

**POSGRADO**

**MAESTRÍA EN CIENCIAS**

**INTRUMENTACIÓN Y CONTROL AUTOMÁTICO**

**MATERIA: Control Inteligente**

**PROFESOR:**

Rivas

**Project:**

**AUTOR:**

Sergio Aldo Lechuga Ensastiga

**FECHA: 05/07/2020**

[Abstract 3](#_Toc45280125)

[Motivation 3](#_Toc45280126)

[Introduction 3](#_Toc45280127)

[Flask 3](#_Toc45280128)

[GitHub 3](#_Toc45280129)

[What is a filter? 4](#_Toc45280130)

[FIR 4](#_Toc45280131)

[Spectral window 5](#_Toc45280132)

[Frequency response. 6](#_Toc45280133)

[Development 8](#_Toc45280134)

[Filter Coefficients 8](#_Toc45280135)

[Sample signals 10](#_Toc45280136)

[Web server implementation 11](#_Toc45280137)

[Tests and results 12](#_Toc45280138)

[Conclusions 15](#_Toc45280139)

Abstract

The aim of this project is to apply knowledge of git\_hub, linux, python and signal processing. That's why the project consists of a python web service of the appi Flask for low-pass filtering of a signal with 4000 Hz cutoff frequency. The web service simulates the data collection of an AD by reading a txt file, and by reading a text file containing the pass filter coefficients it will perform the signal filtering. The result for the user is a data file of the filtered signal and a comparison chart of the before and after filtering.

Motivation

A large part of my thesis project consists of the acquisition of biosignals which are at low frequencies, so they need to be filtered and processed for better handling. Likewise, the acquired data must be stored and displayed in a graph, so it is convenient not to depend on the memory of a micro-controller for its storage and instead to be stored in a local server.

Introduction

Flask

It is a framework written in Python that allows to create web applications quickly and with a minimum number of lines of code. It has the facility to debug, if you have any error you can debug that error and you can see the values of the variables.

You don't need an infrastructure with a web server to test the applications but in a simple way you can run a web server to see the results you get.

GitHub

It's a collaborative software development platform for hosting projects using the Git version control system, software applied to files with a large number of source code, for the purpose of keeping track of changes made to computer files and coordinating the work that various people do on shared documents, thus having copies of each version without losing previous statuses when the work is updated. Git lets you compare the code in a file to see the differences between versions, restore older versions if something goes wrong, and merge changes from different versions. It also lets you work with different branches of a project, such as development to bring new features into the program or production to debug bugs.

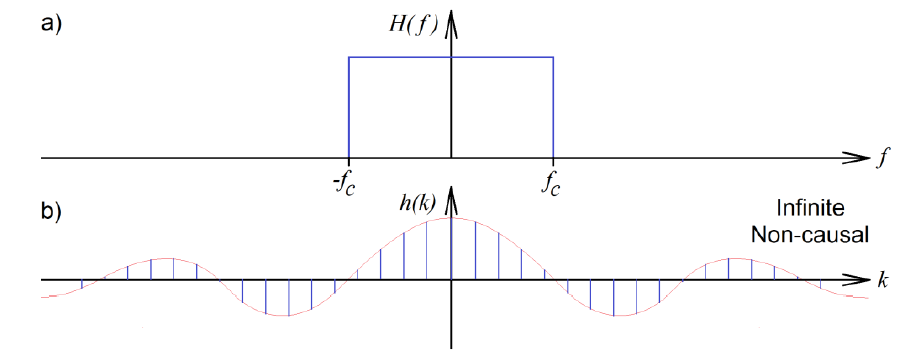
What is a filter?

We can define a filter, either analog or digital, as an electronic device that, given one or several signals at its input, is capable of discriminating at the output the type of signal required according to its frequency. In other words, a filter is capable of selecting which frequencies are required at the output.

FIR

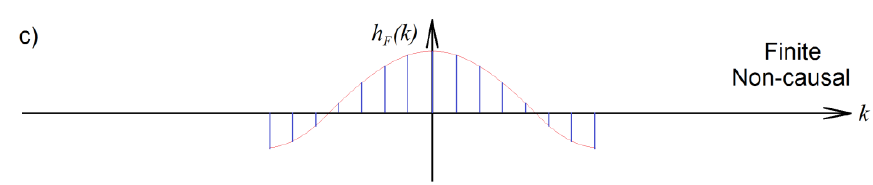
Finite Impulse Response (FIR); its acronym is due to the fact that this type of filter gives a FINITE response in time, meaning that, when faced with an input, the output will respond, but only in a finite way.

The ideal filter does not exist, it can be perfectly stated in a mathematical way in the frequency domain, however, when wanting to implement it in reality it is not possible. The latter is due to the fact that the filter in the frequency domain is infinite and not causal; then we have to adjust certain parameters of the filter window to adapt it to reality.



