Laboratory manual

Digital Signal Processing Laboratory

UbiLAB

Универзитет "Св. Кирил и Методиј" во Скопје
ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И ИНФОРМАЦИСКИ ТЕХНОЛОГИИ

Hochschule Anhalt
Anhalt University of Applied Sciences

University of Maribor

Erasmus+

NATIONAL AGENCY
for European Educational
Programmes and Mobility

4 Digital Signal Processing Laboratory

4.1 Introduction

The Digital Signal Processing (DSP) course provides students with interactive remote-controlled experiments with in the DSP laboratory. Three NI myRIO 1900 devices are programmed to host different experiment each. More about myRIO 1900 can be red in its official manual provided by National Instruments:
https://www.ni.com/docs/en-US/bundle/myrio-1900-getting-started/resource/376047d.pdf

The experiments are based on the exploitation of the Signal Processing Toolkit applied for:

- Real-Time Audio Filtering;
- Real-Time Moving Spectrogram;
- Real-Time PWM Generation.

They have a friendly user interface (UI), from which the remote user can change predefined parameters and observe the response in numerical, graphical and audio format. The LabVIEW software is used both for the experiment's development and for the user front-end and remote control. It should be noted that students do not need any LabVIEW programming knowledge, they just use the UI. However, for enthusiasts who have a basic knowledge of LabVIEW, there is an opportunity for accessing the LabVIEW projects for exploring additional signal processing tools.

The remote performance of the laboratory experiments offers the same possibilities as if the experiments are performed physically in the laboratory. The students need to be remotely connected through the Apache Guacamole Remote Desktop interface, to each workstation (PC computer) to access the UI and myRIO device. Additionally, connection through Zoom will be needed if the students want to process audio signals streamed from their microphone. To do this, the equipment needed from the student's side are a microphone for real-time audio streaming and headphones for hearing the result from the audio processing.

Note: For doing these experiments, students are expected to have a basic knowledge of STFT, spectrograms, IIR filters, and PWM signals. A brief reminder about these topics is given at the beginning of each of the experiments.

4.2 Real-Time Audio Filtering

IIR stands for Infinite Impulse Response, which refers to a type of digital filter commonly used for audio filtering. IIR filters are characterized by feedback, which allows for more efficient implementation compared to FIR (Finite Impulse Response) filters.

In audio filtering applications, IIR filters are used to shape the frequency response of a signal by attenuating or amplifying specific frequency components. They can be useful for tasks such as equalization, low-pass filtering, high-pass filtering, band-pass filtering, and more.

IIR filters are defined by their transfer function, which describes the relationship between the input and output of the filter in the frequency domain. The transfer function typically takes the form of a ratio of polynomials in the z-domain, where z represents the complex variable of the frequency response.

The two main types of IIR filters commonly used in audio processing are:

1. **Butterworth Filters:** Butterworth filters have a maximally flat frequency response in the passband and a gradual roll-off in the stopband. They are commonly used for applications where a smooth frequency response is desired, such as audio equalization. Butterworth filters can be designed as low-pass, high-pass, band-pass, or band-stop filters.
2. **Chebyshev Filters:** Chebyshev filters trade off frequency response flatness for stteper roll-off. They can achieve steeper roll-off rates than Butterworth filters but with some ripple in the passband or stopband. Chebyshev filters are suitable when a more aggressive filtering is needed.

Designing and implementing IIR filters typically involves filter specification: Definition of the desired frequency response characteristics, such as cutoff frequency, passband ripple, stopband attenuation, and filter order.

It's important to note that designing and implementing IIR filters requires careful consideration of stability to avoid issues like filter instability or excessive ringing. Additionally, appropriate filter design and implementation techniques should be employed to meet the desired audio filtering requirements effectively.

4.2.1 Experimental setup

The setup for this experiment, from the laboratory side, is consisted of:

- NI myRIO 1900 device connected to a PC within the DSP laboratory;
- 3.5 mm audio jack male to male cable connected from the PC’s audio output to myRIO’s audio input;
- 3.5 mm audio jack male to male cable connected from myRIO’s audio output to the PC’s audio input;
- LabVIEW user interface.

The setup needed from the student's side is:

- Remote Desktop application;
- Zoom application (optional);
- Microphone (optional);
- Headphones (optional).

The audio signal can be streamed from the PC, for example from YouTube, or from a microphone. If a microphone is used as an audio source, then a Zoom meeting needs to be launched.

The user interface is shown in Figure 4-1. It provides a real-time plot of the raw audio waveform and a real-time raw vs. filtered audio spectrum. Students can change the predefined parameters about IIR filter specifications and the volume of the input audio signal.

