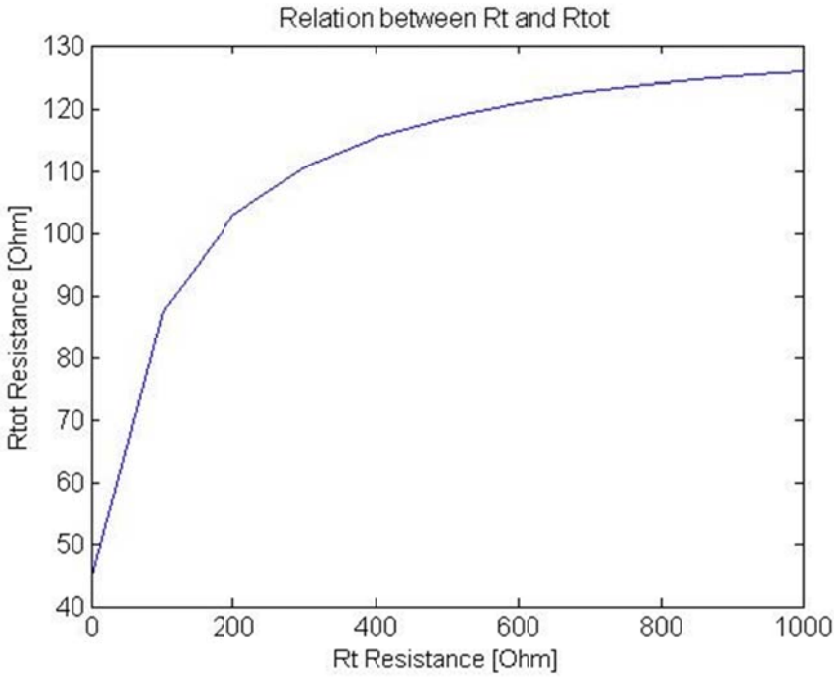


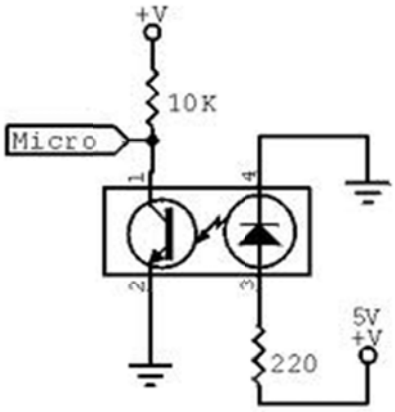
Figure 34: PWM Filter parallel resistance relation.

In Figure 34 R_{Tot} is shown in function of R_T. It is defined as the sum of the filter resistance and the coils resistance:

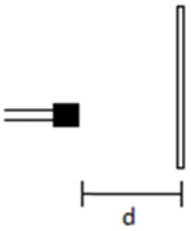
$$R_{Tot} = R_p + R_B \qquad <4.6>$$

After the PWM signal is generated, it is passed through the low-pass filter to obtain the average voltage. The filtered PWM signal defines the distance the WFD generates. To control the position it needs to be characterized and calibrated.

The maximum displacement of the WFD occurs at 5 volts directly, considering that it is a pair of neodymium magnets moving an electromagnet, the distance measurement must be done with a non-contact procedure. For this purpose an optical sensor (QRD1114) was used to identify the relation between the coil current vs. the WFD position. The QRD1114 is a combination of an IR LED and a phototransistor pointing in the same direction. The phototransistor will detect the scattered IR-LED light that is reflected in a nearby frontal object. If this object is closer then the scattered light intensity will be stronger giving a relation between distances vs. intensity. So the QRD1114 may work as an analogic sensor for short distances.



Unfortunately, a phototransistor with a resistor doesn't responds linearly because the circuit works as a voltage divider, because of this the response of the sensor needs to be characterized. First, the sensor was placed in a static position and powered with 5 Votls, and then in front of it a white paper was placed and moved in small distances (.25 mm each).



The results were:

d [mm]	QRD [V]
0	2.26
0.25	2.47
0.5	2.7
0.75	2.93
1	3.14
1.25	3.34
1.5	3.53
1.75	3.66
2	3.8
2.5	4
3	4.17
3.5	4.3
4	4.41
4.5	4.49

5	4.56

Doing a fit analysis with the "Matlab Curve Fitting Tool" suggested a 4th level polynomial equation.

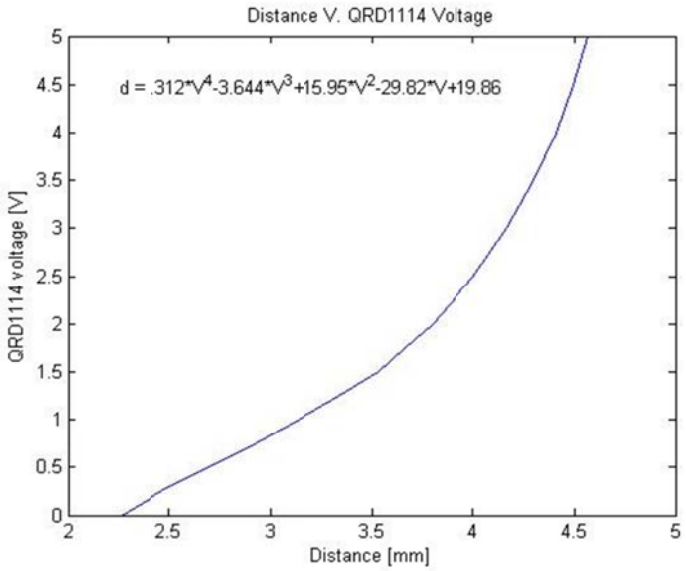


Figure 35: Distance vs. QRD1114 sensor voltage.

The distance measured by the QRD sensor is described by the following equation:

$$d\left(V_{QRD}\right) = .312v^4 - 3.644v^3 + 15.95v^2 - 29.82v + 19.86\left[mm\right] \qquad \text{<4.7>}$$

The next characterization setup was almost the same but this time the WFD was in front of the sensor and the coil was the one that moved towards the sensor. Voltage was supplied to the WFD starting at 0 volts up to 4.5 volts:

Figure 36: The WFD characterization representation.

As the voltage of the coil was varied, the voltage of the QRD sensor and the coil current were measured. With both values it is possible to generate a direct relation between the coil current and the distance traveled obtained with equation <4.7>.

V WFD [V]	I WFD [A]	d [mm]
0.511	0.0083	0.499

1.01	0.0241	0.515
1.528	0.0405	0.528
2.014	0.0553	0.541
2.552	0.0701	0.553
3.025	0.083	0.568
3.511	0.0956	0.583
4.01	0.109	0.599
4.59	0.1244	0.615

The parameter which the microcontroller is able to control is the average voltage. This voltage generates a specific current depending on the resistance of the coil in series with the one calculated for the filter, so the relation between this two factors is calculated using the Matlab curve fitting tool and a new equation is generated. The new equation sets a direct relation between the current along the coil in respect to the distance that it travels. With this relation it is possible to calculate the maximum current desired for a specific maximum displacement generation depending on the length of the ultra-short pulse that is desired to be measured.

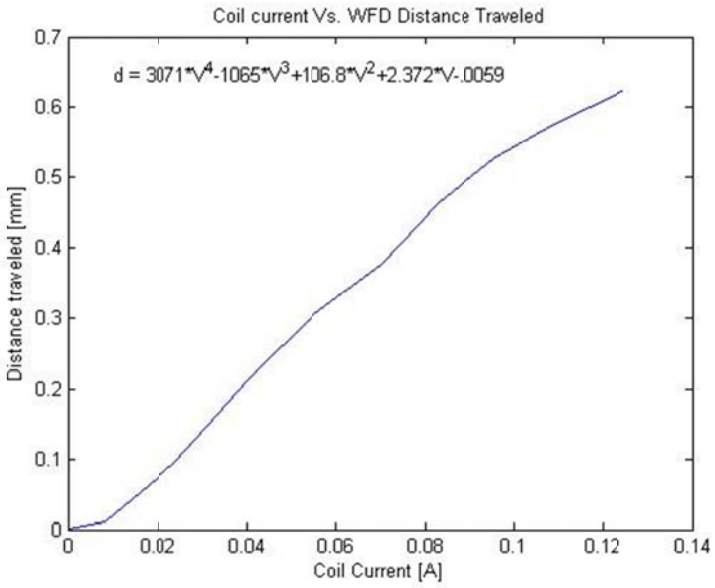


Figure 37: Relation between the WFD coil current and distance generated.

The relation that describes the WFD movement as a function of the applied current is:

$$d\left(I_{QRD}\right) = 3071i_c^4 - 1065i_c^3 + 106.8i_c^2 + 2.372i_c - .0059\left[mm\right] \qquad <4.8>$$

Now that we have the possibility to predict the coils movement with the currents relation, using some other equations it is possible to establish a relation directly to the microcontroller PWM control signal.

Directly from Ohm's law it is possible to define that:

$$i_c = \frac{V_{PWM}}{R_{Tot}} \qquad \text{<4.9>}$$

If equations <4.6>, <4.4>,<4.2> and <4.1> are replaced in equation <4.9> and assuming that $V_{off} = 0[V]$ then:

$$i_c = \frac{V_{on}(D)}{\left(R_B + \frac{(R_s + R_T)R_f}{R_s + R_f + R_T}\right)} \qquad \text{<4.10>}$$

Replacing equation <4.10> on <4.8> generates a final equation that will describe the distance traveled by the WFD that depends on all possible variable factors of the delay system.

$$d = \frac{3071(V_{on}\cdot D)^4}{\left(R_B + \frac{(R_s + R_T)R_f}{R_s + R_f + R_T}\right)^4} - \frac{1065(V_{on}\cdot D)^3}{\left(R_B + \frac{(R_s + R_T)R_f}{R_s + R_f + R_T}\right)^3} + \frac{106.8(V_{on}\cdot D)^2}{\left(R_B + \frac{(R_s + R_T)R_f}{R_s + R_f + R_T}\right)^2} + \frac{2.37}{\left(R_B + \frac{(}{R}\right.}$$ **El tex**

Of course equation <4.11> looks quite horrible and unfriendly, let's re-define it by replacing some fixed values in the equation:

$$V_{on} = 5[V]$$
$$R_B = 35[\Omega]$$
$$R_f = 100[\Omega]$$
$$R_s = 10[\Omega]$$

Now equation <4.11> only depends on R_T value and D. The delay generated only depends on the trimpot adjustable resistance and the duty cycle percentage of the PWM signal, so equation <4.11> is now:

$$d(D, R_T) = \frac{1{,}919{,}375(D)^4}{\left(35 + \frac{(10 + R_T)100}{110 + R_T}\right)^4} - \frac{133{,}125(D)^3}{\left(35 + \frac{(10 + R_T)100}{110 + R_T}\right)^3} + \frac{2{,}670(D)^2}{\left(35 + \frac{(10 + R_T)100}{110 + R_T}\right)^2} +$$ **El tex**

The two independent variables (D and R_T) have the possibility to take a finite distribution of values, in the case of the duty cycle it is a percentage, so the maximum is 100% = 1, and the minimum starts in 0% = 0. Since R_T as it is a variable resistance it starts at 0Ω and ends at 1000Ω. Within these intervals it is possible to analyze the way the WFD will act depending on the different configurations:

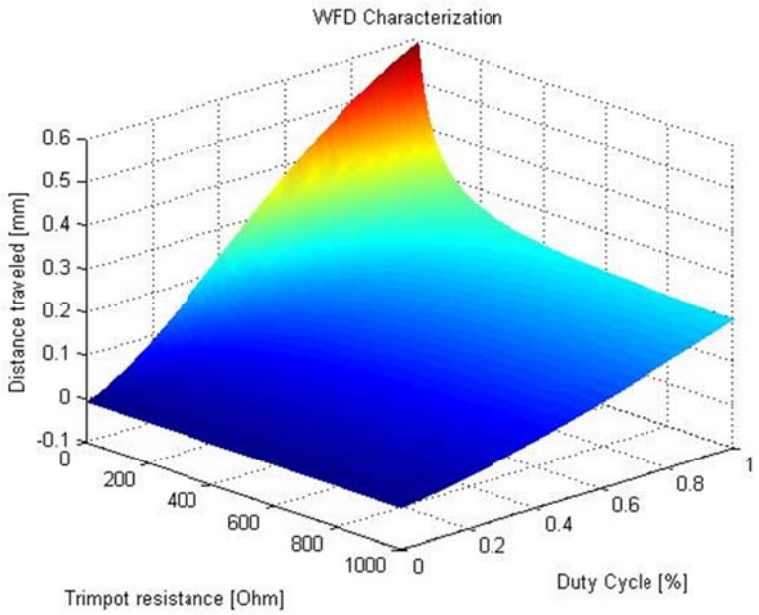


Figure 38: Delay amplitude generated by the WFD in relation with the filter trimpot resistance and the PWM duty cycle.

The WFD characterization graph shows a very particular response of the coil movement. The trimpot resistance results to be a defining variable as it is calibrated an doesn't change while the instrument is working, on the other hand, the duty cycle is the variable factor that changes while the instrument is working. There are several positions where the coil's movement is linear, but there are others, especially at the maximum distances where the behavior is nonlinear.

The configuration of the R_T parameter depends on the length of the pulse to be measured. The SHG-FROG instrument needs a full trace autocorrelation of the pulse, the delay that the WFD needs to generate must be at least three times the pulse length at the PWHM (Full Width Half Maximum). To convert from femto-seconds to micro-meters equation <4.13> is used:

$$d_{delay} = c \cdot t_{pulse} \qquad \text{<4.13>}$$

Where "c" is the speed of light, so the desired delay time is three times $200[fs]$ that means $600[fs]$ of delay:

$$d_{delay} = \left(600x10^{-15}\right)c = .00017988 = 179.88x10^{-6} \approx 180[\mu m]$$

In a "Michelson interferometer" the optical path that a pulse travels is two times the physical distance[7]; the next table shows some different pulse lengths in relation with the required delay travel distance:

[7] See chapter 3 for detailed explanation.

Time [fs]	Distance [mm]	Distance [µm]	Distance x2 [µm]
20	0.006	6	12
100	0.030	30	60
200	0.060	60	120
300	0.090	90	180
600	0.180	180	360
800	0.240	240	480
1000	0.300	300	600

Finally, the chosen configuration was decided to be with $R_T = 40[\Omega]$ and the duty cycle to travel from 21% to 57% centered at 39%, this gives the needed distance and the linearity needed from the WFD.

- **Temperature Control System**

This device is basically an instrumented digital thermometer and a power transistor controlling two power resistors with a PWM signal. The PWM signal is generated by the PIC microcontroller (notice that this PWM signal is different from the one that is used for the delay) which controls the power transistor in an On/Off switching mode quite similar to the delay. The difference in this case is that the PWM is used to control the average voltage over the power resistors. The two resistors are connected in series, one is of 8.2Ω and the other of 15Ω giving a total sum of 23.2 Ω and both considered to support a maximum of 2W.

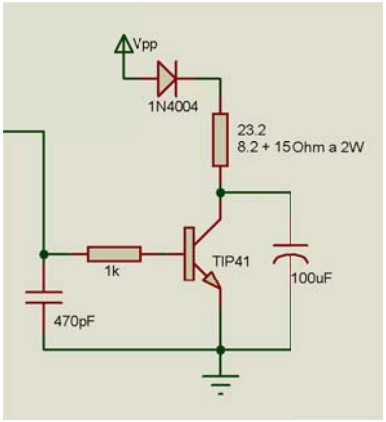


Figure 39: Interfacing circuit for the power resistors.

The digital thermometer is a LM35 centigrade thermometer, in the way it is implemented it is able to measure from 2 degrees up to 150 degrees with a resolution of 10 milli-amperes per degree, so when it reaches the maximum temperature this device gives only 1.5 volts. Because the PIC microcontroller has an internal ADC with the range between 0 to 5 volts the thermometer circuit requires a signal adaptation circuit.

Some calibration details were established before, a consideration that the maximum temperature the thermometer will sense is 50 degrees. The power resistors won't be able to generate upper

temperatures than the ones taken in the consideration and because the ambient temperature will never reach the established maximum (or at least an ultra-short laser pulse measurement in a distant extremist hot desert is still far from happening) the LM35 amplification was calculated for a maximum temperature of 50 degrees.

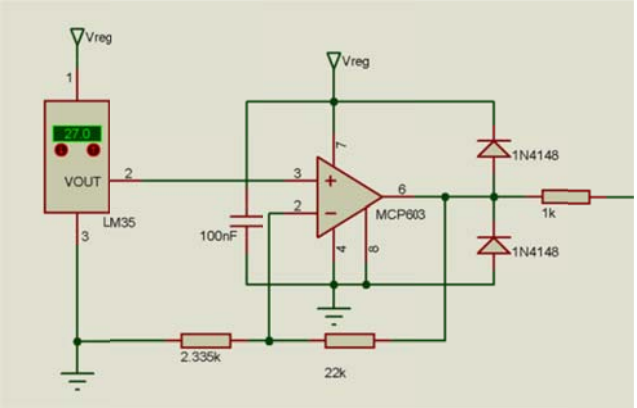


Figure 40: LM35 temperature sensor voltage amplifier circuit.

