Miguel Hernandez University
Electronic Engineering and Industrial Automation

Bachelor Thesis Spring 2021

Graphical User Interface for the control of an experimental ultrasonic device

Authors

M. Burak KIRTAY

Supervisor

Dr. Alberto Rodriguez Martinez

Abstract

In the 21st century, various devices are used to view the organs and vessels in our body, like sonography, tomography and infrared spectroscopy. Yet, these devices are much complex and more expensive. This work is part of a research project devoted to the development a less complex and cheaper system that can detect the location and thickness of blood vessels or organs in the human body with ultrasound. In the project, an experimental ultrasonic device, including a pulser and acquisition device, will be used in order to produce the electrical signal and detect the ultrasound.

The main objective of this thesis is to design and implement a graphical interface in order to control the device, design and select the excitation signal and analyze the ultrasound waves. The program is developed in Python, and different python libraries will be used in order to create the graphical user interface (GUI). This study could be useful to both practitioners and academics. The study shares information and knowledge of ultrasound electronic devices and the software (Python) used for the analysis of the signals. The information about the excitation and acquisition of ultrasonic signals via software will be of special interest.

Key Words: *Ultrasound, Analyzing the Signals, Detecting Objects, Software, Python*

Resumen

En el siglo XXI, se utilizan varios dispositivos para ver los órganos y vasos de nuestro cuerpo, como la ecografía, la tomografía y la espectroscopia de infrarrojos. Sin embargo, estos dispositivos son mucho más complejos y costosos. El objetivo del proyecto es desarrollar un sistema menos complejo y más económico que pueda detectar la ubicación y el grosor de los vasos sanguíneos u órganos en el cuerpo humano con ultrasonido. En el proyecto, se utilizará un transductor y un dispositivo llamado Sedaq para producir la señal eléctrica y detectar el ultrasonido.

Además, la idea principal del informe es producir un programa de interfaz gráfica para controlar Sedaq y el transductor y analizar las ondas de ultrasonido. Se utilizarán un par de bibliotecas Python para crear un programa de interfaz gráfica (GUI). Este estudio podría ser útil tanto para profesionales como para académicos. El estudio comparte información y conocimiento de los dispositivos electrónicos de ultrasonido y el software (Python) que se utiliza para analizar las señales. Especialmente la información sobre el análisis de señales ultrasónicas a través del software se puede encontrar más en este estudio.

Palabras Clave: Ultrasonido, Análisis de señales, Detección de objetos, Software, Python

Acknowledgment

I will be forever grateful to my supervisor Alberto Rodriguez Martinez for all valuable advice during the writing of this dissertation and studying. Moreover, I would like to thanks him for his continued support and allowing me to take a part in the project that he has been working on for years.

Additionally, special thanks go to Miguel Hernandez University for giving me an opportunity to study my final thesis at UMH and permission to using laboratory and its items.

I have been studying at Miguel Hernandez University for 5 months and I am grateful for every single day.

Thank you

Burak KIRTAY,2021

Table of Contents

1.Introduction

The variation in the physical characteristics of blood vessels, as well as the flow through them, provides numerous and valuable information on the vascular status of people. The diameter of the vessels such as thickness and elasticity of their vessel walls, the arterial pressure, and the pulsatility of the tissues, to name a few examples, are closely related to a large number of pathologies that affect the cardiovascular system. Of special relevance is the study of the cerebrovascular system, whose hemodynamics is closely associated with numerous pathologies, not only because it is indicative of the complete system, since the brain is an organ directly fed by the central vascular system, but also because of the correct functioning of the cerebrovascular system profoundly affects its state of health, by regulating the contribution to the brain of oxygen, glucose and other nutrients, transporting waste metabolites, regulating brain pressure and temperature, etc. On the other hand, knowing the state of certain parameters associated with the cerebrovascular system, opens a window to the diagnosis of pathologies associated with the state of the same or its aging, such as inflammatory processes associated with migraine, the reduction of pulsatility due to degenerative diseases such as Alzheimer's or senile dementia, or changes in hemodynamics due to stroke or other cerebrovascular accidents, to name a few of the examples that are attracting the most attention today.

From all of the above it is clear, and it is well known in the neurological field, that continuous monitoring of changes, however subtle, in the parameters associated with the cerebrovascular system, would provide a significant advantage in the diagnosis and prognosis of diseases cardiovascular, cerebrovascular and those associated with the aging of brain tissue or other vital organs, in addition to helping to understand the functioning of these systems, organs and / or tissues, by providing additional information that until now was not available to medical researchers, either due to not having the necessary technology, either because of its high cost, or because of the practical difficulty that continuous monitoring requires.

The measurements and signals required to quantify the parameters and magnitudes described above are traditionally acquired using techniques based on magnetic resonance (2D / 3D PC-MRI, Phase-Constrast Magnetic Resonance Imaging), tomography (PET, Positron Emission Tomography, and SPECT, Single-Photon Emission Computed Tomography), and infrared spectroscopy (fNIRS, functional Near Infra-Red Spectroscopy). Since all of them require expensive equipment, large infrastructures and, above all, immobilization of the patient, they are inappropriate for the purpose of having continuous measurements over time. The only viable non-invasive alternatives to acquire the signals are therefore techniques based on external mechanical or optical sensors, such as ultrasound, photoplethysmography and tensiometer, since they are less expensive, easy to use and require devices that can be portable. The techniques used that would be useful to acquire signals from the cerebrovascular system under these conditions are the following:

• Photoplethysmography, provides detailed information on blood pressure, but at a shallow depth (<8 mm), and also does not provide information on the dynamics of the vessels independently.

• Tonometry, which requires the use of tension sensors, but its reliability in motion is very low and only provides blood pressure measurements.
• Ultrasound, used to estimate the dimensions and motility of the vessels by analyzing the ultrasound images, as well as the pulsatility of the brain tissue, which is capable of penetrating the tissues and obtaining high-resolution images, but it is very sensitive to artifacts due to movement, in addition to the fact that the probes are large, cumbersome, and require constant manipulation to achieve proper coupling. In addition, to achieve correct measurements, the tissues must be pressed, which modifies their behavior and that of the flow of the vessels that pass through them.
• Transcranial Doppler Ultrasonography (TCD-U), which allows the blood flow velocities in the anterior, middle and posterior cerebral arteries to be recorded, but does not allow measuring the dimensions of the vessels or their dynamics and, as in in the case of echoscopy, for pulsatility measurements they require very specific conditions because the transmission windows at their working frequencies are very small.
• Elastography, which allows the analysis of the state of the vessels and their walls, but which cannot be used in motion since the transducers have to be fixed and is very sensitive to noise and artifacts, like other ultrasound-based techniques.

Of the previous techniques, only those based on ultrasound are capable of providing the desired measurements, but due to the problems associated with the coupling and maneuverability of the sensors, as well as the fact that, although relatively manageable, they require the intervention of a specialist in constant, in addition to immobilization of the subject, are not suitable for continuous monitoring of variables. Although it is true that there are some developments of portable autonomous measurement systems, these use a single large transducer, attached to the body by means of clamps, which can only be located in very specific areas (flat and with little movement), which require gels. coupling, and which remain highly sensitive to motion artifacts, as well as offering neither the sensitivity nor the required axial resolution.

From all the above, it follows that it would be desirable to have an ultrasonic device capable of adapting to the skin in different positions of the body and of reduced dimensions, so that it does not interfere with the normal activity of the subject under study, whose acoustic properties are adapted to those of the skin, that generates signals that can penetrate into the tissues until reaching the areas and vessels of interest with sufficient sensitivity and axial resolution, that allows directing the ultrasonic beam towards the area of interest, and that can record the signals autonomously and continuously over time. Despite the challenge that all these demands pose, there is already evidence that it is possible, as some successful cases of devices with characteristics similar to the specifications described above have been reported. In this work, the authors have been able to measure the pulse and blood pressure in different vessels (carotid, brachial and radial arteries).

According to all information below, the aim of the project is to design and build a system that can measure the thickness and location of the vessels on the body. Most important feature of the system is that it easy to use and produce. Yet, there is a lot of research and experimentation that needs to be done to produce this system. Therefore, the first task will be to assess the possibility of acquiring the desired signals with a single transducer. For this, we will have to perform a series of experiments with

different transducers and measurement setups. One of the most important parameters of the analysis is the excitation signal. Both the bandwidth and the waveform are of interest. In order to study these parameters, a special equipment is needed, as it has to be able to produce any digital (rectangular) waveform, with high gain and sensitivity, and of small size to be portable.
In the laboratory, the research group is using an advanced and programable equipment designed specifically for this project, but unfortunately, the user interface and the control software provided with it is not of much use due to its poor usability, and because it lacks functionality.

The objective of this project is therefore to design and program a graphical user interface that allow the use of this device for the analysis of different excitation signals in real time.

2. The Purpose and Goal of the Project

The general aim for the project is that reduce the complexity and cost of the devices like sonography, tomography and infrared spectroscopy in order to measure the parameters such as thickness of vessel. These devices useful to views organs in the body, yet it is much more beneficial to use a small and less expensive device that can show the thickness and location of the vessels. However, there is a lot of experimentation that needs to be done to achieve this goal. For this reason, it is necessary to divide the purpose of the project into several purposes. And the first goal led us to do this project. The aim of our project is to create a program where we can control the electronic equipment we use in the project and also analyze the signals we receive.

As it mentioned before, first of all, to achieve this goal, we must focus and achieve the goal of the project we are doing now. The reason why is the first task should be the one that control ultrasonic transducer and analyzing the signals we receive. In order to do that the program a graphical user interface should be done.

To realize the first task of our project, we first need to investigate how the electronic devices we will use in the project are used and how we can make the program to control these electronic devices. While doing these researches, we should pay attention to parameters such as the convenience of the program we will use and the cheap and easy availability of our electronic devices in the market.

3. The Method of Project

In this part of the project, the research results about what the construction method of the project should be are included. First of all, we need to divide the method by two different field. One of them is the electronic field, and the second is software field. The both fields need to be research very well in order to build a good structure of project. Additionally, since the project is based on the ultrasonic signals, information about how ultrasonic signals works and their history should be investigated.

Yet, the first thing to do should be the research of the electronic equipment.

A. Electronic

In the electronic part, two electronic devices are very important to build the project. One of them is ultrasonic transducer and the other is the device that we are calling pulser/receiver.

- **Ultrasonic Transducer**

Figure 1: Ultrasound Transducer

An ultrasonic transducer is a device used to convert some other type of energy into an ultrasonic vibration. There are several basic types, classified by the energy source and by the medium into which the waves are being generated. Mechanical devices include gas-driven, or pneumatic, transducers such as whistles as well as liquid-driven transducers such as hydrodynamic oscillators and vibrating blades. These devices, limited to low ultrasonic frequencies, have a number of industrial applications, including drying, ultrasonic cleaning, and injection of fuel oil into burners. Electromechanical transducers are far more versatile and include piezoelectric and magnetostrictive devices. A magnetostrictive transducer makes use of a type of magnetic material in which an applied oscillating magnetic field squeezes the atoms of the material together, creating a periodic change in the length of the material and thus producing a high-frequency mechanical vibration. Magnetostrictive transducers are used

primarily in the lower frequency ranges and are common in ultrasonic cleaners and ultrasonic machining applications. (1)

By far the most popular and versatile type of ultrasonic transducer is the piezoelectric crystal, which converts an oscillating electric field applied to the crystal into a mechanical vibration. Piezoelectric crystals include quartz, Rochelle salt, and certain types of ceramic. Piezoelectric transducers are readily employed over the entire frequency range and at all output levels. Particular shapes can be chosen for particular applications. For example, a disc shape provides a plane ultrasonic wave, while curving the radiating surface in a slightly concave or bowl shape creates an ultrasonic wave that will focus at a specific point. (1)

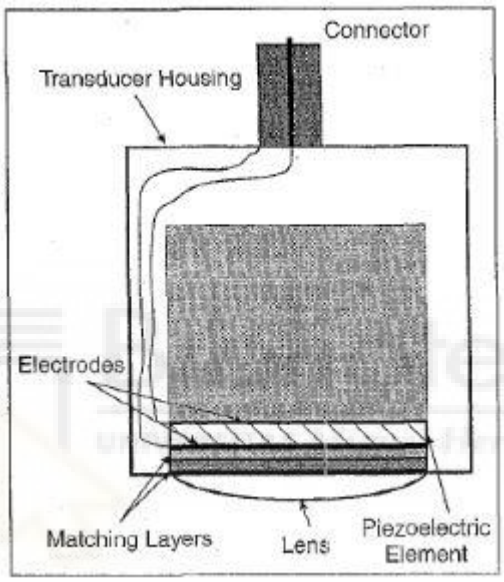

Figure 2: Construction of a single element transducer (2)

As shown in Fig. 2, the most important component of such a device is the piezoelectric element. A number of factors are involved in choosing a proper piezoelectric material for transmitting and/or receiving the ultrasonic wave, including stability, piezoelectric properties, and the strength of the material. The surfaces of the element are electrode with fired-on silver or sputtered chrome-gold. The outside electrode is usually grounded to protect the patients from electrical shock. The housing can be metallic or plastic, and an acoustic isolating material can be placed between the piezoelectric element and the housing to prevent ringing of the housing that follows the vibration of the piezoelectric element itself. (2)

After the explanation of the Ultrasonic Transducers (UT), the more important question is what is the UT working principle?
When we applied electrical signal to this transducer, it vibrates around the specific frequency range and generates a sound wave. These sound waves travel and whenever any obstacle comes, these sound waves will reflect the transducer inform of echo. And at the end of the transducer, this echo converts into an electrical signal.

Additionally, the transducer calculates the time interval between the sending of the sound wave to the receiving the echo signal. The ultrasonic sensor sends the ultrasonic pulse at 40 kHz which travels through the air. These transducers are better than the infrared sensors because these ultrasonic transducer/sensors are not affected by the smoke, black materials, etc. Ultrasonic sensors exhibit excellence in suppressing background interference. (3)

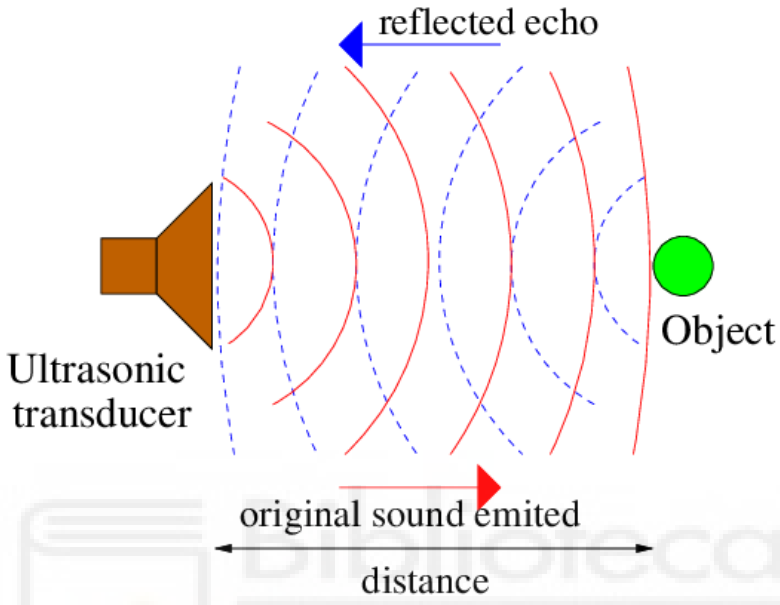

Figure 3: Ultrasonic Transducer Working Principle (Figure 2)

Moreover, ultrasonic transducers are mainly used for finding the distance by using ultrasonic waves. The distance can be measured by the following formula:

$$\text{Distance} = 1/2 \cdot t \cdot C$$

t = Indicates the time difference between sending and reception of ultrasonic waves.
C = The speed of sound in media

Any system has advantages and disadvantages as the ultrasound transducers have. The table 1 below shows us the advantages and disadvantages of converters.

Advantages	Disadvantages
These ultrasonic transducers can able to measure in any type of material.	Ultrasonic transducers are sensitive to temperature variation. This temperature variation may change the ultrasonic reaction.
The ultrasonic transducers are not affected by temperature, water, dust or any.	It will face problems while reading the reflections from small objects, thin and soft objects.
In any type of environment, the ultrasonic transducers will work in a good manner.	-
It can measure in high sensing distances also.	-

Table 1: Advantages and Disadvantages of UT

As can be seen on the Table 1, there are more advantages features of transducer rather than disadvantages. The most important two features are the one that transducer can able to measure in any type of material and it is not affected by temperature, water or any. Yet, there is a valuable disadvantage features which we need to consider it. It is the one that ultrasonic transducers are very sensitive devices which means when the testing and analyzing the signals, attention should be paid to the signal disturbances that may occur due to sensitive.