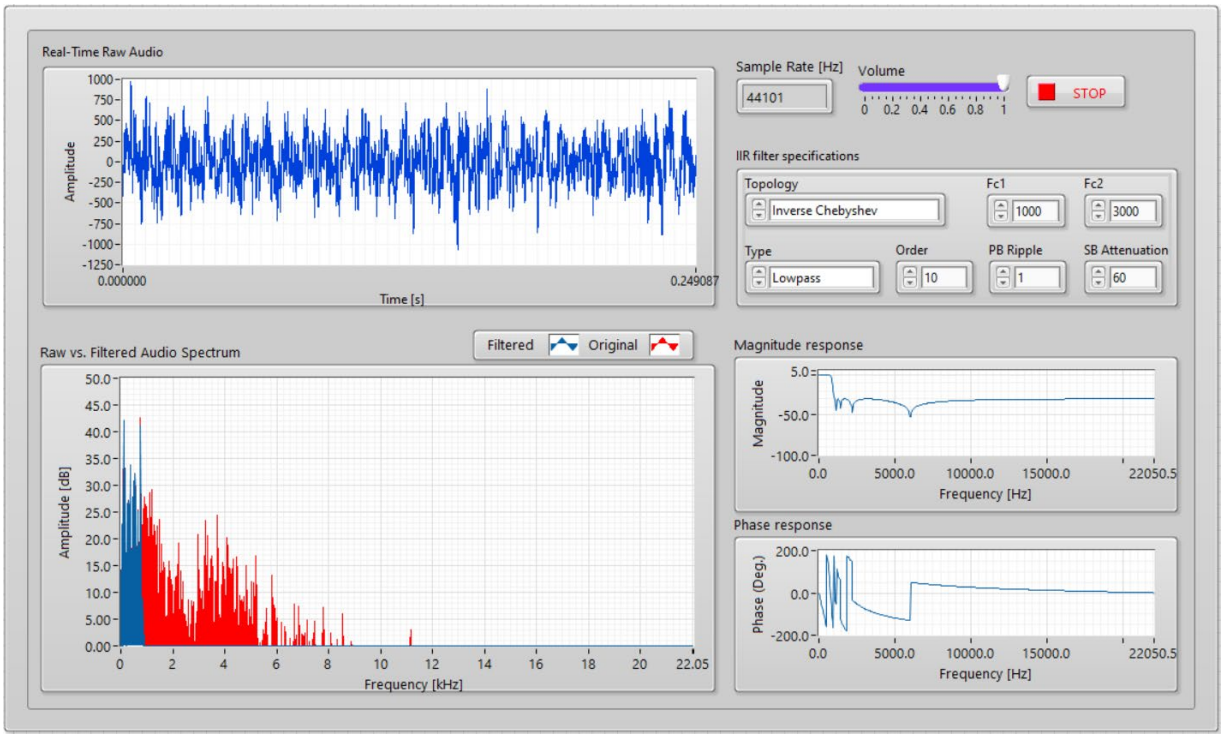

Figure 4-1. User interface for real-time audio filtering.

The offered filter specifications are the following:

- Filter topology:
  - Butterworth;
  - Chebyshev;
  - Inverse Chebyshev;
  - Elliptic;
  - Bessel;
- Filter type:
  - High-pass;

    - Low-pass;
    - Band-pass;
    - Band-stop;
- First cut-off frequency – Fc1;
- Second cut-off frequency – Fc2;
- Filter order;
- Pass band ripple;
- Stop band attenuation.

Note: When either low-pass or high-pass filter is used, the cut-off frequency is Fc1.

4.2.2 Goal of the experiment

The aim of this experiment is to perform real-time IIR filtering on the input audio signal and to observe the change in its spectrum according to the filter responses, as well as to listen to the filtered audio result.

4.2.3 Experimental results

Let's separate the input audio signal into different frequency ranges, called Bass, Midtone and Treble.

Bass is the lowest range of frequencies in an audio signal. To extract them we use a low-pass filter, for example a Butterworth filter, with a cut-off frequency $f_c = 450Hz$ and 3rd order. The result is shown in Figure 4-2. The spectrum of the raw audio signal is given in red, while the spectrum of the filtered audio signals is given in blue.

Midtone is the middle range of frequencies in an audio signal. Here we use a band-pass Butterworth filter, with cut-off frequencies $f_{c1} = 450Hz$ and $f_{c2} = 3000Hz$ and from 3rd order. The result is shown in Figure 4-3.

Treble is the highest range of frequencies in an audio signal. For that aim, we use a high-pass Butterworth filter with cut-off frequency $f_c = 3000Hz$ and from 3rd order. The result is shown in Figure 4-4.

