



**UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH**

**Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa**

Bachelor's degree thesis

Bachelor's degree in Audiovisual Systems

Design of an interface of an automatic equalization system

REPORT

Student: Paula Andrea Chacón Cabrera

Bachelor's degree supervisor: Néstor Berbel Artal

Call: Extraordinary April 2021

Bachelor's degree:

Bachelor's degree in Audiovisual Systems

Student (Name and Surname):

Paula Chacón

Bachelor final degree statement:

Design of an interface of an automatic equalization system

Bachelor's degree supervisor:

Néstor Berbel Artal

Call:

Extraordinary April 2021

ACKNOWLEDGMENTS

First and foremost, I want to thank my research professor, he has been an excellent guide during all this process. His patience has allowed me to learn as much as I could and to bring this project to life.

And want to thank my parents and sister for being my motivation and biggest support.

ABSTRACT

The idea of this project is to create an interface for an automatic equalizer. But first, a general idea of what an equalizer is.

The sole purpose of an equalizer is to bring the different frequencies there are on a range, to the desired value. So, the process is done by sending pink noise to a microphone and then seeing, depending on the place's infrastructure, how the frequencies are affected. An automatic equalizer uses filters to give more or take some dB's, depending on the desired spectrum.

This program already exists, but it is still missing the interface for the user to be able to operate with it. So that is what this project is about, giving the user the easiest interface to interact with to use the program most efficient way. Therefore, one of the main objectives of this project is to simplify the graphic user interface as much as possible.

The execution of this project started with the configuration of a Raspberry Pi, which is where the app is being implemented. The second thing that had to be done was the coding of the app, it was done by using Python as the programming language and Tkinter as the graphical library. The results were as desired, the app is very simple and easy to interact with.

RESUMEN

La idea principal de este proyecto es crear una interfaz para un ecualizador automático. Pero primero, una idea generalizada de lo que un ecualizador es.

El objetivo de un ecualizador es llevar ciertas frecuencias a un rango deseado. Este proceso se logra mediante la emisión de un ruido rosa, que dependiendo del lugar donde haya sido reproducido, se verá afectado y así mismo los distintos rangos de frecuencia que lo conforman. Entonces, el ecualizador se encargaría de hacer esta emisión y captura, para luego modificar, de manera automática los valores iniciales a los deseados, estos estarán determinados por uno de los filtros que conforman el ecualizador.

Este programa ya existe, fue creado en un proyecto previo a este, pero le falta la interfaz para que pueda ser implementado por un usuario. Y de eso se trata este proyecto. El objetivo es crear una aplicación sencilla y eficaz que permita la rápida interacción con el usuario para poder hacer el proceso de ecualización más rápido.

El desarrollo de este proyecto empezó por la configuración de una Raspberry Pi, que es donde se implementará la aplicación. Luego se siguió con la programación del código. Este se programó en Python y con Tkinter como librería gráfica. Los resultados fueron exitosos y la aplicación salió como se esperaba. Sencilla, minimalista y eficaz.

CONTENTS

1. INTRODUCTION	1
2. HARDWARE	3
2.1. Start-Up of the Raspberry	3
2.1.1. OS installation.	4
2.1.2. SSH configuration	4
2.1.3. Raspberry's IP	5
2.1.3.1. Define the static IP address of the Raspberry	5
2.1.3.2. Download PuTTY	6
2.1.3.3. Enable VNC Server at the command line	7
2.1.4. VNC Configuration	10
2.1.5. Fixed IP	12
3. SOFTWARE	17
3.1. Tkinter	18
3.1.1. OOP and EDP	18
3.1.2. Classes and Objects with Tkinter	18
3.1.2.1. Basics of Tkinter	18
3.1.2.2. Classes and Objects in Tkinter	20
3.1.2.3. Positioning of Widgets.	21
3.2. Code explanation	21
4. EXPERIMENTAL RESULTS	25
4.1. Screenshots of the Results	25
5. CONCLUSIONS	31
6. BIBLIOGRAPHY	33

LIST OF FIGURES

FIGURE 1-1. SOFTWARE OF PROGRAMMING	2
FIGURE 2.1.1-1 RASPBERRY PI IMAGER	4
FIGURE 2.1.3 – 1 RASPBERRY IP	5
FIGURE 2.1.3.2-1 PUTTY CONFIGURATION.....	6
FIGURE 2.1.3.3-1 RASPBERRY'S COMMAND PROMPT	7
FIGURE 2.1.3.3-2 RASPBERRY'S SOFTWARE CONFIGURATION	8
FIGURE 2.1.3.3-3 ENABLING VNC CONNECTION.....	9
FIGURE 2.1.3.3-4 VNC CONNECTION ENABLED	9
FIGURE 2.1.4-1 VNC LOGO.....	10
FIGURE 2.1.4-2 VNC INTERFACE	11
FIGURE 2.1.4-3 STARTING VNC CONNECTION.....	11
FIGURE 2.1.4-4 SUCCESSFUL VNC CONNECTION.....	12
FIGURE 2.1.5-1 RIGHT CLICK WI-FI ICON	13
FIGURE 2.1.5-2 NEW IP ADDRESS INTRODUCED	13
FIGURE 2.1.5-3 REBOOT RASPBERRY	14
FIGURE 2.1.5-4 VNC CONNECTION WITH NEW IP	14
FIGURE 2.1.5-5 NEW IP ADDRESS CHECKED.....	15
FIGURE 3-1. TKINTER	17
FIGURE 3.1.2.1-1 BASIC EXAMPLE TKINTER	19
FIGURE 3.1.2.1-2 FRAGMENT OF CODE	20
FIGURE 3.1.2.2-2 BASIC EXAMPLE WEB PAGE.....	20
FIGURE 3.1.2.2-2 WEB EXAMPLE.....	21
FIGURE 3.2-1 CONCEPTUAL MAP	23
FIGURE 4.1-1. PAGE 1 – INIT PAGE	26
FIGURE 4.1-2. PAGE 2 - RECORDING PAGE	27
FIGURE 4.1-3. PAGE 3 - EQUALIZATION PAGE	28
FIGURE 4.1-4 PAGE 4 - CLOSING PAGE.	29

1. INTRODUCTION

Although the program that does the automatic equalizations already exists, an interface that allows a good experience between the software and the client is still needed. The intention is for it to be simple and intuitive for the user. That is going to enable the process to be much faster and more efficient, which is the goal of the equalizer and therefore it is also a goal of the interface.

On a previous final degree project¹, an automatic equalization system was designed and implemented, employing a Teensy 3.6 microprocessor and a graphical interface. The automatic equalization system receives the audio signal, it's processed using an analog preamplifier to adequate the signal levels and sampled at a sampling frequency higher than 40 kHz. The system performed correctly but due to the lack of time, its interface was not designed properly.

When working on a live show or in the acoustics of a place, trying to get them into a specific range or dB value can be challenging and time-consuming, that is why one of the main objectives of this project is to make the interaction between the user and the program as simple and as fast as possible. As the saying goes, sometimes less is more.

The interface design is going to be implemented employing a Raspberry Pi 3B, and a capacitive display of 7" is connected to the computer. Therefore, to design and implement the final degree project, the Raspberry Pi must be initialized with its operating system, connect to the internet, and finally, programme the interface.

It will be necessary to have software that allows the remote connection of the raspberry with a computer, in this case it is going to be “Visual Studio Code”², and the language will be Python



Figure 1-1. Software of programming

2. *HARDWARE*

For the hardware there were three options, the first one was a computer, but because of the pricing it was discarded and one of the objectives of this project is to make it low cost. The second option was a microcontroller/microprocessor, but since the programming was in C and in need of an assembly, more options had to be considered, and that led to the C/Python language. In search of suitable pricing, the best option was a mini PC, in this case, the Raspberry Pi.

The Raspberry Pi has many advantages, one of them being the low cost, it is approximately 40 €, and has a large amount of processing power in a compacted board. Since we are working with a tablet, it is also a perfect fit because it has many interfaces, such as HDMI, USB, Ethernet, and of course, Wi-fi and BlueTooth. And the reason that the chosen language was Python, is because for computer apps it's the widely accepted language.³

2.1. Start-Up of the Raspberry

To start the Raspberry, the OS was needed to be installed, then the wifi and Virtual Network Connection (VNC) had to be configured.

2.1.1. OS installation.

The Raspberry already has an OS, Raspbian OS⁴. It can be downloaded using the official page or by using the Raspberry Pi Imager⁵ which downloads the OS and performs the installation on the SD card. After the installation of the raspbian OS on the SD card, it can be inserted on the Raspberry Pi.



Figure 2.1.1-1 Raspberry Pi Imager

2.1.2. SSH configuration

The SSH is the remote connection that works by terminal commands that will allow the control of the Raspberrypi without it having a keyboard or even a screen.