- **Pulser / Receiver**

Figure 4: Pulser / Receiver

A pulser/receiver is an electronic device that can produce high voltage electrical pulses. Driven by the pulser, the transducer generates high frequency ultrasonic energy.

In this project, we are using advance device as a pulser / receiver which we named as a Sedaq. As seen in the Fig. 4, this device made it by some researches from Lithuania. As can be seen, there are two same devices which are connected each other. One of them uses for producing different electrical signal for different situation, and the other one uses for the controlling 3D scanner.

The most important one of them is the one that produce electrical signal. The reason why is that device is pretending like bridge between the program and transducer. The device can produce three different types of input voltage with different frequency, in the same time it can receive the electrical signals. These different types of electrical signals are Chirp, Burst and Pulse which are using for different situations.

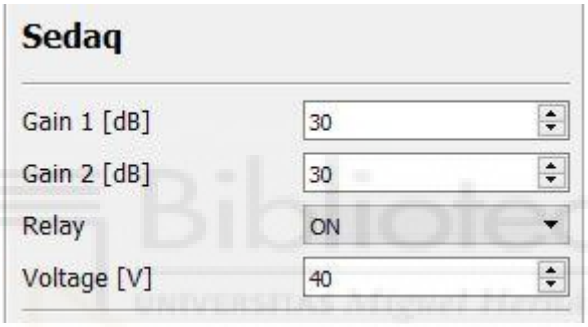

Figure 5: Parameter of Sedaq

In the Fig. 5, there are some parameters which are using for the controlling the Sedaq. The first of the parameter is Gain that needed for the amplify the receiver signal in order to analyze signal better. Second one is Relay and it is using as a switch that controls the input of the transducer. Thirdly, the voltage helps us the adjust the voltage of the input signal for the transducer. Lastly, the delay controls the delay time between the signals that we are sending continuously.

Additionally, in order to operate this device, we need to applied 12V.

B. Software

In the software part of the project lead us to which programming language that we need to use in order to create the program a graphical user interface (GUI). According scientific researches, the programming language of Matlab was using for the academic programming language in 10 years ago. However, today it has been replaced by the python programming language. The reason why is the phyton is more convenient and easier to use. Yet, it cannot be the reason to choose the phyton programming language for our GUI. First of all, it should be investigated which programming languages are better options for creating user interfaces.

A graphical user interface is very different and also it depends on visuals rather than text, making it easier for a user to operate his system or device. When applications and operating systems include a graphical user interface, actions and commands are performed via direct manipulation of on-screen graphical elements.

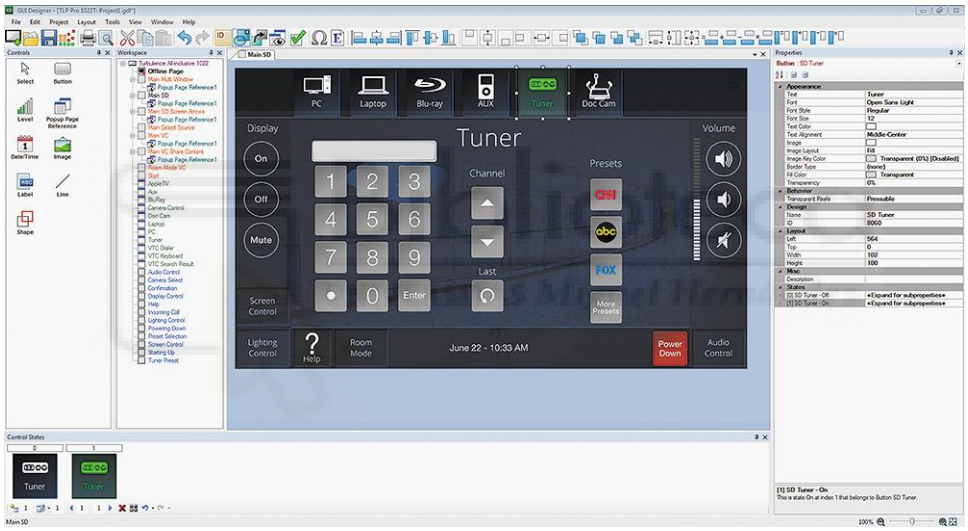

Figure 6: GUI

The principles of graphical user interface comply with the model-view-controller software pattern that separates information's internal representation from how information is provided to the user. It results in a platform where users are provided with functions that are possible instead of needing the input of command codes. Users can interact with the given information through manipulation of visual widgets that are designed to provide responses according to the data type they hold and provide support to the actions that are necessary to complete the task of a user. (4)
The visual appearance/skin of an application or an operating system might be redesigned if wanted, because of the independent nature of graphical user interfaces. Applications usually implement their unique graphical user interface display elements besides the elements of a graphical user interface that already exist in the operating system. (4)

Here are some languages that are suitable for building GUI program.

- C/C++

  C and C++ are considered to be the most suitable languages for building a graphical user interface program. While many cross-platforms libraries are available for both C and C++, they cannot be considered as the easiest to use because of their complicated nature as compared to other programming languages. Hence, you might want to make use of C and C++ in a graphical gaming environment when speed is the main goal. However, if you only aim to use a few buttons, lists, and edit boxes, this might not be the ideal language for your program. (4)

- JAVA

  Java's focuses on becoming a cross-platform programming language. The graphical user interface packages that are available are supported by most Java-enabled devices. Two main packages available for Java Graphical User Interface are AWT, which is now deprecated and is not recommended, and the other is Swing. One important feature, when making use of an IDE like eclipse, is that a user can graphically design a graphical user interface application that can be helpful to save more time spent during the designing stage. (4)

- Python

  Pythons prove to be an interesting language as it is known for its ease of use and better readability. However, a graphical user interface application can be unexciting to create using Python. It is known as an interpreted language as well as it is all written in code, and so creating programs using Python is harder. However, the idea of using a graphical user interface library can be exciting as well as surprising. (4)

  To be precise, Python is still perceived as an easy-to-use language as compared to using languages such as C/C++, since the names of libraries are readable and a few coding lines are required. Python's nature is being less prone to errors.

If the information of the 3 most used programming languages for GUI today, which is written above, is compared, it will be seen that the python programming language is much easier to use. Therefore, we are using a python programming language in order to create the program a graphical user interface. Before starting to create the program, it is necessary to decide which of the hundreds of python libraries to use.

In python language, there are nearly 10 libraries for GUI programming. Yet, according the author of the 2020 only three of them most used for GUI. (5) These are:

1. Pyqt5

- Developed by: Riverbank Computing
- Website Link: https://riverbankcomputing.com/software/pyqt/intro

The PyQt package is built around the Qt framework, which is a cross-platform framework used for creating a plethora of applications for various platforms. The PyQt5 package includes a detailed set of bindings for Python based on the latest version v5 of the Qt application framework.

2. Tkinder

- Developed by: Fredrik Lundh
- Website Link: https://wiki.python.org/moin/TkInter

Often referred to as the go-to GUI toolkit by a majority of Python developers, Tkinder was created to equip modern developers with a standard interface to the Tk GUI toolkit with its Python bindings. In Tkinter's world, most of the visual elements that we're familiar with are called widgets, and each of these widgets offers a different level of customizability.

3. Kivy

- Developed by: Kivy Orginization
- Website Link: https://kivy.org

Written with a mix of Python and Cython**,** Kivy is an open-source GUI framework for building some of the most intuitive user interfaces encompassing multi-touch applications that implement Natural User Interface (NUI)**.**

Among these three libraries of phyton, we must choose the one that works most compatible with the program we will code. In order to fine this one, we need to use trial and error method. The most critical point in the trial-and-error method is to see that the graphic part we create in the program to analyze the signals we perceive works smoothly. As a result of three experience with three different libraries, we decided that it would be better to use Pyqt5 library.

After deciding which programming language and library that we need to use to create the program, we need to code the program which helps us to control Sedaq. In order to do that we need to import some files in our program that belongs to Sedaq which are written with using C++. Later, when the program is done, we need to combine GUI program with controlling program with functions. These functions will help us to call every button on the GUI in the program.

C. Ultrasound

Third part of the method of project is the most valuable part in the project. The reason why is that the whole project based on the ultrasound. Therefore, it would be better to learn some information regarding to ultrasound waves.

Firstly, if we would like to look at the history of ultrasonics, we will have to go back to the time of the second world war. Prior to World War II, sonar, the technique of sending sound waves through water and observing the returning echoes to characterize submerged objects, inspired early ultrasound investigators to explore ways to apply the concept to medical diagnosis. In 1929 and 1935, Sokolov studied the use of ultrasonic waves
in detecting metal objects. Mulhauser, in 1931, obtained a patent for using ultrasonic waves, using two transducers to detect flaws in solids. Firestone (1940) and Simons (1945) developed pulsed ultrasonic testing
using a pulse-echo technique. (6)

Shortly after the close of World War II, researchers in Japan began to explore the medical diagnostic capabilities of ultrasound. The first ultrasonic instruments used an A-mode presentation with blips on an oscilloscope screen. That was followed by a B-mode presentation with a two-dimensional, gray scale image. (6)

Figure 7: The water-bag B-mode scanning system, the SSD-1, from Aloka in 1960

Japan's work in ultrasound was relatively unknown in the United States and Europe until the 1950s. Researchers then presented their findings on the use of ultrasound to detect gallstones, breast masses, and tumors to the international medical community. Japan was also the first country to apply Doppler ultrasound, an application of ultrasound that detects internal moving objects such as blood coursing through the heart for cardiovascular investigation. (6)

Moreover, over the past 40 years, ultrasound has become an important diagnostic modality. Its potential as a leader in medical diagnostic imaging was recognized in the

1930s and 1940s, when Theodore Dussik and his brother Friederich attempted to use ultrasound to diagnose brain tumors. It was not until the 1970s, however, that the work of these and other pioneers of ultrasound research truly came to fruition. (7)

After the historical part of the ultrasound, the biggest question of this section is “what is the Ultrasound? “.

Ultrasound is defined by the American National Standards Institute as "sound at frequencies greater than 20 kHz"[1].

Ultrasound is sound waves with frequencies higher than the upper audible limit of human hearing. Ultrasound is not different from "normal" (audible) sound in its physical properties, except that human cannot hear it. This limit varies from person to person and is approximately 20 kilohertz (20,000 hertz) in healthy young adults. Ultrasound devices operate with frequencies from 20 kHz up to several gigahertz. (8)

Ultrasound is used in many different fields. Ultrasonic devices are used to detect objects and measure distances. Ultrasound imaging or sonography is often used in medicine. In the nondestructive testing of products and structures, ultrasound is used to detect invisible flaws. Industrially, ultrasound is used for cleaning, mixing, and accelerating chemical processes. (8) Animals such as bats and porpoises use ultrasound for locating prey and obstacles[2].

Yet, we are using ultrasound in order to detect objects, object’s thickness and distance in this project. Therefore, we need to understand basic principle of ultrasonic testing.

Ultrasonic Testing (UT) uses high frequency sound energy to conduct examinations and make measurements. Ultrasonic inspection can be used for flaw detection/evaluation, dimensional measurements, material characterization, and more. To illustrate the general inspection principle, a typical pulse/echo inspection configuration as illustrated below will be used. (9)

Typical UT inspection system consists of several functional units, such as the pulser/receiver, transducer, and display devices. A pulser/receiver is an electronic device that can produce high voltage electrical pulses like Sedaq. Driven by the pulser, the transducer generates high frequency ultrasonic energy. The sound energy is introduced and propagates through the materials in the form of waves. When there is a discontinuity (such as a crack) in the wave path, part of the energy will be reflected back from the flaw surface as we can see in the Fig. 8. The reflected wave signal is transformed into an electrical signal by the transducer and is displayed on a screen. In this project, with help of the Sedaq device we are using GUI on computer as a display screen. In the applet below, the reflected signal strength is displayed versus the time from signal generation to when an echo was received. Signal travel time can be directly related to the distance that the signal traveled. From the signal, information about the reflector location, size, orientation and other features can sometimes be gained. (9)

[1] *https://en.wikipedia.org/wiki/Ultrasound*
[2] *Novelline R (1997). Squire's Fundamentals of Radiology (5th ed.). Harvard University Press. pp. 34–35. ISBN 978-0-674-83339-5*

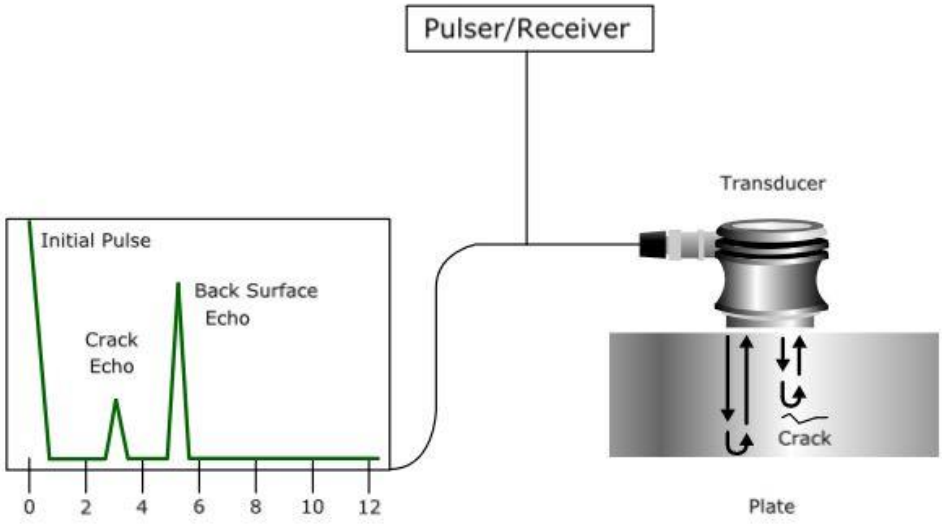

Figure 8: Basic Principles of Ultrasonic Testing

Basically, this project based on this principle. In the project, Sedaq is used instead of pulser / receiver and also instead of display screen, the GUI program is used. GUI program will help us to create an ultrasound on transducer in order to detect the objects. After receiving the signal on the GUI graph, we will be able to analyze the signal.

4. Activities Carried Out in the Project

This part of the project contains information explaining the activities carried out in the project.

A. Design the GUI Program

As it mentioned before, the library of the Phyton, which is PyQt5, will be used in order to build the GUI program. Before starting to code the program, we need to design it. Therefore, we will be using different kind of program which is called Qt Designer. This program helps us to designed GUI program without writing any code.

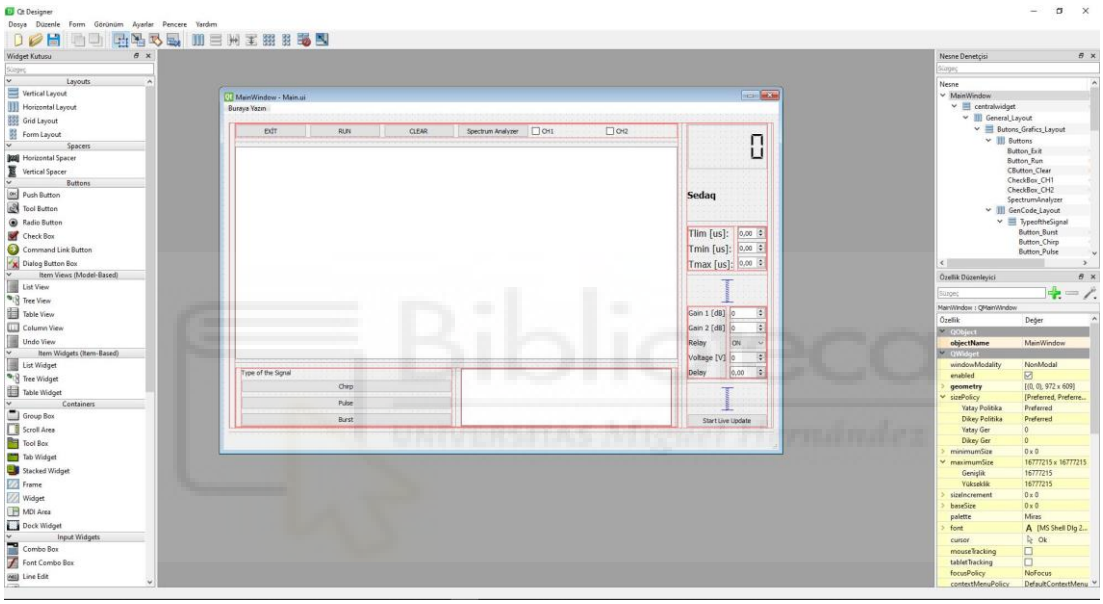

Figure 9: GUI Program on Qt Designer

As we can see on the Fig.9, the left side includes some layouts, buttons, containers and display widgets. We will add them to the main program that we have opened with drag and drop logic. Yet, during this process, it would be better to careful about the structure of layouts. Layouts helps us to keep buttons that we add to the program smoothly. Otherwise, our buttons will disappear on the screen when we enlarge and shrink the desktop application. Therefore, we need to use layouts for every button, widget or line on the GUI.