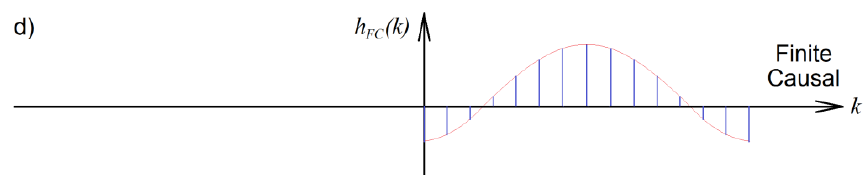
*Figura 1. Ideal filter infinite no causal.*

As can be seen in the previous image, b) is infinite, however, as we move forward and backward in time the energy of the signal h(k) decreases, having values close to 0, therefore, we can only take a part of the signal that contributes significantly to energy.



*Figura 2. No ideal filter, finite no causal.*

The non-causality problem is solved by shifting the signal to start at 0.



*Figura 3. Real filter, finite causal.*

It is clear that doing these movements makes our filter not perfect, however, for practical applications it is enough. The number of coefficients that the filter has is called the order of the filter. It should be noted that the higher the order, the better the performance of the filter, but the longer the execution time, and the great disadvantage of FIR filters is that a considerable number of coefficients are required.

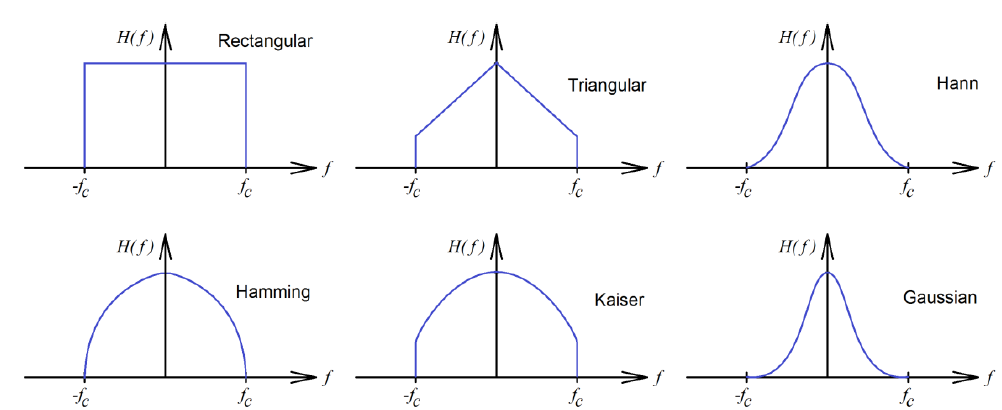
Spectral window

We can consider the first filter sample as the second as and the n sample as and if we take into account that a filtered signal can be obtained with the convolution operation, we have the following equation.

If we apply the Z transform to the above equation, we get the transfer function of the filter.

The a\_n coefficients determine the shape of the spectral window of the, and the cutoff frequencies. The more a\_n coefficients are obtained, the higher will be the filter order and at the same time it will approach the behaviour of an ideal filter, however increasing the order will increase the number of filter delays.

In the following image we can see common spectral windows.

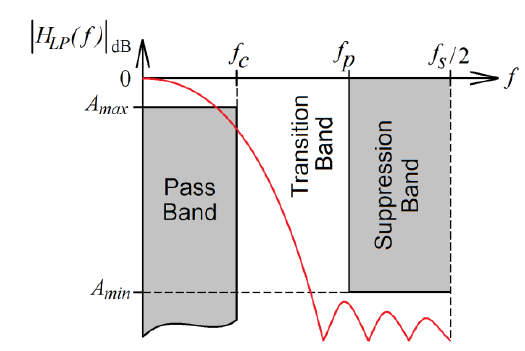


*Figure 4. Spectral windows.*

Frequency response.

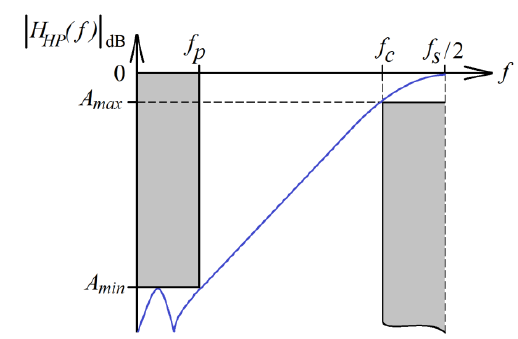
Depending on the cut-off frequency and pass band of a filter, it can be classified into 4 types:

1. Low Pass: Frequencies above the cutoff frequency do not pass.



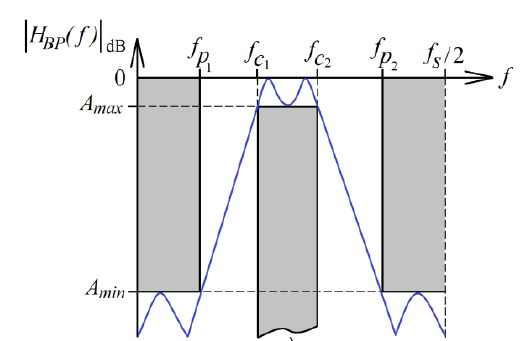
*Figure 5 Low pass filter*

1. High Pass: Frequencies below the cutoff frequency do not pass.



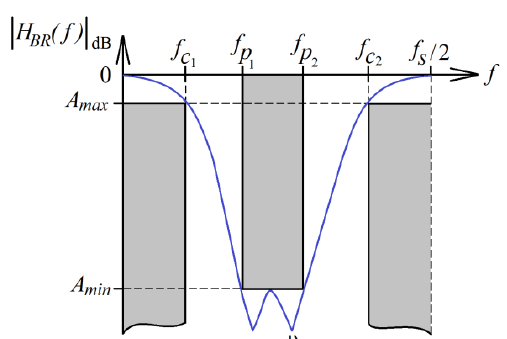
*Figure 6 High pass filter*

1. Bandpass: Is determined by two cutoff frequencies; any signal outside those frequencies does not pass.



*Figure 7 Bandpass filter*

1. Stop Band: Is determined by two cutoff frequencies; any signal within those frequencies does not pass.



*Figure 8 Stop Band filter*

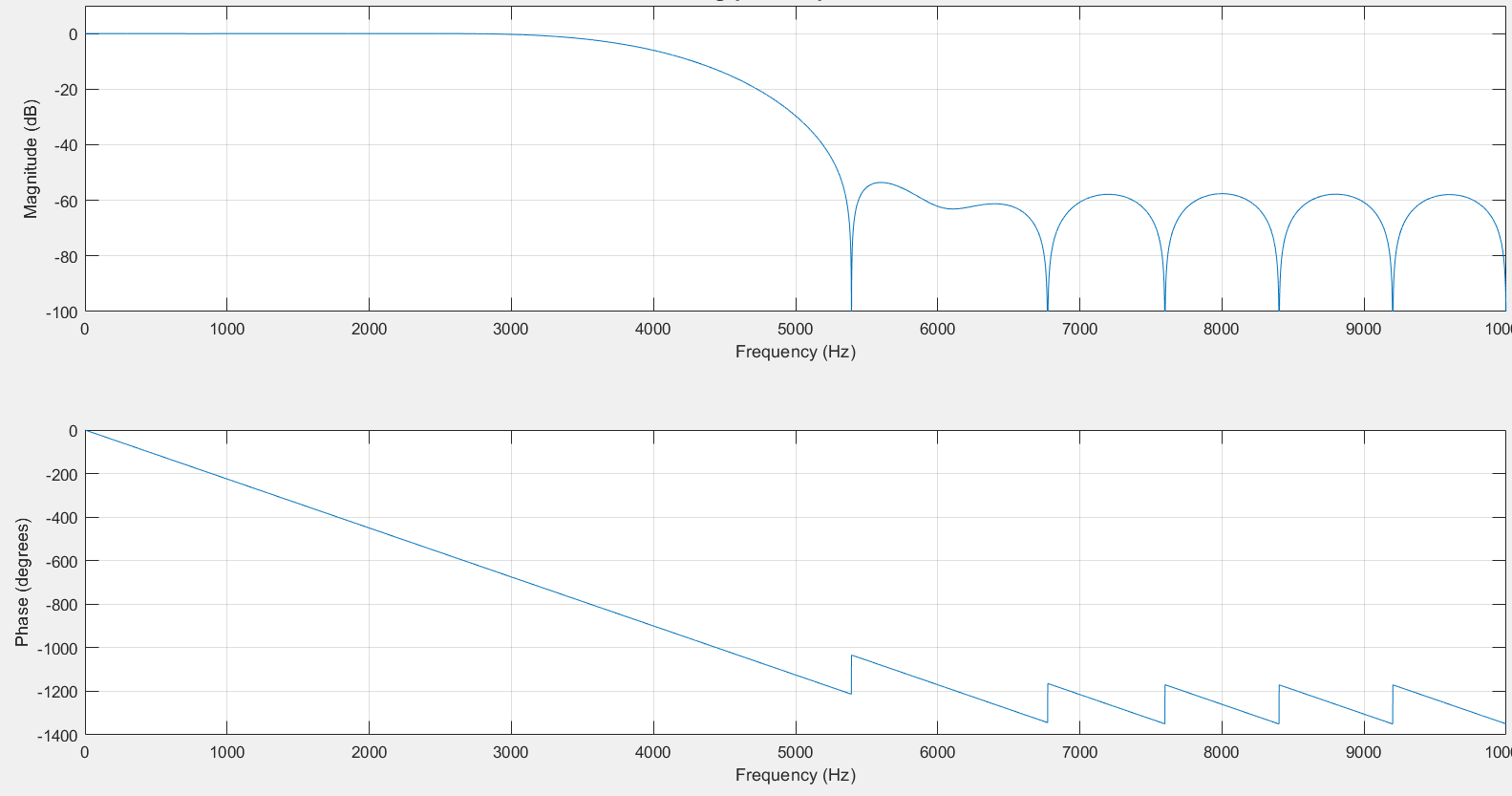
Development

Filter Coefficients

The first step in filtering a signal is to obtain the filter coefficients. It is not the objective of the project to obtain these coefficients, but rather to load them into a web server and filter a signal. However, a matlab program will be left in the github repository if the user wishes to modify it to obtain another type of filter or to perform various tests.

