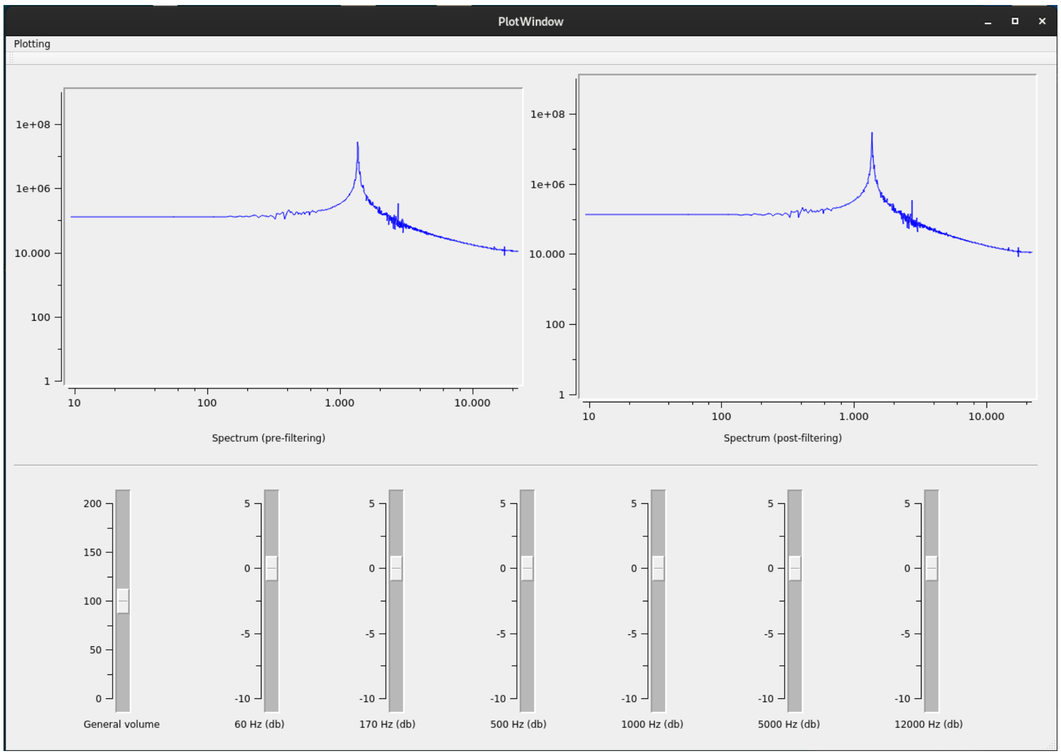
Embedded Computing Systems

Component based software design

CbsdMixer



Name: Matteo Zini

Student ID: 533197

E-mail: matteozini96@gmail.com

1. General description

The project consists in implementing an audio mixer that, taken as input the sound from the microphone, applies a filter to its spectrum and plays it back with the speakers.

The software is developed with the Qt framework for the graphical user interface and the timed events. The Qwt extension has been added to the Qt environment to allow the creation of plots and more customizable sliders. The system also utilizes the ALSA library for Linux to implement the audio input and output part and the fftw3 library to compute the Discrete Fourier Transform of the input signal.

The functionalities offered by the system are:

- Input to output audio mixer
- Plot of the spectra of the signals before and after the processing
- Slider to adjust the general volume
- 6 sliders to adjust the amplification of six different frequency intervals
- An additional window, accessible from the menu bar, to visualize the shape of the filter

Additional customizations that are possible with minor modifications of the source code are (**default options**):

- Modification of the frequencies around which the filtering is applied
- Modification of the shape of the filter (rectangular, triangular, **cosinusoidal**)
- **Enabling** or disabling the windows overlap feature (if the option is enabled, two consecutive audio windows contain a portion of data in common and a final fusion of the consecutive buffers is performed; this is a possible strategy to avoid annoying sounds due to the distortion of the wave induced by the spectrum modification)

In addition, the system utilizes a double buffering system to avoid interferences between the input and the output thread. The exchange is performed using an atomic variable.

IMPORTANT NOTE: the system should be used with earphones to avoid a strong echo due to the fact that the played sound is heard again by the microphone. If the earphones also have an embedded microphone, it is advisable to deactivate it and use another one in its place, since the fact that the two devices are connected can induce a lighter but audible echo too.

