ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 1 | January 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 5.614||ISI Value: 1.188

ANALYSIS AND IMPROVISATION IN EXTRACTING AUDIO SIGNAL AMPLITUDE USING LABVIEW

Adarsh V Srinivasan
Department of Electronics and Communication Engineering
PSG College of Technology
Coimbatore, India

Mr. N. Saritakumar
Assistant Professor (Senior Grade)
Department of Electronics and Communication Engineering
PSG College of Technology
Coimbatore, India

Article DOI: https://doi.org/10.36713/epra3965

ABSTRACT

In this paper, either a pre-recorded audio or a newly recorded audio is processed and analysed using the LabVIEW Software by National Instruments. All the data such as bitrate, number of channels, frequency, sampling rate of the Audio are analyzed and improvising the signal by a few operations like Amplification, De-Amplification, Inversion and Interlacing of Audio Signals are done. In LabVIEW, there are a few Sub Virtual Instrument's available for Reading and Writing Audio in .wav formats and using them and array Sub Virtual Instrument, all the processing are done.

KEYWORDS: ***Virtual Instrumentation (VI), LabVIEW (LV), Audio, Processing, audio array.***

I. INTRODUCTION

Audio Signal

An Audio Signal is a representation of sound, typically as an electrical voltage. Audio signals have frequencies in the frequency range of roughly 20 to 20,000 Hz (the limits of human hearing). Audio signals may be synthesized directly, or may originate at a transducer such as a microphone, musical instrument pickup, phonograph cartridge, or tapehead. Loudspeakers or headphones convert an electrical audio signal into sound. Digital representations of audio signals exist in a variety of formats.

Signal Flow

Signal flow is the path an audio signal will take from source (microphone) to the speaker or recording device. It is most frequently in a recording studio setting, where the signal flow is often very long and convoluted as the electric signal may pass through many sections of a large Analog console, external audio equipment, and even different rooms.

Parameters

Audio signals may be characterized by parameters such as their bandwidth, power level in decibels (dB), and voltage level. The relation between power and voltage is determined by the impedance of the signal path, which may be single-ended or balanced.

Audio signals have somewhat standardized levels depending on application. Outputs of professional mixing consoles are most commonly at line level. Microphones generally output at a lower level, commonly referred to a "mic level". Consumer audio equipment will also output at a lower level.

Digital Audio is a technology that can be used for sound recording and reproduction using audio signals that have been encoded in digital form. Following significant advances in digital audio technology during the 1970s, it gradually replaced Analog audio technology in many areas of audio engineering and telecommunications in the 1990s and 2000s.

In a digital audio system, a microphone converts sound to an analog electrical signal, then an analog-to-digital converter (ADC) typically using pulse-code modulation—converts the analog signal into a digital signal. This digital signal can then be recorded, edited and modified using digital audio tools. When the sound engineer wishes to listen to the recording on headphones or loudspeakers (or when a consumer wishes to listen to a digital sound file of a song), a digital-to-analog converter (DAC) performs the reverse process, converting a digital signal back into an analog signal, through an audio power amplifier and send to a loudspeaker.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 1 | January 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 5.614||ISI Value: 1.188

Digital audio systems may include compression storage processing and transmission components. Conversion to a digital format allows convenient manipulation, storage, transmission and retrieval of an audio signal. Unlike analog audio, in which making copies of a recording results in generation loss, a degradation of the signal quality, when using digital audio, an infinite number of copies can be made without any degradation of signal quality.

Digital Audio Formats

A few of the Digital Audio Formats used currently are listed below:

1. MP3
2. AAC
3. WMA
4. FLAC
5. 3GP
6. DVF
7. M4A
8. RAW
9. WAV

Digital Audio Processing

Audio signal processing or audio processing is the intentional alteration of audio signals often through an audio effect or effects unit. As audio signals may be electronically represented in either digital or Analog format, signal processing may occur in either domain. Analog processors operate directly on the electrical signal, while digital processors operate mathematically on the digital representation of that signal.