To get the SSH running, a file must first be created in the boot partition.⁶ For the headless setup of the Raspberry Pi, SSH can be enabled by placing a file named “ssh”, without any extension, onto the boot partition of the SD card from another computer. When the Pi boots, it looks for the file that has been created previously. If the file is found, SSH is enabled and it is deleted. The content of the file does not matter; it could contain text or nothing at all.

If you have loaded Raspberry Pi OS onto a blank SD card, you will have two partitions. The first one, which is the smaller one, is the boot partition. The file should be placed into the smaller partition.

2.1.3. Raspberry's IP

To find the IP address of the Raspberry, an external software called "Advanced IP Scanner"⁷ was used, it can scan a whole network or just a part of it. The implementation is quite easy, once the software was installed all that had to be done was to let the app explore to find all the devices working in the network.

As it can be seen on the screen (Fig 2.1.3-1), the IP is 192.168.1.105

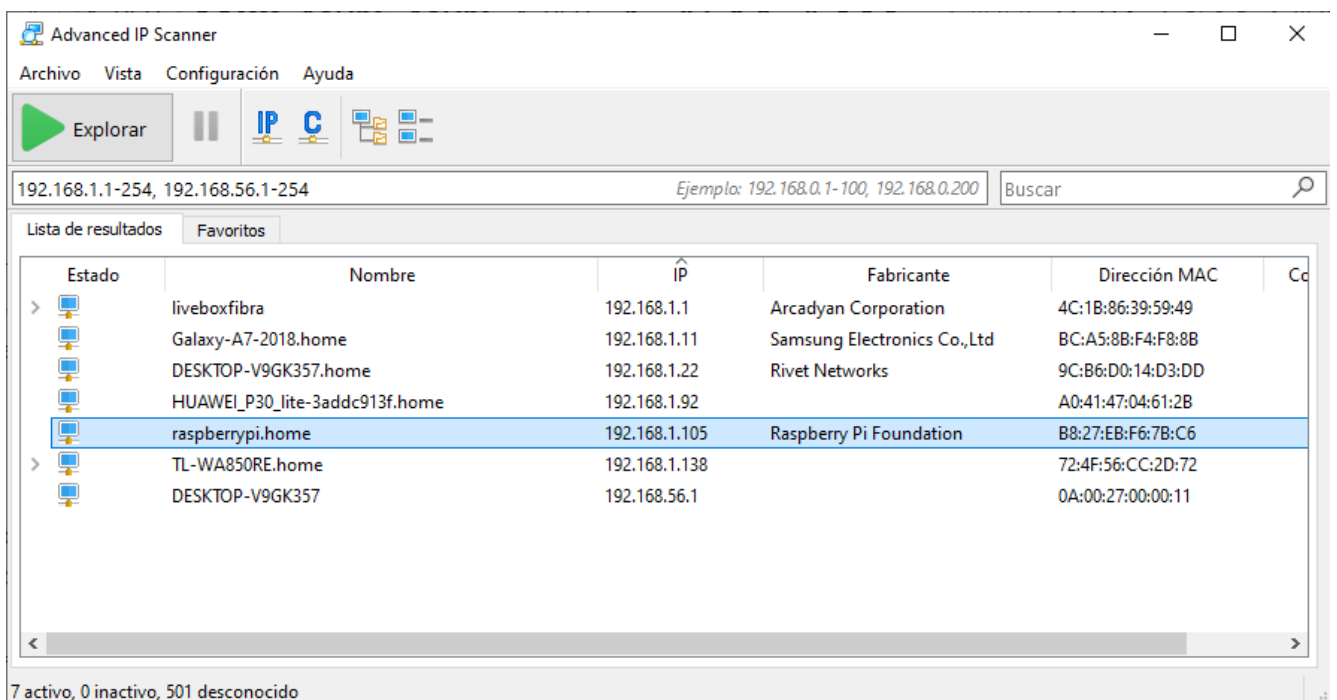


Figure 2.1.3 – 1 Raspberry Ip

2.1.3.1. Define the static IP address of the Raspberry

Every time a connection with the Raspberry is needed, if the IP depends on the DHCP (Dynamic Host Configuration Protocol), the IP Scanner has to be executed. To make that

process faster in the next connections, the Raspberry's IP is going to be configured static and the future connections can be done just by executing the VNC application and introducing the fixed one.

2.1.3.2. Download PuTTY

To first interact with the Raspberry, remote access is needed and for that, the PuTTY app is used.

The IP previously obtained by the "Advanced Ip Scanner" is used in the first window of the PuTTY executable.(Fig 2.1.3.2-1)

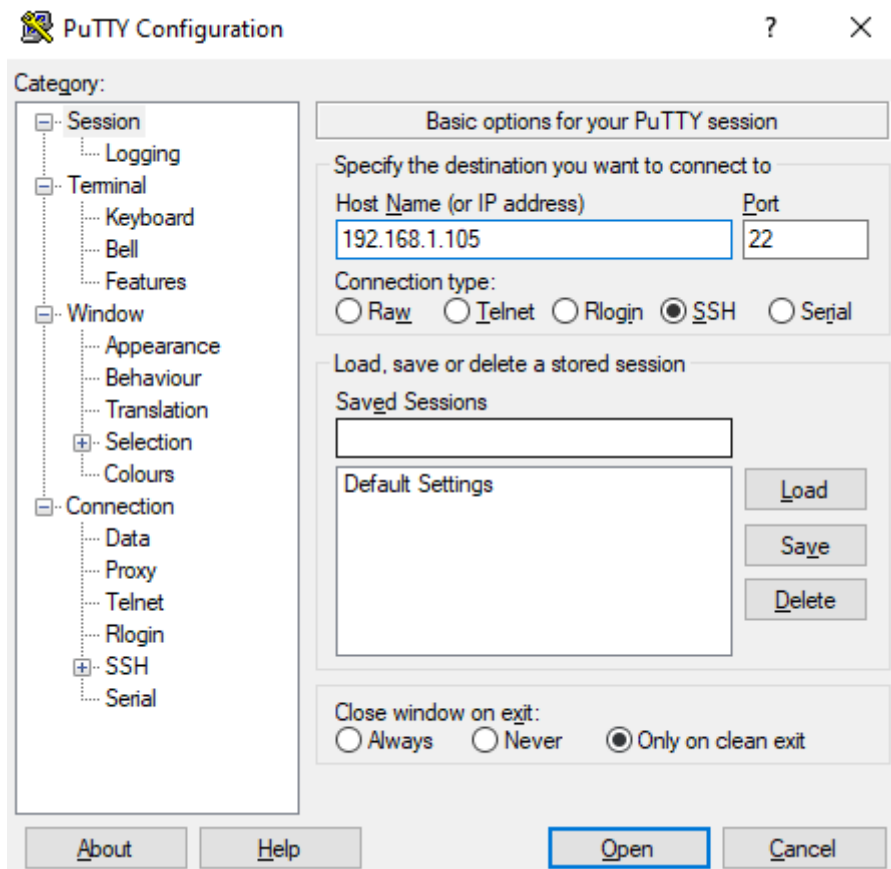
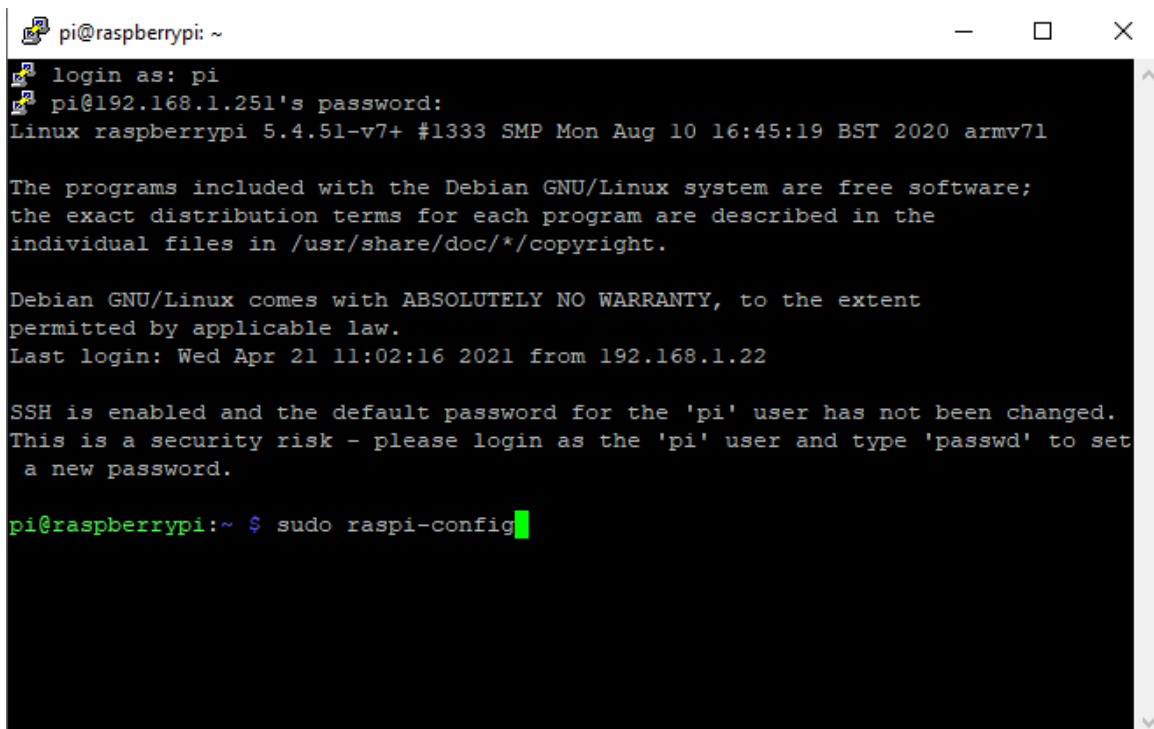