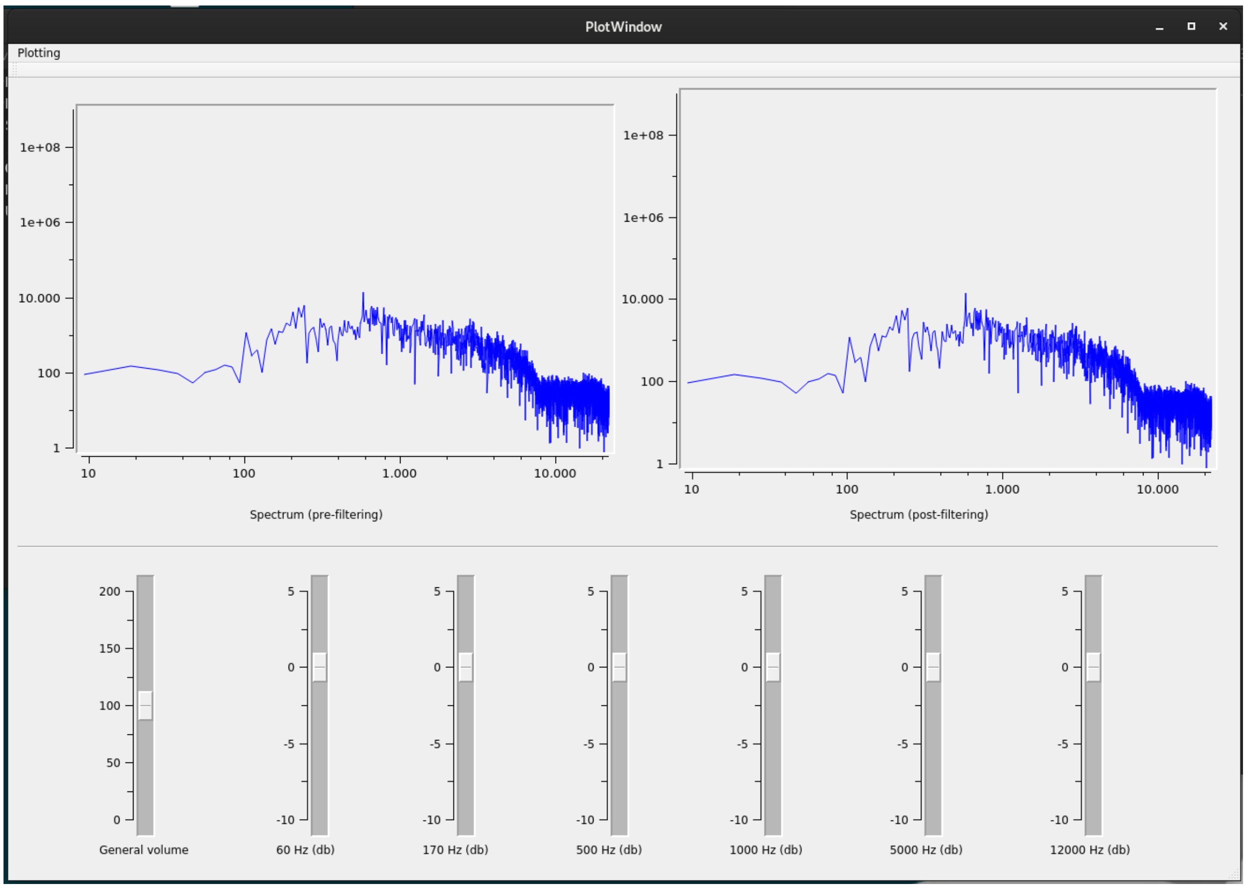
2. Source code structure

The source code is divided in the following files and folders:

- **audioio.cpp, audioio.h:** these files contain the implementation of the AudioIO class, which abstracts the ALSA library offering a simpler interface for the upper layers. It offers some methods to read and write audio samples and can be configured indicating the input and output devices to utilize, the sampling frequency and the number of samples to read/write at every function call
- **Filter folder:** this folder contains the implementation of the filtering functions, organized as subclasses of a common Filter class. The specific derived classes only implement a different function for the computation of the filter. The folder also contains an additional header file which includes all the other ones in the folder.
- **mixer.cpp, mixer.h:** these files contain the implementation of the Mixer class. The class instantiates a filter, depending on the constructor options, and has a method to start the acquisition and reproduction of the sound in separate threads. It handles the Fourier transforms, the double buffering, the window overlapping and offers some setters and getters for the caller
- **plotwindow.cpp, plotwindow.h:** these files contain the setup of the graphical interface through the usage of a ".ui" file and some manual commands. The class also instantiates and starts the mixer, performs the plot updates using a timer and handles the user inputs.
- **filterwindow.cpp, filterwindow.h:** these files contain the setup of the window that shows the filter shape. It uses a ".ui" file and some manual commands. The class contains a method through which the PlotWindow class can update the input data.
- **main.cpp:** this file contains the instantiation of the PlotWindow class and starts the application.