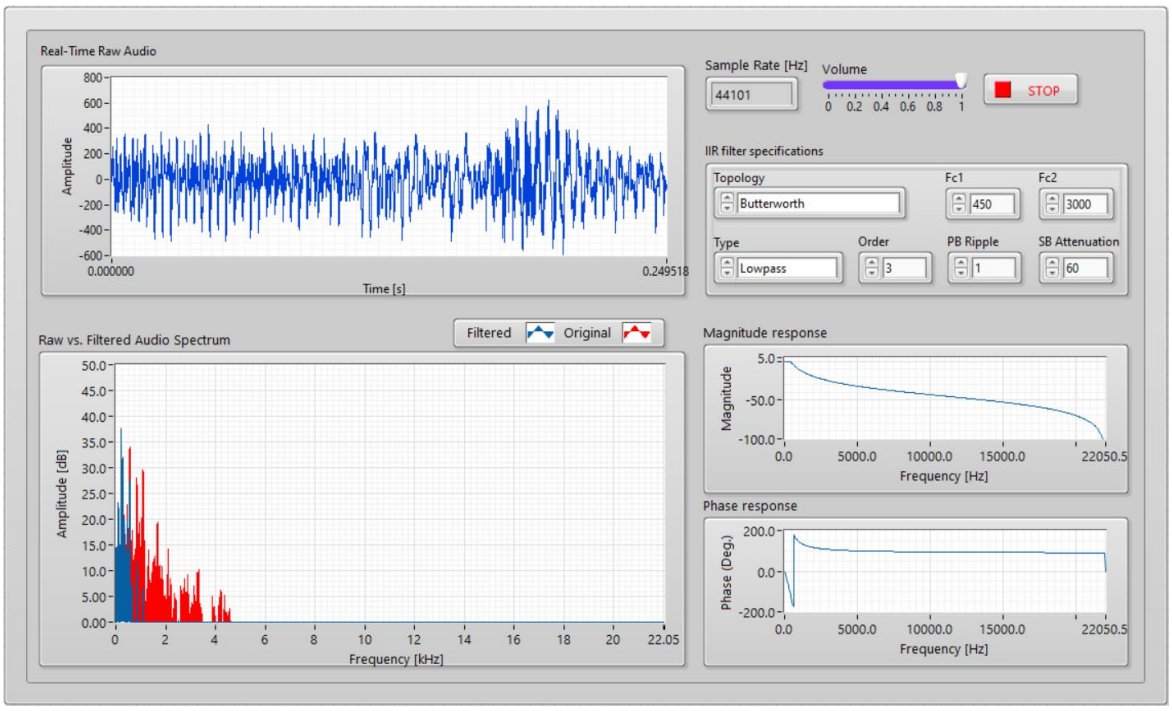

Figure 4-2 Bass frequency range from an input audio signal.