LABVIEW

Laboratory Virtual Instrument Engineering Workbench (LabVIEW) is a system-design platform and development environment for a visual programming language from National Instruments.

The graphical language is named "G"; not to be confused with G-code. Originally released for the Apple Macintosh in 1986, LabVIEW is commonly used for data acquisition, instrument control, and industrial automation on a variety of operating systems(OS), including Microsoft Windows, various versions of Unix, Linux, and MacOS.

The latest versions of LabVIEW are LabVIEW 2017 and LabVIEW NXG 1.0, released in May 2017.

Graphical Programming

LabVIEW integrates the creation of user interfaces (termed front panels) into the development cycle. LabVIEW programs-subroutines are termed virtual instruments (VIs). Each VI has three components: a block diagram, a front panel, and a connector panel. The last is used to represent the VI in the block diagrams of other, calling VIs. The front panel is built using controls and indicators. Controls are inputs: they allow a user to supply information to the VI. Indicators are outputs: they indicate, or display, the results based on the inputs given to the VI. The back panel, which is a block diagram, contains the graphical source code. All of the objects placed on the front panel will appear on the back panel as terminals. The back panel also contains structures and functions which perform operations on controls and supply data to indicators. The structures and functions are found on the Functions palette and can be placed on the back panel. Collectively controls, indicators, structures, and functions will be referred to as nodes. Nodes are connected to one another using wires, e.g., two controls and an indicator can be wired to the addition function so that the indicator displays the sum of the two controls. Thus a virtual instrument can be run as either a program, with the front panel serving as a user interface, or, when dropped as a node onto the block diagram, the front panel defines the inputs and outputs for the node through the connector pane. This implies each VI can be easily tested before being embedded as a subroutine into a larger program.

The graphical approach also allows nonprogrammers to build programs by dragging and dropping virtual representations of lab equipment with which they are already familiar. The LabVIEW programming environment, with the included examples and documentation, makes it simple to create small applications. This is a benefit on one side, but there is also a certain danger of underestimating the expertise needed for high-quality G programming. For complex algorithms or large-scale code, it is important that a programmer possess an extensive knowledge of the special LabVIEW syntax and the topology of its memory management. The most advanced LabVIEW development systems offer the ability to build stand-alone applications. Furthermore, it is possible to create distributed applications, which communicate by a client–server model, and are thus easier to implement due to the inherently parallel nature of G.

Interfacing to Devices

LabVIEW includes extensive support for interfacing to devices, instruments, camera, and other devices. Users interface to hardware by either writing direct bus commands (USB, GPIB, Serial) or using high-level, device-specific, drivers that provide native LabVIEW function nodes for controlling the device.

LabVIEW includes built-in support for NI hardware platforms such as CompactDAQ and CompactRIO, with a large number of device-specific blocks for such hardware, the *Measurement and Automation eXplorer* (MAX) and *Virtual Instrument Software Architecture* (VISA) toolsets.

National Instruments makes thousands of device drivers available for download on the NI Instrument Driver Network (IDNet).

Code compiling

LabVIEW includes a compiler that produces native code for the CPU platform. This aids performance. The graphical code is translated into executable machine code by interpreting the syntax and by compiling. The LabVIEW syntax is strictly enforced during the editing process and compiled into the executable machine code when requested to run or upon saving. In the latter case, the executable and the source code are merged into a single file. The executable

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 1 | January 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 5.614||ISI Value: 1.188

runs with the help of the LabVIEW run-time engine, which contains some precompiled code to perform common tasks that are defined by the G language. The run-time engine reduces compiling time and provides a consistent interface to various operating systems, graphic systems, hardware components, etc. The run-time environment makes the code portable across platforms. Generally, LabVIEW code can be slower than equivalent compiled C code, although the differences often lie more with program optimization than inherent execution speed.