3. GUI description

The GUI consists in the following window:

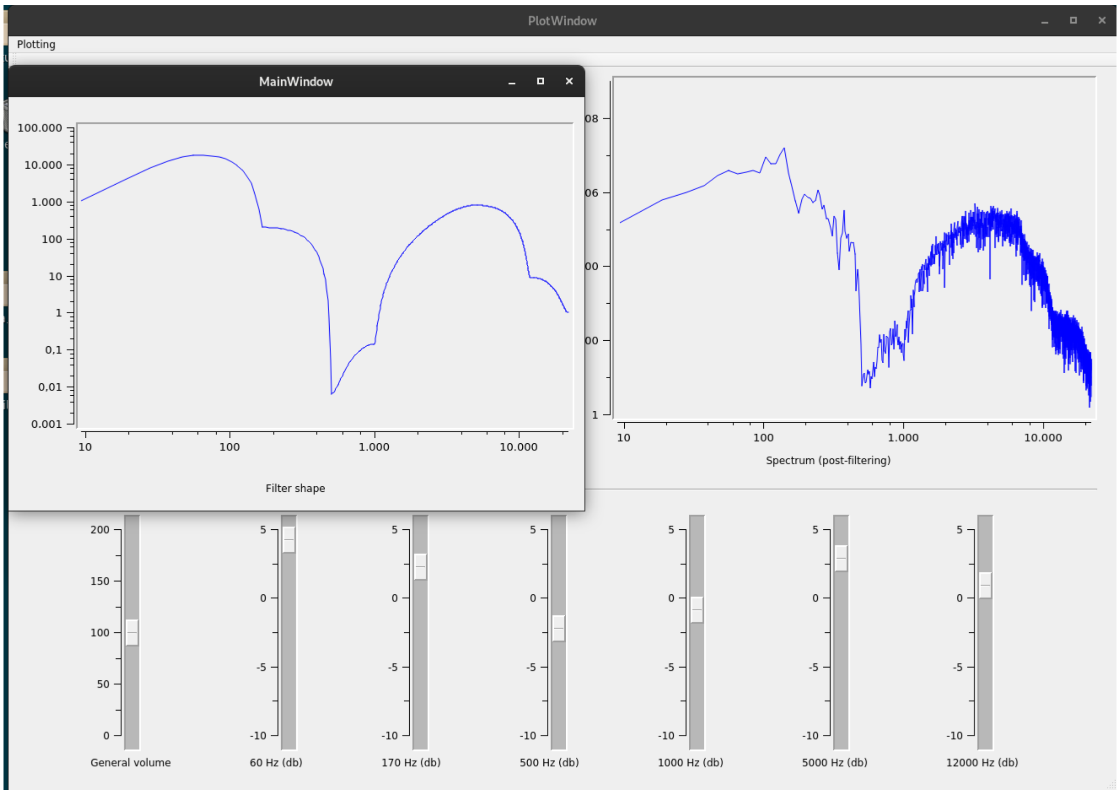


The two plots show the spectrum of the signal before and after the filtering operation.

The user can adjust the filter values utilising the 7 sliders present in the bottom part; the first one modifies the general volume, the others correspond to specific frequency bands.

All the plots and the sliders, except the general volume one, utilize a logarithmic scale.

Utilizing the “Plotting” button in the menu bar and selecting the “Show filter” option, the window showing the shape of the filter is displayed. The plot show in this graphic also uses a logarithmic scale.


PlotWindow
Plotting
MainWindow
100.000
10.000
1.000
100
10
1
0,1
0,01
0.001
10
100
1.000
10.000
Filter shape
08
06
00
00
1
10
100
1.000
10.000
Spectrum (post-filtering)
200
150
100
50
0
General volume
5
0
-5
-10
60 Hz (db)
170 Hz (db)
500 Hz (db)
1000 Hz (db)
5000 Hz (db)
12000 Hz (db)