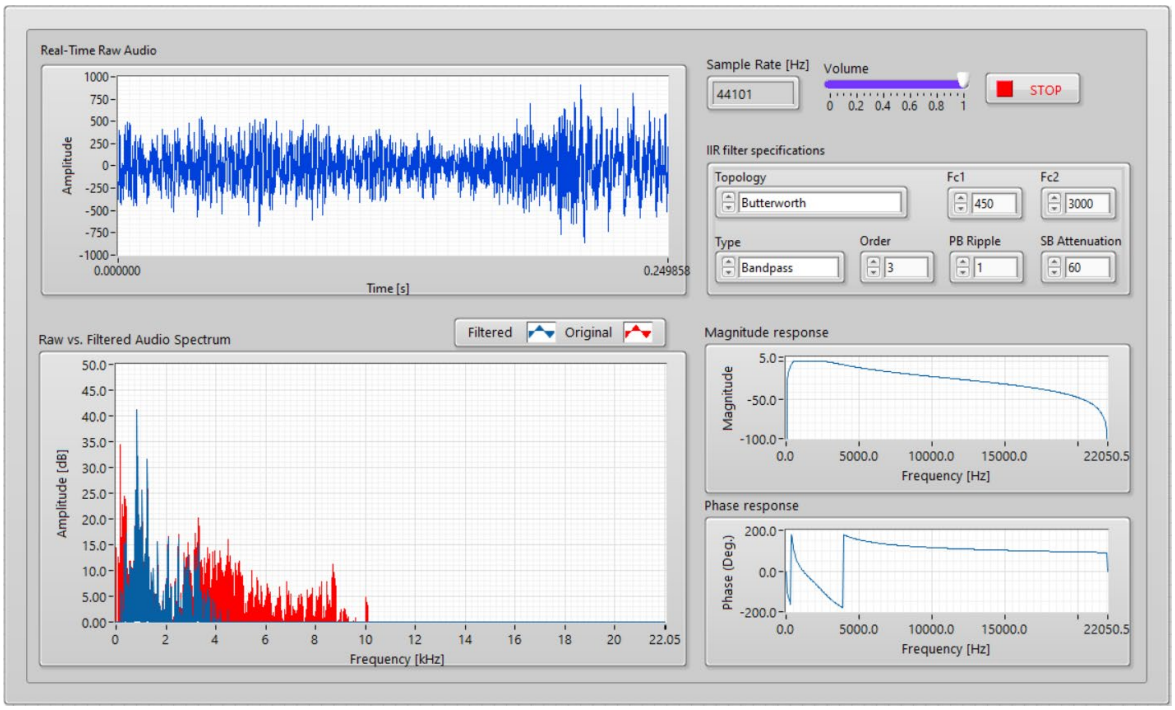

Figure 4-3. Midtone frequency range from an input audio signal.

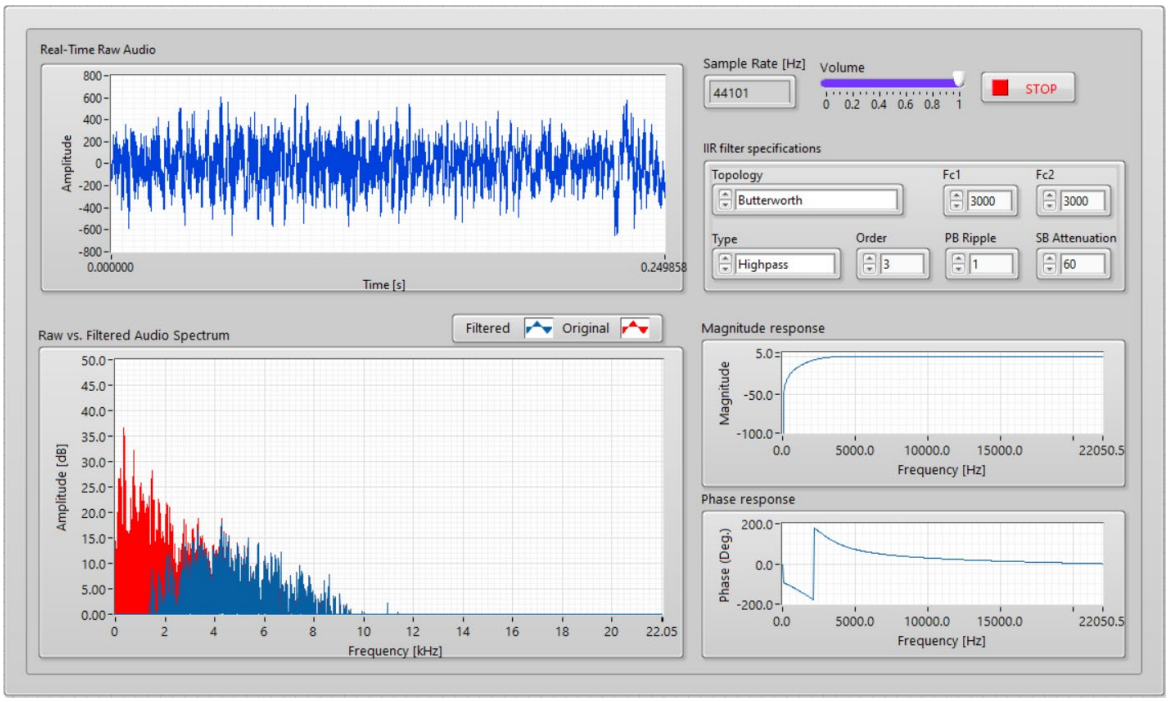

Figure 4-4. Treble frequency ranges from an input audio signal.

Task 1: Make a comparison between the spectrums and the responses of the filters.

Task 2: Try different filter specifications, observe and listen the result.

4.3 Real-Time Moving Spectrogram

STFT, which stands for Short-Time Fourier Transform, is a widely used signal processing technique for analyzing and visualizing the frequency components of a time-varying signal.

The STFT breaks down a signal into a series of short overlapping windows and performs a Fourier transform on each window to obtain its frequency content. This allows us to observe how the frequency content of the signal changes over time.

Here's how the STFT spectrogram is computed:

1. The input signal is divided into small segments or windows. The choice of window length and overlap between windows depends on the specific application and desired time-frequency resolution.
2. Each windowed segment of the signal is multiplied with a window function (e.g., Hamming, Hanning, Blackman …) to reduce spectral leakage and improve frequency resolution.
3. A Fourier transform (typically the Fast Fourier Transform, or FFT) is applied to each windowed segment, producing a frequency spectrum for that specific time interval.