Additionally, some features of widget box appear on the right side of the screen. These things help us to set some features of the widgets like widget size, location, font, font size etc. Moreover, we may able to see the name of every widget and layouts that we are using on the program on the right side of the Fig. 9.

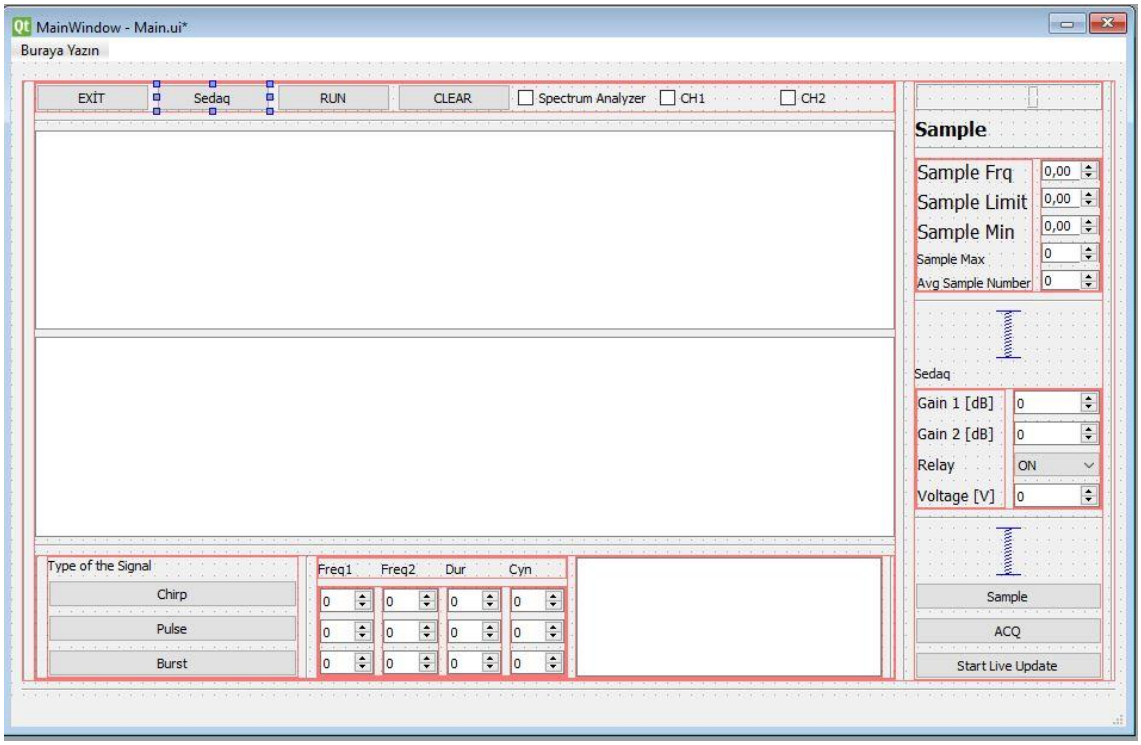

Figure 10: Graphical User Interface Program of Ultrasound Project

As seen in the Fig. 10, the latest version of graphical user interface program has designed with using Qt Designer. As we can see there many buttons and two graphics in the program. These are the explanation of buttons on program below:

- Type of Signal:

The type of the signal part is using for setting the electrical signal that we need to send it to transducer. As it mentioned before, we are using three different types of electrical signals which are Burst, Chirp and Pulse.

- Sedaq:

The buttons of the Sedaq part are using for the set features of the Sedaq device such as the gains, voltage and delay time. Additionally, we need to choose which channel we are working on Sedaq. The reason why is that the Sedaq has to different channel. Usually, we will be working on channel 2.

- Graphic:

This part of the program will be using to see the result on the graph in order to analyze it.

- Spectrum Analyzer Button:

SA (Spectrum Analyzer) part will be using in the future in order to see the receiving signal on SA.

- Buttons of Menu:

In this part of program, there are like four buttons which are Exit, Run, Clear and Live buttons. Exit button will be using for the quit the program properly and the run button

for the run the program again and again. Moreover, the purpose of clear button is to clear the whole graph in order to reset all valuables on the Sedaq or graph. Finally, the Live button made it for refresh the graph.

B. Coding the GUI Program

After finishing the design of program, the coding part will start. To code the program, we will use the PyCharm program, which we can easily import the python libraries. Before starting to code program, we need to import some libraries into our project.

```
1
2   import sys
3   sys.path.insert(0,r"C:\Users\User\Desktop\Final Project\Main Program\QtDesignerWindows")
4
5   from PyQt5 import QtCore, QtGui, QtWidgets
6   from PyQt5.QtGui import QIcon
7   from PyQt5.QtWidgets import QMessageBox, QApplication
8   from pyqtgraph import PlotWidget, plot
9   import pyqtgraph as pg
10  from SpectrumAnalyzer import Ui_SpectrumAnalyzer
11  import SeDaq as SD
12  import GenCode_ToolBox as gc
13  import numpy as np
14  import time
15
```

Figure 11: Libraries of Phyton

In the Fig. 11, the libraries and files that we need to import into project can be seen. In order to import the libraries, we need to download packages with using button of PyCharm setting.

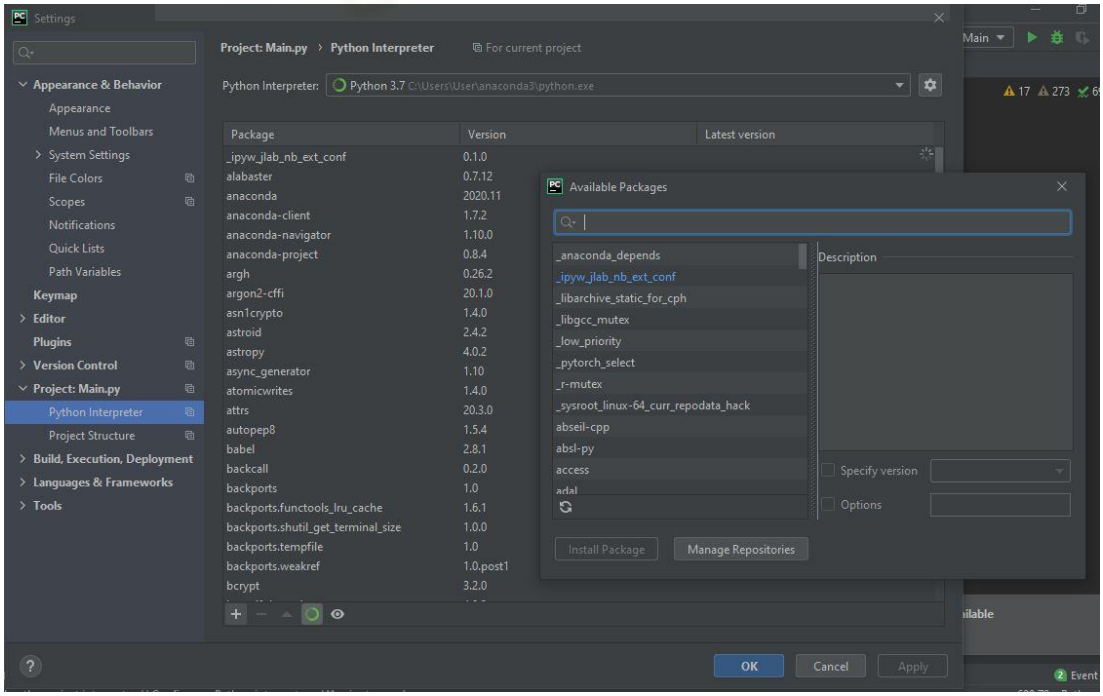

Figure 12: Download the Packages

After click on File>Settings>Python Interpreter, the window like in the Fig. 12 will shows up on the screen. That is how we install packages that we need. Later on, when the package installation process is finished, we need import the libraries in the code like Fig.11. Every line of code represents something:

- import sys
  sys.path.insert(0,r"C:\Users\User\Desktop\Final Project\Main Program\QtDesignerWindows")

  The first line of the code helps us to import all files that we choose in the specific file we aimed.

- from PyQt5 import QtCore, QtGui, QtWidgets
  from PyQt5.QtGui import QIcon
  from PyQt5.QtWidgets import QMessageBox, QApplication

  This code above helps us to import PyQt5 libraries which we are using most.

- from pyqtgraph import PlotWidget, plot
  import pyqtgraph as pg

  Importing the pyqtgraph library in order to see the receiving signal on the graph. Additionally, we are importing pyqtgraph as a 'pg' in order to use pyqtgraph library easily in the code.

- from SpectrumAnalyzer import Ui_SpectrumAnalyzer

  These codes help us to import another script in order to use functions in the script. Additionally, this process calling like object-oriented programming[3] which helps to avoid complexity and ensures that the code written is developable.

- import SeDaq as SD
  import ACQ_ToolBox as ACQ

  In this part of the code, we are importing some scripts which are includes some function regarding to control Sedaq device.

- import numpy as np

  One of the most useful libraries for the python is numpy[4]. NumPy is a Python library used for working with arrays.

  It also has functions for working in domain of linear algebra, Fourier transform, and matrices.

[3] *https://realpython.com/python3-object-oriented-programming/*
[4] *https://numpy.org/*

- import time

Lastly, we are importing the time library in order to use time function in the program.

Before starting to write the code, four parameters should be explained. These are:

1. Self:

   The word 'self' is used to represent the instance of a class. By using the "self" keyword we access the attributes and methods of the class in python. (10)

2. __init__

   "__init__" is a reserved method in python classes. It is called as a constructor in object-oriented terminology. This method is called when an object is created from a class and it allows the class to initialize the attributes of the class. (10)

3. class

   Classes provide a means of bundling data and functionality together. Creating a new class creates a new *type* of object, allowing new *instances* of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by its class) for modifying its state. (11)

4. Function

   A function is a block of code to carry out a specific task, will contain its own scope and is called by name. All functions may contain zero(no) arguments or more than one argument. On exit, a function can or cannot return one or more values. (12)

```python
#global Sedaq = SD.SeDaqDLL()

class Ui_MainWindow(object):

    def __init__(self):
        self.Sedaq = None
        #First you need to connect with the Sedaq in order to use the other function

    def Popup(self):
        message = QMessageBox()
        message.setWindowTitle("Exit")
        message.setText("Are you sure you want to Exit?")
        message.setIcon(QMessageBox.Critical) #Critical Icon Added
        #Buttons
        message.setStandardButtons(QMessageBox.Yes | QMessageBox.No)
        message.setDefaultButton(QMessageBox.No)#Defaoult button with the Highlite
        message.setDetailedText("Details") #Details Button
        message.buttonClicked.connect(self.PopupButton)
        x = message.exec_()  # Show the message box


    def PopupButton(self, i):
        print(i.text())
```

PyCharm 2020.3.5 available
Update...

Figure 13: Class Method

On the top of the Fig. 13, we are implementing the class method. Afterwards, we will write every function and method in this class.

First of all, we will define a function which name is setupUI in order to create main window like in the Fig. 14.

Main.py

```python
    ######################################################################

    def setupUi(self, MainWindow):

        #MainWindow
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(972, 609)


        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")

        #Main Layout
        self.verticalLayout = QtWidgets.QVBoxLayout(self.centralwidget)
        self.verticalLayout.setObjectName("verticalLayout")

        #Lines of Vertical Layout
        self.line_15 = QtWidgets.QFrame(self.centralwidget)
        self.line_15.setFrameShape(QtWidgets.QFrame.VLine)
        self.line_15.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line_15.setObjectName("line_15")
        self.verticalLayout.addWidget(self.line_15)

        #General Layout
        self.General_Layout = QtWidgets.QHBoxLayout()
        self.General_Layout.setObjectName("General_Layout")
```

Figure 14: Define Function for Main Window

After defined function for main window, we will use this code line below in order to create empty window. Moreover, we determine the width and length of the window.

```
MainWindow.setObjectName("MainWindow")
MainWindow.resize(972, 609)
```

After this section, what we need to do is write code for each object that we will add to the GUI program. Hence, the following is an explanation of the codes required to identify each object.

- Layouts

```
#Main Layout
self.verticalLayout = QtWidgets.QVBoxLayout(self.centralwidget)
self.verticalLayout.setObjectName("verticalLayout")
```

Figure 15: Layout

In this part of the code, we are defining a layout for the main window. In order to do that we are using Qt.Widger.QVBoxLayout function and adding layout into central widget. Additionally, we named the layout using funtion which name is setOjectName.

- Lines

```
#Lines of Vertical Layout
self.line_15 = QtWidgets.QFrame(self.centralwidget)
self.line_15.setFrameShape(QtWidgets.QFrame.VLine)
self.line_15.setFrameShadow(QtWidgets.QFrame.Sunken)
self.line_15.setObjectName("line_15")
self.verticalLayout.addWidget(self.line_15)
```

Figure 16: Lines

In order to create a line for separate some objects on the program and to make the program more organized, we need to use these codes. The code of "QtWidget.QFrame(self.centralWidget)" creates the line in the main window. Second line of code defind the types of line. Last line determines the which layout the line will be in.

- Button

```
#Exit Button
self.Button_Exit = QtWidgets.QPushButton(self.centralwidget)
self.Button_Exit.setEnabled(True)
self.Button_Exit.setObjectName("Button_Exit")
self.Buttons.addWidget(self.Button_Exit)
```

Figure 17: Button

We are using “QtWidget.QPushButton(self.centralWidget)” function in order to create any button that we want. “self.Button.setEnabled(True)” is using for make button clickable.

- Box Button

```
#Check Box Button (CH1)
self.CheckBox_CH1 = QtWidgets.QCheckBox(self.centralwidget)
self.CheckBox_CH1.setObjectName("CheckBox_CH1")
self.Buttons.addWidget(self.CheckBox_CH1)
```

Figure 18: Box Button

In this part of the code, we are creating box button for CH1 and CH2.

- Graphic

```
#Grafic
self.Grafic = PlotWidget(self.centralwidget)
self.Grafic.setObjectName("Grafic")
self.Butons_Grafics_Layout.addWidget(self.Grafic)


#Grafic1
self.Grafic1 = PlotWidget(self.centralwidget)
self.Grafic1.setObjectName("Grafic1")
self.Butons_Grafics_Layout.addWidget(self.Grafic1)
```

Figure 19: Graphic

Fig. 19 shows us that how we can create graphic in the program.

- Plain Text

```
#Output of the Script = PlainText
self.plainTextEdit_2 = QtWidgets.QPlainTextEdit(self.centralwidget)
sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Ignored)
sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)
sizePolicy.setHeightForWidth(self.plainTextEdit_2.sizePolicy().hasHeightForWidth())
self.plainTextEdit_2.setSizePolicy(sizePolicy)
self.plainTextEdit_2.setObjectName("Output")
self.verticalLayout_2.addWidget(self.plainTextEdit_2)
self.GenCode_Layout.addLayout(self.verticalLayout_2)
```

Figure 20: Plain Text (Output)

These bunch of lines are using for creating a plain text. The first line of the code represented to creating the plain text. Moreover, the sizePolicy function helps to organized the font and location of the text in the plain text. After that, we are determining the name of the plain text as an “Output”.