A linear calibration[8] was calculated with the help of a unipolar Microchip OpAmp in a non-inverting amplification configuration.

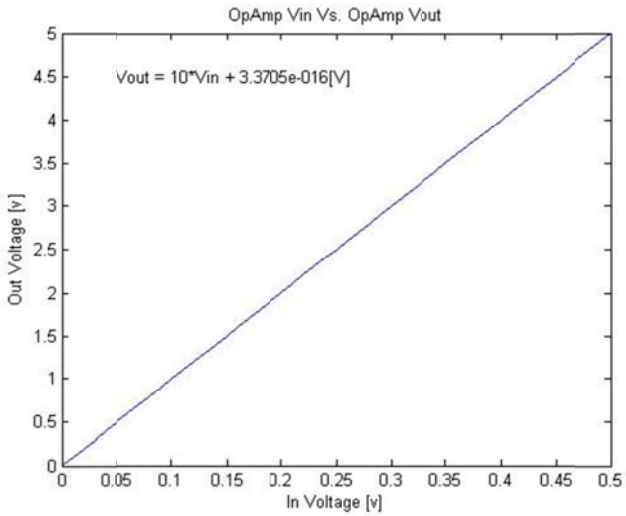


Figure 41: Relation between the OpAmp input voltage vs. the OpAmp output voltage.

$$V_{out} = 10V_{in}\left[V\right] \qquad \text{<4.14>}$$

In equation<4.14>, a linear relation with a scaling factor of 10 multiplying the input voltage to obtain a maximum output voltage of 5 volts. Depending on the magnitude of the voltage read by the ADC form the PIC microcontroller the duty cycle is set to add or subtract power to the resistors thus controlling the temperature. The controlling program is simply a PID algorithm that will be explained in chapter 5.

[8] For detailed explanation on the calculation refer to Annex B.

5. PC and Microcontroller Software

Nowadays almost every instrument has some kind of software related to it, personal telephones, cameras, oscilloscopes, etc. The portable SHG-FROG instrument has two different software's working together synchronizing the data acquisition with the physical delay and the real time calculations performed by the PC.

The main PC software was written in C# in Visual Studio 2010 environment, because of this, the software is able to work in any actual PC (Win XP, Vista, 7 and 8 at x32 & x64 environment). The microcontroller's software was created in C language using the PIC CCS compiler over the Microchip MPLAB 8 platform.

The main PC software was designed in blocks, because C# is an OOP (Object Oriented Programming) language. The possibility of reusing "program methods" shortened the software, especially in complex data processing (like FFT transforms, normalizing, complex numbers, etc.).

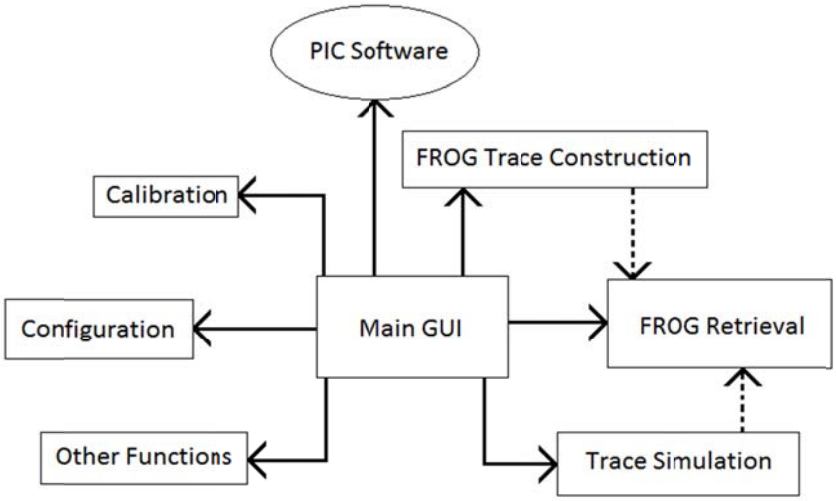
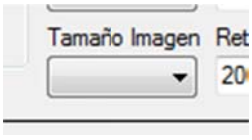


Figure 42: Complete interaction diagram.

- **The Main PC Software**

Main GUI

The main graphical user interface is the principal window where all the other functions are controlled, it doesn´t matter if the current process is done in the same space or in another one, the data always returns to the main interface.



mage size

As soon as the software starts loading, almost all the variables are initialized with standard values. Some of these values are loaded from a

".dat" file that stores the last configuration used (if the first execution, the file is created with the standard default values). Then the initialization of the buttons and checkbox parameters is performed so that the software starts at the same point every time.

There are further initializations that are performed only when some events happen, the most important of them is the one performed by the webcam and the creation of the "principal methods" class object. The webcam works via a C# object, it is created at the main start but isn't initialized until the "Start" button is pressed for the first time. A control flag is then activated so that the initialization of the webcam occurs just once, if the acquisition process is stopped or paused it may be re-started but the initialization of the webcam doesn't occur again until the complete software is closed and opened again. The "principal methods" object is also created at the beginning of the execution and is initialized when the "image size" tab is changed; this action is performed every time the size menu changes because the size of the 2D arrays that are used for the numerical processing and the creation of the image bitmaps depend directly on this number.

.: Aquisition

The main GUI was designed to show the real-time webcam image and the retrieved FROG trace as it is being constructed, these two images were placed at the left side of the window (See Figure 45, pointer 1) with the same webcam proportions but scaled in size, in the middle, the filtered recovered/simulated trace and the calculated trace can be found (See Figure 45, pointer 2). There are two charts, one corresponding to the simulated/acquired trace and the other for the recovered trace (See Figure 45, pointer 3). The upper chart only shows the simulated intensity profile, if the software is acquiring a trace this chart is disabled. The other chart shows the intensity and phase profiles in time or frequency (depending on the user selection) of the recovered trace.

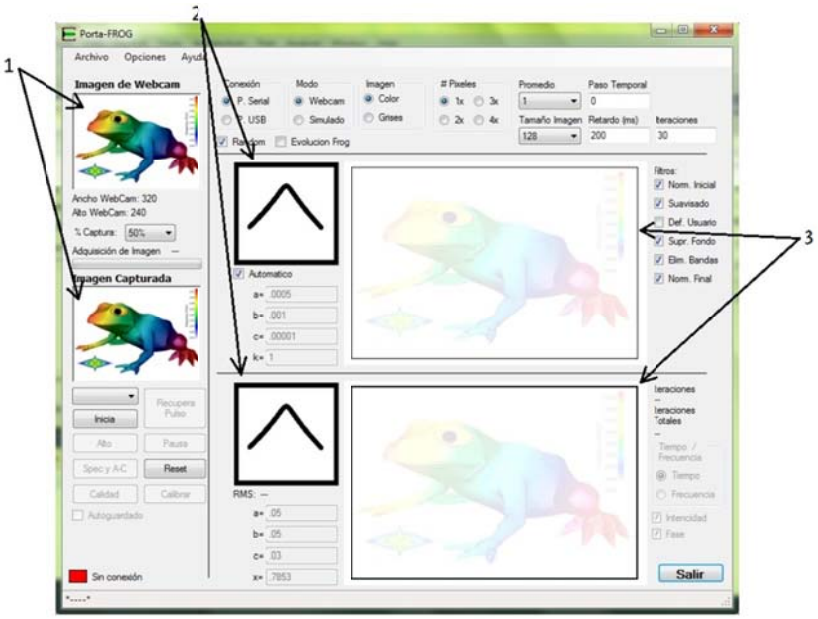


Figure 45: Main Window.

Trace Simulation

The synthetic pulse trace was implemented to test the software in further updates and to aloud the user to calibrate or to test the software with intentional chirped pulses. The simulation is forced over a certain type of pulse architecture, the simulated pulse parameters can build up a simple Bandwidth Limited (BL) Gaussian pulse or add linear and second order chirp (also a possibility of adding a second delayed pulse with different amplitude). By default the software loads some specific parameters, nevertheless, the controls in the main GUI give the possibility to either select randomly or manually the input parameters.

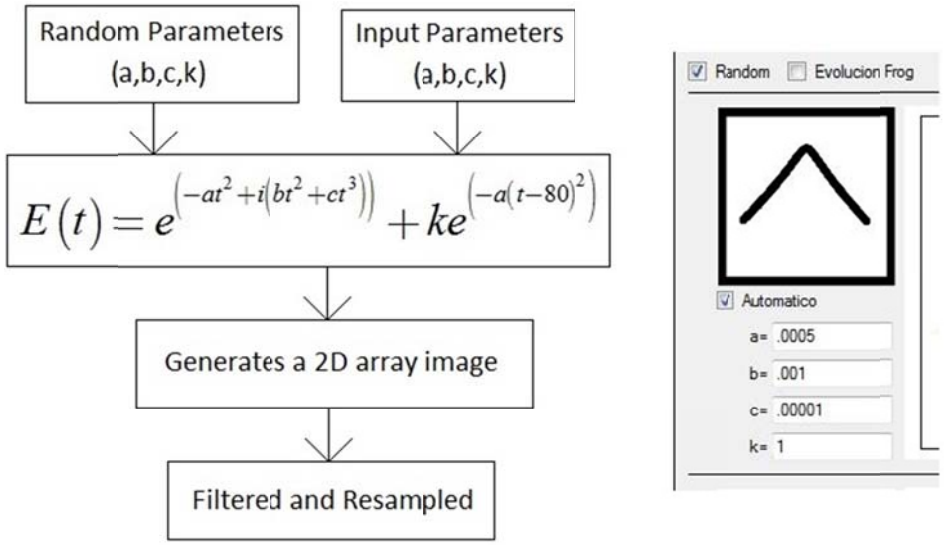


Figure 46: Simulated pulse architecture and input parameters menu.

Trace Acquisition/Construction

The acquisition and construction of the SHG-FROG trace (time delay vs. frequency) is a simple but long process (See Figure 53). It may be divided in steps:

- *Webcam Image acquisition*
- *Color to Grayscale conversion*
- *Mean calculation of the webcam image height*
- *Rotation and interpolation of the height to width*
- *Repetition of the process*
- *Resampling of the acquired image*
- *Filtering of the acquired image*
- *Storage in memory*

The complete process with simple configuration parameters lasts about two minutes. After an acquired image has been captured, the system is able to start the acquisition of a new SHG-FROG trace without erasing the last one until the new trace is completed, this allows the user to save the acquired trace or work with it before deciding what to do.

- *Webcam Image acquisition*: As it sounds, it takes one picture and temporally stores it in the memory.

- *Color to Grayscale conversion*: The stored image is converted to B&W with a powerful algorithm that converts each component of the color (Red – Green – Blue) to its luminance representation in one same pixel and dimension, it's the same method used to convert images in standard color TV and some video formats like PAL or NTSC[9]. The model used for each B&W pixel is:

$$Y = .299R + .587G + .114B$$

$$Y = .3R + .59G + .11B \tag{4.15}$$

> "*The coefficients represent human perception of colors, in particular that humans are more sensitive to green and least sensitive to blue.*"[10]

- *Mean calculation on the image height (*Figure 47*, section a)*: Because each delay generated by the WFD corresponds to a different spectra and the SHG-FROG trace is the concatenation of this spectra's, each captured image needs to be adjusted before the concatenation process. First, the mean of each column is calculated, this results in just one height mean line as long as the width of the image.

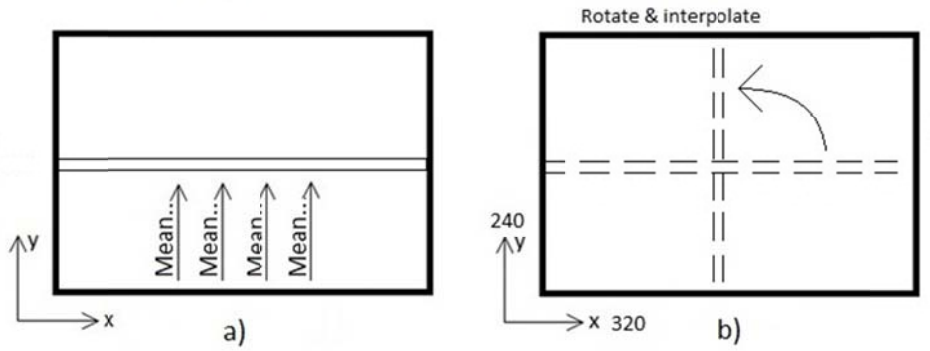


Figure 47: a) Mean calculation of the columns. B) Rotation and interpolation of the mean line.

- *Rotation and interpolation of the image height to width*: Each mean line (that is as long as the width of the image) is rotated and interpolated so it can fit in the image height (Figure 47, section b). The SHG-FROG final trace must be symmetric in the "X" axis (delay time) but not necessarily symmetric in the "Y" axis (frequency):

[9] Reference from http://www.switchonthecode.com/tutorials/csharp-tutorial-convert-a-color-image-to-grayscale
[10] Reference from http://en.wikipedia.org/wiki/Grayscale

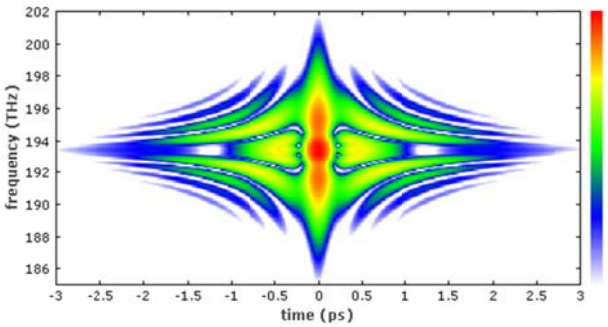


Figure 48: Complete FROG Trace.[11]

The webcam captures just one line of spectrum for each corresponding delay. All involved frequencies will be distributed along the "X" axis:


Figure 49: One SHG spectra cached by the CCD.

After calculating the mean of all the columns ("Y" axis) the resulting line spectrum needs to be rotated and interpolated so that it can fit the height of the constructed image and place the frequencies in the vertical axis. A simple interpolation algorithm is implemented, a linear relation. In the equation (4.16), the parameter "m" depends directly on the size of the image and the "b" parameter is always cero because the first pixel on the horizontal line will always be the first pixel of the vertical line.

The resulting algorithm is:

$$y = mx + b$$

$$m = \frac{I_{Width}}{I_{Height}} \;\&\; b = Offset = 0 \qquad (4.16)$$

[11] Reference from http://www.rp-photonics.com/frequency_resolved_optical_gating.html

$$Y_{Pixel} = \frac{I_{Width}}{I_{Height}} X_{Pixel} \tag{4.17}$$

Where:

- Y_{Pixel} *is the position of the pixel in the vertical axis (Height).*
- X_{Pixel} *is the position of the pixel in the horizontal axis (Width).*
- I_{Width} *is the total width of the image.*
- I_{Height} *is the total height of the image.*

- *Repetition of the process*: The above process is repeated as many times as needed to fill the constructed image width with interpolated mean lines. So, if the FROG is working with a webcam with 320x240 pixels resolution it will repeat the process 320 times.

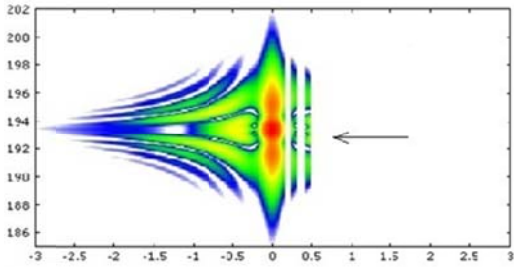


Figure 50: Simulation of the construction performed by the software of a FROG trace.

- *Resampling of the acquired image*: When the repeating process is finished, a SHG-FROG trace will be built up with the webcam dimensions. The FROG retrieval algorithm works with square images of fixed sizes that are multiples of 2^n, (in this software it is possible to work with 32, 64 and 128 pixels resolution in square arrays) so that a complete resampling of the captured trace is needed. This resampling is done directly with the interpolation methods of the graphics C# library. From all the possible algorithms the HighQualityBicubic option was selected, it takes longer time to compute than any other option, but since it is only one image per process, this "longer time" doesn´t really affect as it takes more less 180 milliseconds.

- *Filtering of the acquired image*: The software may work with five different filters, and, considering that all of the filters can't work directly over a "Bitmap", the resampled Bitmap is converted to a 2D "double type" array. Each filter may be activated or de-activated depending on the user's convenience. The first and last filters are equal, just normalization algorithms. The second filter is a convolution matrix filter algorithm between the desired pixel and the eight surrounding it. It's a relation of weights between the surrounding pixels and the central pixel giving a smoothing factor to the resized image; by default this filter is activated.