Large libraries

Many libraries with a large number of functions for data acquisition, signal generation, mathematics, statistics, signal conditioning, analysis, etc., along with numerous for functions such as integration, filters, and other specialized abilities usually associated with data capture from hardware sensors is enormous. In addition, LabVIEW includes a text-based programming component named MathScript with added functions for signal processing, analysis, and mathematics. MathScript can be integrated with graphical programming using *script nodes* and uses a syntax that is compatible generally with MATLAB.

Parallel programming

LabVIEW is an inherently concurrent language, so it is very easy to program multiple tasks that are performed in parallel via multithreading. For example, this is done easily by drawing two or more parallel while loops and connecting them to two separate nodes. This is a great benefit for test system automation, where it is common practice to run processes like test sequencing, data recording, and hardware interfacing in parallel.

II. DESIGN AND METHODOLOGY

A. Design

The design is done in a way such that an available audio in .wav format or a newly recorded audio in .wav format is analysedto find the Audio Elements and store them in an Array for further Analysis and Processing.

The Elements are manipulated as a whole for both Analysis and Processing using different array operation Sub VI's. And again, the Audio elements are stored as a waveform and is written into the Hard Disk (HDD).

The Design has Sub VI's related to Array Operations, Waveform Operations and Components, Laplace Operations, Sound Sub VI's and reading and writing Sub VI's.

In case of a new audio file to be recorded and analysed and processed, either a laptop microphone or an external microphone. If the laptop does not have an inbuilt microphone or if is damaged, a cheap and cost effective microphone is built and is used.

Fig 1. Control flow the program

B. Methodology

The Program is Initialized and is run on LabVIEW by National Instruments. There are various steps in the execution.

Following are the steps involved in this Program.

Input

A pre-recorded or a Live recorded Audio File in .wav format in read into LabVIEW using 'Read' Sub VI under Sound by giving a path to the Source Audio. For a pre-recorded Audio, the Source of the .wav file can directly be given in the path. To incorporate a live recorded Audio, an external mic is built using a Condenser mic in an amplifier circuit. The mic is connected to the Laptop and using 'Audacity' software, the audio is recorded lively and is saved in the computer in .wav format. The saved audio file's path can be used in the LabVIEW to Analyse and Process the newly recorded Audio. It is then passed onto a Sub VI called 'Index Array' to store all the Audio Elements in an array for further Analysis and Processing. This array forms the basis for all other operations.

Fig 2. Input path of the source audio

Different Processes are done in the VI.

Displaying and Playing the Original Audio

Using a 'Waveform Graph' Sub VI, the Original Audio Signal is displayed as a graph.

The Waveform is fed into a Sub VI called 'Play Sound' to play the Original Audio File.

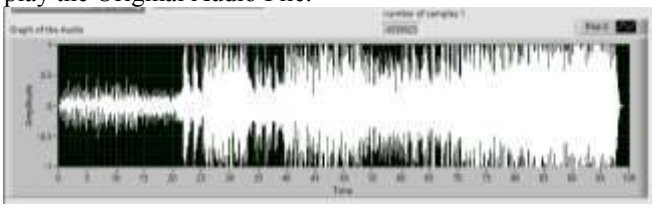


Fig 3. Original audio waveform

Amplifying the Audio

The Audio element are obtained from the Index Array and is fed into a 'Get Waveform Components' Sub VI which converts all the elements into Integer format and stores it as a separate Array.

The Array is then multiplied by a constant value '5' in order to increase the value of each sample by 5 times which in turn increases the Amplitude of the Audio by 5 times. Hence, the Audio is Amplified.

The Amplified array elements are fed into a Sub VI called 'Build Waveform' and is given as an input to 'Waveform Graph' Sub VI and 'Play Sound' Sub VI.