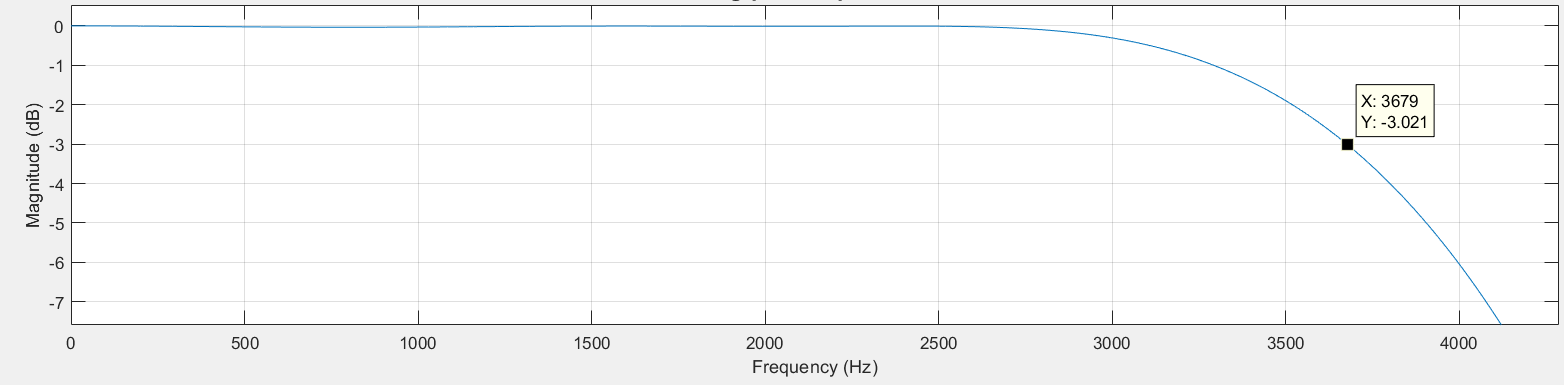
For this project we want to design a filter to filter human voice which has a frequency between 250 Hz and 4000Hz, so a low pass filter with a cut-off frequency of 4000Hz is sufficient for this purpose. A filter of order 25 was chosen for this case.

In the following image we can see the frequency response of the filter described



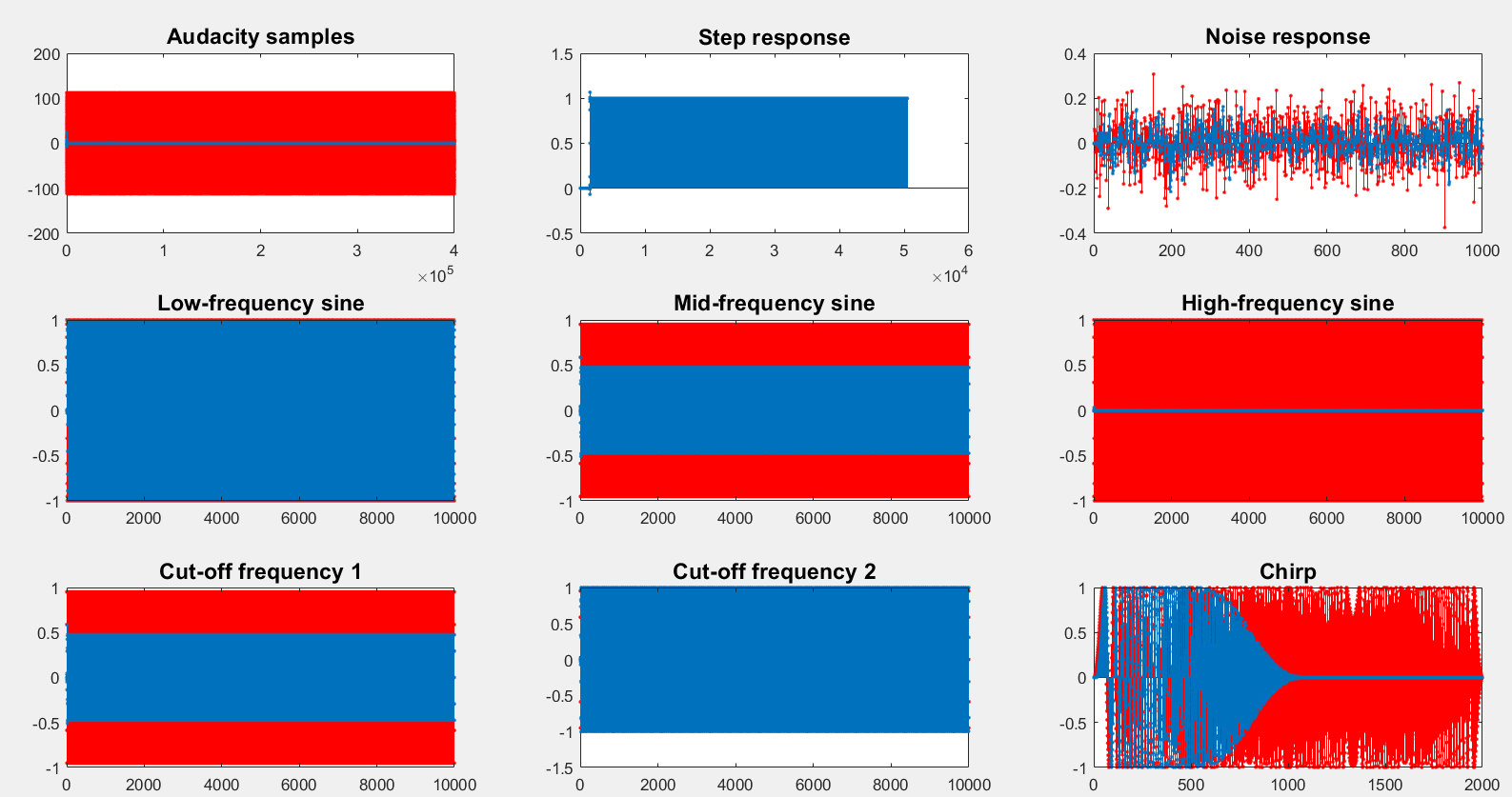
*Figure 9 Frequency response of bandpass filter.*