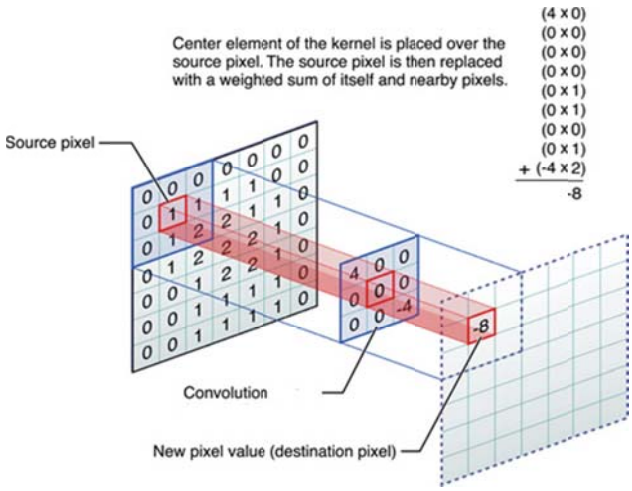


Figure 51: Matrix convolution filter algorithm as a graphic diagram.[12]

The third filter is also a convolution filter, but all the weights are user defined; by default this filter is de-activated.

The fourth filter is for suppressing the background noise of the image. To do this, a complete column of the left side of the image (the column may be calibrated between the 2nd and the 10th) is taken and the mean calculated. This mean value is subtracted from all the pixels in the image, a condition of a minimum value set as cero is checked so that any pixel in the image can never be less than cero. With this filter any noise in the background of the image is removed leaving only the important information of the pulse in it; by default this filter is activated.

The fifth filter is for removing a fixed percentage of the captured trace from the top and bottom of the image. This filter may be adjusted depending on the nature of the measured pulse; by default this filter is activated.

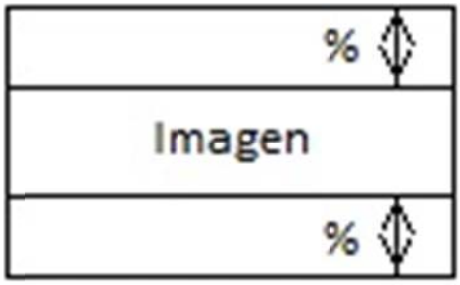


Figure 52: Fifth filter active zone diagram.

- *Storage in memory*: As mentioned before, all of the filters were applied to a 2D "double type" array, but the image can't be printed in this format and also the 2D array memory will be reused in the next measurement process. Consequently it is temporally converted to a Bitmap and immediately saved in a "Memorystream". The Memorystream stores directly in memory the image so it can be accessed without losing any reference while the new image is being built.

[12] Image from https://developer.apple.com/library/mac/#documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html

The complete FROG trace construction process may be resumed with the next image:

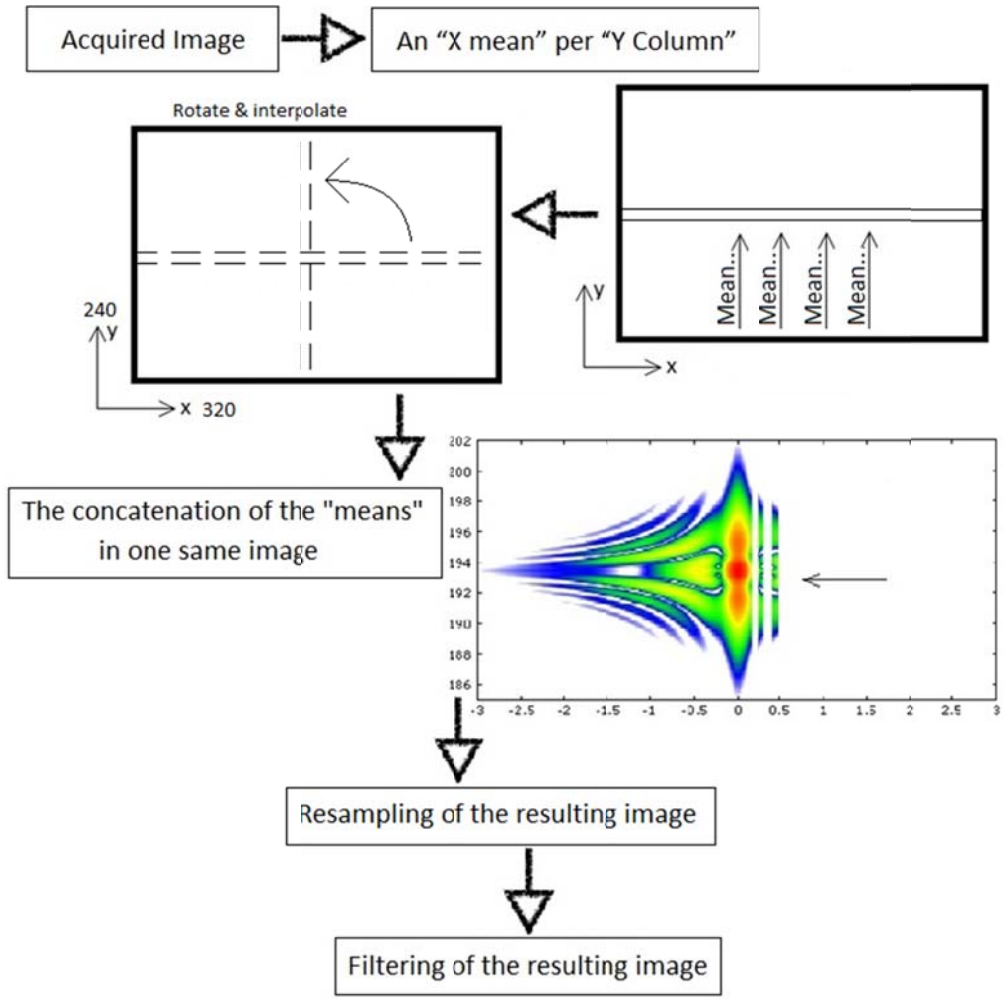


Figure 53: FROG trace construction steps.

FROG Retrieval

The FROG retrieval algorithm is in charge of retrieving the pulse from the acquired trace or the simulated trace. Basically, the algorithm generates a random pulse and creates a FROG trace.

In the general algorithm the random created pulse is computed to create a “random trace”. Then this trace is transformed into the frequency domain using the Fourier transform. As the acquired/simulated trace also is in the frequency domain, both can be compared. The random trace is treated with a mathematical algorithm that will try to achieve the matching between the traces. Finally, the inverse Fourier transform is performed and once again the pulse is recovered in time. This process is continuously repeated until the comparison between both traces (the random one and the acquired/simulated one) reaches an acceptable percentage of error.

In order to force the algorithm to converge, there are different numerical methods which can be implemented. The generic FROG algorithm starts by generating an initial guess of the electric field,

then it is Fourier transformed in order to generate the field in frequency domain. The measured FROG trace is then compared with the guess with some mathematical methods, the objective is to equal the guess trace with the measured trace. A schematic of the algorithm is described below.

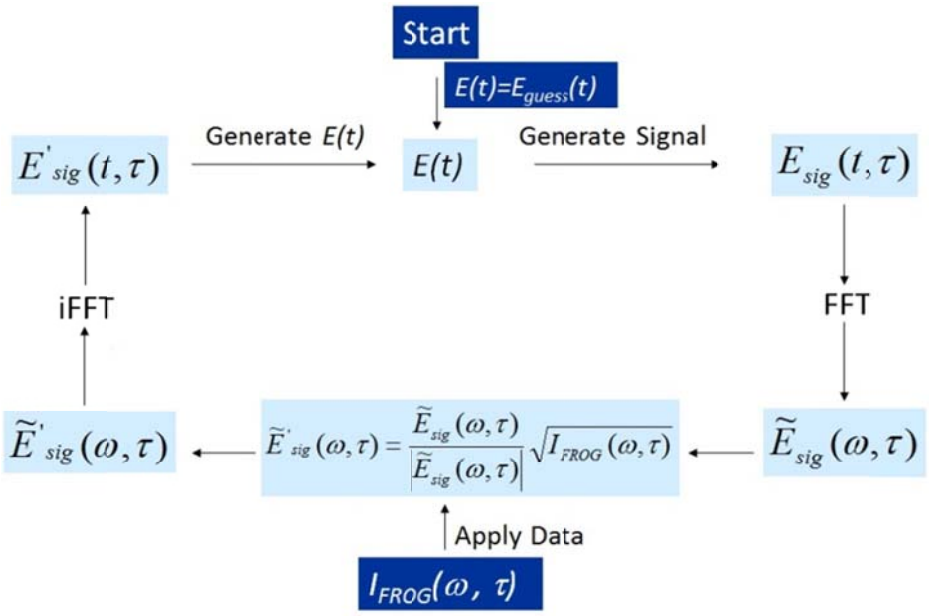


Figure 54: Generic FROG Algorithm

In this software the used mathematical method applies an approach of "Singular Value Decomposition" (SVD)[13]. The algorithm is known as Generalized Projections Method[14].

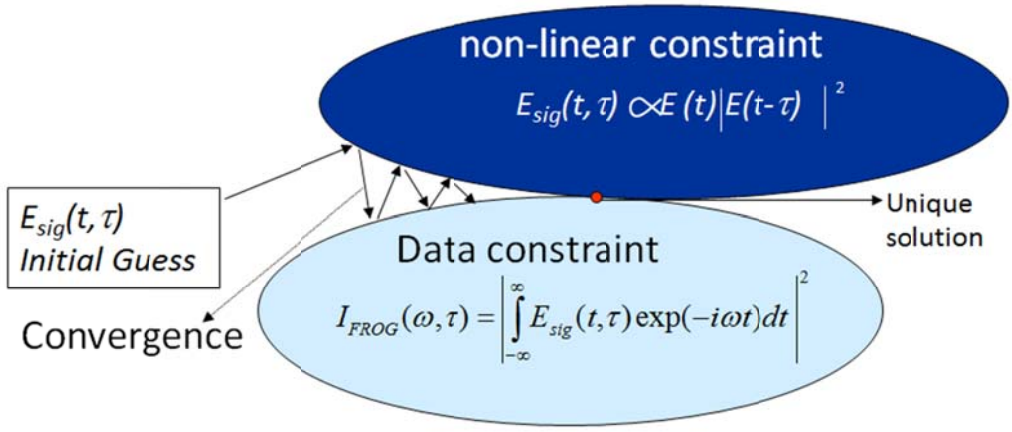


Figure 55: Generalized Projections Mathematical Method.

Calibration

The calibration window is available after a FROG trace has been acquired or simulated, in this window the trace is analyzed so that the intensity and the spectrum are retrieved separately.

[13] "Singular Value Decomposition" (SVD).
[14] Generalized Projections Method.

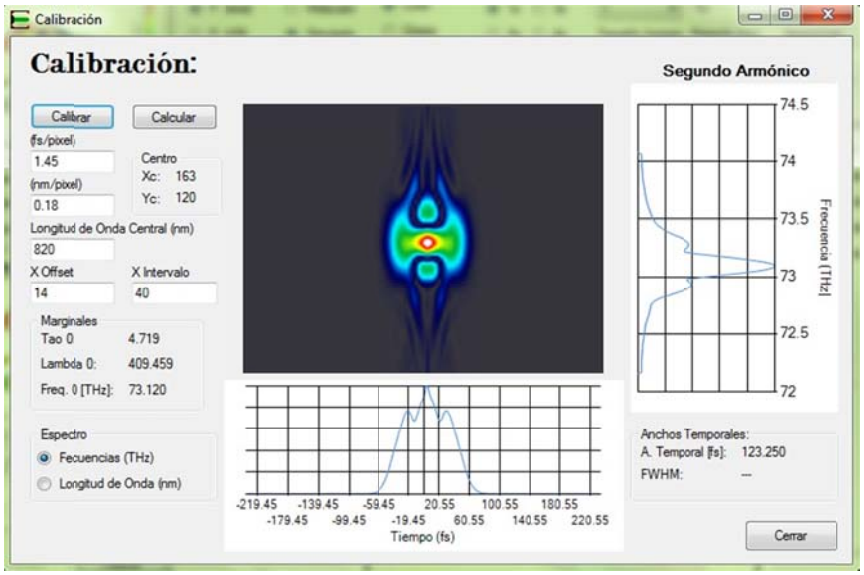


Figure 56: Pulse calibration window.

The trace that the calibration window shows is the acquired/simulated trace; this means that the image represents the second harmonic pulse trace. Knowing this, the calibration must be done with some considerations:

- *The time scale is the same for the fundamental pulse and the SHG pulse.*
- *The SHG Frequency is twice the fundamental frequency.*
- *The SHG wavelength is half the fundamental wavelength.*
- *The relation between the frequency and the wavelength is not linear.*

The calibration window requires some information from the user to compute the real values of both pulses (the SHG Trace and the fundamental pulse values). For more information on how to acquire this information refer to Annex F. The required data is:

- *The relation of femtoseconds per pixel.*
- *The relation of nanometers per pixel.*
- *The central fundamental wavelength of the measured pulse.*
- *The offset factor for the plotting cero reference to fit with the graph.*
- *The desired interval of the plotted graph.*

After this information is introduced the calibration is possible, the first step is to re-scale the wavelength:

$$\Delta\lambda_{SHG} = \frac{\Delta\lambda}{2} \tag{4.18}$$

$$\lambda_{SHG_0} = \frac{\lambda_0}{2}$$

The FROG Trace image has the frequencies in the "Y" axis and the delays in the "X" axis, the relation between the wavelength and the frequency isn't linear, a non-lineal relation is obtained.

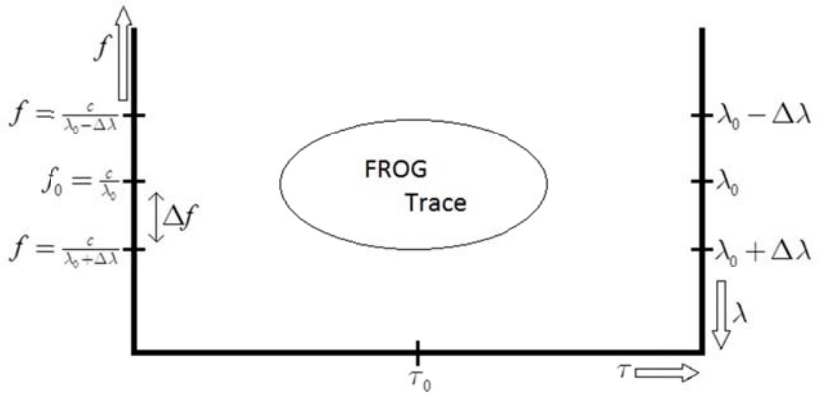


Figure 57: Image configuration of the calibration window.

$$\Delta f = \frac{c}{\lambda_0} - \frac{c}{\lambda_0 \pm \Delta\lambda}$$

$$\Delta f = c\frac{\Delta\lambda}{\lambda_0\left(\lambda_0 \pm \Delta\lambda\right)} \tag{4.19}$$

With these relations between wavelength and frequency the software uses the $^{\Delta\lambda}/_{Pixel}$ to calculate the $^{\Delta f}/_{Pixel}$.

$$^{\Delta f}/_{Pixel} = c\frac{^{\Delta\lambda}/_{Pixel}}{\lambda_0\left(\lambda_0 \pm {}^{\Delta\lambda}/_{Pixel}\right)} \tag{4.20}$$

With this information it is now possible to generate the frequency equivalent scale in the second harmonic space:

$$f = f_0 + \left(n - {}^{N}/_{2}\right){}^{\Delta f}/_{Pixel} \tag{4.21}$$

An interpolation is done from the frequency scale to match the $i \times i$ resampled trace. With the interpolation done, the frequencies are returned to wavelengths:

$$\lambda = \frac{c}{f_0 + \left(n - {}^{N}/_{2}\right){}^{\Delta f}/_{Pixel}} \tag{4.22}$$

Finally, the obtained wavelength scale is multiplied by 2 so that the scale returns from the second harmonic to the fundamental scale. This scale is the one used by the software to display the data of the retrieved pulses. If the calibration is not done, general time units are used and no frequency units are showed.

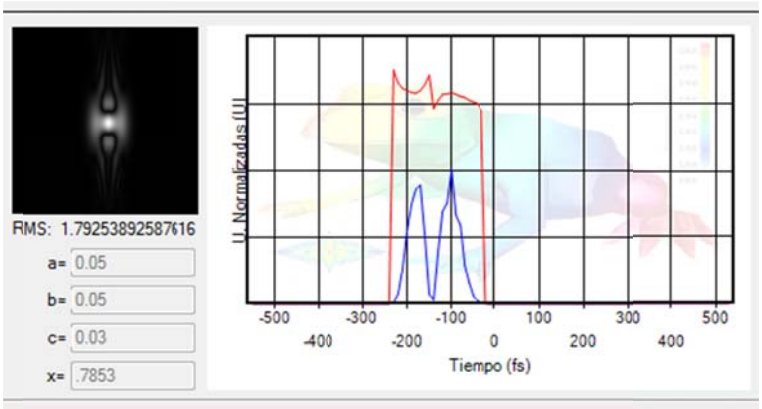


Figure 58: Retrieved data graph.