- Label

```
        #TimLim Label
self.label_Timlim_2 = QtWidgets.QLabel(self.centralwidget)
font = QtGui.QFont()
font.setPointSize(12)
self.label_Timlim_2.setFont(font)
self.label_Timlim_2.setLayoutDirection(QtCore.Qt.LeftToRight)
self.label_Timlim_2.setTextFormat(QtCore.Qt.AutoText)
self.label_Timlim_2.setScaledContents(True)
self.label_Timlim_2.setObjectName("label_Timlim_2")
self.Text_TimeLayout_2.addWidget(self.label_Timlim_2)
```

Figure 21: Label

These codes help us to create a label in the program. As we can see on the Fig.21., with the help of the function of font we can set the font of the text. Moreover, the setLayoutDirection(QtCore.Qt.AutoText) determine text to be written from right to left. Also, the function of setTextFormat(QtCore.Qt.AutoText) is using for the setting format of text automatically.

- Spin Box

```
        #Spin of Timemax
self.Spin_Tmax_2 = QtWidgets.QDoubleSpinBox(self.centralwidget)
self.Spin_Tmax_2.setObjectName("Spin_Tmax_2")
self.Spin_TimeLayout2_2.addWidget(self.Spin_Tmax_2)
self.Spin_Tmax_2.setRange(-1000, 1000)  # Set the Range
```

Figure 22: Spin Box

The first line on the Fig.22. is creating the spin box in the central widget. Further, the third line of the code add spin box into time layout in order to organize the structure of program. Lastly, the last code is setting the range of the spin box.

- Combo Box

```
        #Combo Box of Relay
self.ComboBox_Relay_2 = QtWidgets.QComboBox(self.centralwidget)
self.ComboBox_Relay_2.setObjectName("ComboBox_Relay_2")
self.ComboBox_Relay_2.insertItem(1,'ON')
self.ComboBox_Relay_2.insertItem(0,'OFF')
self.Spin_ControlLayout_2.addWidget(self.ComboBox_Relay_2)
```

Figure 23: Combo Box

These codes using for determine the combo box for relay button. The only differences from code of the button to define 'ON' and 'OFF' items.

- Menu Bar

```
#Menu Bar
MainWindow.setCentralWidget(self.centralwidget)
self.menubar = QtWidgets.QMenuBar(MainWindow)
self.menubar.setGeometry(QtCore.QRect(0, 0, 972, 21))
self.menubar.setObjectName("menubar")
MainWindow.setMenuBar(self.menubar)
self.statusbar = QtWidgets.QStatusBar(MainWindow)
self.statusbar.setObjectName("statusbar")
MainWindow.setStatusBar(self.statusbar)
```

Figure 24: Menu Bar

In this part of the code representing to determine the menu bar into the program. Yet, this is not necessary for the GUI program for now.

- Rename the Objects

We need to open a separate method to determine the names of the objects that appear in the program. The function of "setObjectName" that we used for setting name of the object above only named the object in the code. Therefore, we need to determine the names for the program. After writing the method of 'retranslateUi', we need to call it in the other function which name is 'setepUI'.

```
    self.retranslateUi(MainWindow)
    QtCore.QMetaObject.connectSlotsByName(MainWindow)


#Rename the Everything
def retranslateUi(self, MainWindow):
    _translate = QtCore.QCoreApplication.translate
    MainWindow.setWindowTitle(_translate("MainWindow", "Ultrasound Project"))
    self.Button_Exit.setText(_translate("MainWindow", "EXİT"))
    self.Button_Run.setText(_translate("MainWindow", "RUN"))
    self.CButton_Clear.setText(_translate("MainWindow", "CLEAR"))
    self.Button_Start.setText(_translate("MainWindow", "SeDaq"))
    self.SpectrumAnalyzer.setText(_translate("MainWindow", "Spectrum Analyzer"))
    self.CheckBox_CH1.setText(_translate("MainWindow", "CH1"))
    self.CheckBox_CH2.setText(_translate("MainWindow", "CH2"))
    self.TypeLabel.setText(_translate("MainWindow", "Type of the Signal"))
    self.Button_Chirp.setText(_translate("MainWindow", "Chirp"))
    self.Button_Pulse.setText(_translate("MainWindow", "Pulse"))
    self.Button_Burst.setText(_translate("MainWindow", "Burst"))
    self.label_Sedaq_2.setText(_translate("MainWindow", "Sedaq"))
    self.label_Timlim_2.setText(_translate("MainWindow", "Tlim [us]:"))
```

Figure 25: Rename the Objects

- Showing Main Window on the Screen

```
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

Figure 26: Sowing Main Window on the Screen

The function that equals with ‘app’ helps to determine the widget as an application. We also drop the setupUI method we created into the MainWindow function we created. Further, we are showing our MainWindow function with MainWindow.show(). The function of sys.exit(app.exec_()) is using to quit window.

When we use the objects written with the explanations above, we create a GUI without a function like in Fig. 27.

Figure 27: GUI Program

Additionally, every object on the program that it can be seen in Fig. 27 coded with using the explained code above. Yet, these buttons are dysfunctional. In order for each button or object in the program to have a function, a function must be defined for all of them.

C. Connection Sedaq and GUI with Functions

In this part of the project, we will implement a function into each object into program in order to use it. These functions explained below:

1. Exit Button

In order to create function for the exit button of the program, the code in the Fig.28. is used. The popup message created in order to ask a warning to avoid incorrect button presses. Because the user may have pressed this button by mistake.

```
26      def Popup(self):
27
28          message = QMessageBox()
29          message.setWindowTitle("Exit")
30          message.setText("Are you sure you want to Exit?")
31          message.setIcon(QMessageBox.Critical) #Critical Icon Added
32          #Buttons
33          message.setStandardButtons(QMessageBox.Yes | QMessageBox.No)
34          message.setDefaultButton(QMessageBox.No)#Defaoult button with the Highlite
35          message.setDetailedText("Details") #Details Button
36          message.buttonClicked.connect(self.PopupButton)
37          x = message.exec_()  # Show the message box
38          if x == 16384:
39              QApplication.quit()
40
41      def PopupButton(self, i):
42          print(i.text())
```

Figure 28: Code of Exit Button

The QMessageBox library of the Pyqt5.QtWidget is used in order to create popup message with 'QMessageBox()' method. 'setText()' method used to implement a message in the popup message. Further, The yes and no button created with 'setStandartButtons(QMessage.Yes | QMessage.No)' method.

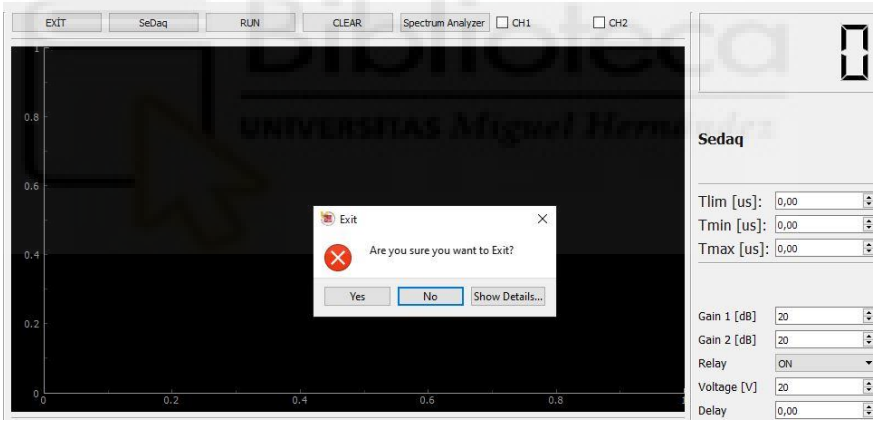

Figure 29: Popup Message

The user is given the option to click on two buttons named yes and no. If the user chooses the 'yes' button, then program create a number which is '16384'. All we need to do is equalize this number with 'QApplication.quit()' method in order to close the program. If the user chooses the 'No' button, then we need to set this button as a default with 'setDefaultButton' method in order to just close the popup message.

```
298         #Exit Button
299         self.Button_Exit = QtWidgets.QPushButton(self.centralwidget)
300         self.Button_Exit.setEnabled(True)
301         self.Button_Exit.setObjectName("Button_Exit")
302         self.Buttons.addWidget(self.Button_Exit)
303         self.Button_Exit.clicked.connect(self.Popup)
304
```

Figure 30: Connection between Popup Function and Exit Button

As it can be seen in the Fig.30, the red line of the code represents the connection between popup message function and exit button.

2. Sedaq Button

```
45      '''
46      First Connect with Sedaq
47      '''
48      #Sedaq Connection
49      def SedaqConnection(self):
50          self.Sedaq = SD.SeDaqDLL()
51
52          time.sleep(1)
53          Connection = "Connected"
54          Connection1 = "Not Connected"
55          if self.Sedaq == True :
56              print("Sedaq is", Connection)
57          else:
58              print("Sedaq is", Connection1)
59
60      def ClearButton(self):
61          self.Grafic.clear()
```

Figure 31: Sedaq Button

First thing to do in the program should be to connect with Sedaq device. In order to connect with Sedaq device, we need to use the functions from the Sedaq script which is 'SedaqDLL'. In order to use this function, first we need to import Sedaq script as we did in the previous part. After that, we need to set the Sedaq as 'None' in order to use this method in every function like in the Fig. 31.

```
21
22      def __init__(self):
23          self.Sedaq = None
24          #First you need to connect with the Sedaq in order to use the other function
```

Figure 32: Sedaq = None

Additionally, the function in Fig. 32 tells us that we need to connect Sedaq in order to use this function in other functions.

Moreover, we are using an if method in order to understand if we are connected with Sedaq.

```
305         #Sedaq Connection
306         self.Button_Start = QtWidgets.QPushButton(self.centralwidget)
307         self.Button_Start.setObjectName("Button_Start")
308         self.Buttons.addWidget(self.Button_Start)
309         self.Button_Start.clicked.connect(self.SedaqConnection)  # Call Sedaq
310
```

Figure 33: Connection between Sedaq Function and Sedaq Button

As it can be seen in the Fig.33, the red line of the code represents the connection between Sedaq function and Sedaq button.

3. Relay Combo Box

```
def Setrelay(self, index):
    selected_Text = self.ComboBox_Relay_2.currentText()
    try:
        if selected_Text == "ON":
            self.Sedaq.SetRelay(1)
            print("text :", selected_Text)

        if selected_Text == "OFF":
            self.Sedaq.SetRelay(0)
            print("text :", selected_Text)
    except Exception as e:
        print("Please Connect with the Sedaq",e)
```

Figure 34: Relay Combo Box

The Fig.34 shows the function and methods are used for the setting relay combo box ‘ON’ or ‘OFF’. Once again, we are using different function to control relay from Sedaq script which is ‘SetRelay’ in the SedaqDLL. We are connecting SetRelay function with ‘self.Sedaq.SetRelay(1)’ method.

```
764             #Combo Box of Relay
765         self.ComboBox_Relay_2 = QtWidgets.QComboBox(self.centralwidget)
766         self.ComboBox_Relay_2.setObjectName("ComboBox_Relay_2")
767         self.ComboBox_Relay_2.insertItem(1,'ON')
768         self.ComboBox_Relay_2.insertItem(0,'OFF')
769         self.Spin_ControlLayout_2.addWidget(self.ComboBox_Relay_2)
770         self.ComboBox_Relay_2.currentIndexChanged['int'].connect(self.Setrelay)
771
```

Figure 35: Connection between Relay Combo Box and Relay Function

As it can be seen in the Fig.35, the red line of the code represents the connection between Relay function and Relay button.

4. Gain1 and Gain2 Spin Boxes

```
77      def Setgain1(self):
78          try:
79              self.Selected_Gain1 = self.spin_Gain1_2.value()
80              print("Gain 1:", self.Selected_Gain1)
81              #self.Sedaq.SetGain1(self.Selected_Gain1)
82          except Exception as e:
83              print("Please Connect with the Sedaq", e)
84
85
86      def Setgain2(self):
87
88          try:
89              self.Selected_Gain2 = self.Spin_Gain2_3.value()
90              print("Gain 2:", self.Selected_Gain2)
91              #self.Sedaq.SetGain2(self.Selected_Gain2)
92          except Exception as e:
93              print("Please Connect with the Sedaq", e)
94
```

Figure 36: Gain1 and Gain 2 Spin Boxes

The functions of Gain1 and Gain2 is used to setting Gains value in the Sedaq device. In this section, unlike other buttons, we only get the value in the spin box. Because we want to send the values entered in the spin box to our Sedaq device with the ACQ button.

```
750             #Spin of Gain 1
751         self.spin_Gain1_2 = QtWidgets.QSpinBox(self.centralwidget)
752         self.spin_Gain1_2.setObjectName("spin_Gain1_2")
753         self.Spin_ControlLayout_2.addWidget(self.spin_Gain1_2)
754         self.spin_Gain1_2.setRange(20, 60) #Set the Range
755         self.spin_Gain1_2.valueChanged.connect(self.Setgain1)
756
757             #Spin of Gain 2
758         self.Spin_Gain2_3 = QtWidgets.QSpinBox(self.centralwidget)
759         self.Spin_Gain2_3.setObjectName("Spin_Gain2_3")
760         self.Spin_ControlLayout_2.addWidget(self.Spin_Gain2_3)
761         self.Spin_Gain2_3.setRange(20,60) #Set the Range
762         self.Spin_Gain2_3.valueChanged.connect(self.Setgain2)
```

Figure 37: Connection between Gains Functions and Gain Spin Boxes

With help of the 'Gain.valueChanged.connect(self.SetGain)' method, we can make connection between function and spin box.

5. Voltage Spin Box

```
96      def SetVoltage(self):
97          try:
98              self.Selected_Voltage = self.Spin_Voltage_3.value()
99              print("Voltage:", self.Selected_Voltage)
100             #self.Sedaq.SetExtVoltage(self.Selected_Voltage)
101         except Exception as e:
102             print("Please Connect with the Sedaq",e)
103
```

Figure 38: Voltage Spin Box

As we did in the previous Gain section, here we only call the value entered in the voltage spin box with using 'self.Spin_Voltage_3.value()' method.

```
772             #Spin of Voltage
773         self.Spin_Voltage_3 = QtWidgets.QSpinBox(self.centralwidget)
774         self.Spin_Voltage_3.setObjectName("Spin_Voltage_3")
775         self.Spin_ControlLayout_2.addWidget(self.Spin_Voltage_3)
776         self.Spin_Voltage_3.setRange(20, 60) #Set the Range
777         self.Spin_Voltage_3.valueChanged.connect(self.SetVoltage)
778
```

Figure 39: Connection between Voltage Spin Box and Voltage Function

With help of the 'Voltage.valueChanged.connect(self.SetVoltage)' method, we can make connection between function and spin box like in the Fig.39.

6. ACQ Button

```
112     def ACQButton(self):
113         self.Setgain1()
114         self.Setgain2()
115         self.SetVoltage()
116         self.Setdelay()
117         try:
118             self.Sedaq.SetGain1(self.Selected_Gain1)
119             self.Sedaq.SetGain2(self.Selected_Gain2)
120             self.Sedaq.SetExtVoltage(self.Selected_Voltage)
121             self.Sedaq.SetBankDelay(self.Selected_Delay)
122         except Exception as e:
123             print("Please Connect with the Sedaq",e)
124             #There may be error in the SetBankDelay Check it
125
```

Figure 40: ACQ Button

The ACQ Button helps us to send the values that we get from different spin boxes (Gain1, Gain2, Voltage) to Sedaq script with using 'Sedaq' function like in the Fig.40. But, first thing to do should be call the function where the values calling.

```
799             #ACQ Button
800         self.ACQ = QtWidgets.QPushButton(self.centralwidget)
801         self.ACQ.setObjectName("ACQ")
802         self.Sedaq_Layout_2.addWidget(self.ACQ)
803         self.ACQ.clicked.connect(self.ACQButton)
```

Figure 41: Connection between ACQ Button and ACQ Function

Additionally, as seen in functions, try and exception functions are used. The reason why is that the try function first tries to run the code inside the try part. However, if this code fails to run, it runs the exception section. The reason we use this method is to ensure that the code we write works properly. If we get an error when one of these buttons is pressed, the entire application may fail and cause the program to crash.

For instance, ACQ button is connected with Sedaq function in order to send the values to SedaqDLL function. Yet, we implemented Sedaq function as 'None'. That means is that first thing to do should be connect with Sedaq. If the user presses the ACQ button before connected with Sedaq, the method cannot be sent to Sedaq and the program gets error. In order to avoid this error, we are using try-exception method. Thanks to this method, if the user did not connect with Sedaq first, the user will receive a 'Please Connect with Sedaq' warning.

7. Chirp Button

The purpose of the chirp button is to connect with MakeGenCode function from GenCode_Toolbox script in order to create chirp signal and send it to UpdateGencode function from Sedaq Script. With this way, we are ordering to create chirp signal to Sedaq device. The 'gc.MakeGenCode' function helps to send the values that we get from Chirp Spin Boxes to GenCode_Toolbox script. Moreover, the 'Sedaq.UpdateGenCode' function is using for sending the signal that we created from another script to Sedaq.