As we can see in figure 9, the filter does not cut exactly in 4000Hz but we can say that the filter has a good performance since as we can see in the zoom of the previous figure, from the frequency 3679 Hz the signal loses 3dB which means that signals that have that frequency will lose half of its power and from 4000Hz will have lost 6dB which translates into the loss of a quarter of the power of the signal and beyond that frequency are practically indictable for our interest.



*Figure 11 Zoom of figure 9.*

We can see in figure 12 that the filter response was tested with multiple signals at different frequencies.

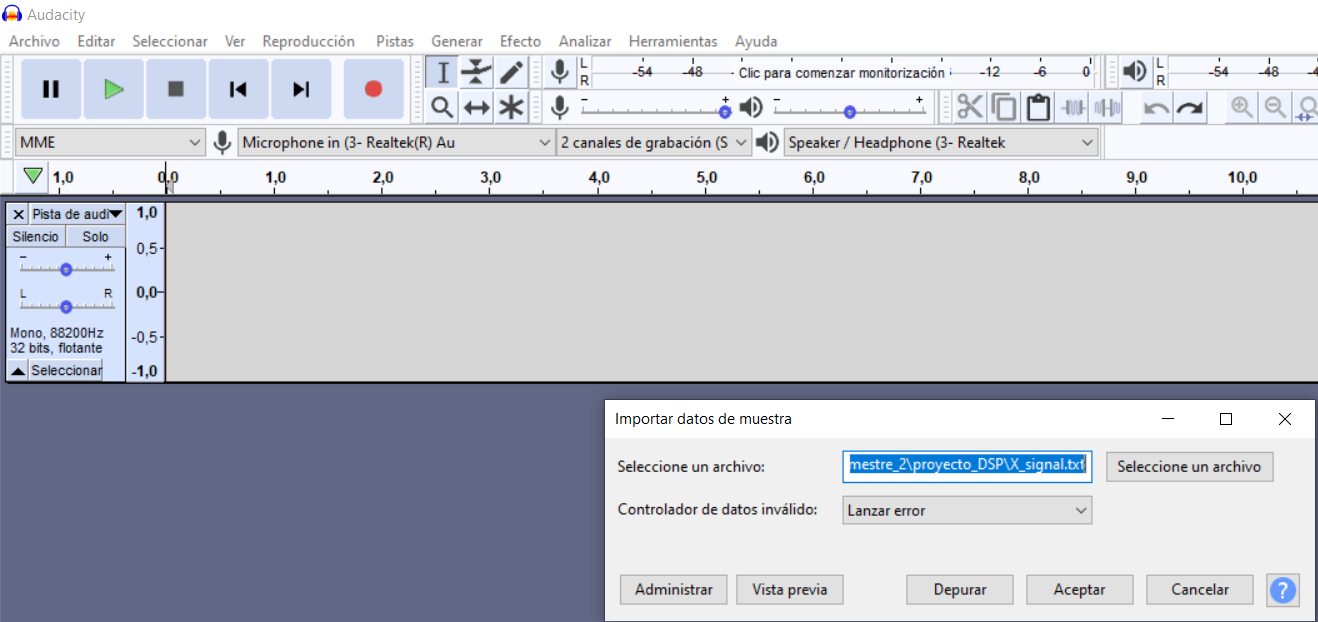


*Figure 12 Tests over different signals.*

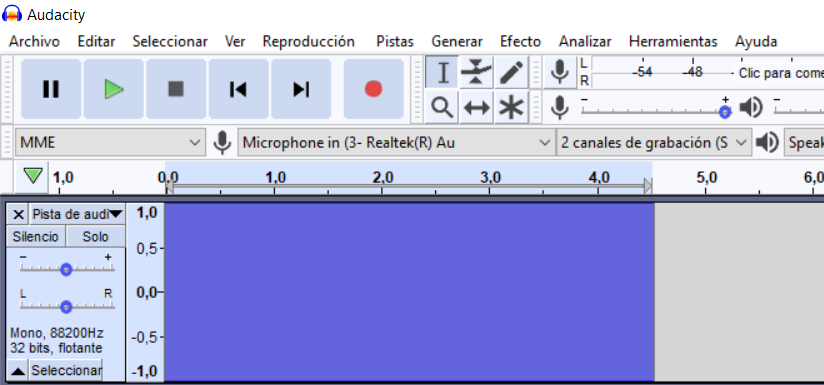
Since it has been verified that the filter works, we save the coefficients obtained in a text file which is available in the GitHub repository.

Sample signals

For the generation of a signal with noise, first we used the audacity program which is specialized in sound manipulation, what was done with that program was recorded a voice and saved in a text file the voice samples, then in the matlab program these samples were added several sinusoidal signals between 4,000Hz and 10,000 Hz of different amplitudes the obtained signal was saved in a text file. The same audacity program has the possibility of importing sample data for reprehension, which was done; if we reproduce this data, we do not hear anything of the voice, instead we hear a strong high tone.