Figure 2.1.3.2-1 PuTTY configuration

2.1.3.3. Enable VNC Server at the command line

To work in the Raspberry from another device by remote control, VNC is used. It is a graphical desktop sharing system that allows to remotely control the desktop interface of the computer (running VCN Viewer). The Raspberry's OS has VNC but needs to be enabled. This can be done graphically or at the command line.

When connecting with PuTTY, it gives access to the Raspberry by the Command Prompt(Fig 2.1.3.3-1), and there the `sudo raspi-config` command is run.



```
pi@raspberrypi: ~  
login as: pi  
pi@192.168.1.251's password:  
Linux raspberrypi 5.4.51-v7+ #1333 SMP Mon Aug 10 16:45:19 BST 2020 armv7l  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Wed Apr 21 11:02:16 2021 from 192.168.1.22  
  
SSH is enabled and the default password for the 'pi' user has not been changed.  
This is a security risk - please login as the 'pi' user and type 'passwd' to set  
a new password.  
  
pi@raspberrypi:~ $ sudo raspi-config
```

Figure 2.1.3.3-1 Raspberry's Command Prompt

It will show the Software Configuration Tool (Fig 2.1.3.3-2), clicking on the Interface Option and saying yes will allow the VCN to be enabled.

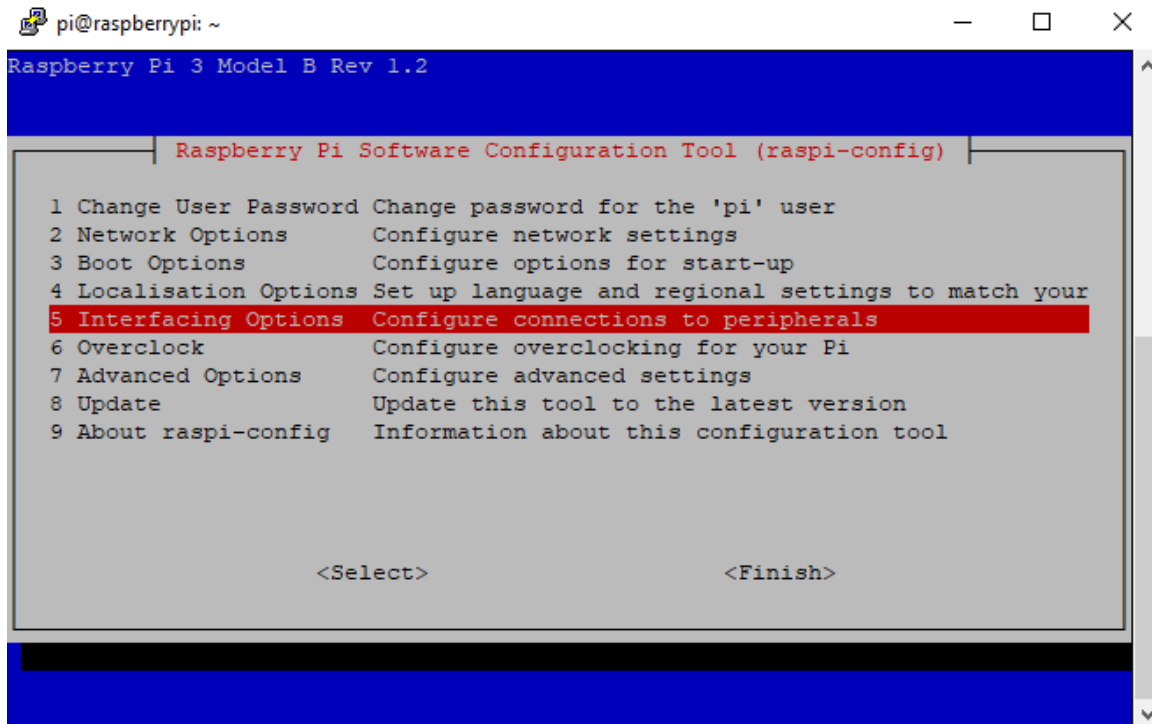