```
140     def ChirpSignal(self):
141         try:
142             self.Chirp()
143             F1_int = self.CFreq1 #More like f1 2MHZ
144             F2_int = self.CFreq2 #More like f2 8MHz
145             Dur = self.CDur  # ChirpLength 1-10 us
146             ProbingGencode = gc.MakeGenCode(Excitation='Chirp',
147                                             ParamVal=[F1_int, F2_int, Dur, 'linear', 270])  # crear
148             self.Sedaq.UpdateGenCode(ProbingGencode)
149         except Exception as e:
150             if self.Sedaq == None:
151                 print("First click Sedaq and Connect with Sedaq Script")
152
```

Figure 42: Chirp Button

```
def MakeGenCode(Excitation='Pulse', Param='frequency', ParamVal = 5e6, SignalPolarity = 2, Fs = 200e6,
                DeadTime_Samples=0, CancelDuration=0, AddZerosInFront_Samples=0):
    """
    Make excitation gencodes.
    Inputs
        Excitation = {'Pulse','Chirp','Burst'}
        Param = {'Frequency','Duration'}: use only for pulse
        ParamVal =
            {Fo or Duration} for Pulse
            {Fstart,Fend,Duration,Method,Phase} for Chirp
            {Fo,NoCycles} for Burst
        SignalPolarity = {2, or 1 or -1}, usually 2 (bipolar)
        Fs = Sampling frequency (200e6 for usual KTU pulser/receiver)
        DeadTime_Samples = Not sure, leave 0
        CancelDuration = Not sure, leave 0
        AddZerosInFront_Samples = Not sure, leave 0
    Outputs
        GenCode = Produced GenCode
    """
    if Excitation.upper()=='PULSE':
        Signal = GC_MakePulse(Param = Param, ParamVal = ParamVal, SignalPolarity = SignalPolarity, Fs = Fs)
    elif Excitation.upper()=='CHIRP':
        Signal = GC_MakeChirp(Fstart=ParamVal[0], Fend=ParamVal[1], Duration=ParamVal[2], method=ParamVal[3], Phase=P
    elif Excitation.upper()=='BURST':
        Signal = GC_MakeBurst(Fo=ParamVal[0], NoCycles=ParamVal[1], SignalPolarity = SignalPolarity, Fs = Fs)
    GenCode = Signa2GenCode(Signal, SignalPolarity = SignalPolarity, DeadTime_Samples=DeadTime_Samples, CancelDurati
```

Figure 43: MakeGenCode Function of GenCode_ToolBox Script

The Fig.43. shows the 'MakeGenCode' function of GenCode_ToolBox scritp that we are using in Fig.44.

```
# Chirp GenCode
def GC_MakeChirp(Fstart=2e6,Fend=8e6,Duration=3e-6, method='linear', Phase=270, SignalPolarity = 2, Fs = 200e6):
    """
    Make Chirp gencode
    Inputs
        Fstart = start (lower) frequency
        Fend = end (higher) frequency
        Duration = duration of the pulse in seconds
        Phase = phase of the pulse in degrees
        method = sort of chirp {'linear', 'quadratic', 'logarithmic', 'hyperbolic'}
        SignalPolarity = [2 or 1 or -1], polarity of the gencode
        Fs = Sampling frequency in Hz
    Outputs
        Signal
    """
    t = np.arange(0,Duration-1/Fs,1/Fs)
    SignalChirp=signal.chirp(t, Fstart, Duration, Fend, method, Phase);
    nr=0
    Signal = np.zeros(len(SignalChirp))
    for Value in SignalChirp:
        if Value>0:
            Signal[nr]=1
```

Figure 44: GC_MakeChirp Function of GenCode_ToolBox Script

After sending the parameters to the GenCode page, the function we use sends these parameters to the GC_MakeChirp function in order to create Chirp Signal.

```
89      def UpdateGenCode(self, gencode):
90          gencode_len = len(gencode)
91          N = ClosestPowerOf2(gencode_len)
92          BytesTot = 2**N
93          GenArrayTo = (c_uint8 * BytesTot)()
94          for i in range(BytesTot):
95              if i<gencode_len:
96                  GenArrayTo[i] = c_uint8(int(gencode[i]))
97              else:
98                  GenArrayTo[i] = c_uint8(0)
99          BytesTot = len(GenArrayTo)
100         self.SeDaqDLL_SetExcWave(byref(GenArrayTo),BytesTot,0)
101
```

Figure 45: UpdateGenCode Funtion of Sedaq Script

```
173     # Chirp
174     def Chirp(self):
175         self.CFreq1 = self.spinBox_1.value() * 1000000
176         self.CFreq2 = self.spinBox_4.value() * 1000000
177         self.CDur = self.spinBox_7.value() * 0.000001
178         print(self.CFreq1,self.CFreq2,self.CDur)
```

Figure 46: Getting Values from Chirp Spin Boxes

In Fig. 46 shows that how we can get the values of Chirp spin boxes. Additionally, as we can see on the Fig.46 the values multiply with numbers. The reason why is that the frequency of chirp signal is too high to write into spin boxes. Therefore, we need to multiply it. Moreover, the frequencies that we are using in the project are too high because the ultrasound waves are producing with high frequencies.

Further, we are using a 'spinbox.valueChanged.connect(self.function)' method in order to connect with Chirp function and spin boxes.

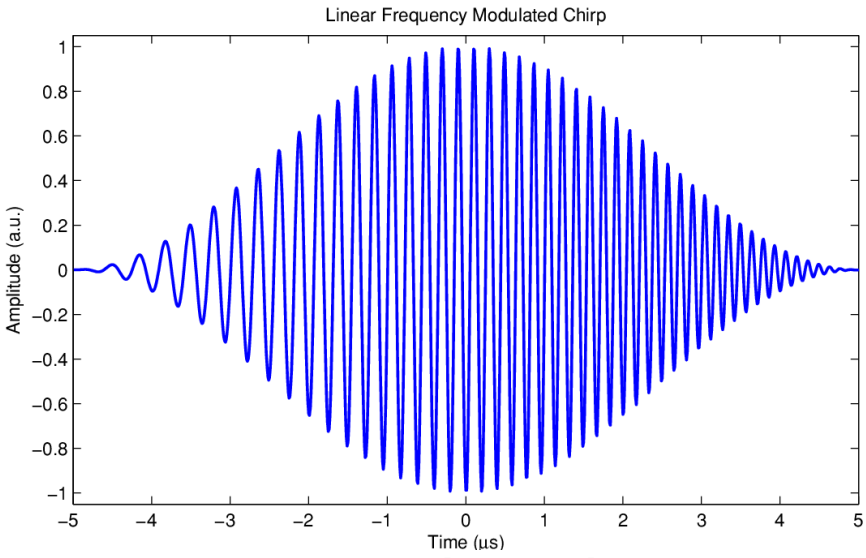

Figure 47: Chirp Signal

In order to create chirp signal like in the Fig.47, we need to use same parameters. These are two frequency values and duration value. That is the reason why we need use three parameter in the function 'gc.MakeGenCode' in the Fig.43.

8. Burst Button

```
166     def BurstSignal(self):
167         try:
168             #We may need to set Cyn as 5 like in the MakeBurst function of GenCode_ToolBox
169             self.Burst()
170             BF1_int = self.BFreq1 #5MHz
171             #BDur = self.BDur # We dont need it
172             BCynn = self.BCyn # 3 -5 cycles
173             BProbingGencode = gc.MakeGenCode(Excitation='Burst',
174                                              ParamVal=[BF1_int, BCynn])  # crear
175             self.Sedaq.UpdateGenCode(BProbingGencode)
176         except Exception as e:
177             if self.Sedaq == None:
178                 print("First click Sedaq and Connect with Sedaq Script")
179
```

Figure 48: Burst Button

In the burst button function, we apply the same operations as in the chirp button function. We are using function from GenCode_Toolbox script in order to create burst signal and send it to UpdateGencode function from Sedaq Script. With this way, we are ordering to create burst signal to Sedaq device.

The 'gc.MakeGenCode' function helps to send the values that we get from Burst Spin Boxes to GenCode_Toolbox script. Moreover, the 'Sedaq.UpdateGenCode' function is using for sending the signal that we created from another script to Sedaq.

```
80  # Chirp GenCode
81  def GC_MakeBurst(Fo=5e6, NoCycles=5, SignalPolarity=2, Fs=200e6):
82      """
83      Make Burst gencode
84      Inputs
85          Fo = Central frequency Hz
86          NoCycles = Number of cycles (integer)
87          SignalPolarity = [2 or 1 or -1], polarity of the gencode
88          Fs = Sampling frequency in Hz
89      Outputs
90          Signal
91      """
92      PeriodSamples = int(np.round(Fs/Fo))
93      Cycle = np.zeros(PeriodSamples)
94      Cycle[0:int(np.round(PeriodSamples/2))-1] = 1
95      Signal = np.zeros(1)
96      for i in range(0,NoCycles):
97          Signal = np.concatenate((Signal,Cycle))
98      # signal polarity
99      if SignalPolarity == -1 or SignalPolarity == 1:
100         Signal = Signal*SignalPolarity
101     else:
102         Signal=(Signal*2)-1
103         Signal[0]=0
```

Figure 49: GC_MakeBurst Funtion of GenCode_ToolBox Script

After sending the parameters to the GenCode page, the function we use sends these parameters to the GC_MakeBurst function in order to create Burst Signal.

```
191        #PulseWindow
192        def Burst(self):
193            self.BFreq1 = self.spinBox_3.value() * 1000000
194            self.BDur = self.spinBox_9.value() * 0.000001
195            self.BCyn = self.spinBox_12.value()
196            #print(Breq1,BDur,BCyn)
197
```

Figure 50: Getting Values from Burst Spin Boxes

In Fig. 50 shows that how we can get the values of Burst spin boxes. Additionally, as we can see on the Fig.52 the values multiply with numbers as we did in the Chirp button part.

Further, we are using a same method that we used in previous section 'spinbox.valueChanged.connect(self.function)' in order to connect with Burst function and spin boxes.

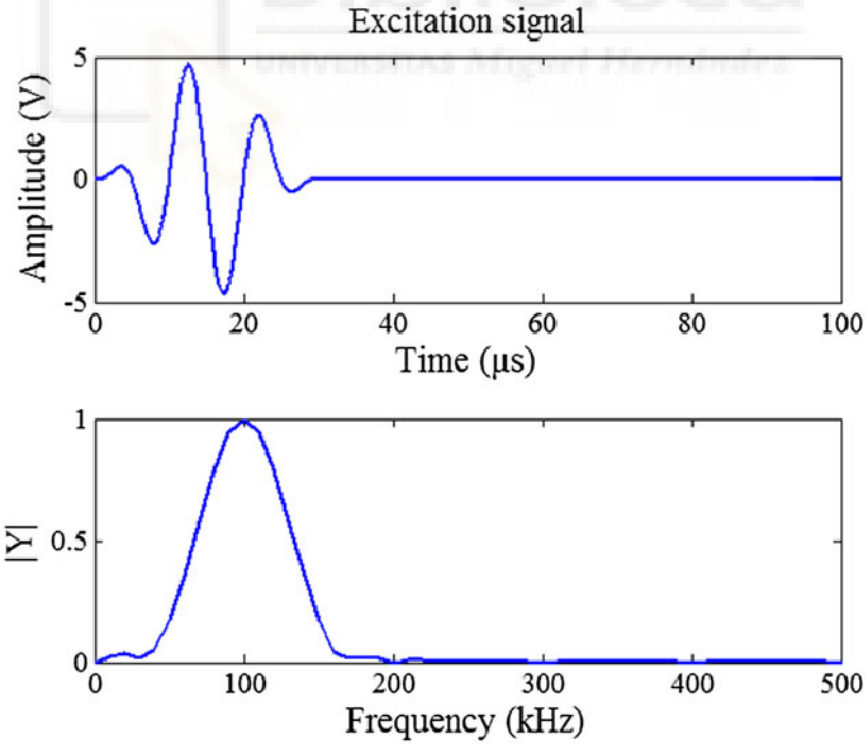

Figure 51: Burst Signal

In order to create burst signal like in the Fig.51, we need to use same parameters. These are frequency values, duration value and cycles value. That is the reason

why we need use three parameter in the function ‘gc.MakeGenCode’ in the Fig.49.

9. Pulse Button

```
153     def PulseSignal(self):
154         try:
155             self.Pulse()
156             PF1_int = self.PFreq1 #5 MHz
157             PDur = self.PDur # it depends on the Freq1 it s not really necesseraly
158             PProbingGencode = gc.MakeGenCode(Excitation='Pulse',
159                                              ParamVal=PF1_int)  # crear
160             self.Sedaq.UpdateGenCode(PProbingGencode)
161         except Exception as e:
162             if self.Sedaq == None:
163                 print("First click Sedaq and Connect with Sedaq Script")
164
```

Figure 52: Pulse Button

In the Pulse button function, we apply the same operations as in the chirp and burst button function. We are using function from GenCode_Toolbox script in order to create pulse signal and send it to UpdateGencode function from Sedaq Script. With this way, we are ordering to create pulse signal to Sedaq device.

The ‘gc.MakeGenCode’ function helps to send the values that we get from Burst Spin Boxes to GenCode_Toolbox script. Moreover, the ‘Sedaq.UpdateGenCode’ function is using for sending the signal that we created from another script to Sedaq.

```
21  def GC_MakePulse(Param = 'frequency', ParamVal = 5e6, SignalPolarity = 2, Fs = 200e6):
22      """
23      Make Pulse gencode
24      Inputs
25          Param = ['Frequency' or 'Duration' or 'Samples']
26          ParamVal = value, units must be [Hz or sec or Samples]
27          SignalPolarity = [2 or 1 or -1], polarity of the gencode
28          Fs = Sampling frequency in Hz
29      Outputs
30          Signal
31      """
32      if Param.upper() == 'FREQUENCY':
33          samples = int(np.round(1./ParamVal*Fs/2))
34      elif Param.upper() == 'DURATION':
35          samples = int(np.round(ParamVal*Fs))
36      elif Param.upper() == 'SAMPLES':
37          samples = int(ParamVal)
38      Signal = np.zeros(samples+2)
39      Signal[1:samples+1]=1
40      # signal polarity
41      if SignalPolarity == -1 or SignalPolarity == 1:
42          Signal = Signal*SignalPolarity
43      return Signal
```

Figure 53: GC_MakePulse Funtion of GenCode_ToolBox Script

After sending the parameters to the GenCode page, the function we use sends these parameters to the GC_MakePulse function in order to create Burst Signal.