*Figure 13. Data import.*



*Figure 14. Non filtered signal in audacity.*

Web server implementation

To run the main flask file the following commands are activated in the linux terminal:

* *source .venv/bin/activate*

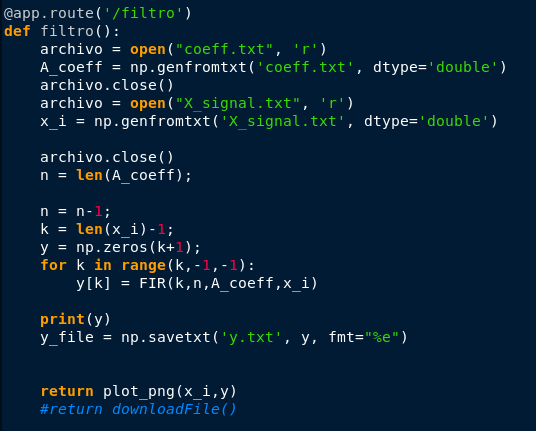
which runs the virtual python environment.

* *FLASK\_APP=service.py flask run*

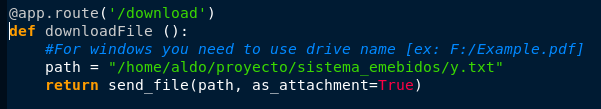
The service.py file is where all the functions of our web server such as redirecting requests or downloads are stored. This command compiles and executes this file

The service.py file has the following functions:

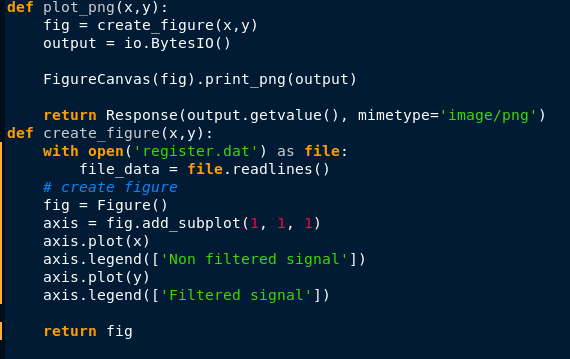
* Filter: first opens the coefficient file and saves the coefficient values in an array. It does the same with the sample file "X\_i.file". This step of the code simulates the sampling of an embedded system. Both arrays are sent to the FIR function. The filtered signal is saved (in the local server) in a text file and finally a graph is displayed where the filtered signal (orange) is compared with the original signal (blue).
* FIR: it runs the discrete time equation of a FIR-type filter.
* downloadFile: if the user enters with the url: *http://127.0.0.0:5000/download*, this function will be executed which downloads from the local server to the user the text file of the filtered signal.
* plot\_png and create\_figure: They help us create a graph of the original signal (X\_i) and the sampled signal (y)