Figure 2.1.3.3-2 Raspberry's Software Configuration

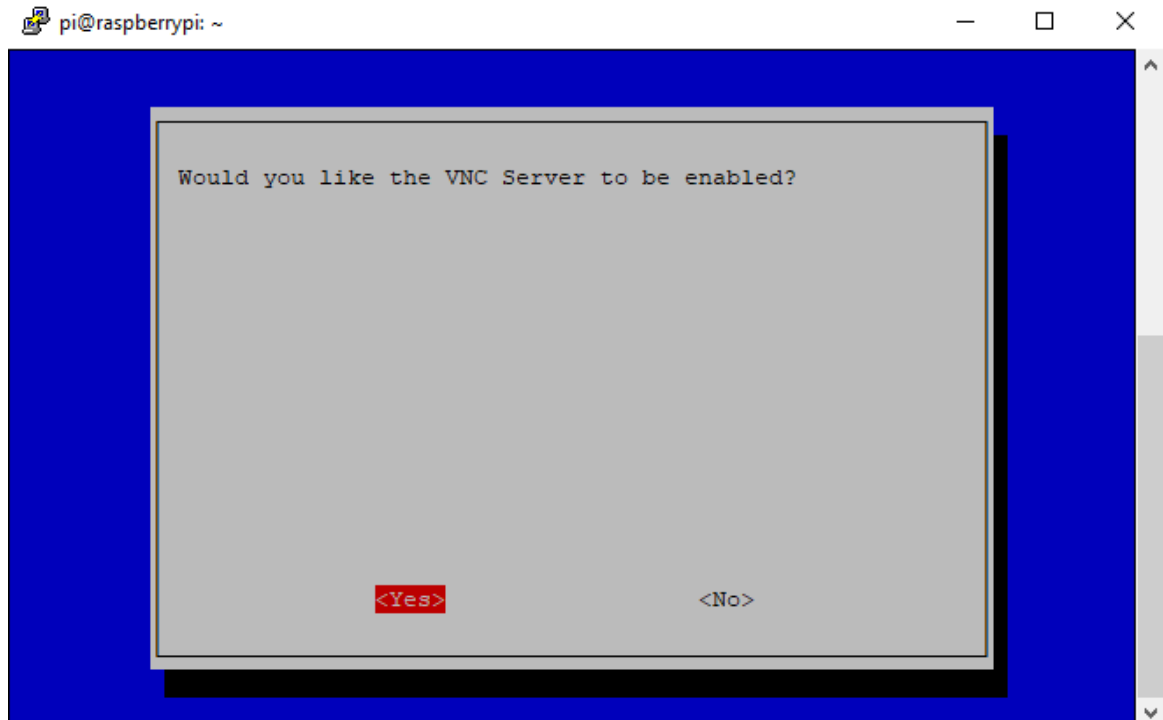


Figure 2.1.3.3-3 Enabling VNC Connection

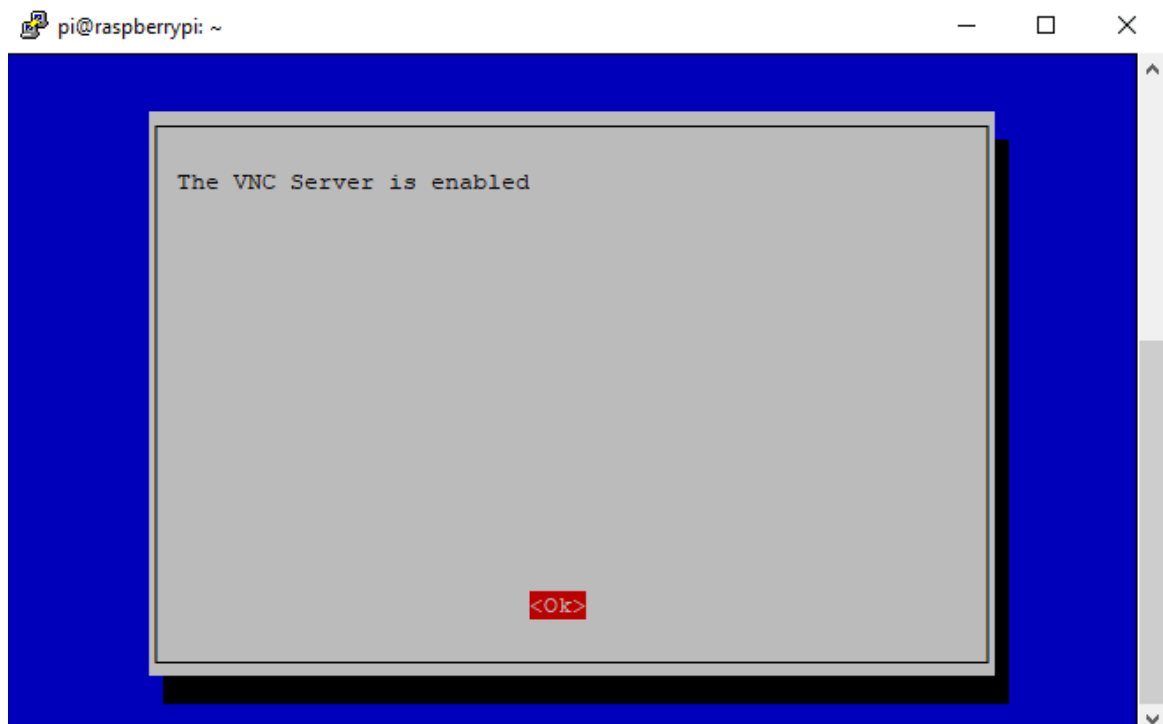


Figure 2.1.3.3-4 VNC Connection Enabled

2.1.4. VNC Configuration

After the VCN connection has been enabled, the next step is to download the VNC Viewer



Figure 2.1.4-1 VNC Logo

Step two is to run the VCN Viewer application(Fig 2.1.4-2) and use the IP address that the IP Scanner has previously given.

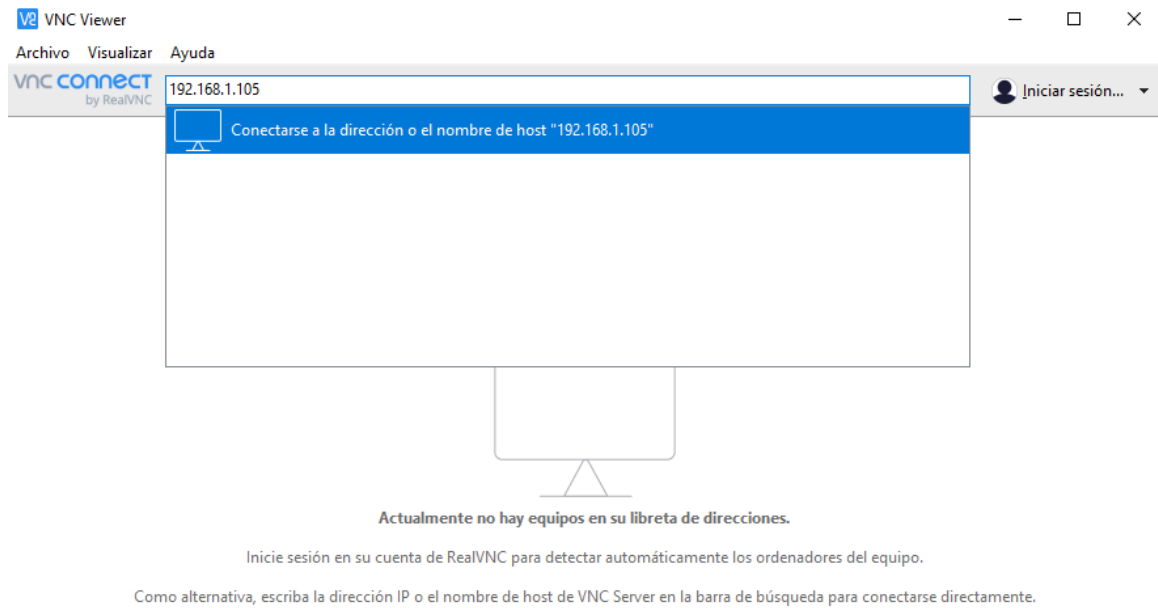


Figure 2.1.4-2 VNC interface

The last step is to introduce the user (pi) and the password (raspberry) and it will connect to the Raspberry's interface, as can be seen in (Fig 2.1.4-4)

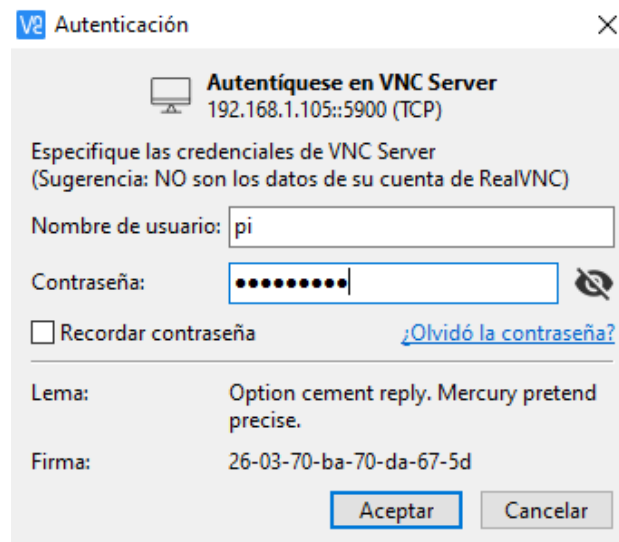


Figure 2.1.4-3 Starting VNC connection

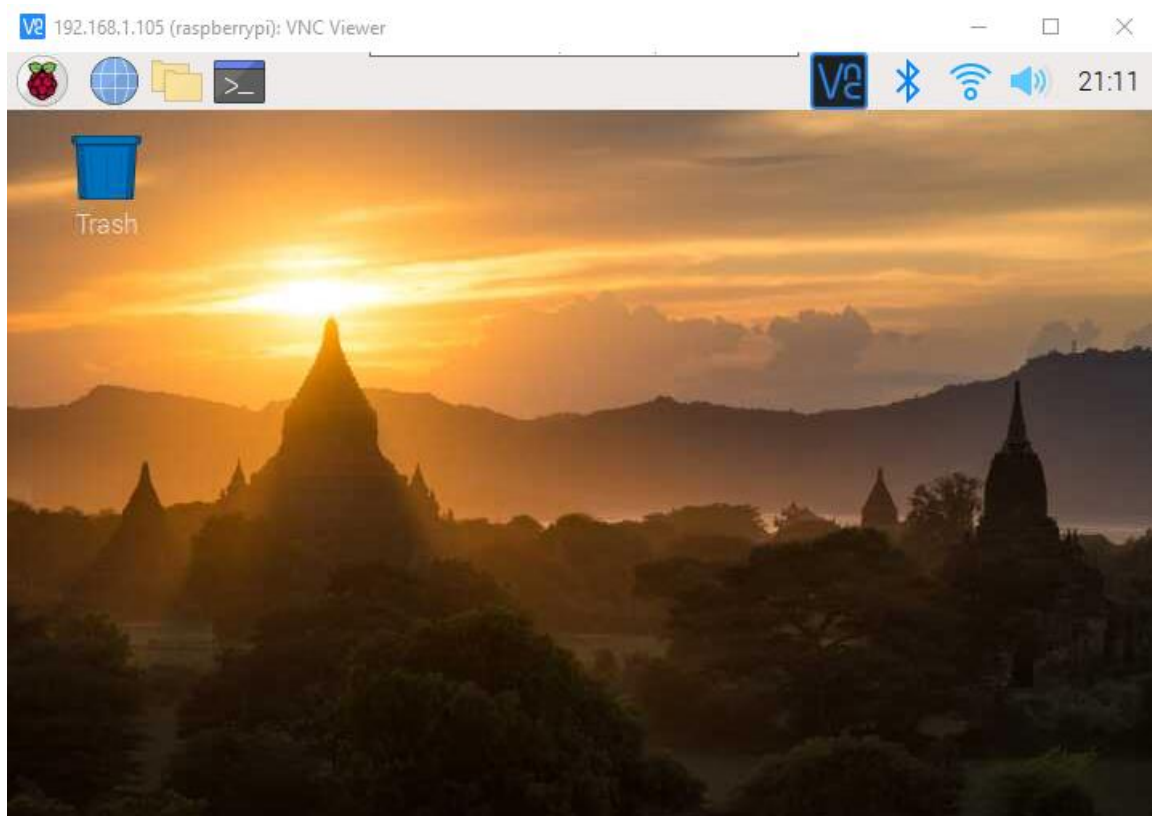


Figure 2.1.4-4 Successful VNC Connection

2.1.5. Fixed IP

Once access has been granted to the Raspberry, it can connect to the Wi-Fi in the area and the static IP address can be established. This can be done in two ways, graphically or in the command prompt. In this case, the graphical one was used. To do so, right-click on the Wi-Fi icon, various options will appear, and then click on the “Wireless & Wired Network Settings” (Fig 2.1.5-1)