When the Amplified Audio file is written in the specified location is played outside of LabVIEW, the difference can be viewed and noticed.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 1 | January 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 5.614||ISI Value: 1.188

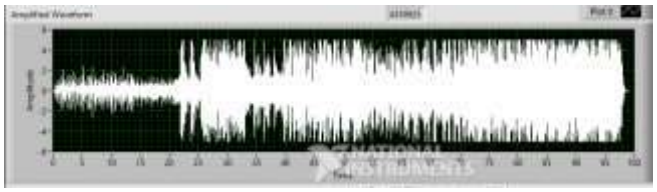


Fig 4. Amplified waveform

De-Amplifying the Audio

The Audio element are obtained from the Index Array and is fed into a 'Get Waveform Components' Sub VI which converts all the elements into Integer format and stores it as a separate Array.

The Array is then divided by a constant value '5' in order to decrease the value of each sample by 5 times which in turn decreases the Amplitude of the Audio by 5 times. Hence, the Audio is De-Amplified.

The De-Amplified array elements are fed into a Sub VI called 'Build Waveform' and is given as an input to 'Waveform Graph' Sub VI and 'Play Sound' Sub VI.

When the Amplified Audio file is written in the specified location is played outside of LabVIEW, the difference can be viewed and noticed.

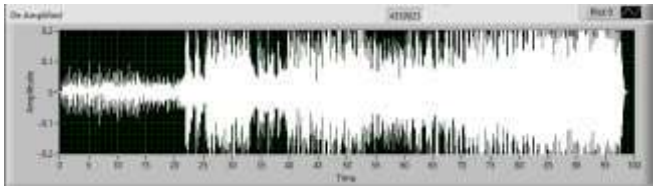


Fig 5. De-amplified waveform

Inversion of Audio

The Audio element are obtained from the Index Array and is fed into a 'Get Waveform Components' Sub VI which converts all the elements into Integer format and stores it as a separate Array.

The Array is then inverted using Sub Vi under Array Operations called 'Reverse 1D Array' which Inverts the complete 1D Array.

The Inverted array elements are fed into a Sub VI called 'Build Waveform' and is given as an input to 'Waveform Graph' Sub VI and 'Play Sound' Sub VI.

Fig 6. Inverted waveform

Interlacing of Audios
Original and Inverted

Using the Sub VI 'Add' under 'Mathematical Operations', the integer array of both the Original Audio and the Inverted Audio are added and is fed to 'Build Waveform' Sub VI and then to 'Play' Sub VI.

This built waveform will contain both the Original and Inverted Audio in the same file, i.e, the audio elements are interlaced.

When this interlaced Audio is played, both the files play as a single audio and the difference can be clearly noted.

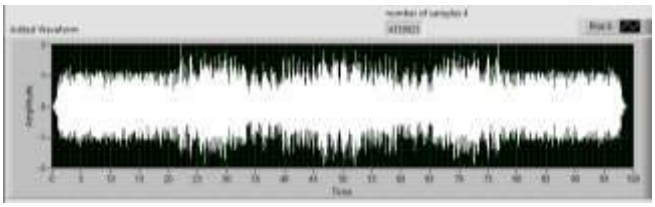


Fig 7. Original and inverted – interlaced waveform

Amplified and Inverted

Using the Sub VI 'Add' under 'Mathematical Operations', the integer array of both the Amplified Audio and the Inverted Audio are added and is fed to 'Build Waveform' Sub VI and then to 'Play' Sub VI.

This built waveform will contain both the Amplified and Inverted Audio in the same file, i.e, the audio elements are interlaced.

When this Audio is played, the Audio playing in the Original direction is heard more clearly than the file which is being played in backward since the forward playing Audio is Amplified.