Configuration

The available configurations in the software are to control and manage different filters and/or parameters that are related to some peripherals (Webcam & Microcontroller) and internal configurations, such as:

- *Webcam resolution*
- *Delay movement relation*
- *Zero Delay position*
- *SHC temperature control*
- *Connection Status*

The movement relation is the amount of times the WFD will move for each picture or couple of pictures to be taken. This shortens the "time delay" generated by the WFD but lowers the noise ratio of the image as more pictures of the same point are taken for each delay position. The software supports 4 different relations, the next table illustrates the case under the assumption of a webcam that works at a resolution of 320 x 240 pixels:

Relation	# Pictures/Change	# Delay steps
1/1	1	320 (+160, -160)
1/2	2	160 (+80, -80)
1/3	3	106 (+53, -53)
1/4	4	80 (+40, -40)

The webcam's resolution -especially the width- directly affects the maximum displacement that the WFD will generate, especially when the movement ratio is 1/1, a 320x240 resolution webcam will generate 320 steps and a 640x480 resolution webcam will generate 680 steps. The movement relation configuration may be variable for every measure so it is located directly at the main GUI, but for the other configurations that must remain constant during the measurements, an especial window was created. This window is really simple, it only gives the possibility to change the temperature at which the SHC heating system will work and the central point of the WFD. As mentioned previously in the electronics chapter, the central position of the WFD depends directly on the PWM duty cycle. This duty cycle resolution is of 10 bits defined by the microcontroller

hardware, so to position the WFD the configuration window uses the 10 bits value converted into its decimal equivalent:

Binary	Decimal
110001110	398
110010000	400
110010010	402

The calibration of the central position may be done directly from the instrument or with this feature. Both methods are totally independent, this helps the user decide which one to calibrate depending on the situation.

Finally, the external configuration window contains an indication of the actual state of the connection between the computer and the instrument.

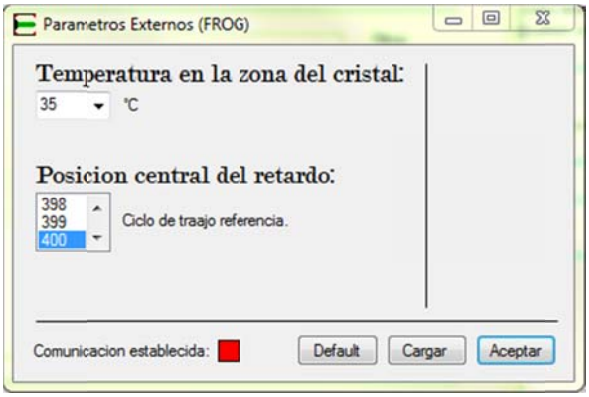


Figure 59: External parameters configuration window.

As mentioned before, there are four filters applied to each image, it doesn´t matter if it's opened from a previous stored session or if it's a new generated trace. Each filter affects the image in a different way; the configuration of each filter is pre-defined but may be adjusted depending on the results that the user wants to achieve.

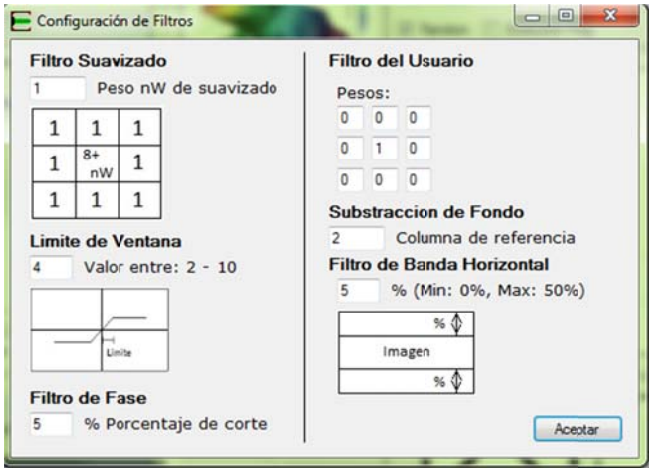


Figure 60: Filter parameters configuration window.

Other functions

The software includes some other helpful functions that give the user the ability to process the acquired or simulated data with different tools. These functions are basic in almost any software, they include:

- *Save simulated data*
- *Save acquired data*
- *Open stored data*
- *Quality of the trace*
- *Webcam Configurations*
- *Microcontroller communication*

Save simulated data: As the subtitle describes, it gives the user the possibility to save the simulated trace, the retrieved trace and the retrieved data. The recovered data means four different charts (the pulse intensity, the pulse phase in time, the pulse spectrum and the pulse phase in frequency) thus the time/frequency scale and the intensity/phase data are stored in separate ".txt" files. This menu also gives the possibility to store the data separately, in case the user desires to store only specific portions of it.

Save acquired data: Has exactly the same function of the past menu, but with the acquired data generated after the retrieval of a real measured pulse.

Open stored data: When the user wants to experiment or study again a stored pulse, only the B&W image is required. This image is loaded as a simulated pulse but may be a stored image of a captured pulse, after loaded the image is converted into a process-able format and the retrieval may be done, acquiring again the pulse data.

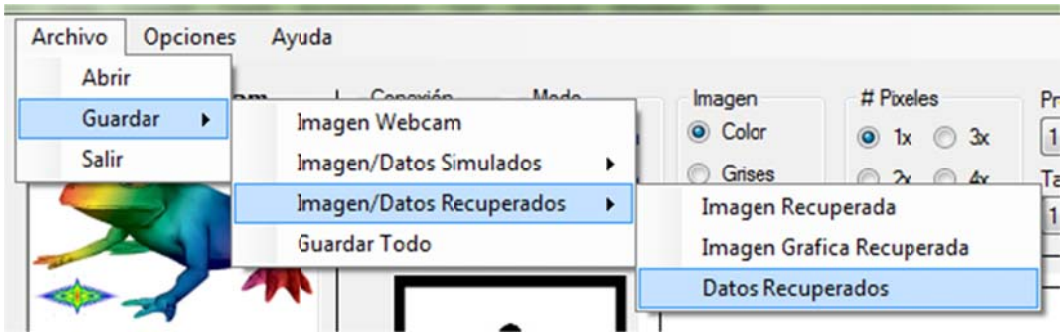


Figure 61: Tooltip menu example.

Quality of the trace: It only gives the user a number representing the RMS value. This value is calculated making a comparison between the retrieved trace and the original trace (it doesn't matter if it's a simulated or a recovered trace). The better the retrieval process the lower the RMS value gets.

Webcam Configurations: Since the image acquisition is performed by a simple webcam, some configurations may be adjusted by the user to calibrate the acquired image. This menu has

controls to change the brightness, contrast, color correction and the RGB saturation parameters of the webcam.

Microcontroller communication: This isn't a menu, it's only an indicator of the status of the communication between the microcontroller and the PC. There are three possible status of this indicator:

- *Disconnected → Red color*
- *Connected: Stand by → Yellow color*
- *Connected: Acquiring data → Green color*

This indicator may be found at the bottom of the main GUI window or at the bottom of the external configuration parameters window.

- **The Microcontroller software**

The microcontroller is a PIC18 chip from Microchip. The PIC18F4550 is a very popular device due its capability's and included peripherals. For the FROG instrument, the important peripherals are the ADC port, the double PWM generation port, the serial communication port and the included USB 2.0 port. The PIC18F4550 software is based on interruption routines; almost all of the time it is waiting for a command to be sent from the PC to perform an action. The only action that the PIC does continuously is the temperature control, it starts at a preset set-point and only modifies it if the PC software sends a request, but it never stops.

The software is also divided in sections that work together:

- *Initialization of the system*
- *Connection scanning*
- *Configuration of parameters*
- *Temperature control*
- *Main acquisition process*

Initialization of the system – As in any other software, the initialization is required to configure the initial states of the peripherals and the variables used by the processor. The selection of the form of communication between the PIC and the PC needs to be defined before the instrument is turned on (with an external switch), if this channel of communication is changed after the initialization, a forced reset is needed to configure it.

Connection scanning – This is an infinite loop waiting for the computer to send an acknowledge signal. When these acknowledge is received, the PIC sets to connected state.

Configuration of parameters – The configuration is a process that may be called from the FROG instrument at any time and is always called at the initialization of the instrument. This process calls all the configuration parameters that were configured at the main software:

- *Delay movement relation*
- *Delay cero position*
- *SHC temperature control*

Temperature control – The temperature control is a simple "automatic PID control", a default set-point is stored in the PIC memory and only changes if the user programs it via the configuration process.

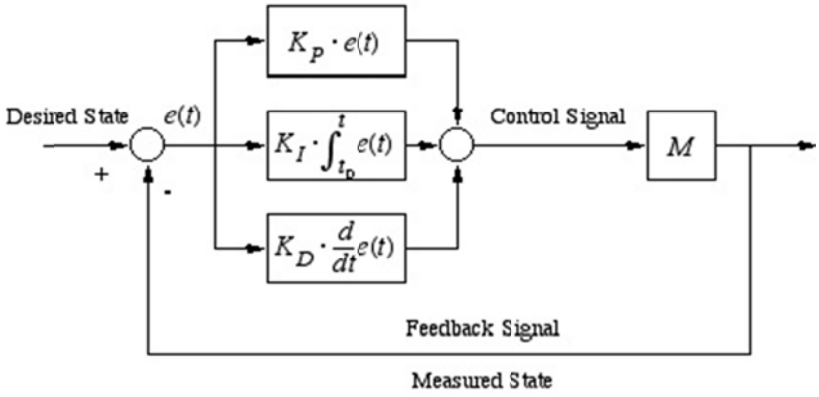


Figure 62: Simple closed loop PID controller [15]

Main acquisition process – The main process is quite simple, it only moves the delay "one bit position", meaning that the duty cycle of the PWM signal is changed in one unit of it resolution so that a minor displacement is generated. Then the PIC sends one acknowledgement to the PC so it can take a picture with the webcam. The delay won't move until the PC sends another acknowledgement to the PIC asking for a new delay position. The main acquisition process communication protocol works like:

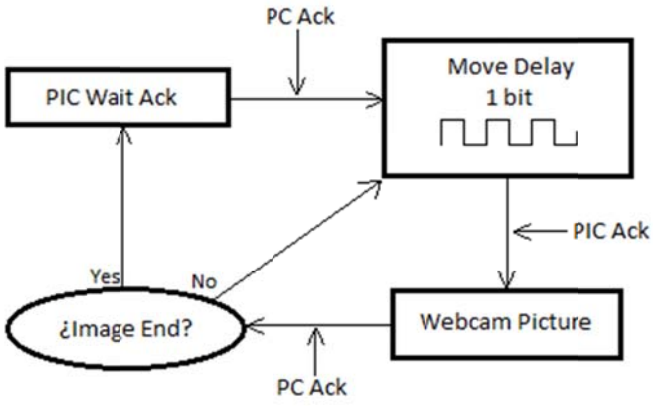


Figure 63: PIC-PC Communication process.

[15] Image from http://commons.wikimedia.org/w/index.php?title=File:PID.svg&page=1

The process repeats “n” times until the complete trace is built in the PC. If the “continuous reading” option is checked in the main GUI, the process starts again immediately, otherwise the instruments stops and get back to “stand-by” waiting for another acquisition.

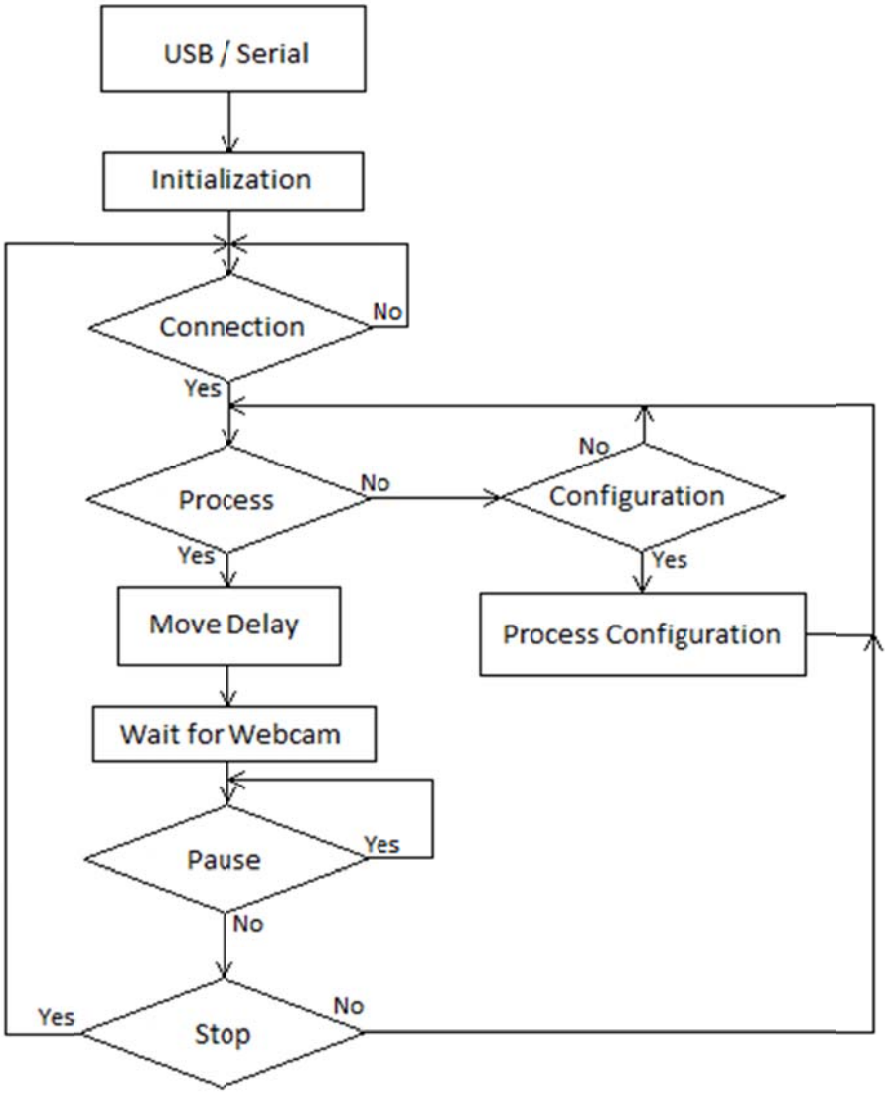


Figure 64: PIC general software diagram.

6. Economic Review

The SHG-FROG instrument has some interesting economic advantages. A complete list of prices and quantities of the components and manufacturing processes costs is included.

Optics:

# Parts	Name	Price (DLS)/U	Total
1	Near IR Achromat AC254-030-B	$ 83.00	$ 83.00
1	Unmounted Visible Achromat AC254-030-A	$ 77.00	$ 77.00
1	Unmounted Visible Achromat AC254-050-A	$ 71.30	$ 71.30
3	Fixed Optical Mount FMP1/M	$ 14.70	$ 44.10
1	Pellicle Beamsplitter BP145B1	$ 160.40	$ 160.40
1	Kinematic Mount KM100BP	$ 75.50	$ 75.50
3	Adjustable support (KMSS/M)	$ 31.75	$ 95.25
2	Precision Plane Mirror 16-MX-16	$ -	$ -
1	Protected Silver Mirror PF03-03-P01	$ 26.00	$ 26.00
1	Rotation Mount RSP05	$ 69.00	$ 69.00
1	Comar	$ -	$ -
1	Diffraction Grating 1200-DG-500	$ -	$ -
1	Iris diaphragms	$ -	$ -
	Optics Sub-total (DLS)		$ 701.55
	Optics Sub-total (MXN)[16]		$ 8,769.38

Mechanics:

# Parts	Name	Price (MXN)/U	Total
1	Main base aluminum block	$ 400.00	$ 400.00
80	CNC Machining hour	$ 350.00	$ 28,000.00
1	Platform for fixed delay mirrors	$ 50.00	$ 50.00
1	Diffraction grating support	$ 50.00	$ 50.00
1	Dynamic support for lens and webcam	$ 50.00	$ 50.00
1	Main webcam support	$ 50.00	$ 50.00
1	Adjustable support for the "diaphragm"	$ 30.00	$ 30.00
1	Dynamic delay support	$ 70.00	$ 70.00
1	Adjustable base for the dynamic support	$ 70.00	$ 70.00
1	Main SHC and lens base	$ 100.00	$ 100.00
1	Adjustable base for the SHC	$ 50.00	$ 50.00
8	1/8" thick 1/4" long screws	$ 5.00	$ 40.00
2	1/8" thick 2" long screws	$ 5.00	$ 10.00
	Mechanics Sub-total		$ 28,970.00

[16] The calculation was made pretending the change equivalent as 12.50 per dólar.