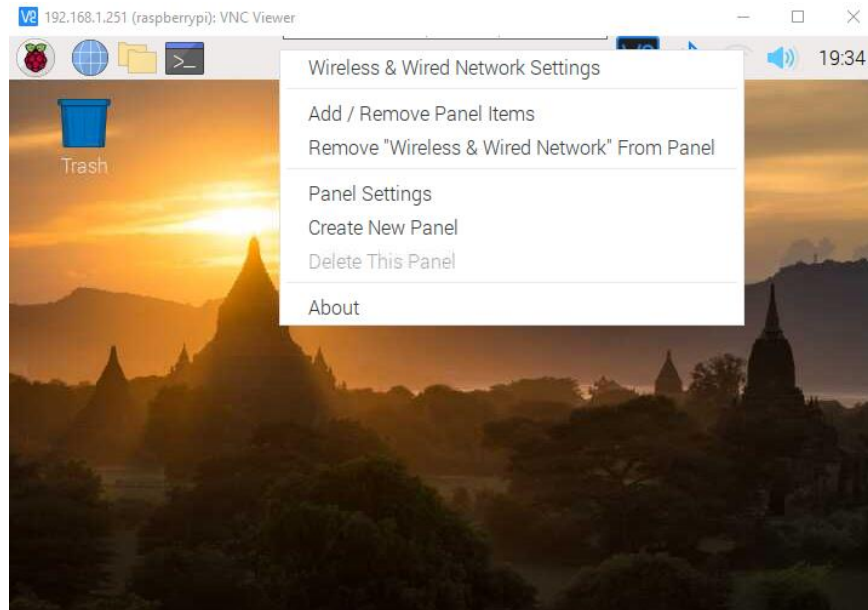


Figure 2.1.5-1 Right Click Wi-Fi Icon

Since the connection is Wi-Fi, the Wlan0 needs to be configured. And all that needs to be done is introduce the desired IP (Fig 2.1.5-2) that is inside the range of the network, apply the changes and reboot the Raspberry.

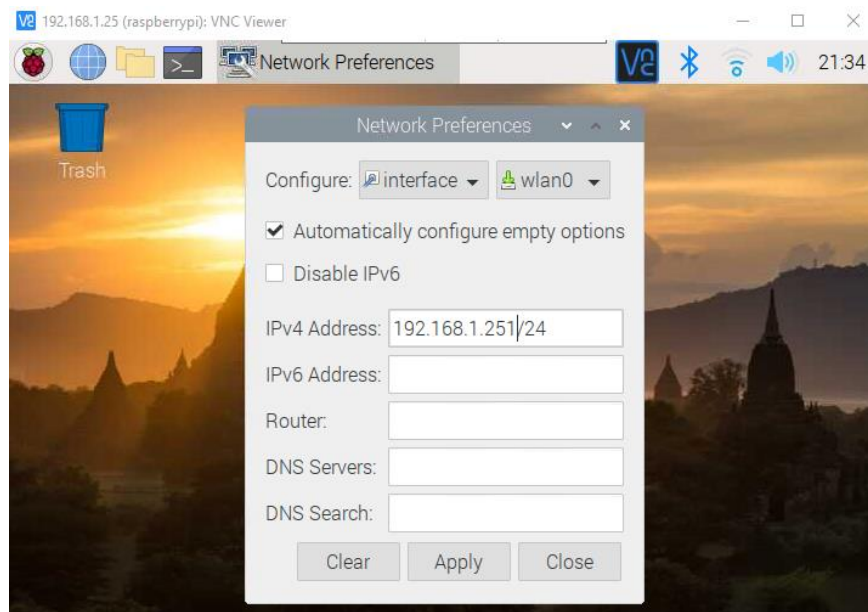


Figure 2.1.5-2 New IP Address Introduced

Reboot the Raspberry as sudo; `sudo reboot`. (Fig 2.1.5-3) It will disconnect, but just by opening the VNC Viewer again and introducing the new IP, the connection will be established once more.

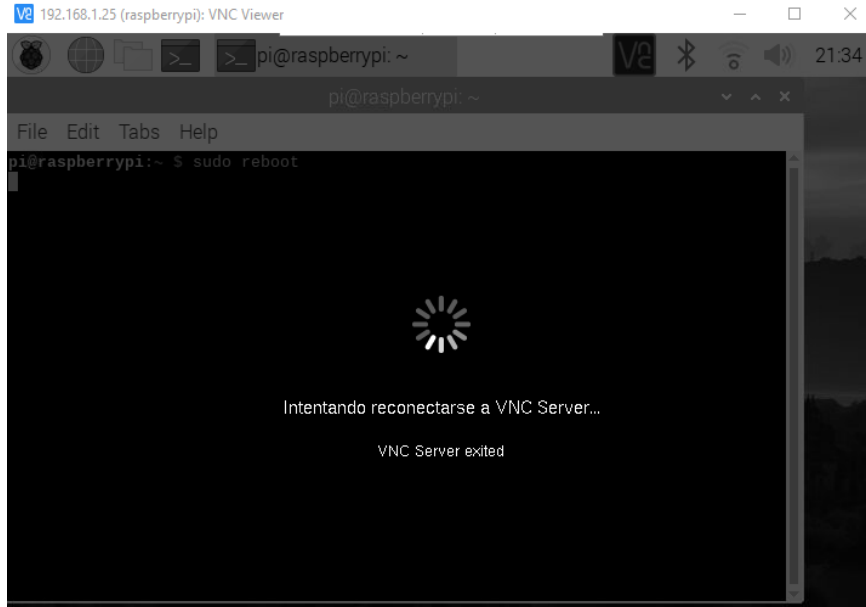


Figure 2.1.5-3 Reboot Raspberry

The VCN connection is started all over again with the new Ip address (Fig 2.1.5-4). The user and the password are the same.

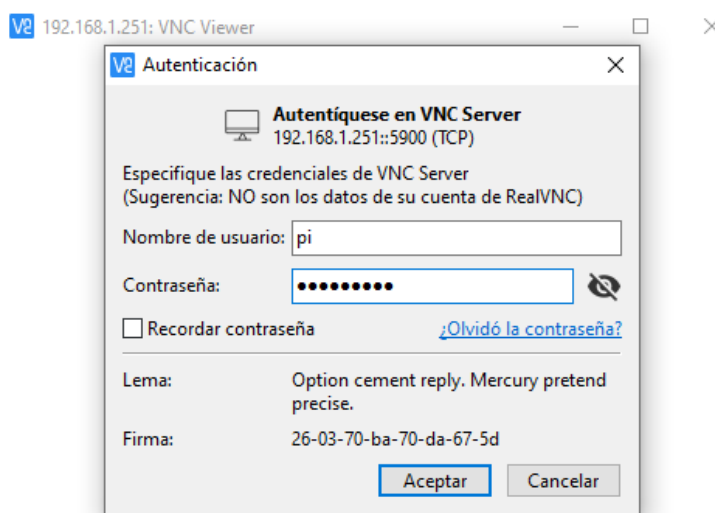
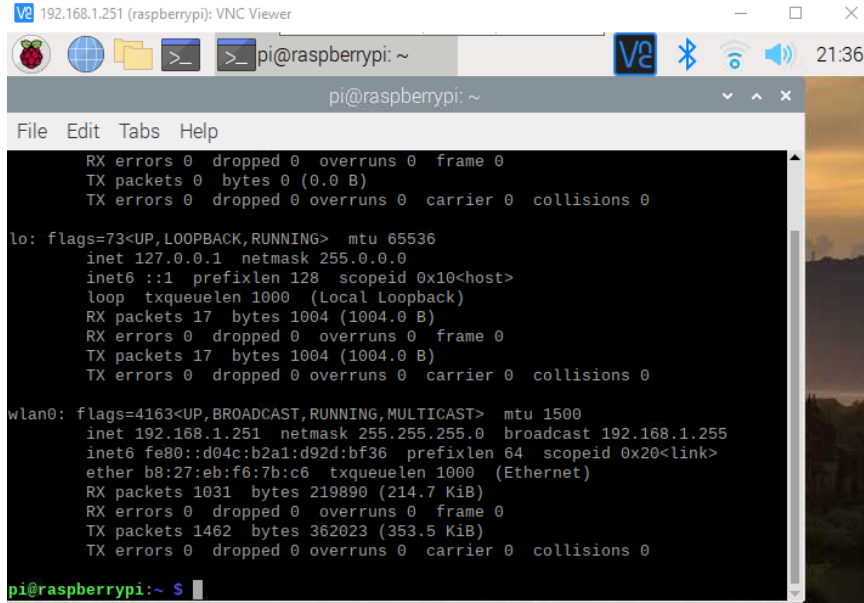