4. The resulting spectra are stacked together to form a two-dimensional representation called the spectrogram. The x-axis represents time, and the y-axis represents frequency. The color intensity or grayscale value at each time-frequency point corresponds to the magnitude or power of the frequency component present in that segment.

By analyzing the spectrogram, you can identify the dominant frequencies at different points in time and observe how they evolve. Spectrograms are commonly used in various fields, including audio processing, speech analysis, music analysis, vibration analysis, and more.

It's worth noting that the STFT spectrogram is a time-frequency representation, providing valuable information about the spectral characteristics of a signal over time. However, it has limitations in terms of temporal and frequency resolution, and some details may be lost due to windowing and overlap. Different choices of window size, type and overlap can trade-off between time and frequency resolution to suit specific analysis requirements.

4.3.1 Experimental setup

The setup for this experiment, from the laboratory side, consists of:

- NI myRIO 1900 device connected to a PC within the DSP laboratory;
- 3.5 mm audio jack male to male cable connected from the PC's audio output to myRIO's audio input;
- LabView user interface.

The setup needed from the student's side is:

- Remote Desktop application;
- Zoom application (optional);
- Microphone (optional);
- Headphones (optional).

The audio signal can be streamed from the PC, for example from YouTube, or from a microphone. If a microphone is used as an audio source, then a Zoom meeting needs to be launched. In the following example the audio file sound1.wav is used, attached as s source file to this experiment.

The user interface is shown in Figure 4-5. It provides a real-time raw audio graph and a real-time moving spectrogram. Student can change the predefined parameters of the window type, and window length.

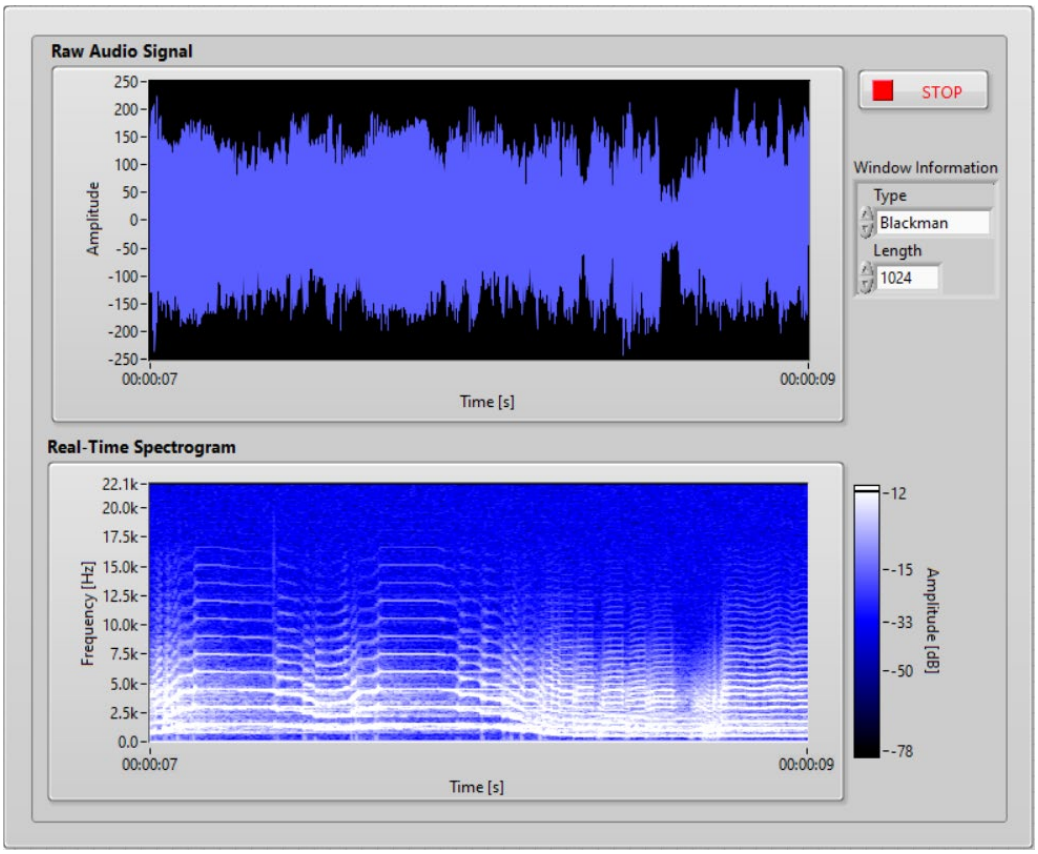

Figure 4-5. User interface for real-time moving spectrogram.