*Figure 15. function “Filtro()”.*



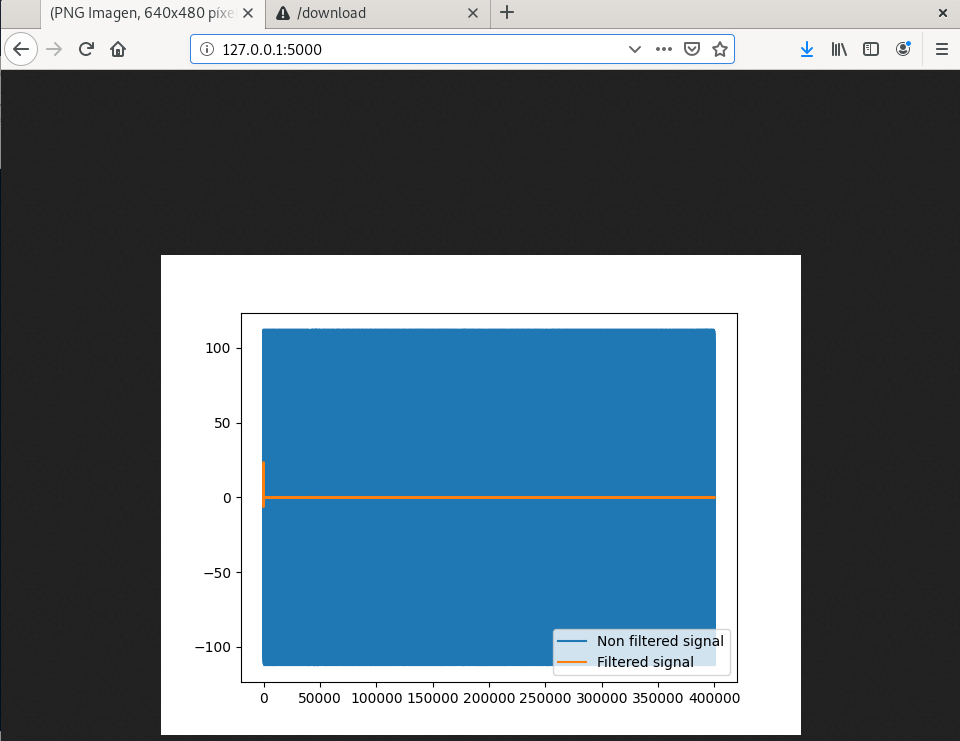
*Figure 16. function “downloadFile()”.*



*Figure 17. functions “plot\_png ()” and “create\_figure()”.*

Tests and results

In the following figure we can see a small graph in orange the filtered signal while the blue signal is the original signal. It should be noted that it looks like this because the original signal had large amplitudes compared to the filtered speech signal.



*Figure 18. Compare signals.*

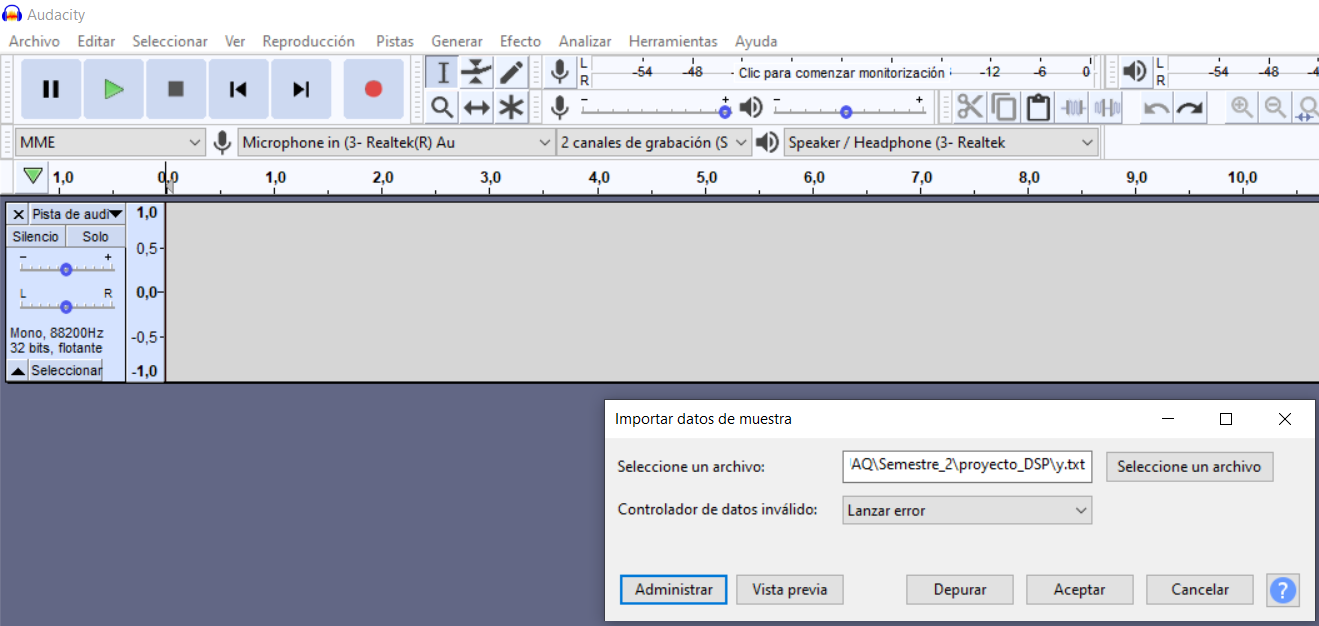
However, a video has been included in the following link: [*https://vimeo.com/user119281215/review/437217015/e89110ca27*](https://vimeo.com/user119281215/review/437217015/e89110ca27)