Figure 2.1.5-4 VNC connection with new IP

After the connection is established, the wlan0 direction is checked. (Fig 2.1.5-5)



```
192.168.1.251 (raspberrypi): VNC Viewer
pi@raspberrypi: ~
File Edit Tabs Help
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 17 bytes 1004 (1004.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 17 bytes 1004 (1004.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.1.251 netmask 255.255.255.0 broadcast 192.168.1.255
inet6 fe80::d04c:b2a1:d92d:bf36 prefixlen 64 scopeid 0x20<link>
ether b8:27:eb:f6:7b:c6 txqueuelen 1000 (Ethernet)
RX packets 1031 bytes 219890 (214.7 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 1462 bytes 362023 (353.5 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

pi@raspberrypi:~$
```

Figure 2.1.5-5 New IP address Checked



3. SOFTWARE

The software used to program the app was “Visual Studio Code”. As mentioned before, the language used was Python and its de-facto GUI (Graphic User Interface) library, Tkinter.⁸



Figure 3-1. Tkinter

3.1. Tkinter

The Tkinter package is the standard Python interface to the Tk GUI. Tkinter is implemented as a Python interpreter⁹, wrapped around a complete Tcl language¹⁰.

3.1.1. OOP and EDP

As is well known, Object-Oriented Programming¹¹ is based on the concept of objects that contain data and code; data in attributes and properties, and code in methods, which are equal as functions. Event-Driven Programming¹² is determined by user actions, that are considered the events. These actions can be a mouse click, keypresses, sensor outputs, or message passing. In this project, the only event used was the mouse click for the buttons, this was used so the interactions were quick and efficient.

3.1.2. Classes and Objects with Tkinter

Working with objects means working with classes. In this case, the principal Class is the TkinterApp, this is where the dictionary of the different pages is declared. That class is the one that allows the application to have “multiple windows” and depending on the buttons clicked by the user, the page that is raised varies. Also, classes are a more organized way to program and easier to understand when reading the code.

The widgets¹³ that the Tkinter library gives access to are various, the ones used in this project were; Label, Button, and Frame. The implementation of each one is intuitive. The frame contains a desired group of widgets, the label displays text and the button performs an action, which is the one that makes the app EDP.

3.1.2.1. Basics of Tkinter

To give some background and minimal knowledge in Tkinter a basic example from a web page¹⁴ (Fig 3.1.2.1-1) is going to be used as a reference. First of all, it is important to know which version of Python is being used, the difference between these two is the way the

module is written. In version 2.x is *from Tkinter import ** and in version 3.x is *from tkinter import **. If not written correctly it will generate an error.

```
from tkinter import *  
  
window = Tk()  
  
window.title("Welcome to LikeGeeks app")  
  
window.mainloop()
```

Figure 3.1.2.1-1 Basic Example Tkinter

The first thing to do is import the Tkinter package that will immediately create an empty window. This window will not be shown until the main loop is called, *window.mainloop()*. This will tell Python to run the Tkinter event loop and will be listening for any event, such as button clicks or keypresses.

This is the easiest way to create a window with Tkinter. And now is ready to be packed with widgets, labels, and buttons.

In the case of the coding for this project (Fig 3.1.2.1-2), since we are using classes and not making individual windows, the way the initial window is created is by first passing it as an argument in the main class, which means that the main class inherits the Tk() class, and is initialized in the second init of the class, by writing *tk.Tk.__init__(self)*. And how the *__init__* method defines a master, the self argument is indicating that the master is itself the main class.

```

class tkinterApp(tk.Tk): #Inheritance from another Tkinter class

    #Init method that initialise the tkinterApp class
    #Function that gets called everytime the class is called.
    def __init__(self):

        #This will initialize the tkinter, the inherited class
        tk.Tk.__init__(self)
  
```

Figure 3.1.2.1-2 Fragment of Code

3.1.2.2. Classes and Objects in Tkinter

A better way to explain the class's definition and the use it is given in this project is by presenting a simpler example, this is another basic example from a web page¹⁵ (Fig 3.1.2.2-2).

```

from tkinter import Tk, Label, Button

class MyFirstGUI:
    def __init__(self, master):
        self.master = master
        master.title("A simple GUI")

        self.label = Label(master, text="This is our first GUI!")
        self.label.pack()

        self.greet_button = Button(master, text="Greet", command=self.greet)
        self.greet_button.pack()

        self.close_button = Button(master, text="Close", command=master.quit)
        self.close_button.pack()

    def greet(self):
        print("Greetings!")

root = Tk()
my_gui = MyFirstGUI(root)
root.mainloop()
  
```

Figure 3.1.2.2-2 Basic Example Web Page

The first thing that is done, is writing the class. The name of the main class is MyFirstGUI. The second line is initializing the class, as it can be shown in the image (Fig 3.1.2.2-2) the method is receiving a parameter that is going to be the master. Then, there are some

widgets added; a label and two buttons. The label is simple, but it indicates that the master of that label is the same one that passed as a parameter before and it means that the label is going to be positioned in the master.

The buttons are the same, just different text. And in this case, the method used to add them to the master is `.pack()`. The result of that code is as follows:

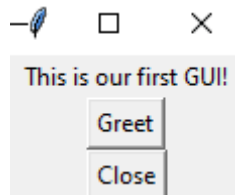


Figure 3.1.2.2-2 Web Example

3.1.2.3. Positioning of Widgets.

Regarding the layout of the app, there are three ways that Tkinter allows the positioning of widgets; `.place()`, `.grid()`, and `pack()`. The most common one is the grid, this is because normally the apps perform in different devices, so their sizes will vary and the best way to make sure the layout is the same proportion and position in every screen is by using the grid method. But since this app is going to be implemented on a tablet and the screen size is fixed, what worked best was the place method, just by playing with the numbers everything got perfectly positioned.

3.2. Code explanation

As mentioned before, this code is based on OOP, so classes are the foundation of it. There are two types of classes, the main one, and all the secondary ones. The main one can be considered the back-end of the app. This class is where the “root” is created, the geometry of the frame is fixed, and most importantly, where the dictionary of the multiple windows of the applications is implemented.

The code (Annex A), as shown in the concept map (Fig 3.2-1), begins by running the first class, which has all the initial information, this will create the window of 1024x600 size and

will load the StartPage, which is indicated in the *self.show_frames(StartPage)* function. On that page a button is going to be placed at the center, if pressed it will lead to the second Page, and there the recording and display of the sound will begin. On this page, the user just has to press the finish button once they want to end it and continue. If they want to start again all that needs to be done is press the “Back to Start Page” button. On the third page, is where the Equalization is done, once again all the user has to do is wait until it is finished. In case they want to record again the “Back to Start Page” button will be available. If they finish all they have to do is press the “Finish” button and it will close the App.

The code (Annex A) is commented on, and the methods and classes are better explained in it. The union that this project has with the other one it is not shown in this code, but the equalization filter would have to be added on the third page