Electronics:

# Parts	Name	Price (MXN)/U	Total
1	PIC18F4550 Microchip Microcontroller	$ 350.00	$ 350.00
2	22pf ceramic capacitors	$ 4.00	$ 8.00
1	20 MHz Crystal	$ 20.00	$ 20.00
1	H-Bridge L293D	$ 75.00	$ 75.00
1	Serial Driver MAX232	$ 28.00	$ 28.00
1	Transistor PN2222	$ 12.00	$ 12.00
1	Transistor TIP41	$ 9.00	$ 9.00
1	Temperature Sensor LM35		$ 10.00
1	5 Volts Regulator LM7805	$ 12.00	$ 12.00
1	12 Volts Cooling FAN VN2-012P	$ 77.00	$ 77.00
1	Female Serial DB9 Connector	$ 9.00	$ 9.00
1	USB Type B Jack		$ -
1	9mm Female Jack		$ -
1	2 Position Small Switch SCMM-122	$ 5.00	$ 5.00
1	Microchip OpAmp MCP603	$ 25.00	$ 25.00
1	Chip Mounting Base 40 Pin	$ 5.00	$ 5.00
2	Chip Mounting Base 16 Pin	$ 5.00	$ 10.00
1	Chip Mounting Base 8 Pin	$ 5.00	$ 5.00
1	Diode 1N4004	$ 2.00	$ 2.00
6	Diode 1N4148	$ 2.00	$ 12.00
1	5K Ohm Trimpot	$ 45.00	$ 45.00
1	1K Ohm Trimpot	$ 45.00	$ 45.00
4	Pushbutton NO AU-1012	$ 4.00	$ 16.00
2	Red LED	$ 3.00	$ 6.00
1	Green LED	$ 3.00	$ 3.00
1	Yellow LED	$ 3.00	$ 3.00
7	2 Terminal T-Block TRT-02	$ 5.00	$ 35.00
1	Electrolytic Capacitor 2200 uf, 16V	$ 8.00	$ 8.00
1	Electrolytic Capacitor 1000 uf, 16V	$ 6.00	$ 6.00
1	Electrolytic Capacitor 470 uf, 16V	$ 4.00	$ 4.00
1	Electrolytic Capacitor 100 uf, 25V	$ 3.00	$ 3.00
1	Electrolytic Capacitor 10 uf, 25V	$ 6.00	$ 6.00
4	Electrolytic Capacitor 1 uf, 100V	$ 2.00	$ 8.00
8	Ceramic Capacitor 100 nf	$ 4.00	$ 32.00
2	Ceramic Capacitor 470 pf	$ 4.00	$ 8.00
1	22K Ohm Resistor	$ 1.00	$ 1.00
6	10K Ohm Resistor	$ 1.00	$ 6.00
3	1K Ohm Resistor	$ 1.00	$ 3.00
4	330 Ohm Resistor	$ 1.00	$ 4.00

1	120 Ohm Resistor	$ 1.00	$ 1.00
1	10 Ohm Resistor	$ 1.00	$ 1.00
1	Main Circuit PCB	$ 800.00	$ 800.00
1	Filter PCB	$ 800.00	$ 800.00
1	Power Source PCB	$ 800.00	$ 800.00
1	Serial & USB Connectors PCB	$ 800.00	$ 800.00
1	Temperature Controller PCB	$ 800.00	$ 800.00
		Electronics Sub-total	$ 4,918.00

The process for machining the aluminum pieces is the most expensive. The machined pieces may serve as base models for the production of replicas with other methods like melting.

The cost of producing the first SHG-FROG instrument is:

Optics Subtotal	$ 8,769.38
Mechanics Subtotal	$ 28,970.00
Electronics Subtotal	$ 4,918.00
Total	$ 42,657.38

A “Swamp Optics” FROG instrument reaches the cost of $17,000 DLS[17] that is the equivalent to $220,000 MXN. In comparison to the one shown in this work (assuming that it will be just a single production) it is 75% more expensive.

The cost of producing the next SHG-FROGs wouldn’t include the machining cost, this advantage reduces the production cost significantly:

Optics Subtotal	$ 8,769.38
Mechanics Subtotal	$ 5,170.00
Electronics Subtotal	$ 4,918.00
Total	$ 18,857.38

A resulting rounded cost of $20,000.00 MXN that is the equivalent to a 10% cost of a “Swamp Optics” FROG instrument.

[17] Price referenced from http://www.swampoptics.com/PDFs/SwampOpticsPriceList.pdf

7. Experimental Results

The final built instrument is:

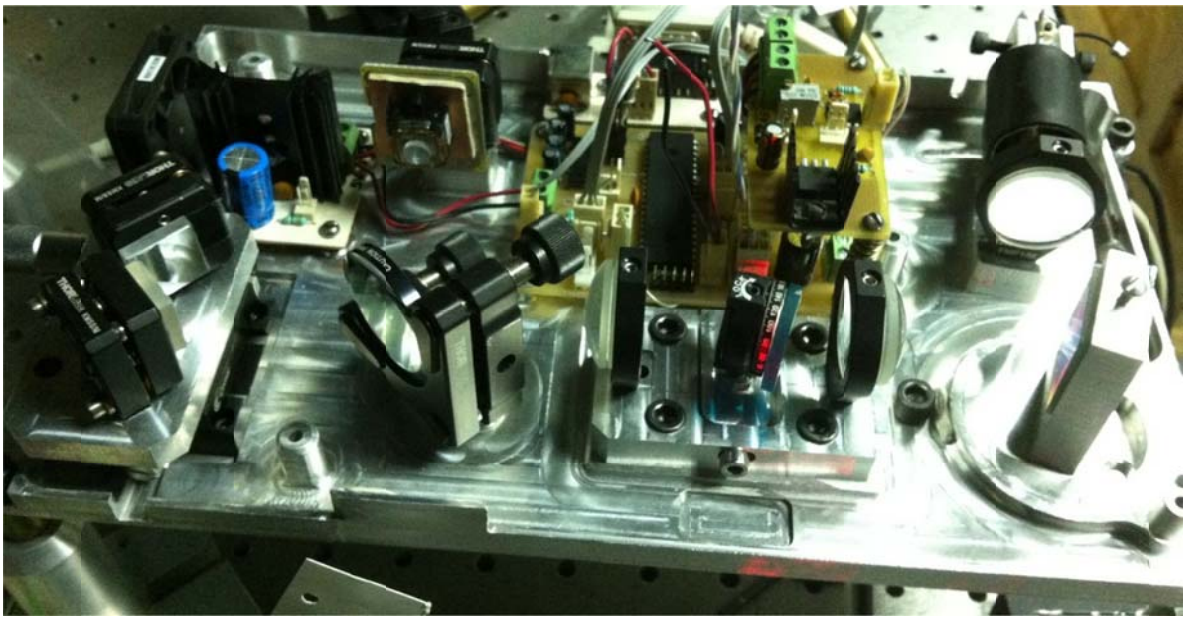


Figure 65: Complete SHG-FROG Instrument.

The different obtained results through the construction process of the instrument show a notorious evolution of the quality and veracity of the generated traces.

- **Retrieved traces considerations**

The first trace acquired (it doesn't mean that it was useful or right) was:

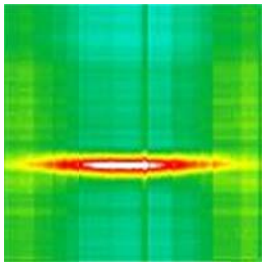


Figure 66: First trace retrieved.

After this, a lot of changes and improvements were applied to the instrument and to the software. There are important factors that must be taken into consideration with every retrieved trace to be considered useful.

- *The analyzed trace must be centered in both axes.*
- *The background of the trace must be cleared.*
- *The trace must be symmetric.*

The FROG algorithm would not be able of recovering traces that have one or more of the above listed problems.

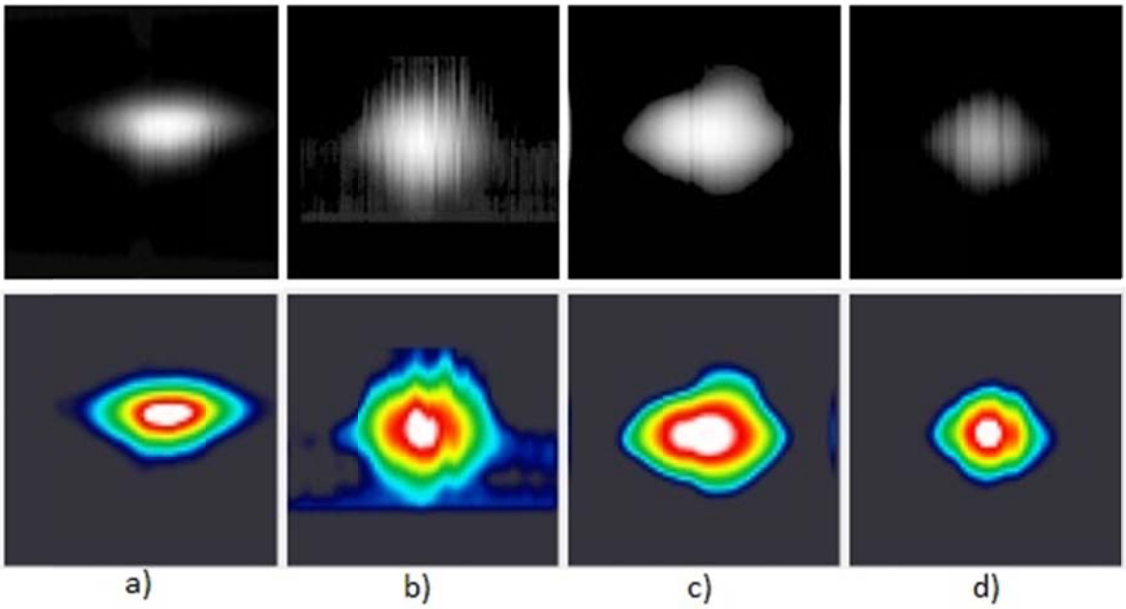


Figure 67: a) Bad centered trace. b) Noisy background trace. c) Not symmetrical trace. d) Bad relation of size trace.

The images in Figure 67 show four same traces, each in grayscale and color with different problems. The “a” trace has the centered problem, the “b” trace has a lot of noise in the background, the “c” trace is not symmetric and the “d” trace may look better than the other ones, but is too big in relation with the complete image.

- **Recovered traces**

The obtained results when a simulated trace is being measured are similar to an acquired trace. For example, the next trace is default in the software:

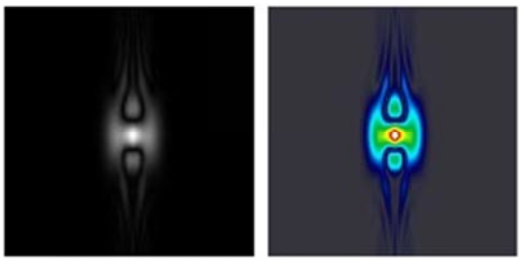


Figure 68: Default simulation trace.

In the simulation the software plots the pulse profile in time as reference:

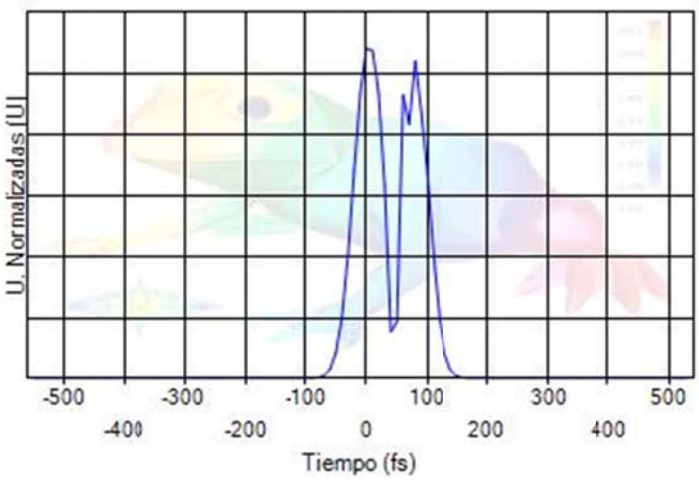


Figure 69: Pulse intensity in time.

The FROG algorithm retrieves:

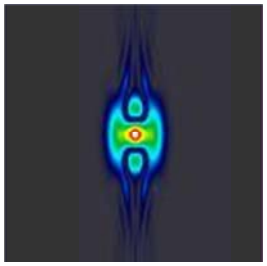


Figure 70: Retrieved trace.

The software calculates an average RMS value of 4.5, an acceptable retrieved trace has this value under an average of 15. The retrieved data is then divided and plotted in time and frequency (intensity and phase each). In time domain the retrieved data is:

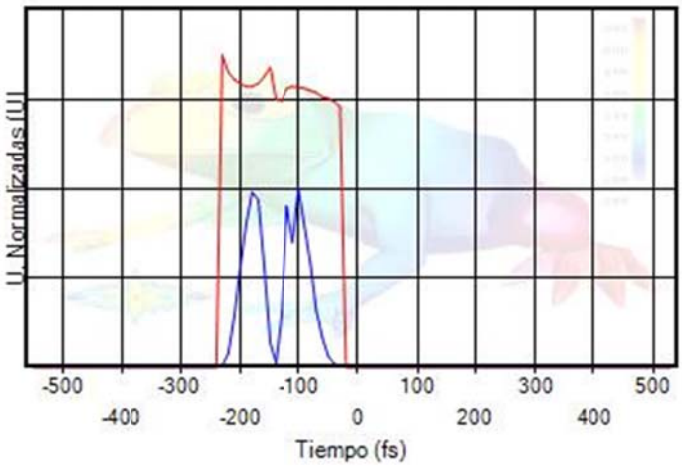


Figure 71: Time domain retrieved data.

In separated graphs the intensity and phase are:

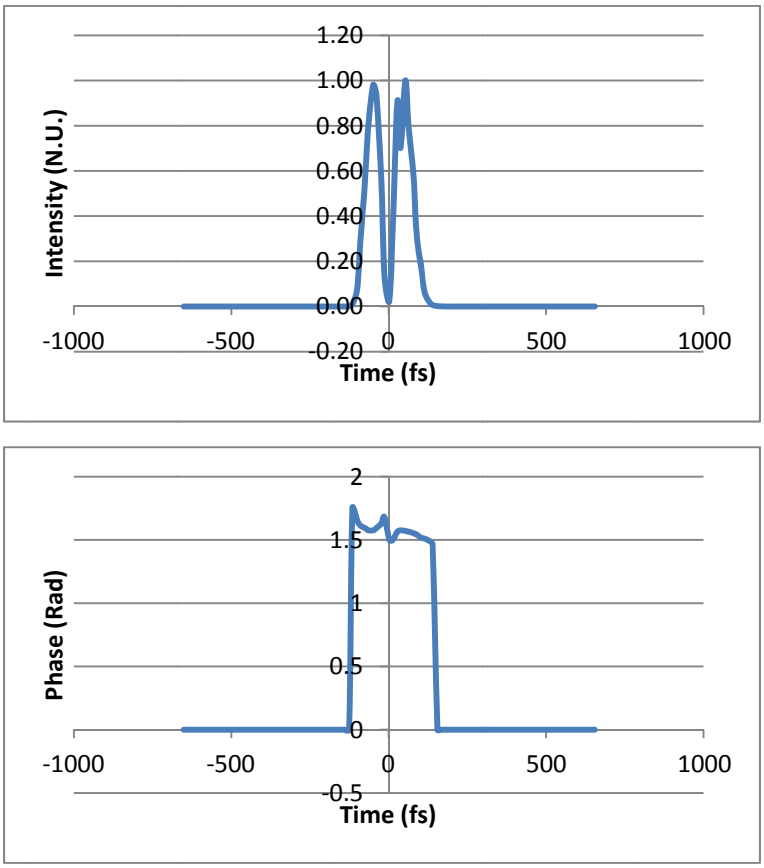


Figure 72: Intensity (Up) and Phase (Down) in Time.

The same simulated pulse in frequency handles:

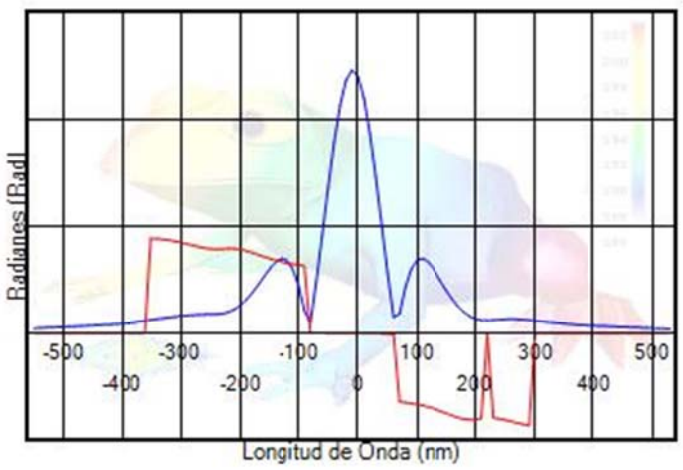


Figure 73: Frequency domain retrieved data.