The offered window types are the following:

- Hanning;
- Hamming;
- Blackman – Harris;
- Exact Blackman;
- Blackman;
- Flat Top.

4.3.2 Goals of the experiment

The goal of this experiment is to observe the impact of the different window type and different window length on the quality of the audio spectrogram.

4.3.3 Experimental results

Choose the window type to be "Hamming". Let's try two different window lengths and observe the obtained results.

4.3.3.1 Narrow window

In the first case we can use a narrow window with a length equal of 64. The result is shown in Figure 4-6. It can be seen that this window gives good time localization but very poor frequency resolution.

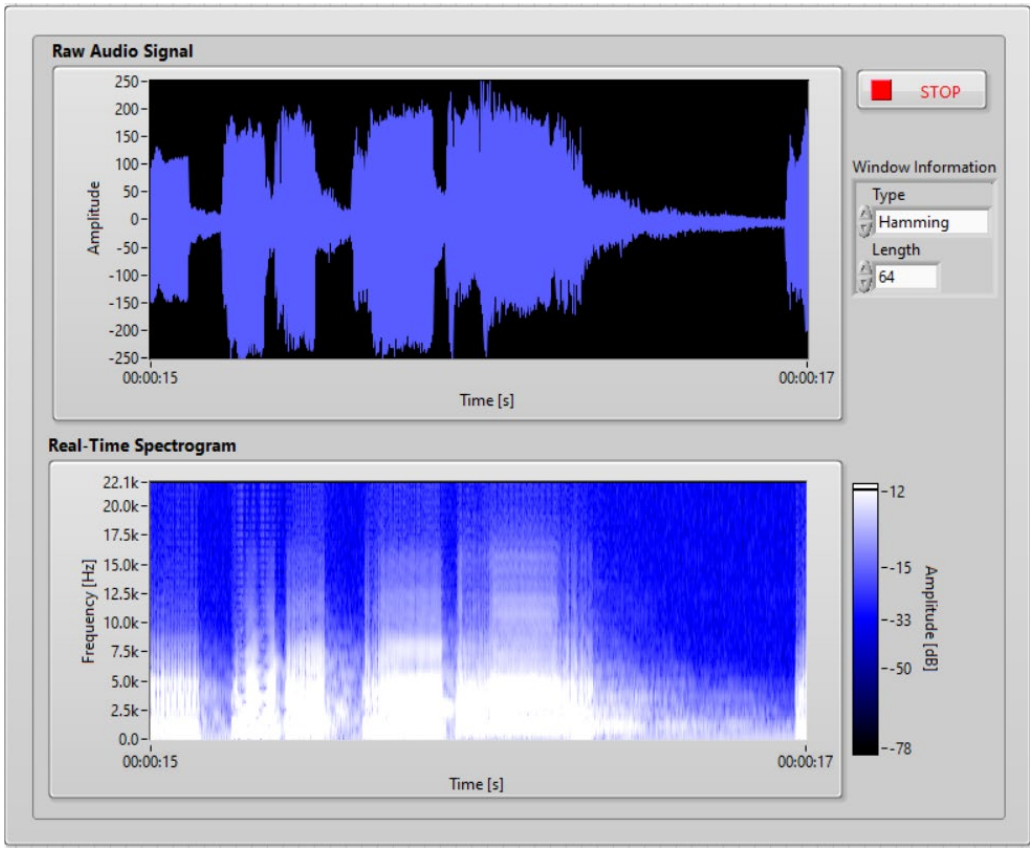

Figure 4-6. Narrow window.

4.3.3.2 Wide window

In the second case we will set the length of the window to 512. From Figure 4-7, we can conclude that this window gives better frequency resolution, but now the time localization is worse.