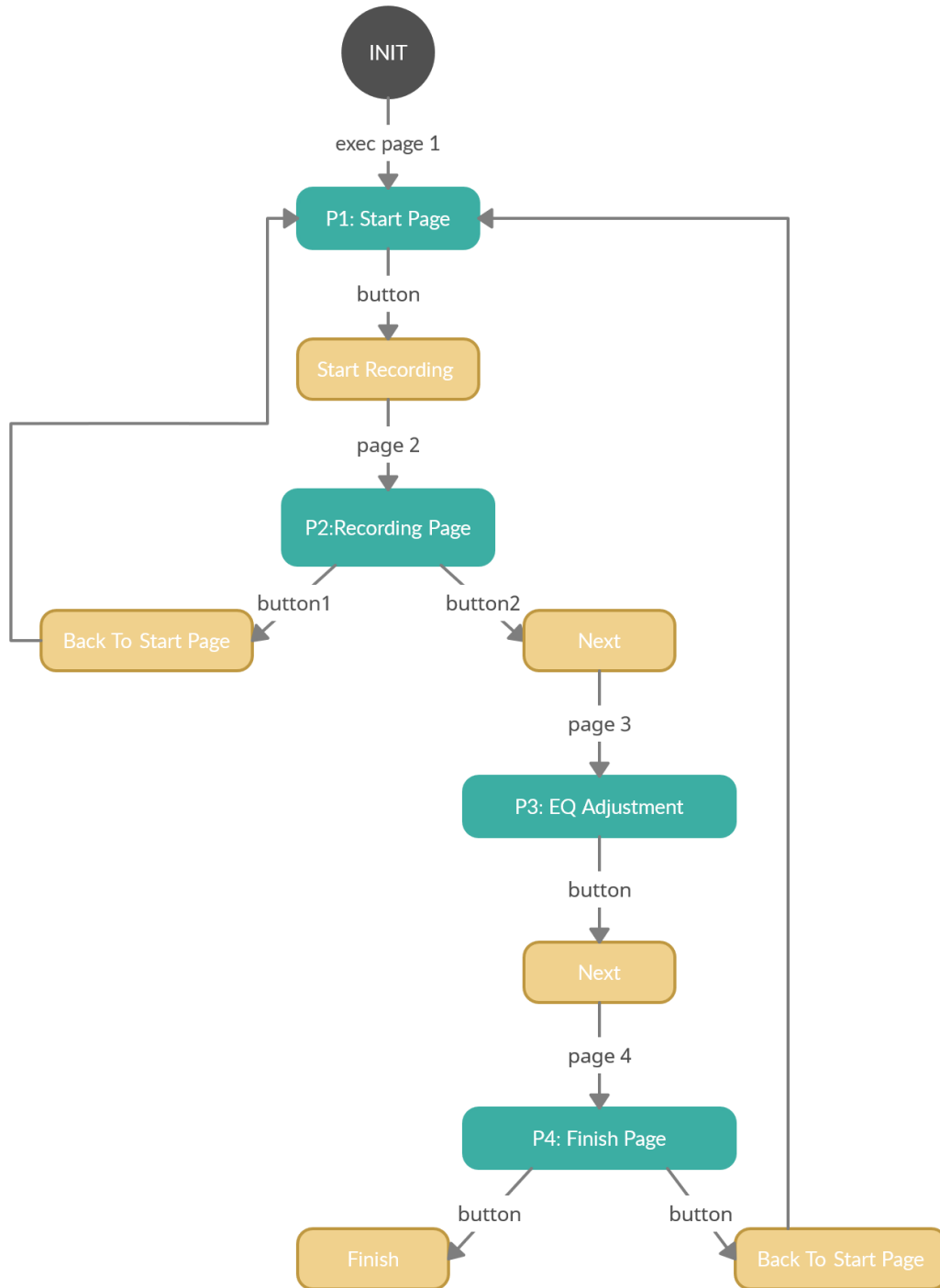


Figure 3.2-1 Conceptual Map



4. **EXPERIMENTAL RESULTS**

As planned, the App was successfully programmed, the result was as desired and allows an intuitive and simple interaction with the client. The layout of the pages is minimalist, there is not a necessity for more widgets, and the size of the labels is designed to make the interaction fast and clear.

4.1. **Screenshots of the Results**

Some screenshots of the different pages are going to be added.

The first one (Fig. 4.1-1) is the Start Page, this is the first thing the code opens. It has the button to start the recording.

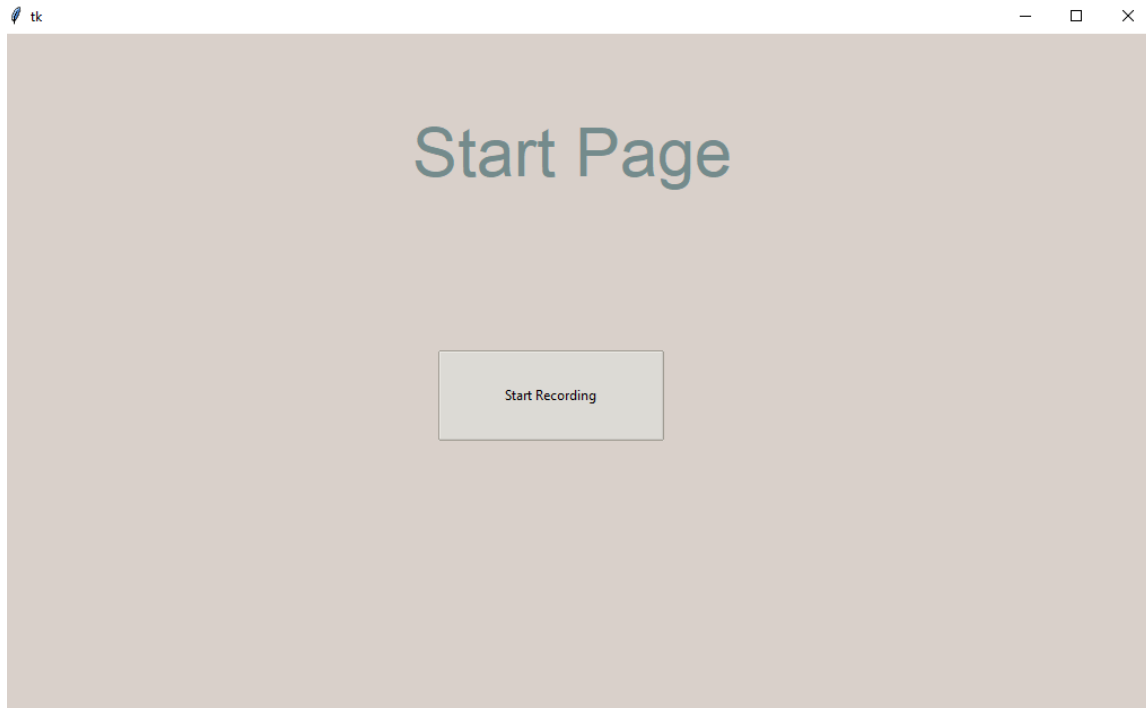


Figure 4.1-1. Page 1 – Init page

The following image (Fig. 4.1-2) is the window that is shown after pressing the “Start Recording” button. Here is where the recording and reproducing of the sound is done.

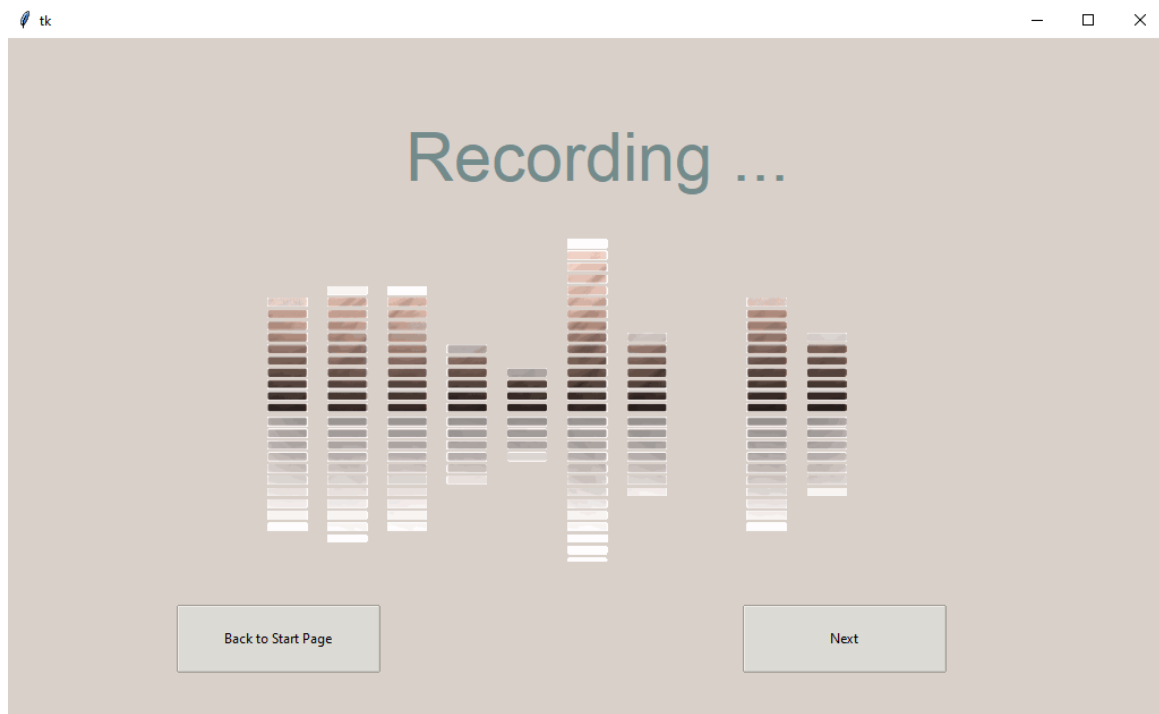


Figure 4.1-2. Page 2 - Recording Page

Once the user wants to finish the recording, the “Next” button on the “Recording Page” is going to be pressed, and the window that is going to be shown is the one that does the equalization.