Again, in separated graphs the intensity and phase are:

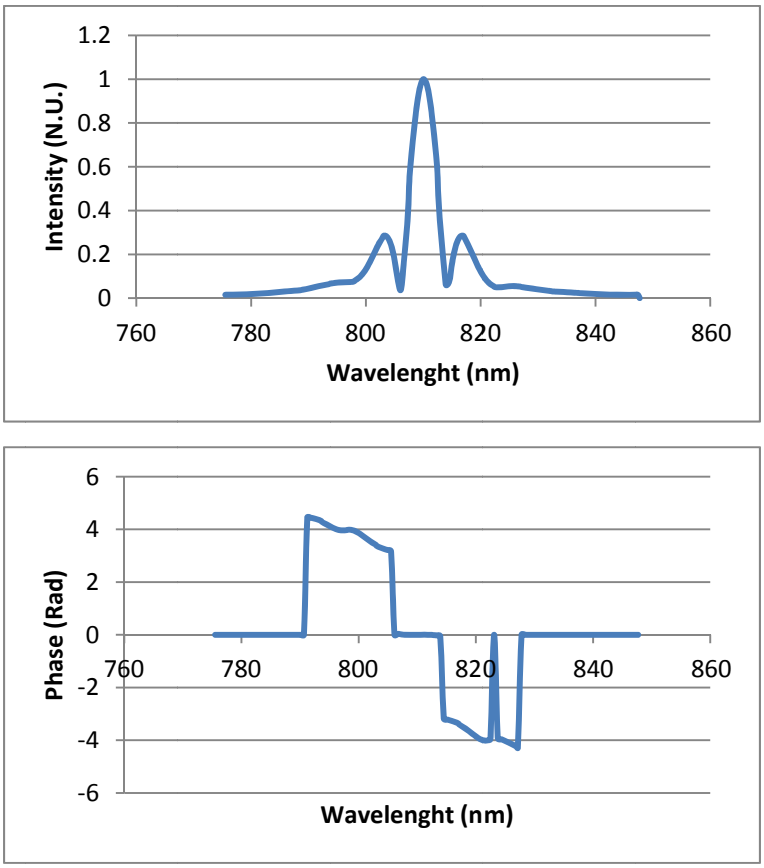


Figure 74: Intensity (Up) and Phase (Down) in Frequency.

With an acquired trace in good conditions, the FROG algorithm is able of recovering the trace and the information of the measured pulse. An example of a good pulse measurement is:

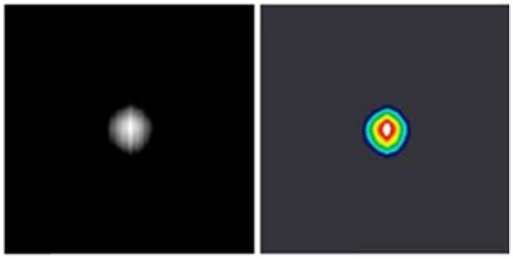


Figure 75: Good condition trace.

Because this is a measured pulse, there is no reference intensity. The FROG algorithm retrieves:

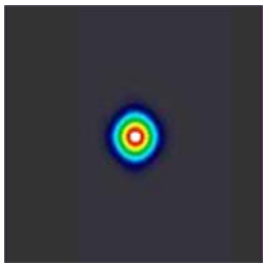


Figure 76: Favorable retrieved trace.

An average RMS value of 9 indicates that the retrieved trace is similar to the acquired one. The In time domain the retrieved pulse looks:

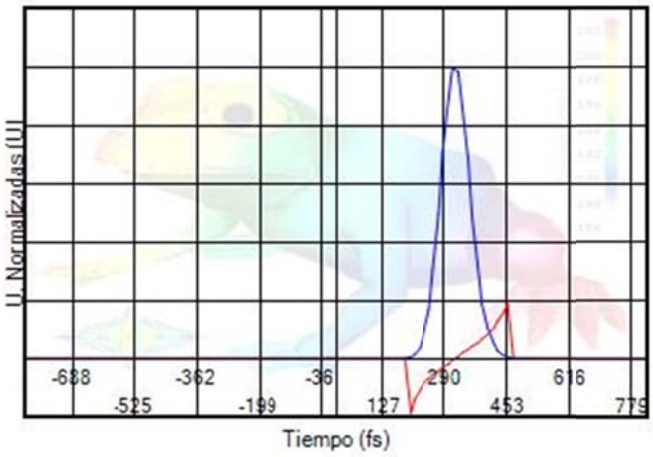
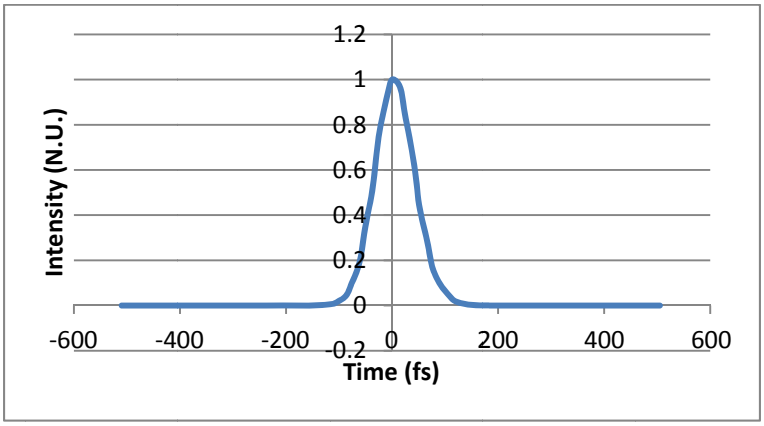


Figure 77: Retrieved pulse in time domain.

When intensity and phase in time domain are plotted separately it is easier to analyze the retrieved data:



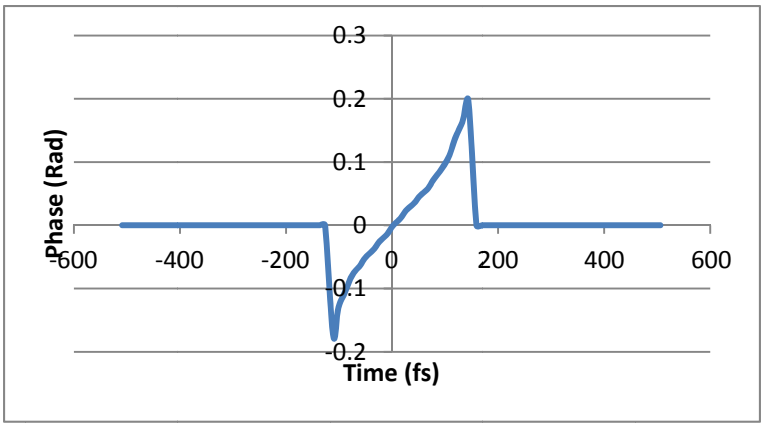


Figure 78: Intensity (Up) and Phase (Down) in Time.

The same retrieved pulse in frequency domain looks:

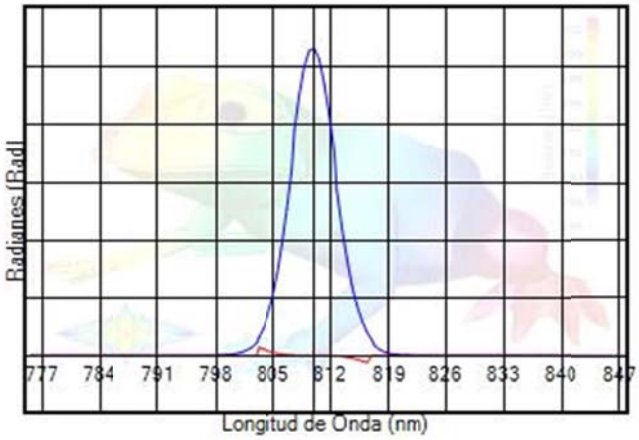
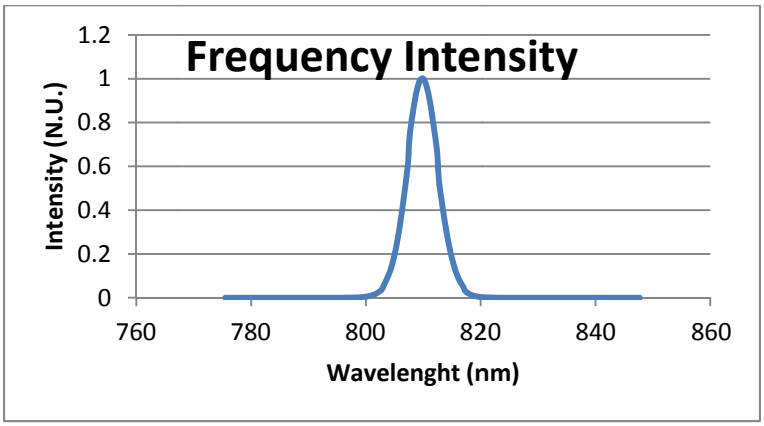


Figure 79: Retrieved pulse in frequency domain.

And when phase and intensity are plotted separately:



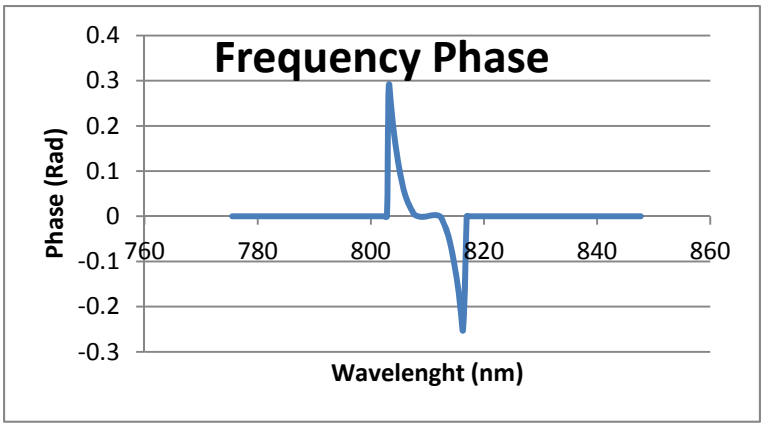


Figure 80: Intensity (Up) and Phase (Down) in Frequency.

With the retrieved data it is possible to analyze the information of the pulse in time and frequency, now it is possible to use the same pulse in other systems and measure changes in the pulse.

8. Conclusions

In the present work it has been developed a Second Harmonic Generation – Frequency Resolved Optical Gating (SHG-FROG) instrument for full characterization of femtosecond laser pulses (amplitude and phase).

The instrument includes the following features:

- Portable “Plug-&-Play” system with communication through serial or USB port.
- Synchronized control between a webcam’s HD CCD and a *Webcam-Focusing-Device* (WFD) working as a delay line.
- Less than 12 Watts power consumption at full operation.
- Includes friendly GUI software.
- The software is completely developed under Visual C# environment.
- Pulse retrieval time from 2 minutes at low resolution to 10 minutes at HD resolution.

The instrument is able to retrieve experimental and simulated traces and estimates the phase and amplitude in time and frequency domain. Retrieved data provides full information to reconstruct the pulse that is being measured. The actual results are in good agreement with technical specifications of a commercial femtosecond laser system (Coherent MIRA 900) and alternative temporal-only characterization systems: a second order autocorrelator (Coherent Femtochrome).

The advantage of developing GUI software is that gives researchers and students the ability to use this instrument without wasting time in first studying the software functionality. The SHG-FROG instrument is easy to calibrate and to install in recent computers (Win-XP, Vista, Win7, Win8 in x32 and x64).

A small analysis of the cost demonstrated that the economic advantages of the designed instrument are notorious over a commercial one. Especially, the easy replication of this instrument gives other users the possibility to apply advance methods of ultrashort pulse retrieval in their own investigations.

The above characteristics cover the objective mission of this work giving the fundamental problem of a small user friendly instrument a solution. The next step will be the perfection and debugging of the system that for sure will lead to new useful instruments.

Annex A – Parallel Resistance Formula

The parallel resistance formula that was used was obtained from the simplification of a simple resistors array:

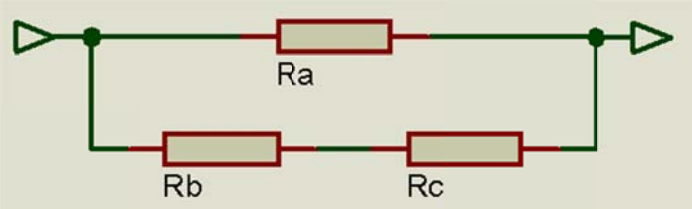


Figure 81: Mixed series-parallel array.

By definition the formula for calculating the equivalent resistance for a series array is:

$$R_{eq} = R_1 + R_2 + R_3 + ... + R_n \qquad \text{<A.1>}$$

And for calculating the equivalent resistance for a parallel array is:

$$R_{eq} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + ... + \frac{1}{R_n}} \qquad \text{<A.2>}$$

So using both equations and simplifying in just one equation it is possible to minimize de expression:

$$R_p = \frac{1}{\frac{1}{R_a} + \frac{1}{(R_b + R_c)}} = \frac{1}{\frac{R_a + R_b + R_c}{R_a (R_b + R_c)}}$$

$$R_p = \frac{R_a (R_b + R_c)}{R_a + R_b + R_c} \qquad \text{<A.3>}$$

Annex B – Temperature Sensor Calibration

The calibration of the sensor was done pretending that the temperature will never reach more than 50° degrees. The temperature sensor is a LM35 centigrade thermometer, it supply's 10 mV per degree and has a maximum range of 150° that is translated in 1.5 volts. The ADC input port of the microcontroller works with 10 bits resolution and has an input range that goes from 0 to 5 volts, so some scaling must be done to use the complete range of the ADC with the limited range of the sensor; the next table shows some values:

Temperature (°C)	Sensor Output [V]	Desired Voltage [V]	Bits Value (10 Bits)
0	0	0	0
5	0.05	0.500	102
10	0.1	1.000	205
15	0.15	1.500	307
20	0.2	2.000	410
25	0.25	2.500	512
30	0.3	3.000	614
35	0.35	3.500	717
40	0.4	4.000	819
45	0.45	4.500	922
50	0.5	5	1024

The graph that describes the sensor output against the desired voltage is:

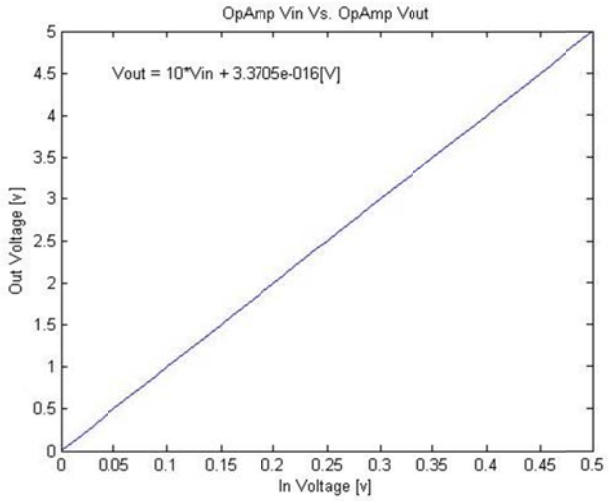


Figure 82: Temperature sensor amplification function.

The graphs equation was obtained by a linear approximation computed by Matlab in the form:

$$y = mx + b \qquad \text{<B.1>}$$

Where $y = V_{out}$, $m = 10$, $x = V_{in}$ and $b = V_{off}$. So the graph equation is:

$$V_{out} = 10V_{in} + 3.3705x10^{-16}\left[V\right] \qquad \text{<B.2>}$$

Since the offset voltage is really small it can be neglected so the scaling equation is defined as:

$$V_{out} = 10V_{in}\left[V\right] \qquad \text{<B.3>}$$

A non-inverting operational amplifier configuration was chosen to do the scaling; the gain of the system is defined by the R_2 and R_1 resistors following the next expression:

$$V_{out} = \left(1 + \frac{R_2}{R_1}\right)V_{in} \qquad \text{<B.4>}$$

So the factor R_2/R_1 plus 1 must be equal to 10 and following equation <B.3> there must be no offset voltage.

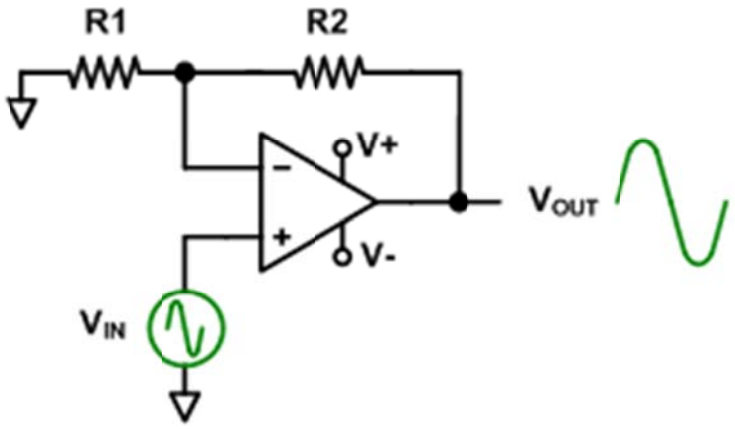


Figure 83: Non-Inverting OpAmp amplifier circuit.