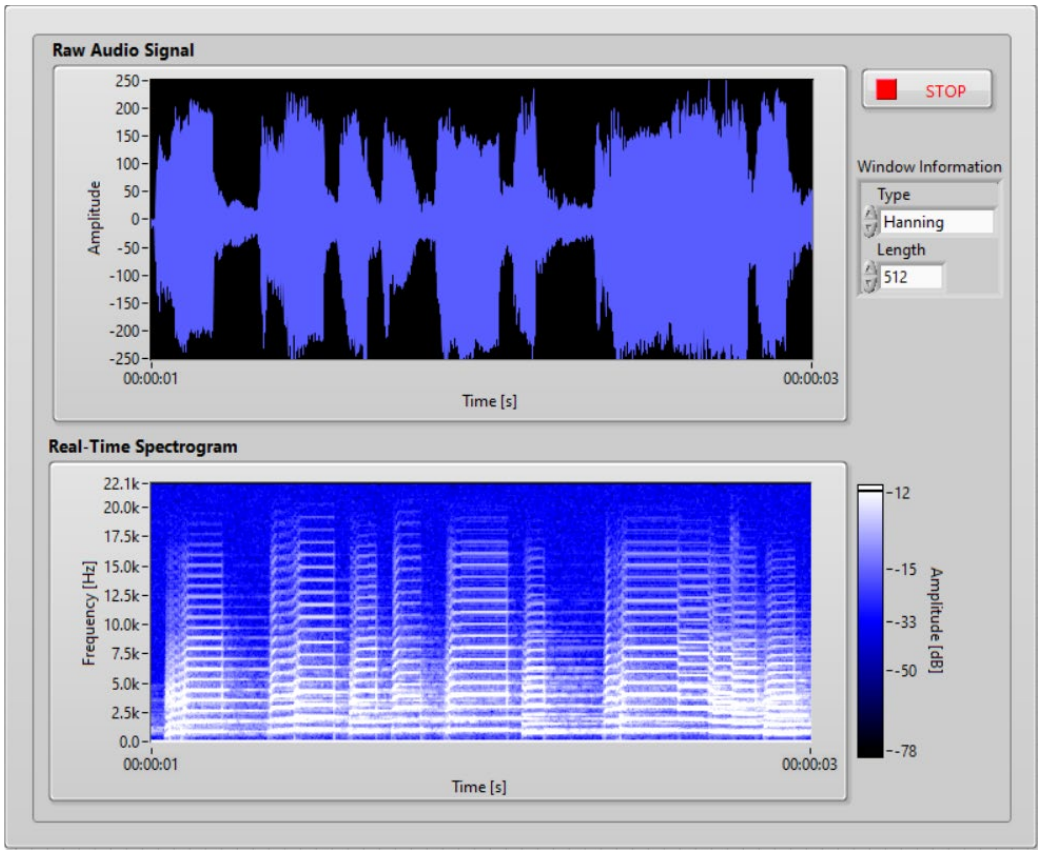

Figure 4-7. Wide window.

We can conclude that a narrow window gives better time localization but a poor frequency resolution, while a wide window gives better frequency resolution, but a worse time localization.

Task 1: Try different window lengths and observe the results.

Task 2: Make a comparison on the impact of the window type on the audio spectrogram.

4.4 Real-Time PWM Signal Generation

PWM stands for Pulse Width Modulation, which is a modulation technique used in electronics to encode information in the form of a pulsing signal. It is widely used in various applications, including controlling the speed of motors, dimming LEDs, generating analog signals, and more.

In PWM, the signal is typically a square wave with a fixed frequency and variable duty cycle. The duty cycle refers to the ratio of the pulse width (ON time) to the total period of the signal. By changing the duty cycle, the average power delivered to a device or component can be controlled.

For example, in motor control, a PWM signal can be used to regulate the speed of a motor. By adjusting the duty cycle of the PWM signal, the effective voltage and power applied to the motor

can be varied. A higher duty cycle will result in a higher average voltage and power, thus increasing the motor speed, while a lower duty cycle will reduce the average voltage and power, slowing down the motor.

PWM signals are generated using specialized circuits or microcontrollers capable of producing such signals. The frequency of the PWM signal determines how quickly it switches between high and low states, while the duty cycle determines the percentage of time the signal spends in the high state.

The advantage of PWM is that it allows for precise control of power levels or analog-like signals using digital components. By rapidly switching the signal on and off, the average power delivered can be adjusted smoothly, creating the effect of a variable voltage or analog control.

4.4.1 Experimental setup

The setup for this experiment, from the laboratory side, is consisted of:

- NI myRIO 1900 device connected to a PC within the DSP laboratory;
- LabView user interface.

The setup needed from the student's side is:

- Remote Desktop application.

The user interface is shown in Figure 4-8. It provides real-time PWM signal generation. Student can change the predefined parameters about the frequency and duty cycle of two PWM signals and observe their graphical representation and their control over two LEDs. The firs PWM signal also controls the blinking of LED 0 on myRIO.

4.4.2 Goals of the experiment

The aim of this experiment is to observe how a PWM signal is interpreted by digital and analog outputs, to which LEDs are connected.

4.4.3 Experimental results

Generate two identical PWM signals and observe how these signals are interpreted by the digital and the analog output, both used to control LED lightning. The light intensity of the LED is between 0 and 255.

In the first case, in the case of the digital output, the generated signal is interpreted as it has only two values or states, "0" and "1" or "OFF" and "ON". Hence, in this case the LED is blinking.