We can ask the server to download the filtered signal into a text file.

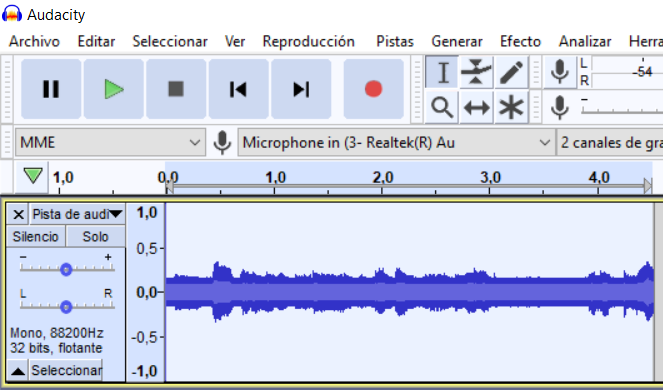


*Figure 19. Download txt file.*

Finally, the filtered signal was imported to audacity to verify that the test voice is heard, in the image 21 you can see a voice signal, if you want to hear the result you can see the video in the provided link.

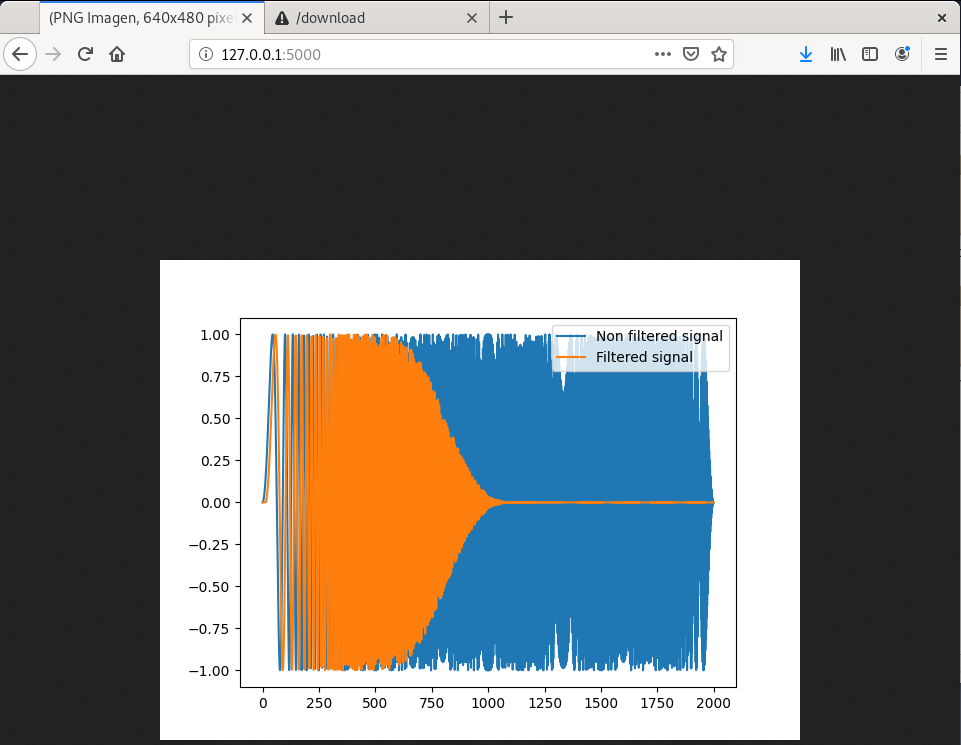


*Figure 20. Import Filtered signal.*



*Figure 21. Signal voice filtered.*

For a more visual test, it was tested with a chirp signal which is a sine wave that changes its frequency as time increases.



*Figure 22. Signal voice filtered.*

As we can see in figure 22, the higher frequencies are suppressed while lower frequencies pass without any problems.

Conclusions

The big disadvantage that some microcontrollers can have is that they depend on a limited size of bits so the result of a floating number can be affected and consequently decrease the quality of the filter, with the implementation of a web server this disadvantage goes away so this can help to choose a cheaper microcontroller or can save resources of it.

The great advantage that was observed in github during the development of the project is that you can have several versions of the same project which translates into multiple tests to choose the best one or converge into one.