```
181        #Pulse
182        def Pulse(self):
183            self.PFreq1 = self.spinBox_2.value() * 1000000
184            self.PDur = self.spinBox_8.value() * 0.000001
```

Figure 54: Getting Values from Pulse Spin Boxes

In Fig. 54 shows that how we can get the values of Pulse spin boxes.

Further, we are using a same method that we used in previous section ‘spinbox.valueChanged.connect(self.function)’ in order to connect with Burst function and spin boxes.

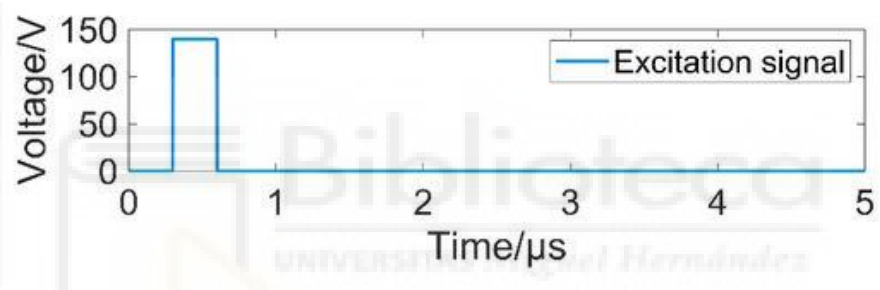

Figure 55: Pulse Signal

In order to create pulse signal like in the Fig.57, we need to use two parameters. These are frequency values and duration value. That is the reason why we need use two parameter in the function ‘gc.MakeGenCode’ in the Fig.52.

To give a footnote in this section, we must first connect to the Sedaq device to send Chirp, Burst and Pulse signals to the Sedaq device. Otherwise, we will receive the 'please connect with Sedaq' warning, which we added to the exception section.

10. Sample Spin Boxes

```
204     #Sample Frequency
205     def SampleFreq(self):
206         try:
207             self.SampleFreq1 = self.Spin_SFreq.value() * 1000000
208             print("Sample Frequency:", self.SampleFreq1)
209 
210         except Exception as e:
211             print("Please Connect with the Sedaq",e)
212 
213     #Sample Limit
214     def SetSamplim(self):
215         try:
216             self.SampleLimit = self.Spin_Slim_2.value()
217             print("Limit of Sample:", self.SampleLimit)
218 
219             self.Sedaq.SetRecLen(self.SampleLimit)
220 
221         except Exception as e:
222             print("Please Connect with the Sedaq",e)
223 
224     #Sample Min
225     def SetSampMin(self):
226         try:
227             self.SampleMinumum = self.Spin_Smin_2.value()
228             print("Sample Minimum :", self.SampleMinumum)
229 
230         except Exception as e:
231             print("Please Connect with the Sedaq",e)
232 
233     #Sample Max
234     def SetSampMax(self):
235         try:
236             self.SampleMaximum = self.Spin_Smax_2.value()
237             print("Sample Maximum:", self.SampleMaximum)
238         except Exception as e:
239             print("Please Connect with the Sedaq",e)
240 
```

Figure 56: Sample Spin Boxes

As we can see in the Fig 56, these codes help to reach the values of sample spin boxes. There are four types of the spin box in the program.
First one is setting the sample frequency of the device. This frequency value helps us to adjust the time axis of our graph. Moreover, this value is MHZ.

Second one is setting the limit of sample. With the value of limit sample, we are setting the max limit and it is around 32*1024.

Third and fourth one is for the setting min and max value of sample in order to see specific points that we want to see on the graph. Because the other values outside this region is not needed. Thus, we can analyze the data we want better.

Moreover, these functions are connected with spin boxes with ‘spinboxes.valueChanged.connect(self.funtion)’ method.

11. Check Boxes

```
241         def CheckBox1(self, checked):
242             if (Qt.Checked == checked):
243                 print('CH1 is ON')
244             else:
245                 print('CH1 is OFF')
246 
247         def CheckBox2(self,checked):
248             if (Qt.Checked == checked):
249                 print('CH2 is ON')
250             else:
251                 print('CH2 is OFF')
252
```

Figure 57: Check Boxes

There are two types of Check Boxes in the program. One of them is connected with Channel 1 of Sedaq and the other Channel 2 of Sedaq. The codes in the Fig. 57 helps to define whether the check boxes are on or off.

12. Starts Live Button - Graph

```
#Properties of the Graphs
# Set the Color of line in the Graph (below) and use the pg as pyqtgraph lib
#the With is setting the thicknes of line on the Graph and you can set the style as style method
pen = pg.mkPen(color=(255, 0, 0), width=1, style=QtCore.Qt.DashLine)
pen1= pg.mkPen(color=(0,0,255),width=1,style=QtCore.Qt.DashLine)
# Set the Grid
self.Grafic.showGrid(x=True,y = True)
self.Grafic1.showGrid(x=True,y = True)

#Seeting X and Y range of Graphs
self.Grafic.setXRange(0,self.SampleLimit, padding=0) #Setting the X range (padding is for range of number)
self.Grafic.setYRange(-0.5,0.5, padding=0) #Setting the Y range

self.Grafic1.setXRange(0,self.SampleLimit, padding=0)
self.Grafic1.setYRange(-0.5,0.5, padding=0)

# X Axises for Graphs
time_axis = np.arange(self.SampleMinumum,self.SampleMaximum)#* (1/self.SampleFreq1)
time_axis1 = np.arange(0,self.SampleLimit)
```

Figure 58: Feature of PyQtgraph

The starts live button in the program is using for the plot the graph. In this part, the first thing we did was to code the properties of our graph.

```
pen = pg.mkPen(color=(255, 0, 0), width=1, style=QtCore.Qt.DashLine)
pen1= pg.mkPen(color=(0,0,255),width=1,style=QtCore.Qt.DashLine)
```

These codes above help to set the color and thickness of the line on the graph. (13).

The code of `self.Grafic.showGrid(x=True,y = True)` is using for the set the grid on the graph (13).

Moreover, we need to define the x and y range of the graphs. In order to do that we are using these codes below: (13)

```
self.Grafic.setXRange(0,self.SampleLimit, padding=0) #Setting the X range (padding is for range of number)
self.Grafic.setYRange(-0.5,0.5, padding=0) #Setting the Y range
self.Grafic1.setXRange(0,self.SampleLimit, padding=0) #Setting the X range (padding is for range of number)
self.Grafic1.setYRange(-0.5,0.5, padding=0) #Setting the Y range
```

Additionally, we need to set the x axis of the graph that we mentioned pervious part. In order to define the x axis which is in time (sample) domain we need to use this formula:

```
time_axis = np.arange(self.SampleMinimum,self.SampleMaximum)
time_axis1 = np.arange(0,Sample.Limit)
```

```
self.Ascan_Ch2 = ACQ.GetAscan_Ch2(self.SampleMinumum, self.SampleMaximum, AvgSamplesNumber=self.AvgSampleNumber,
                                  Quantiz_Levels=1024)  # acq Ascan
self.Ascan_Ch1 = ACQ.GetAscan_Ch1(self.SampleMinumum, self.SampleMaximum, AvgSamplesNumber=self.AvgSampleNumber,
                                  Quantiz_Levels=1024)  # acq Ascan

self.Grafic1Ascan_Ch2 = ACQ.GetAscan_Ch2(0, self.SampleLimit, AvgSamplesNumber=self.AvgSampleNumber,Quantiz_Levels
self.Grafic1Ascan_Ch1 = ACQ.GetAscan_Ch1(0, self.SampleLimit, AvgSamplesNumber=self.AvgSampleNumber,Quantiz_Levels
```

Figure 59: CH1 and CH2 Graph Process

The codes in the Fig. 59 are that help to send the values that comes from spins to ACQ_ToolBox script in order to making a graph. Further, we are using this method for both CH1 and CH2.

```python
def GetAscan_Ch2(Smin, Smax, AvgSamplesNumber = 10, Quantiz_Levels = 1024):
    '''Get Ascan fomr channel 2 only, and extract data between Smin and Smax
    and normalizes according to quantization levels, and averages according to
    number of samples acquired at each point
        Inputs
            Sedaq = acq object
            Smin = First sample
            Smax = Last Sample
            AvgSamplesNumber = Number of Ascan to average in each acq
            Quantiz_Levels = Number of levels of acq (2^B)

        Outputs
            AscanCh2 = acquired ascans
    '''
    SeDaq = SeDaqDLL()
    Ascan_Ch2 = np.zeros(Smax-Smin)
    Flag = AvgSamplesNumber

    while Flag > 0:
        SeDaq.GetAScan() #get Ascan
        Aux_Ch2 = np.array(list(map(float,SeDaq.DataADC2[Smin:Smax]))) #get Ascan WP
        Aux_Ch2 = (Aux_Ch2 - Quantiz_Levels/2)/Quantiz_Levels # Normalize
        Aux_Ch2 = Aux_Ch2 - np.mean(Aux_Ch2) # remove mean

        if not(np.all(Aux_Ch2==0.0)):
            Ascan_Ch2 = Ascan_Ch2 + Aux_Ch2
            Flag -= 1

    Ascan_Ch2 = Ascan_Ch2 / AvgSamplesNumber #calculate averaged Ascan
    Ascan_Ch2 = Ascan_Ch2 - np.mean(Ascan_Ch2) #substract mean value
    return Ascan_Ch2

def GetAscan_Ch1(Smin, Smax, AvgSamplesNumber = 10, Quantiz_Levels = 1024):
    '''Get Ascan fomr channel 1 only, and extract data between Smin and Smax
    and normalizes according to quantization levels, and averages according to
    number of samples acquired at each point
        Inputs
            Sedaq = acq object
```

Figure 60: ACQ_ToolBox Script

Figure 60 shows us which function it is trying to achieve with the codes we use.

```python
if self.CheckBox_CH1.isChecked() and self.CheckBox_CH2.isChecked():

    try:
        self.Grafic.plot(time_axis,self.Ascan_Ch1, pen=pen)
        self.Grafic.plot(time_axis, self.Ascan_Ch2, pen=pen1)
        self.Grafic1.plot(time_axis1,self.Grafic1Ascan_Ch1, pen=pen)
        self.Grafic1.plot(time_axis1, self.Grafic1Ascan_Ch2, pen=pen1)
    except Exception as e:
        print('please connect with sedaq',e)
    print('CH1 and CH2 is clicked')


elif self.CheckBox_CH2.isChecked():
    try:
        self.Grafic.plot(time_axis,self.Ascan_Ch2, pen=pen)
        self.Grafic1.plot(time_axis1, self.Grafic1Ascan_Ch2, pen=pen1)


    except Exception as e:
        print('please connect with sedaq',e)
    print('CH2 is clicked')

elif self.CheckBox_CH1.isChecked():
    try:
        self.Grafic.plot(time_axis, self.Ascan_Ch1, pen=pen)
        self.Grafic1.plot(time_axis1, self.Grafic1Ascan_Ch1, pen=pen)
    except Exception as e:
```

Figure 61: Plotting the Graph

With the codes in Fig. 61, we plot the graphics from channel 1 or channel 2 according to whether the CH1 and CH2 boxes are on or off in our program.

As it can be seen in the program, we are using two graphs. While one of the graphs shows the entire signal, the other only shows the data in the sample min and sample max range that we have determined.

13. Clear Button

```
238        def ClearButton(self):
239            self.Grafic.clear()
```

Figure 62: Clear Button

The clear button is using for the clear the graph. The reason why is that the user needs to clean the graph in order to update it.

```
319        #Clear Button
320        self.CButton_Clear = QtWidgets.QPushButton(self.centralwidget)
321        self.CButton_Clear.setObjectName("CButton_Clear")
322        self.Buttons.addWidget(self.CButton_Clear)
323        self.CButton_Clear.clicked.connect(self.ClearButton)
```

Figure 63: Connection between Clear Button and Clear Function

The function above shows that how the connection between clear button and function made it.

14. Setting Icon

```
def SetIcon(self):
    #appIcon = QIcon("Logo1.png")
    appIcon = QIcon(r"C:\Users\User\Desktop\Final Project\Main Program\QtDesignerWindows\Logo1.png")
    app.setWindowIcon(appIcon)
```

Figure 64: Setting Icon of Program

In this section, we define an icon for our program by using the QIcon method by directing to the file where the photo is located. After defining the method, we need to call this function in the SetupUi function.

15. Spectrum Analyzer Check Box

The function in the Fig. 65 shows the how we can make a spectrum analyzer according to signal that we receive. The same graphic features are used in the function. The differences between the other graphic are the x and y ranges.

```
Self.Grafic1.setXRange(0,self.SampleFreq1/4)* Nfft)
Self.Grafic1.setYRange(0,20)

FTCh2 = np.abs(np.fft.fft(self.Ascan_Ch2,Nfft))
```

FTCh1 = np.abs(np.fft.fft(self.Ascan_Ch1,Nfft))

The methods above using for the creating spectrum analyzer based on the Ch1 and Ch2 signal we receiving from the Sedaq. Additionally, we are using special function which name is 'fft'.

```
#Spectrum Analyzer Window
def SA(self):

    self.Grafic1.clear()
    Nfft = 2048
    pen = pg.mkPen(color=(255, 0, 0), width=1, style=QtCore.Qt.DashLine)
    freq_axis = np.arange(Nfft) * (self.SampleFreq1 / Nfft)
    self.Grafic1.setXRange(0, self.SampleFreq1 / 4, padding=0)
    #self.Grafic1.setXRange(self.SampleMinumum, self.SampleMaximum , padding=0)
    self.Grafic1.setYRange(-1,20, padding=0)


    FTCh2 = np.abs(np.fft.fft(self.Ascan_Ch2, Nfft))
    FTCh1 = np.abs(np.fft.fft(self.Ascan_Ch1, Nfft))
```

Figure 65: Spectrum Analyzer Function

```
if self.CheckBox_SA.isChecked():

    # Plots depends on CH1 or CH2 clicked
    if self.CheckBox_CH1.isChecked() and self.CheckBox_CH2.isChecked():

        try:
            self.Grafic1.plot(freq_axis, FTCh2, pen=pen)
            self.Grafic1.plot(freq_axis, FTCh1, pen=pen)
            self.Grafic1.setXRange(0, self.SampleFreq1 / 4, padding=0)
        except Exception as e:
            print('Something Wrong', e)

    elif self.CheckBox_CH2.isChecked():

        try:
            self.Grafic1.plot(freq_axis, FTCh2, pen=pen)
            self.Grafic1.setXRange(0, self.SampleFreq1 / 4, padding=0)
        except Exception as e:
            print('Something Wrong', e)

    elif self.CheckBox_CH1.isChecked():
        try:
            self.Grafic1.plot(freq_axis, FTCh1, pen=pen)