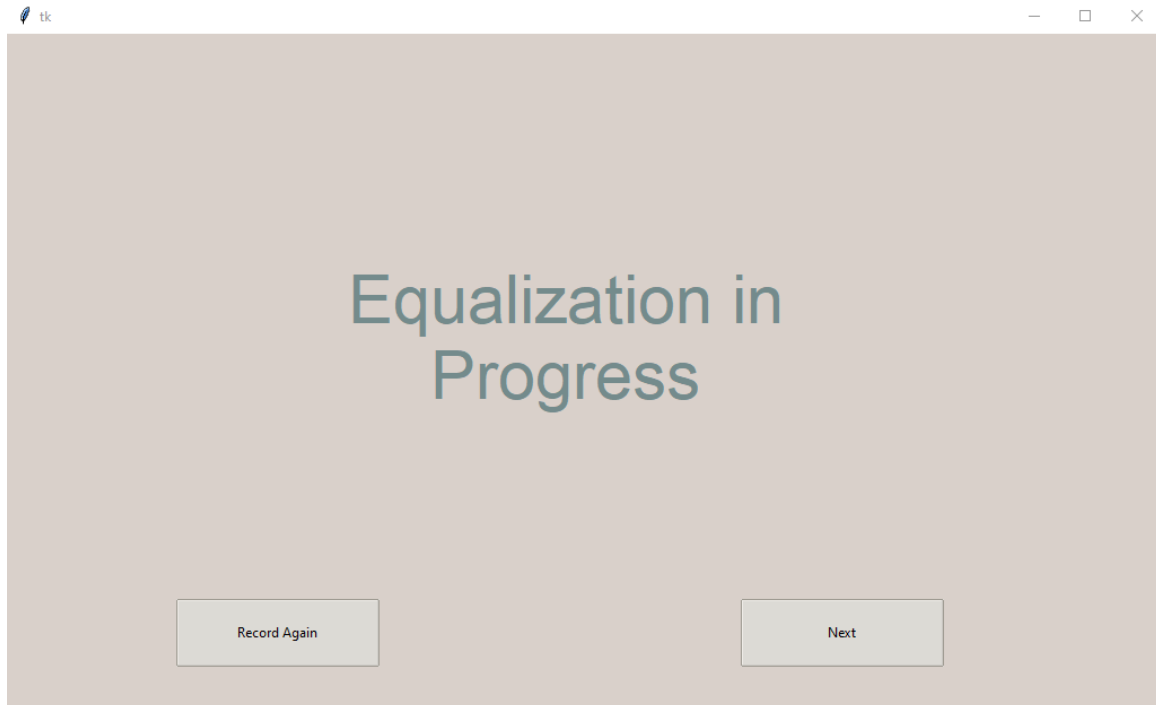


Figure 4.1-3. Page 3 - Equalization Page

And finally, after pressing “Finish” on the preview page, the last page that will be shown (Fig 4.1-4) is the one that allows the user to close the app.

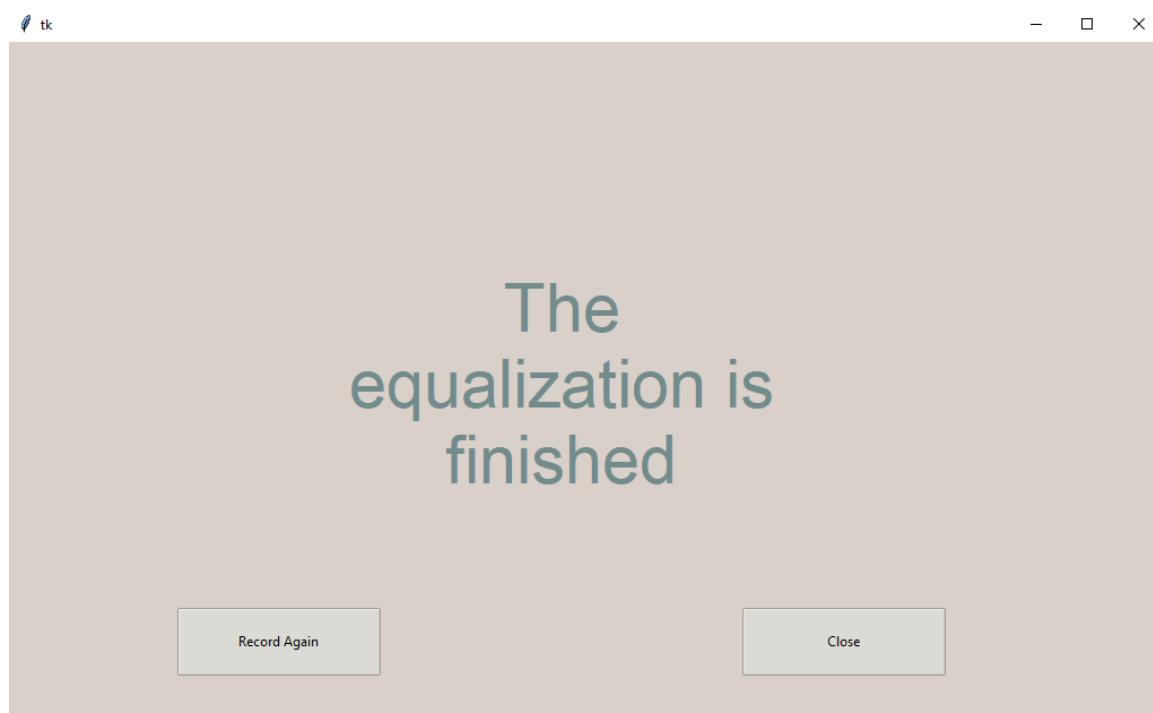


Figure 4.1-4 Page 4 - Closing page.



4. EXPERIMENTAL RESULTS

5. CONCLUSIONS

The first step to take towards the initialization of this project was the configuration of the Raspberry. Starting with the OP, then the SSH, the Wi-Fi, the IP, and finally the VNC connection. These were done successfully and will allow the download of the final app.

The code was the second thing to work on, the implementation of Python, its Tkinter library, and finding ways to implement different ideas so the code was simple and efficient. The main objective of this project was to make it as simple as possible and the final app is very straightforward and intuitive, so anyone can implement it when needed and hopefully, it will make the equalization process easier.

Regarding the code, working with classes instead of just simple windows was a bit complicated at the beginning. The implementation of methods changed since the master was itself. Some information that was key to comprehend, was the use of the parameters that passed as inherited in the classes, the initialization of them, and the interaction with the main class. In spite of the challenges, the outcome of these implementations has permitted an organized and simple code. Now, if the code wants to be expanded, it can be done with more efficiency.

Concerning the possible improvements, showing the user a graphic result of the capturing and equalization of the sound can be considered, this way they have a better idea of what is happening with the different frequencies.

The actual application doesn't allow the user to choose the sound card. It would be interesting to implement a dropdown menu to allow the user to choose the audio input and

the audio output. Also, an advanced configuration tab would be interesting to change the latency, sampling frequency, and buffer size of the sound card.

6. BIBLIOGRAPHY

1. Disseny d'un equalitzador de sistemes de PA (Public Adress) automàtic.
<https://upcommons.upc.edu/handle/2117/129886>.
2. Visual Studio Code - Code Editing. Redefined. <https://code.visualstudio.com/>.
3. 15 best programming languages for mobile app development 2021.
<https://www.spinxdigital.com/blog/mobile-app-development-languages/>.
4. Raspberry Pi OS – Raspberry Pi. <https://www.raspberrypi.org/software/>.
5. Raspberry Pi OS – Raspberry Pi. <https://www.raspberrypi.org/software/>.
6. Habilitar SSH en la Raspberry sin conectar el Monitor. uGeek Blog.
<https://ugeek.github.io/blog/post/2019-10-31-habilitar-ssh-en-la-raspberry-sin-conectar-el-monitor.html>.
7. Advanced IP Scanner – Explorador de redes de descarga gratuita.
<https://www.advanced-ip-scanner.com/es/>.
8. tkinter — Python interface to Tcl/Tk — Python 3.9.2 documentation.
<https://docs.python.org/3/library/tkinter.html>.
9. Introduction to Tkinter - first steps with Tkinter library.

- <https://zetcode.com/tkinter/introduction/>.
10. Languages using Tk - TkDocs . <https://tkdocs.com/resources/languages.html>.
 11. Object-oriented programming - Wikipedia. https://en.wikipedia.org/wiki/Object-oriented_programming.
 12. Event-driven programming - Wikipedia. https://en.wikipedia.org/wiki/Event-driven_programming.
 13. Python GUI Programming With Tkinter – Real Python. <https://realpython.com/python-gui-tkinter/>.
 14. Ejemplos de la GUI de Python (Tutorial de Tkinter) - Like Geeks. <https://likegeeks.com/es/ejemplos-de-la-gui-de-python/>.
 15. Introduction to GUI programming with tkinter — Object-Oriented Programming in Python 1 documentation. https://python-textbok.readthedocs.io/en/1.0/Introduction_to_GUI_Programming.html.