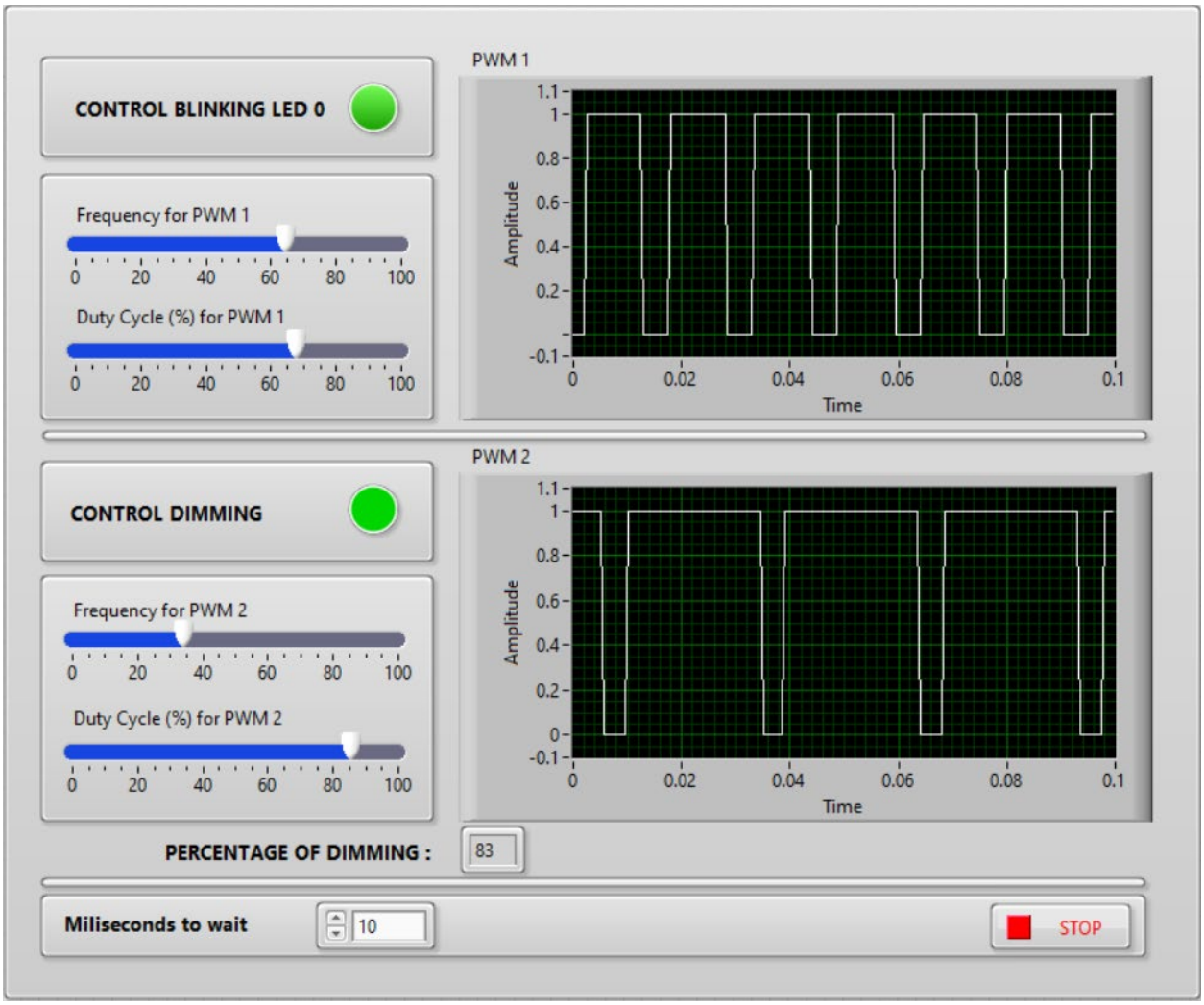

Figure 4-8. User interface for real-time PWM signal generation.

The blinking speed depends on the selected frequency, while the time spent in state "ON" depends on the selected duty cycle.

In the second case, in the case of the analog output, the generated PWM signal regulates the LED dimming (regulates the lightning intensity in the range of values between 0 and 255). By changing the frequency and the duty cycle you can change the percentage of dimming according to the following equation:

$$Percantageofdimming = Dutycycle - Frequency \times 6 \times 10^{-4}$$

Task: Try different frequencies and duty cycles and observe the results.

4.5 Conclusion

These experiments allow students to practically apply their knowledge from the field of IIR filtering, STFT, spectrograms and PWM signal generation, whether they work physically in a laboratory or remotely from home. With the friendly user interfaces, students can try different cases, compare results and draw conclusions.