        except Exception as e:
            print('Something Wrong', e)
```

Figure 66: Plotting Spectrum Analyzer

The codes in the Fig. 66 used for the plot the spectrum analyzer signals. We are using the second graphic on the program in order to plot the spectrum analyzer signal. Moreover, the plot method is connected with the spectrum analyzer checkbox and Ch1,Ch2 checkboxes.

D. Testing the GUI Program

After all the processes we mentioned in the previous sections are done, what we need to do is to test the program. Before testing the program, we need to know what is the instruction of the program. The instructions:

1. Connect with Sedaq
   We need to connect with Sedaq with function of self.Sedaq = SD.SedaqDLL() in order to use other function in the script.

2. Set Relay 'ON'
   If Relay is not ON, then the all we can see in the graph is noise.

3. Setting Samples
   We need to set samples value in order to define max and min limits on the graph. Such as Sample frequency, sample limit, sample min and sample max.

4. Setting ACQ Values
   In this part of instruction, we should set the voltage, gain1 and gain 2 values of the Sedaq device. After setting value is done, we need to click ACQ button in order to send these values to Sedaq.

5. Setting Chirp, Burst or Pulse Electrical Signal
   We must select one of the electrical signals such as chirp, burst or pulse and click one of these buttons and send it to the sedaq device.

6. Choosing CH1 or CH2
   We have to choose which channel we will use in the device.

7. Click Start Button
   This button using for the plot the signal on the graph.

8. Click Clear Button
   Clear button helps to clear signal on the graph.

9. Exit

After following the instructions written above, we can see the signal that receiving from the transducer in the graphics section of the program.

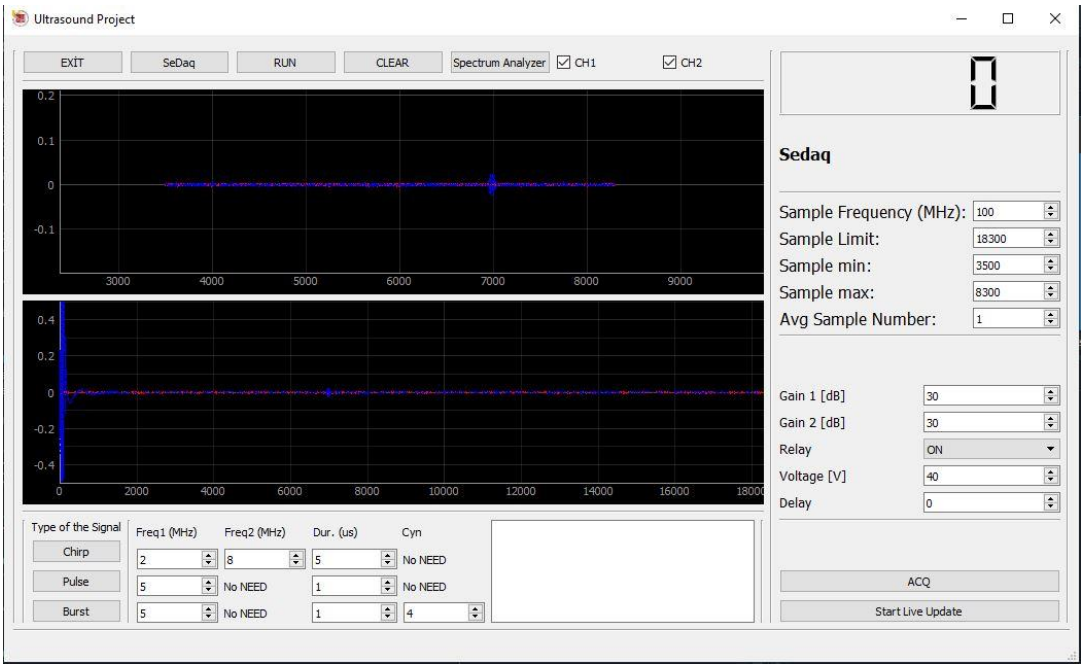

Figure 67: The Signal

In addition, if a problem occurs while following these instructions, the Sedaq device could not be connected. To solve this problem, the Sedaq connection must be checked, the device must be turned off and on, and the connection must be reestablished.

As it can be seen in the Fig. 67, the program is working perfectly. In order to test the program, we send the pulse signal and parameters like in the Fig,67., and two graphs shows the receiving signal without a problem. We tried the same test with burst and pulse signals to see that every button and spin of the program works correctly.

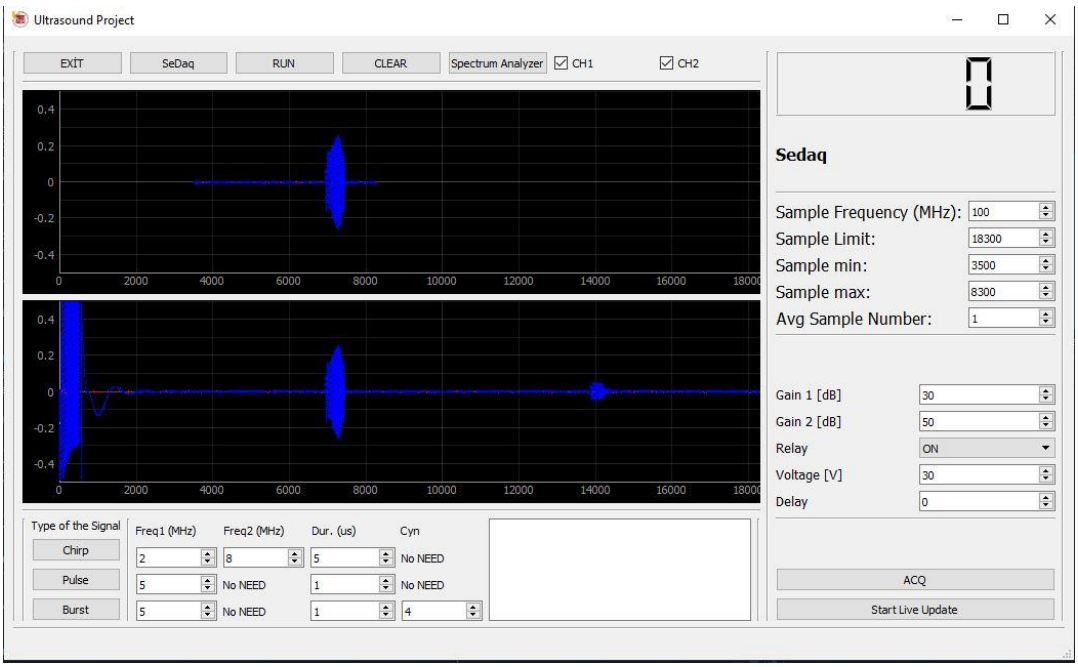

Figure 68: The Chirp Signal Result

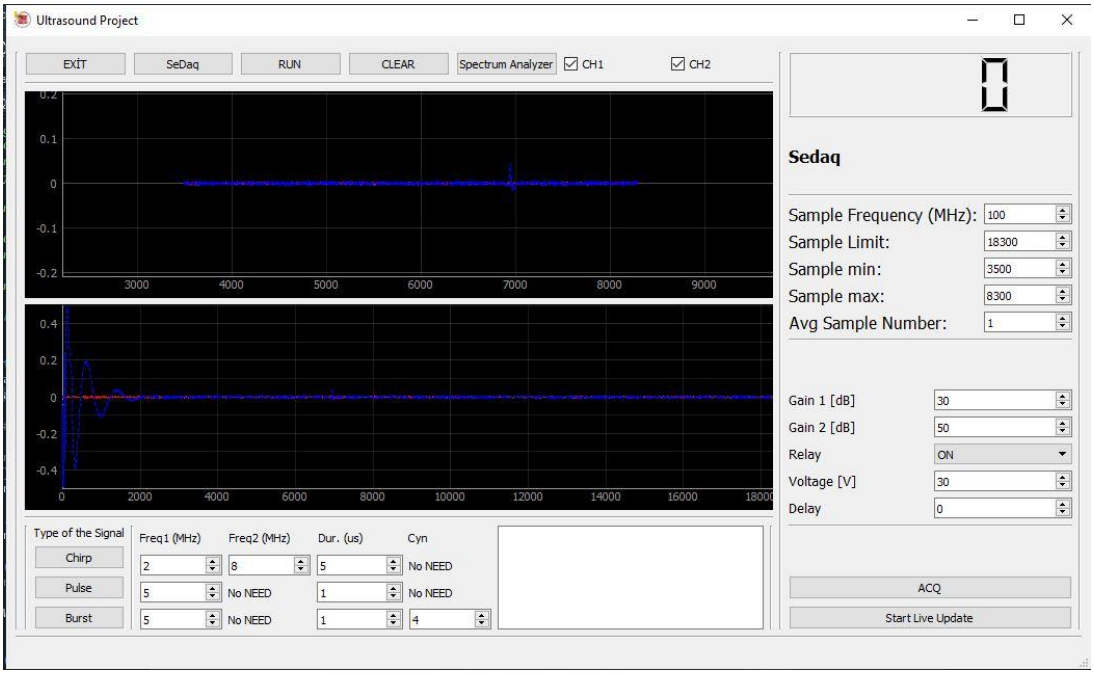

Figure 69: The Pulse Signal Result

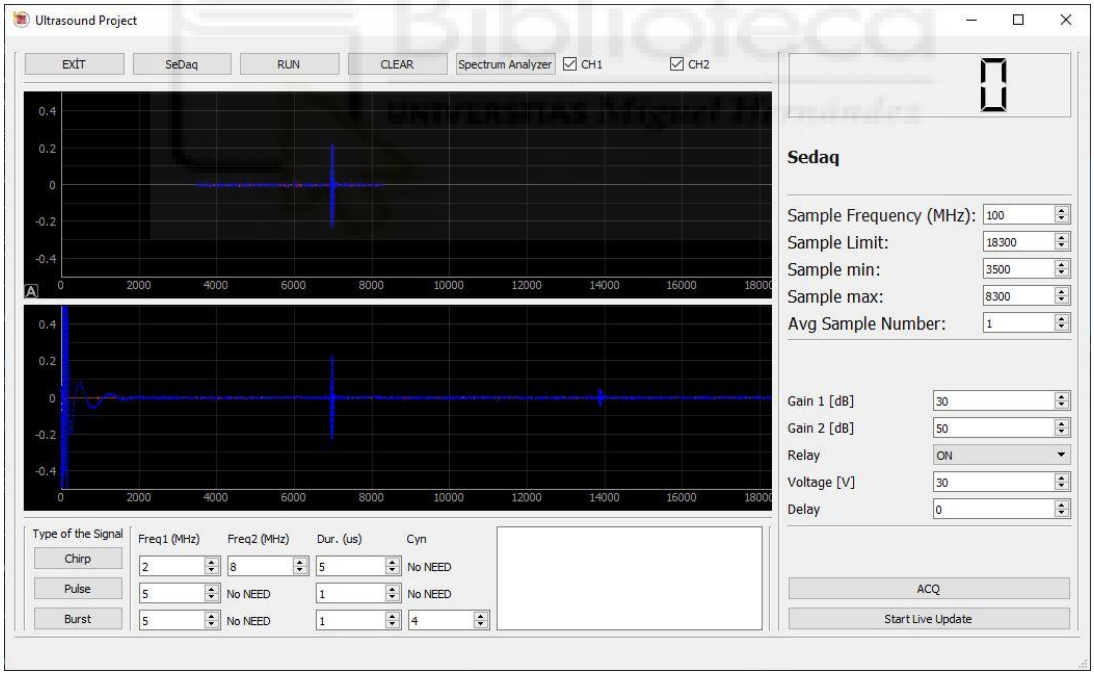

Figure 70: The Burst Signal Result

As we can see in the figures above, our program is working properly.

E. Analyzing the Signals

In the last part of the project, all the steps were followed and carried out successfully. What needs to be done is to check whether the program is working properly with a small experiment. The purpose of this experiment will be to analyze the 3 types of signals (Chirp, Burst, Pulse) we send as electrical signals and to compare their accuracy.

First of all, we fill an empty box with some water. Then we put a small piece of metal inside the water. Further, we place the transducer device in contact with water as shown in the figure. We are using the water inside the box to propagate the ultrasound vibrations from the transducer. Because the ultrasound vibration cannot transmit in the air. That is the reason the doctors are using some kind of liquid when they are using a sonography device.

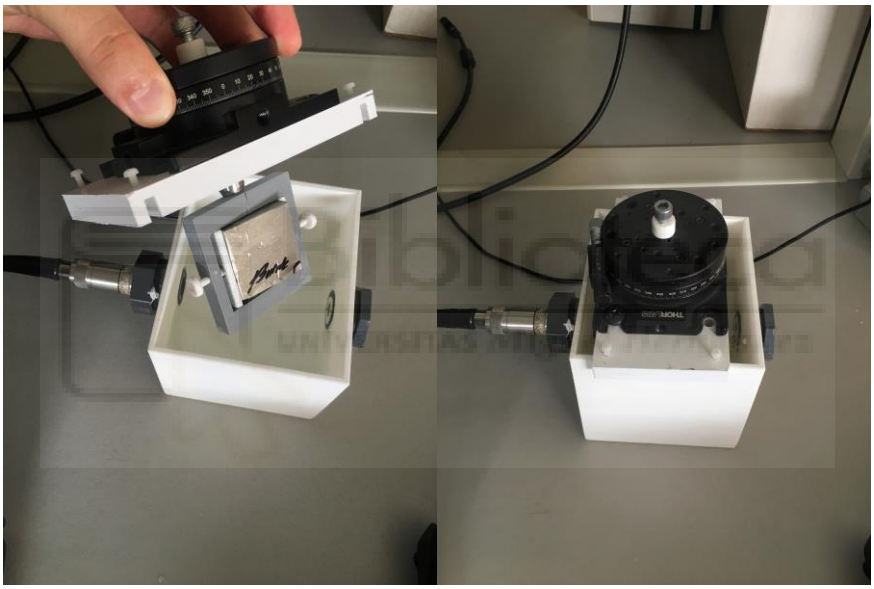

Figure 71: The Box and Transducer

After we connected the transducer with the Sedaq, we need to open our program in the computer. Yet, we need to use 32Bit compiler python program in order to operate the Sedaq. The reason why is that the hardware of the Sedaq is 32 Bit. And, we are using a 32Bit Spyder compiler.

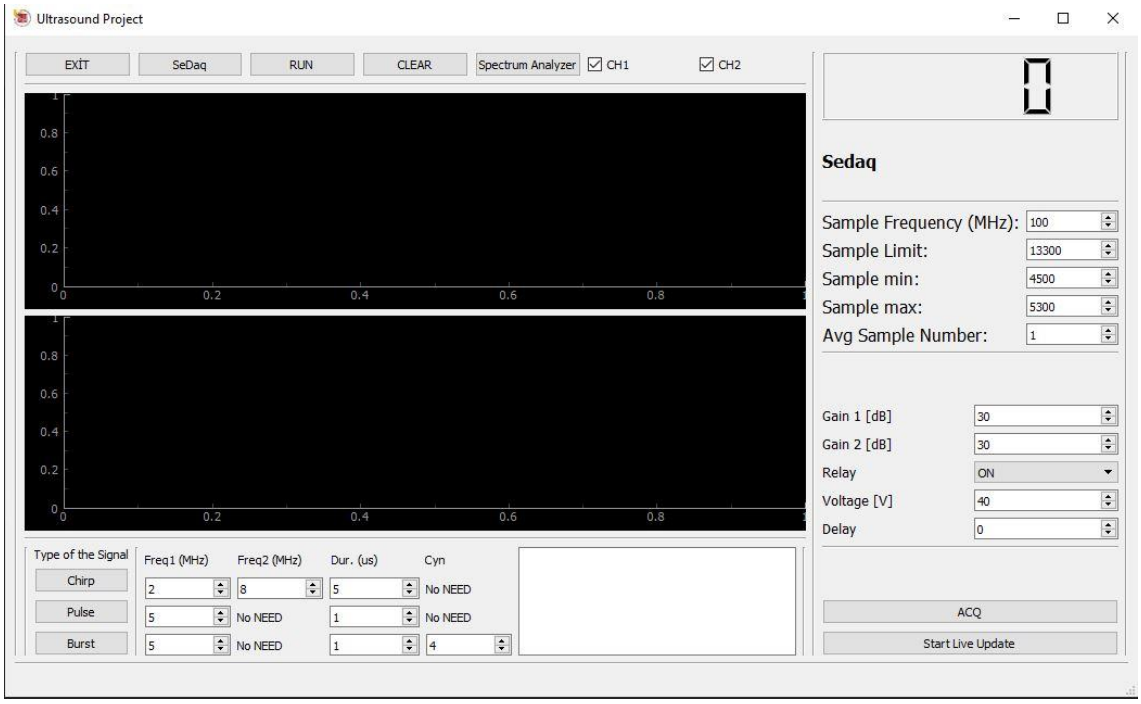

Figure 72: The GUI Program

Moreover, the only thing that we need to do is that follow the instruction of program in order to use properly and analyze the signal that we will receive from the Sedaq and transducer.

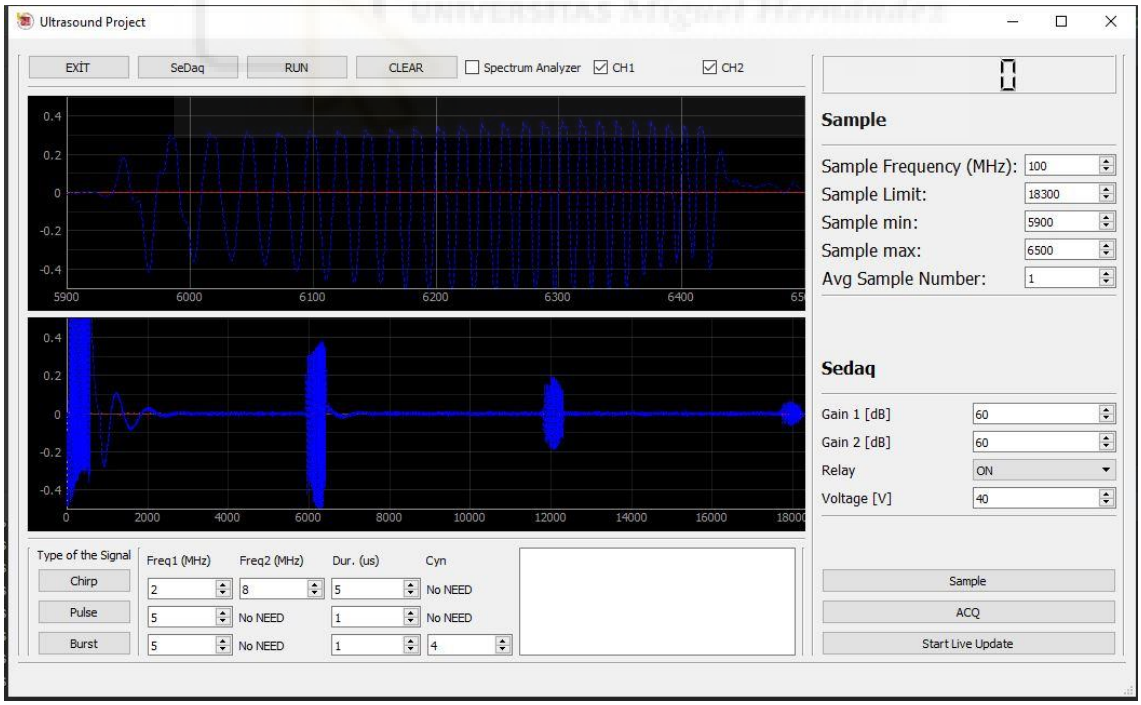

Figure 73: Receiving Chirp Signal

The first thing that we did it in the experiment is that we ordered the program to send a chirp signal. We set the parameters as shown in Fig. 73. Then, we were able to see the whole and part of the received signal with the two graphs that we used in the program.

Figure 74: Spectrum Analyzer of Chirp Signal

After seeing the receiving chirp signal in the graph perfectly, we need to use the spectrum analyzer checkbox in order to see the spectrum analyzing signal for the specific range of the signal. As it can be seen in the Fig. 74 the spectrum analyzer part working perfectly.

Additionally, the amplitude of the spectrum analyzer signal depends on the cycles of the signal that we are receiving in the first graph. That means if we increase the cycles of the signal, then the amplitude of spectrum analyzer will increase in the same time. Further, the bandwidth of the spectrum analyzer depends on the shape of the receiving signal. Therefore, if we change the shape of the signa, then the bandwidth of the spectrum analyzer will be changed.

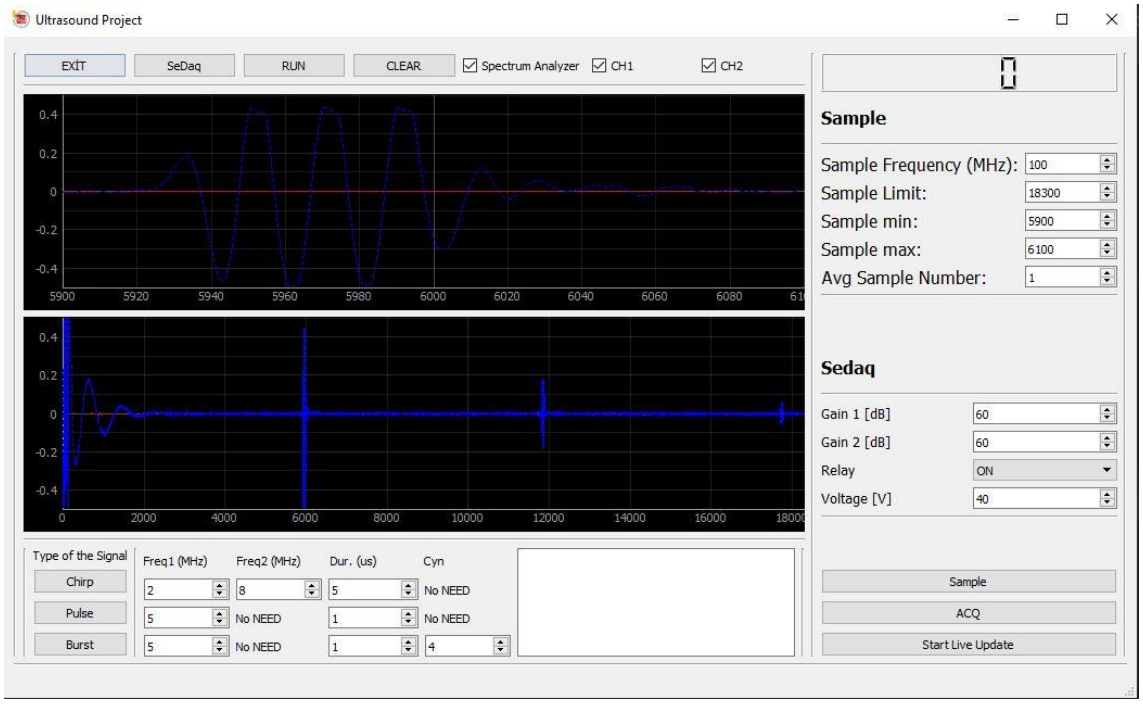

Figure 75: Receiving Burst Signal

Secondly, when we use the Burst signal as an electrical signal in the project, the receiving signal will be like in the Fig. 75. Also, as we can see in the graph the amplitude of the signal is higher. The reason why is that we increase the gain of the Sedaq. Therefore, the amplifier in the Sedaq amplify the receiving signal.

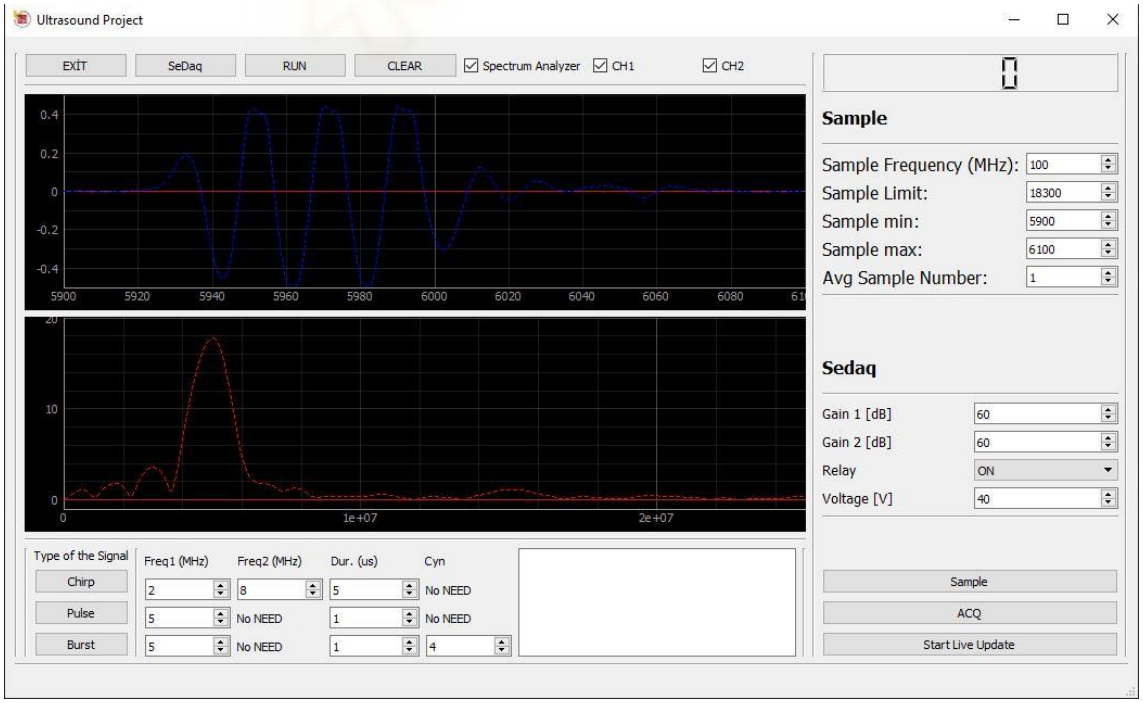

Figure 76: Spectrum Analyzer of Burst Signal

In the Fig. 76, we can see the spectrum analyzer of the specific part of the signal in the graph. Moreover, the amplitude and the bandwidth of the spectrum analyzer signal is different. Because the shape and cycle of the receiving burst signal is different from the previous one. This shows us that our spectrum analyzer graph is running smoothly.

Figure 77: Receiving Pulse Signal

The last part of the analyzing signal is that receiving pulse signal in the graph. We set the frequency of the pulse electrical signal as a 5 MHz and the duration value is 1us. After that we send the signal into the Sedaq. Thus, the Fig.77 shows us the receiving pulse signal occur smoothly.

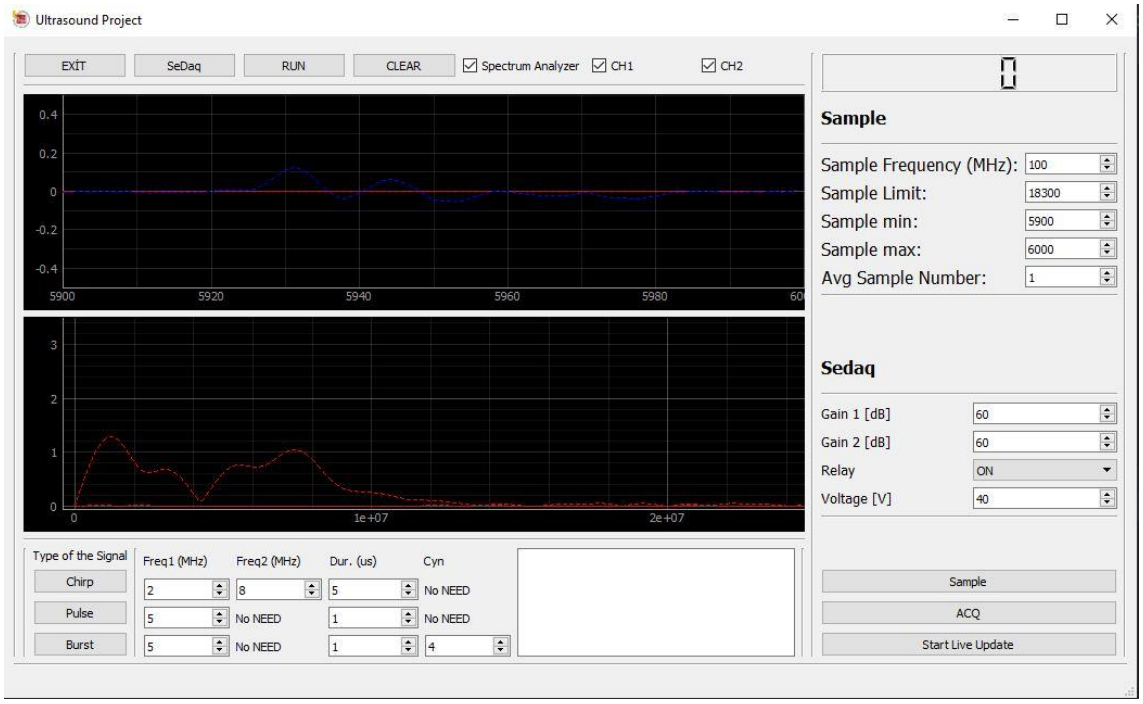

Figure 78: Spectrum Analyzer of Pulse Signal