To calculate the gain factor, first the R_2 resistor is defines as $22k\Omega$. So R_1 is defined as:

$$10 = \left(1 + \frac{22000}{R_1}\right) \therefore R_1 = 2444.444... \approx 2.45k\Omega \qquad \text{<B.5>}$$

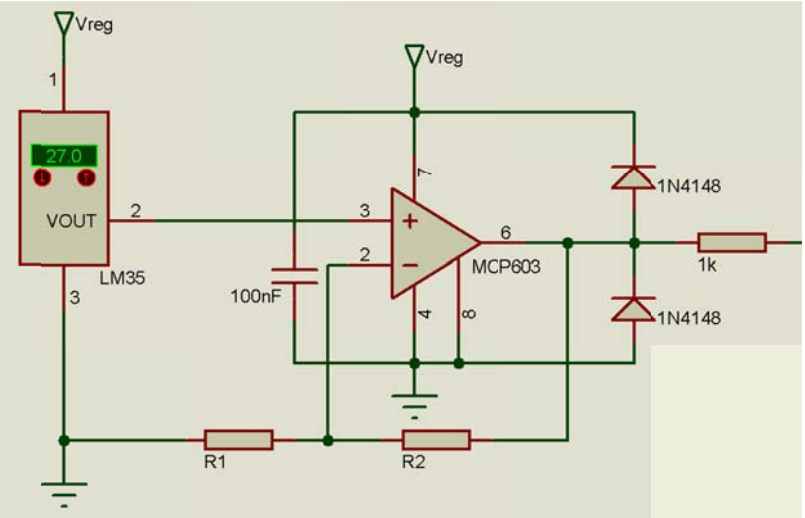
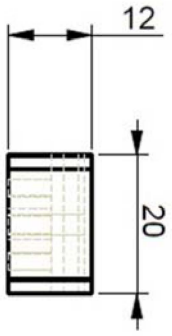
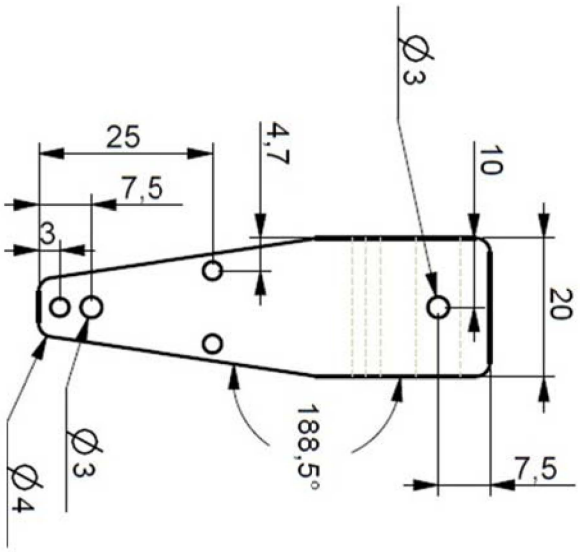


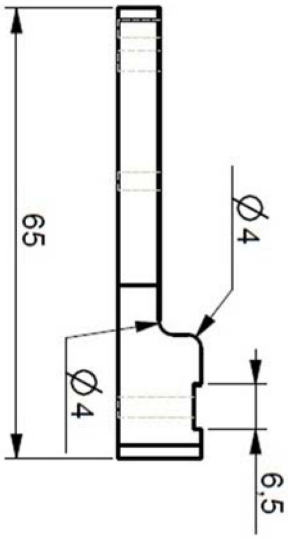
Figure 84: Temperature sensor amplifier circuit.

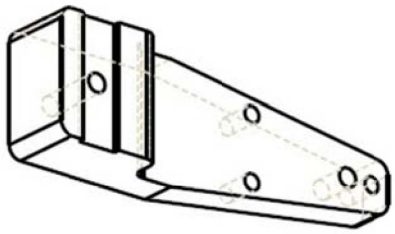
Annex C – CAD/CAM Design Plans

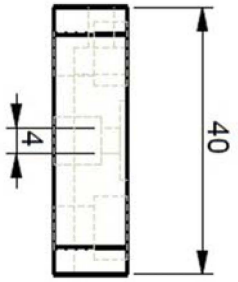
For space reasons, just some plans were printed in this section. To see all the plans refer to the CD included at the end if this work.

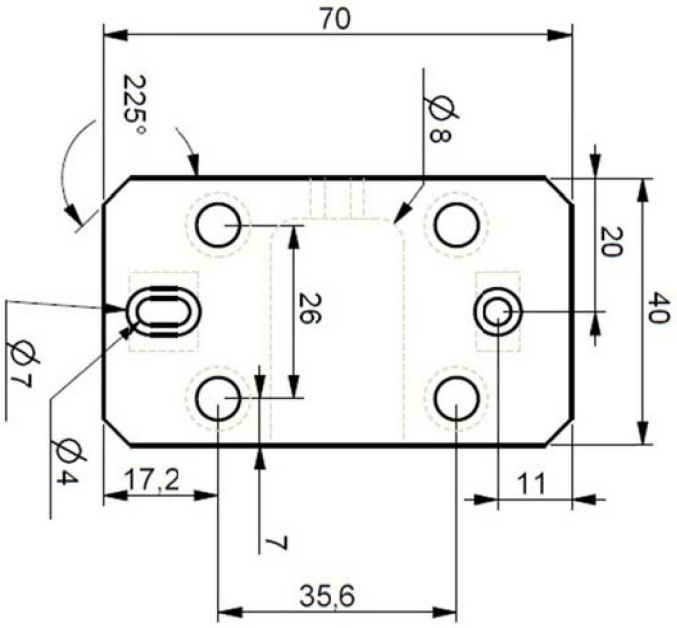


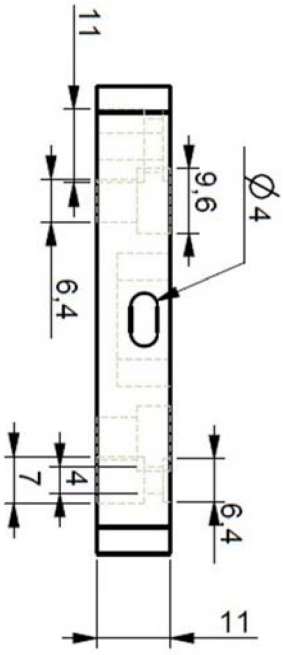


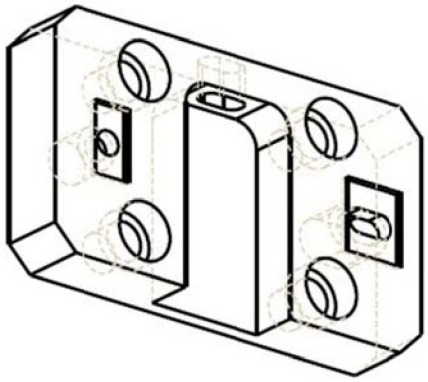


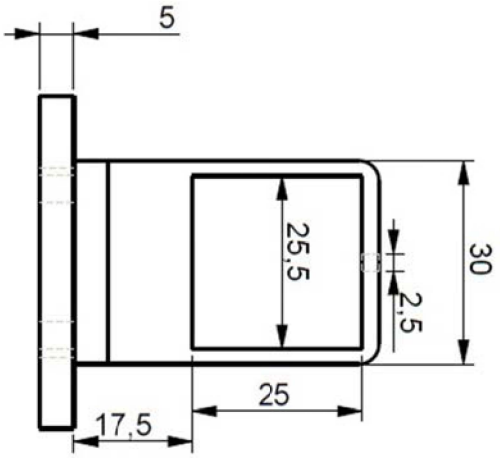


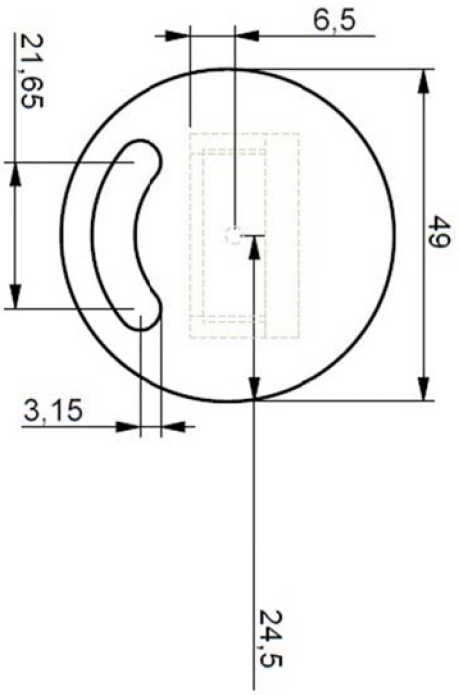

4
40

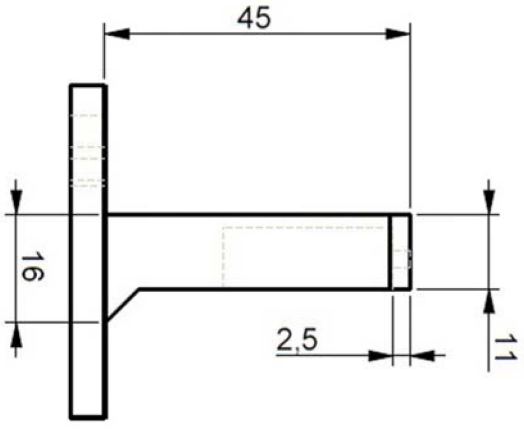

70
225°
Ø8
20
40
26
Ø7
Ø4
17,2
11
7
35,6

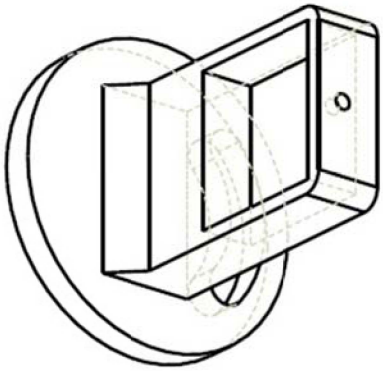

11
9,6
Ø4
6,4
7
4
6,4
11

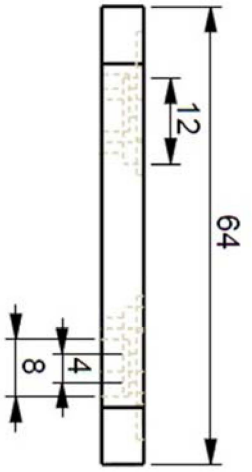


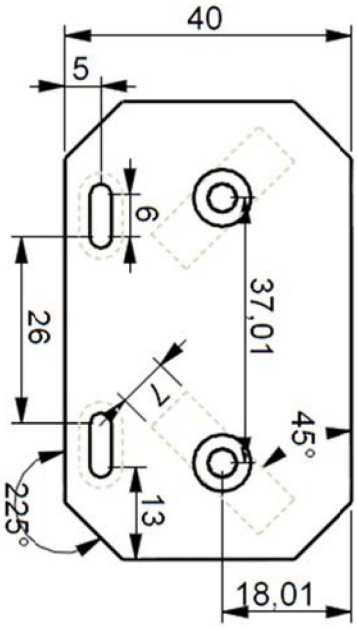

5
25,5
30
2,5
25
17,5

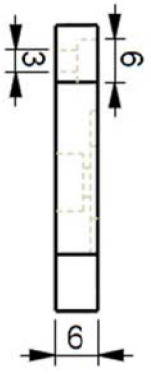

6,5
21,65
49
3,15
24,5

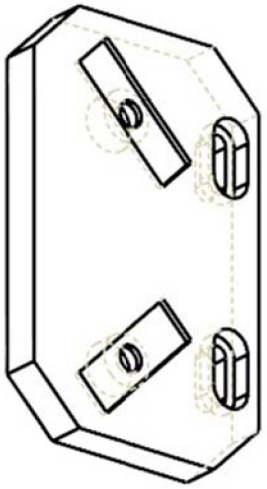

45
16
2,5
11



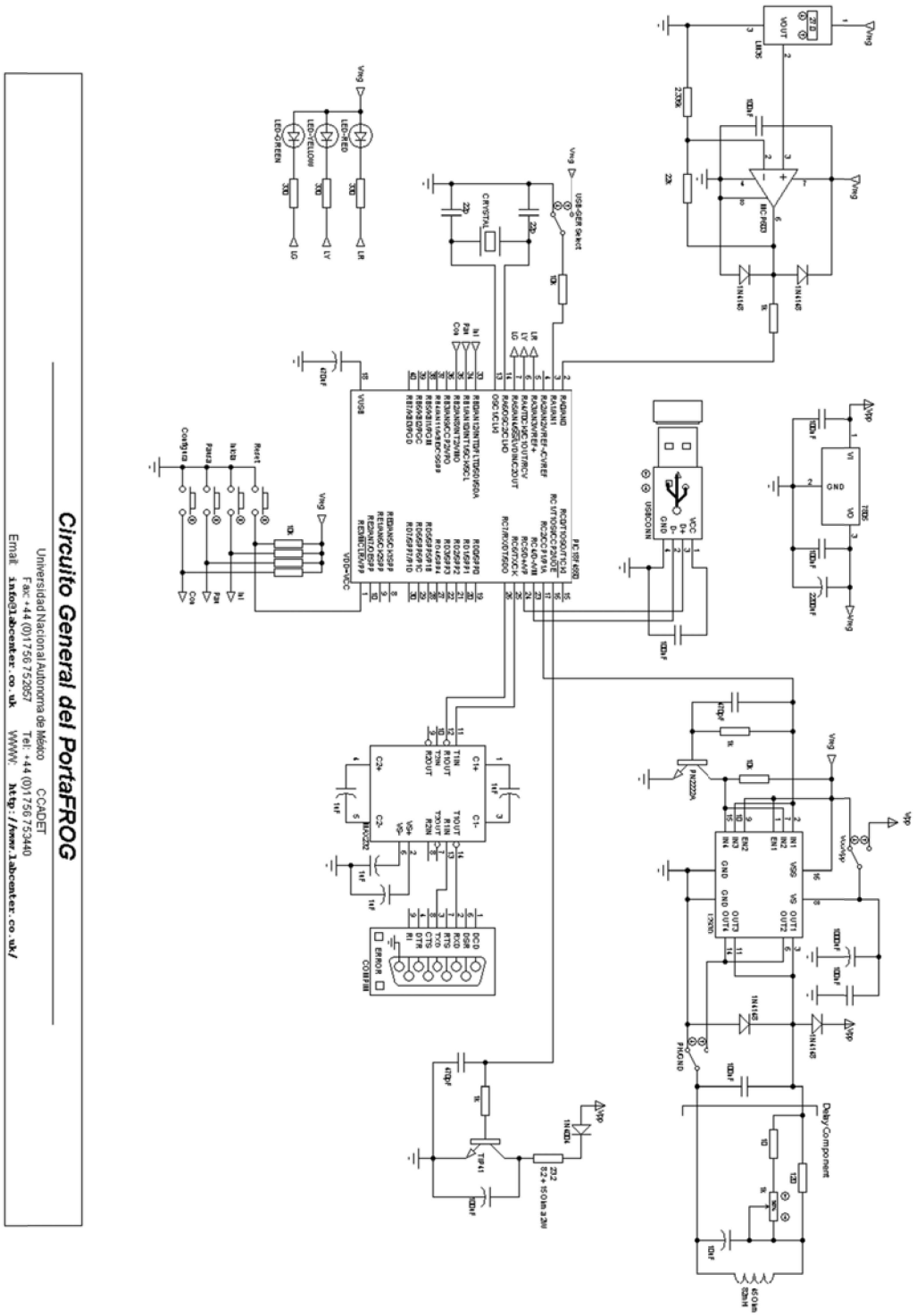

12
64
8
4


40
5
6
37,01
26
7
45°
225°
13
18,01


3
6
6



Annex D – Electronic Diagram



Annex E – PIC & PC Software

For space reasons, just the main methods are printed in this section and the minor methods not shown. To see the complete project software refer to the CD at the end of this work.