The spectrum analyzer of the Pulse signal can be seen in the Fig. 78. Because of the pulse signal shape, the spectrum analyzer shape is more different than other electrical signals. Also, the cycles of the pulse signal are lower than the others. Therefore, the amplitude of the spectrum analyzer is lower.

As can be seen the all figures above, the signal that we are receiving from the Sedaq and transducer are perfectly seen in the graphs. All the buttons and spin values in the program have been tested and proven to work correctly.

The only part that may cause a problem is that the gain part of the Sedaq. The reason why is that the method that we are using in the Fig.36 are not correctly corresponding. Sometime, it may not work. The reason for this problem is that the Sedaq device we use and the program it is installed in are old and the codes cannot be accessed. Thus, the methods that we are using in the program are pretty old. Therefore, this situation may able to create like this problem. Yet, after completing the signal analysis part, which is the last part of the project, we have seen the program works properly.

5. Conclusion

Many developments and innovations have been made in the field of medicine with the developing ultrasound technology since the 20th century. The success of this project is proof that this situation is still ongoing.

With the user interface program, which is the aim of the project, the control of Sedaq and transducer electronic devices has been successfully done. The next step of this project, the electrical signal type, frequency or bandwidth values that will best analyze the ultrasound signals will be investigated with this program.

In the project, first of all, ultrasound signals and electronic devices used in the project were investigated. Further, the process of the user interface program has been explained. After the GUI program was built, the control of the electronic devices and the plotting of the ultrasonic signal on the graphic were made. After the all processes are done, the successful operation of the project has been proven by a small experiment. In this experiment, a small sample of an experiment to be made in the later stages of the project was made it. With this experiment the location and thickness of the material has been found.

At the end of the project, a GUI that can control the Sedaq device and transducer has been made and these devices have been successfully controlled. Therefore, it can examine and investigate the ultrasound signals that receiving from the transducer.

This project is a small part of the big project that aim to determine the thickness and location of the veins in our body with small budget and small devices. The successful conclusion of the project proves that the step of the big project has been taken.

6. Bibliography

Figure 1: Ultrasonic Transducer (November,05,2018)- Pro-Wave Electronics 300HE100 High Frequency Air Ultrasonic Transducer – Retrieved: May,2,2021- From https://www.mouser.es/new/pro-wave-electronics/pro-wave-300he100-transducers/

(1) Richard R.Berg (October,06,2017) – Transducers- Retrieved: May,2,2021 - From https://www.britannica.com/science/ultrasonics#ref64031

(2) K. K. Shung, M. Zippuro(Nov-Dec,1996) – Ultrasonic Transducer and Arrays – Retrieved: May,5,2021 – From https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=544509

Figure 3: Leonardo Romero, Antonio Concha (July,08,2014)- Building a Ring of Ultrasonic and Infrared Sensors For a Mobile Robot – Retrieved: May,5,2021 – From: https://www.researchgate.net/figure/Ultrasonic-sensor-operation_fig3_237722717

(3) Elprocus (2015) – What is Ultrasonic Transducer: Working and its Applications – Retrieved: May,6,2021 – Form: https://www.elprocus.com/ultrasonic-transducer-working-and-its-applications/

Figure 6: Extron (2021) – GUI Designer – Retrieved: May,7,2021 – From: https://www.extron.es/product/software/guidesigner

(4) Nexxytech (May,21,2020) – What is the GUI? – Retrieved: May,10,2021 – From: https://nexxytech.com/2020/05/21/what-is-the-gui-which-language-is-best-to-build-its-program/

(5) Claire D. Costa (Dec,7,2020) – Top 10 Python GUI Frameworks for Developer – Retrieved: May,10,2021 – From: https://towardsdatascience.com/top-10-python-gui-frameworks-for-developers-adca32fbe6fc

(6) IOWA State University () – Ultrasonic Testing – Retrieved: May,10,2021 – From:https://www.nde-ed.org/NDETechniques/Ultrasonics/Introduction/description.xhtml

(7) Paul G. Newman, Grace S. Rozycki (May,25,2005) – The History of Ultrasound – Retrieved: May,11,2021 – from: https://www.sciencedirect.com/science/article/abs/pii/S003961090570308X

(8) Wikipedia (May,6,2021)- Ultrasound – Retrieved: May,11,2021 – From: https://en.wikipedia.org/wiki/Ultrasound#cite_note-squires-1

Figure 7: Dr. Joseph Woo (1998) - A short History of the development of Ultrasound in Obstetrics and Gynecology – Retrieved: May,15,2021 – From: https://www.ob-ultrasound.net/history1.html

(9) , Figure8: IOWA State University () – Ultrasonic Testing – Retrieved: May,10,2021 – From:https://www.nde-ed.org/NDETechniques/Ultrasonics/Introduction/description.xhtml

(10) Rajendra Dharmkar (Jan,12,2018) – What is differences between self and __init__ methods in python class? – Retrieved: May,15,2021 – From: https://www.tutorialspoint.com/What-is-difference-between-self-and-init-methods-in-python-Class

(11) Python (Jume,02,2021) – Classes – Retrieved: May,27,2021 – From: https://docs.python.org/3/tutorial/classes.html

(12) Karthikeya Boyini (Feb,19,2019) – Differences between Method and Function in Python – Retrieved: May,27,2021 – From: https://www.tutorialspoint.com/difference-between-method-and-function-in-python

Figure 45: S.Harput (2012) – Use of Chirps in Medical Ultrasound Imaging – Retrieved: June,01,2021 – From: https://www.semanticscholar.org/paper/Use-of-chirps-in-medical-ultrasound-imaging-Harput/e24663f442226c55e0f9bbb0e069dc7ba6ba0156

Figure 51: Wentao Wang, Hui Zang, Jerome P. Lynch (March,2018) - Experimental and numerical validation of guided wave phased arrays integrated within standard data acquisition systems for structural health monitoring – Retrieved: June,01,2021 – From: https://www.researchgate.net/figure/Basic-tone-burst-signal-in-the-time-top-and-frequency-bottom-domains_fig2_323777530

Figure 57: Lecheng Jia, Bin Xeu, Shili Chen (March, 04, 2019) - A High-Resolution Ultrasonic Ranging System Using Laser Sensing and a Cross-Correlation Method – Retrieved: June,01,2021 – From: https://www.mdpi.com/2076-3417/9/7/1483/htm

(13) John Lim (Oct,12,2019)- Plotting with PyQtGraph – Retrieved: June,02,2021 – From: https://www.mfitzp.com/tutorials/plotting-pyqtgraph/