- **The main PC program (Fragment)**

```
using System;
using System.IO;
using System.IO.Ports;
using System.Collections.Generic;
using System.Text;
using System.Text.RegularExpressions;
using System.Windows.Forms;
using System.Threading;
using System.ComponentModel;
using System.Drawing;
using System.Drawing.Imaging;
using System.Linq;
using System.Data;
using System.Runtime.InteropServices;
using System.Windows.Forms.DataVisualization.Charting;
using System.Numerics;
using LibUsbDotNet;
using LibUsbDotNet.Main;
using WIA;

namespace Imagen_WCam{
    public partial class Form1 : Form{

        const string archdat = @"..\..\myFile.dat";
        public static UsbDevice usbdev;

        #region SET YOUR USB Vendor and Product ID!
        public static UsbDeviceFinder usbfind = new UsbDeviceFinder(0x04D8, 0x000C);
        #endregion                              // USB Configura

        byte[] binf = { 31, 32, 33, 34 }, bgen = { 0, 0, 0, 0 };
        int var_x = 0, wid = 0, hei = 0, cen = 0, despl = 0, dt_pas = 10, ns, lect = 0,
                pindice = 2, smoothnW = 1, cutpor = 20, t_shc = 35, centropwm = 400,
                tamdex = 2, gral = 0, iteraciones = 30, nmcal = 0, calxoff = 0,
                intercalvx = 50, wc_h = 0, wc_w = 0, wc_f = 0, pcut = 5, vlim = 4,
                atnum = 50, wc_width = 640, wc_height = 480, wc_frame = 20;
        int[,] imag_des, img_c, mtxflt;
        double porcen = .5, poradv = 0, calpix = 0, freqpix = 0;
        double[] pulso, pulso2, tiempo, ejegraft, ejegraff, faseadq, fpulso2, ffaseadq;

        ErrorCode ec;                           // USB error
        public static DateTime LastDataEventDate = DateTime.Now;

        Complex[] et_pasa;
        Boolean ini = false, ser1 = false, ser2 = false, flag=true, wflag=false,
            grfx=false, paut = false, frmlod = true;
        Webcam webcam;
        Image foto;
        Bitmap fotoG, perm, permres, bmp1, bmp2, bmpb;
        ColorMatrix mat;
        ImageAttributes atributo;
        MemoryStream iMemoryStream, iMemoryStream2, iMemoryStream3, iMemoryStream4,
            iMemoryStreamI, imagenMemoryStream, iMemres, memopen, memperm, a ,b;
        SerialPort comport = new SerialPort();
        Metodos_Frog frog;
        FileStream fs;

        static AutoResetEvent autoEvent = new AutoResetEvent(false);

        public Form1(){…}

        ~Form1(){…}

        private void Form1_Load(object sender, EventArgs e){
        private void wbcm_parm() {…}
```

```
        private void archexist(){…}
        private void loadconfi(){…}
        static void WorkMethod(object stateInfo){…}
        private double datadatd(string infos, int lindex){…}
        private int datadati(string infos, int lindex) {…}
        private void datastr(){…}
        public void SetPortNameValues(object obj){…}
        private void FillData(Chart grafica, string serie, double[] info, Color tono, Boolean add){
        private void FillData(Chart grafica, string serie, double[] info, double[] ejes, Color tono, Boolean add,
Boolean zerorss){…}
        private void captu(){…}
        private void boton_retpros(){…}
        private void grafica_temp(){…}
        private void grafica_fase(){…}
        private double check_calidad(){…}
        private void graficas(){…}
        private void reset_all(){…}
        private void gimagenWebcam(){…}
        private void gimagenSimuladaAdquirida(){…}
        private void gimagenGraficaSimulada(){…}
        private void gdatosSimulados(){…}
        private void gimagenRecuperada(){…}
        private void gimagenGraficaRecuperada(){…}
        private string nuevostring(string cambio, string referencia, string anadir) {…}
        private void gdatosRecuperados(){…}

        // Metodos de botones...  ******************************************************

        private void inicia_Click(object sender, EventArgs e){…}
        private void button1_Click(object sender, EventArgs e){…}
        private void stop_Click(object sender, EventArgs e){…}
        private void paus_Click(object sender, EventArgs e){…}
        private void salida_Click(object sender, EventArgs e){…}
        private void ret_pros_Click(object sender, EventArgs e){…}
        private void reset_Click(object sender, EventArgs e){…}
        private void calib_Click(object sender, EventArgs e){…}
        private void specac_Click(object sender, EventArgs e){…}
        private void clidad_Click(object sender, EventArgs e){…}
        private void temp_CheckedChanged(object sender, EventArgs e){…}
        private void freq_CheckedChanged(object sender, EventArgs e){…}
        private void intens_CheckedChanged(object sender, EventArgs e){…}
        private void phase_CheckedChanged(object sender, EventArgs e){…}

        // Metodos de menuStrip...  ****************************************************

        private void resolucionToolStripMenuItem_Click(object sender, EventArgs e) {…}
        private void configuracionAvanzadaToolStripMenuItem_Click(object sender, EventArgs e) {…}
        private void imagenSimuladaAdquiridaToolStripMenuItem_Click(object sender, EventArgs e){…}
        private void imagenGraficaSimuladaToolStripMenuItem_Click(object sender, EventArgs e){…}
        private void datosSimuladosToolStripMenuItem_Click(object sender, EventArgs e){…}
        private void imagenRecuperadaToolStripMenuItem_Click(object sender, EventArgs e){…}
        private void imagenGraficaRecuperadaToolStripMenuItem_Click(object sender, EventArgs e){…}
        private void datosRecuperadosToolStripMenuItem_Click(object sender, EventArgs e){…}
        private void imagenWebcamToolStripMenuItem_Click(object sender, EventArgs e){…}
        private void guardarTodoToolStripMenuItem_Click(object sender, EventArgs e){…}
        private void camaraWebToolStripMenuItem_Click(object sender, EventArgs e){…}
        private void salirToolStripMenuItem_Click(object sender, EventArgs e){…}
        private void cOMInfoToolStripMenuItem_Click(object sender, EventArgs e){…}
        private void informacionFROGReaderToolStripMenuItem_Click(object sender, EventArgs e){…}
        private void limpiarBufferToolStripMenuItem_Click(object sender, EventArgs e){…}
        private void abrirToolStripMenuItem_Click(object sender, EventArgs e){…}
        private void parametrosRetardoToolStripMenuItem_Click(object sender, EventArgs e){…}

        // Metodos de eventos no controlados...  ***************************************

        private static void OnRxEndPointData(object sender, EndpointDataEventArgs e){…}
        private void imagen_LoadCompleted(object sender, AsyncCompletedEventArgs e) {…}
        private void sermax_Tick(object sender, EventArgs e){…}
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e){…}
        private void iterations_TextChanged(object sender, EventArgs e){…}
        private void tamn_SelectedIndexChanged(object sender, EventArgs e){…}
        private void prom_lectura_SelectedIndexChanged(object sender, EventArgs e){…}
        private void comav_SelectedIndexChanged(object sender, EventArgs e){…}
        private void time_step_Leave(object sender, EventArgs e) {…}
        private void randguess_CheckedChanged(object sender, EventArgs e){…}
        private void w_cam_CheckedChanged(object sender, EventArgs e){…}
        private void sintetic_CheckedChanged(object sender, EventArgs e){…}
        private void KeyPress_sinp(object sender, KeyPressEventArgs e){…}
        public void comport_DataReceived(object sender, SerialDataReceivedEventArgs e){…}
```

```
        // Metodos para coneccion entre clase Metodos_frog y las Formas, envío y recepcion de propiedades...

        public int nsize{…}
        public int[,] mtxcus{…}
        public string time_steptext{…}
        public Boolean randguessval{…}
        public Bitmap pulso_img_a{…}
        public Complex[] pulso_cplx{…}
        public double calibrapix{…}
        public double[] pulso_dot{…}
        public double[] ejes_grafy{…}
    }
}
```

- **The main microcontroller program**

```
#include "RetCon.h"

#INT_TIMER1
void tmr1_int(){
}

#INT_EXT                        //Instrucción de interrupción externa Int0
void boton_ini(){
}

#INT_EXT1                       // Instrucción de interrupción externa Int1
void boton_pausa(){
}

#INT_EXT2                       // Instrucción de interrupción externa Int2
void boton_config(){
}

#INT_RDA
void inter_serial(){
}

void main(void){

        led_act(1);                             // Prendemos LED Rojo...

        setup_adc_ports(AN0_TO_AN2);
        setup_adc(ADC_CLOCK_INTERNAL);          // Utilizamos el CLK interno...
        set_adc_channel(0);                     // Usamos el AN0...

        set_tris_b(0xF7);
        setup_comparator(NC_NC_NC_NC);
        setup_vref(FALSE);
        setup_ccp1(CCP_PWM);                    // Configuramos el CCP1 como PWM...
        setup_ccp2(CCP_PWM);                    // Configuramos el CCP2 como PWM...

        setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
        setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);         // Timer 1 como base de interrupción para medir temperatura...

        setup_timer_2(T2_DIV_BY_1, 255, 8);     // Timer de frecuencia del PWM...

        Ext_int_edge(0,L_TO_H);                 // Configuramos las interrupciones externas...
        Ext_int_edge(1,L_TO_H);                 // de bajo hacia alto...
```

```
        Ext_int_edge(2,L_TO_H);
        enable_interrupts(INT_EXT);          // Activamos la interrupción externa Int0...
        enable_interrupts(INT_EXT1);         // Activamos la interrupción externa Int1...
        enable_interrupts(INT_EXT2);         // Activamos la interrupción externa Int2...
        enable_interrupts(INT_TIMER1);       // Activamos la interrupción del Timer1...
        enable_interrupts(INT_RDA);          // Activamos la interrupción del puerto Serial...

// ***** Inicia actividad *****
recon:
        if(input(USB_SER)){
                comm=true;                   // Cambio a USB...
        }else{
                comm=false;                  // Default: Com Serial...
        }
        r_eeprom();                          // Se carga el centro del PWM anterior...
        op = 200;                            // Opción de decisión invalida...
        flag = false;                        // Todo en standy-by...
        conect = false;                      // Se especifica un fallo de conexión...
        pwm_uno=true;
        pwmtemp=temp_ini;                    // Se inicializa la temperatura...
        set_pwm2_duty(pwmtemp);              // Iniciamos el control de temperatura...
        pwmval = centro;
        pospwm();                            // Iniciamos PWM...
        conexion();                          // Llamamos al método conexión hasta que se conecte el serial...
        enable_interrupts(GLOBAL);           // Activamos las interrupciones...
        delay_ms(250);                       // Espera de 250ms...
        configura();                         // Método de configuración de parámetros...
        delay_ms(250);                       // Espera de 250ms...

        while(true){
                if(flag==true){
                        led_act(3);                          // LED Verde prendido...
                        if(pwm_uno==true){
                                pwmval = centro - limit;
                                pwm_uno = false;
                                primrun = true;
                        }
                        if(primrun == true){                 // Si es la primer movida...
                                pwmval = centro - limit;
                                pospwm();                    // Movemos el DC del PWM...
                                primrun = false;
                                delay_ms(500);               // Dale tiempo de mover...
                        }
                        if(pwm_cont==pwm_flag){              // La cantidad de veces que espera antes de desplazarse...
                                pospwm();                    // Movemos el DC del PWM...
                                pwm_cont=0;                  // Reiniciamos conteo de movimiento...
                        }
                        pwm_cont++;                          // Sumamos 1 al contador...
                        ser_acnl();                          // Envía confirmación de proseguir...
                }else{
                        led_act(2);                          // LED Amarillo prendido...
                        if(op==10){                          // Solo que se haya pedido un reset...
                                goto recon;                  // Brinca al inicio de todo...
                        }
                }
        }
}
```

Annex F – Software Calibration Method

As mentioned before, the calibration requires specific data to work. Three of these parameters depend directly on the laser and the pulse generated:

- *The relation of femtoseconds per pixel.*
- *The relation of nanometers per pixel.*
- *The central fundamental wavelength of the measured pulse.*

The relation femtoseconds/pixel and nanometers/pixel can be acquired with the calculator included in the calibration window (Figure 85), it can be accessed with the "Calcular" button (See Figure 56). The fundamental wavelength depends on the used laser and is a direct parameter.

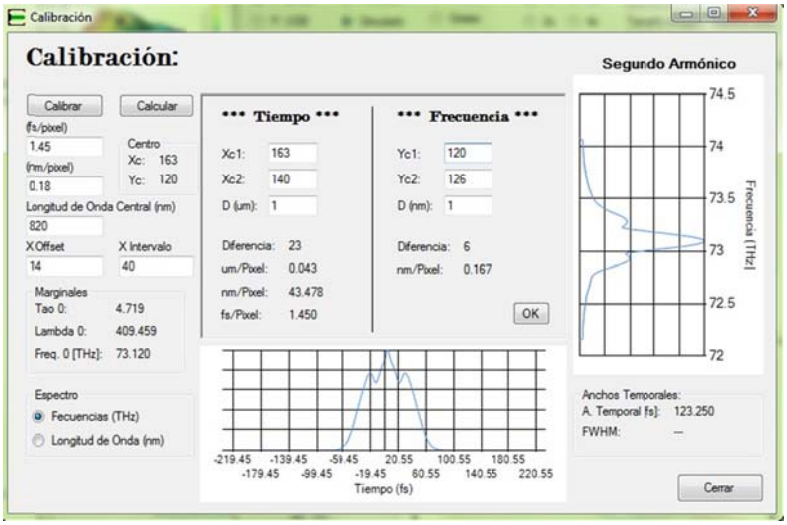


Figure 85: Calculator of the calibration window.

Both relations (femtoseconds/pixel and nanometers/pixel) are obtained by the calculator, the user only needs to introduce a change in distance (when calibrating time) or a change in wavelength (when calibrating frequency) and a change in the center coordinates in pixels.

- **Time Delay Calibration**

To calibrate the time axis a horizontal change in the position of the trace must be generated. To do this, the next method is implemented:

1. *A trace is acquired.*
2. *The pixel* Xc_1 *of the center o mass of the trace is captured.*
3. *The fixed delay of the interferometer is moved a small distance* Δd *.*
4. *Another trace is then acquired.*
5. *The new pixel* Xc_2 *of the center of mass is captured.*
6. *Pixels* Xc_1 *and* Xc_2 *and the distance* Δd *are introduced in the calculator for computing.*

The center of mass position in pixels is automatically calculated by the calibration window. The result is given in femtoseconds/pixel (fs/Pixel) with some other extra parameters.

Figure 86: Calculator window.

- **Frequency Calibration**

The calibration method for the frequency axis is almost the same as in time, but the factor that changes is the fundamental wavelength of the pulse:

1. *A trace is acquired.*
2. *The pixel* Yc_1 *of the center o mass of the trace is captured.*
3. *The fundamental center wavelength of the laser is slightly shifted.*
4. *Another trace is then acquired.*
5. *The new pixel* Yc_2 *of the center of mass is captured.*
6. *Pixels* Yc_1 *and* Yc_2 *are introduced in the calculator for computing.*

The result is directly given in nanometers/pixel (nm/Pixel) (See Figure 86).

References:

Books:

- García B. E., (2008). *Compilador C CCS y simulador PROTEUS para Microcontroladores PIC*. México, Alfaomega.
- Oda N. B., (2009). *Introducción al análisis grafico de datos experimentales*. México, UNAM, La prensa de Ciencias.
- Sinclair I. R. & Dunton J., (2010). *Practical electronics handbook* (6th ed.). Kidlington, Oxford, Newnes.
- Silfvast W. T., (2004). *Laser Fundamentals*, (2nd ed.). Cambridge.
- Trebino R., (2000). *Frequency-Resolved Optical Gating: The Measurement of Ultrashort Laser Pulses*. USA, Atlanta, KAP.

Articles:

- D. T. Reid, C. McGowan, W. E. Sleat, and W. Sibbett, Appl. Opt. 36, 9103 (1997).
- D. J. Kane, IEEE J. Sel. Top. Quantum Electron. 4, 278 (1998).
- P. O'Shea and R. Trebino, "*Extremely simple intensity and phase ultrashort pulse measurement device with no spectrometer, thin crystal or delay line*," Conference on Laser and Electro-Optics, OSA Technical Digest (Optical Society of America, Washington DC, 2000), pp. 587-588.
- K. W. DeLong, D. N. Fittinghoff, and R. Trebino, IEEE J. Quantum Electron. 32, 1253 (1996).
- D. J. Kane, G. Rodriguez, A. J. Taylor, and T. S. Clement, J. Opt. Soc. Am. B 14, 935 (1997).
- J. Garduño-Mejía, E. Ramsey, A. Greenaway, and D. T. Reid, "*Real time femtosecond optical pulse measurement using a video-rate frequency-resolved optical gating system*" Ultrafast Optics Group, School of Engineering and Physical Sciences (Physics). Heriot-Watt University, Riccarton, Edimburgh EH14 4AS, United Kingdom (2003).
- R. Trebino, K. W. DeLong, D. N. Fittinghoff, J. N. Sweetser, M. A. Krumbügel, and D. J. Kane, "Measuring Ultrashort Laser Pulses in the Time-Frequency Domain Using Frequency-Resolved Optical Gating," Review of Scientific Instruments 68, 3277-3295 (1997).

Internet:

http://en.wikipedia.org/wiki/Frequency-resolved_optical_gating
http://www.ee.nthu.edu.tw/~sdyang/Files/C012_2005_CLEO_FROG.pdf
http://www.icfo.eu/images/publications/MT04-001.pdf
http://msdn.microsoft.com/es-mx/ms348103.aspx
http://en.wikipedia.org/wiki/Grayscale
http://www.switchonthecode.com/tutorials/csharp-tutorial-convert-a-color-image-to-grayscale
http://www.swampoptics.com/index.htm