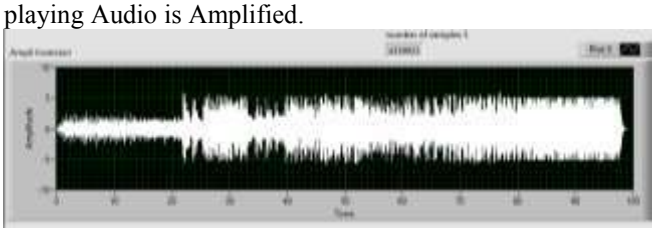


Fig 8. Amplified and inverted – interlaced waveform

Writing the processed Audios in the Hard Disk

To write the processed Audios to the Hard Disk, the processed waveform is fed into a Sub Vi called 'Build Array' which converts the integer array to Audio Elements Array.

The converted array is given as an input to a Sub VI called 'Sound File Write' and a path is given to it for the location.

Fig 9. Output paths of processed audios

Sampling Interval of Audio

To find the Sampling Interval of the given Audio, from the 'Get Waveform Components' Sub VI of the Original Audio, the Sub VI is dropped down to get more Attributes and 'dt' option must be selected.

Adding an 'Indicator' Sub VI would display the Sampling Interval.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 1 | January 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 5.614||ISI Value: 1.188

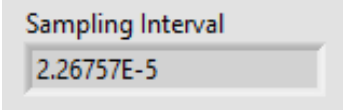


Fig 10. Sampling Interval

Frequency of Audio

To find the frequency of the given audio file, the sampling interval value is just inverted using a '1/x' Sub VI under 'Mathematical Operations'.

A Indicator is created for the Sub VI to display the Sampling Frequency of the Audio.

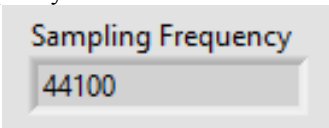


Fig 11. Sampling Frequency

Number of Samples

The number of samples of an audio file is found by using a pre-defined Sub VI called 'Number of Waveform Samples' and an indicator is added to it to display the value.

The more the number of samples, the more accurate the Audio will be.

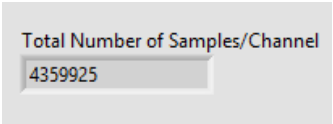


Fig 12. Total number of samples

Maximum and Minimum Amplitude

From the Original, Amplified and De-Amplified integer arrays, they are fed into a Sub VI called 'Max and Min' to obtain the values of Maximum Amplitude and Minimum Amplitude and an Indicator is created to display the values and is compared.

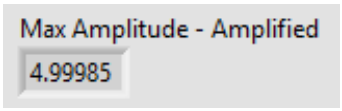


Fig 13. Amplitude of amplified waveform

Analog to Digital Conversion

The Audio is in Analog form and in order to Convert it to Digital, 'Analog to Digital Converter' Sub VI is used.

The converted digital format is in Binary format and is displayed as an array.

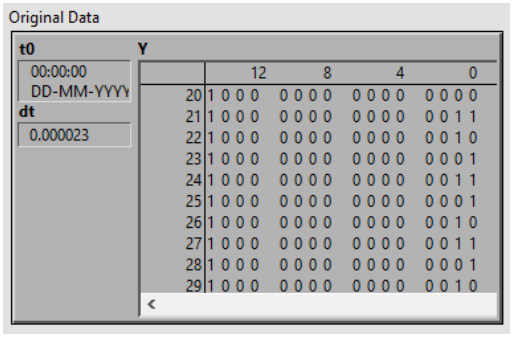


Fig 14. Digital data of original audio file

General Overview

A general overview of the audio can also be viewed by using the Sub VI called 'Sound File Info' and adding another Sub VI called 'Overview' to display the General details.

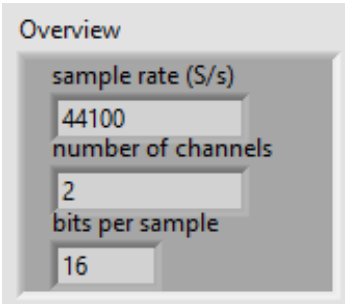


Fig 15. Overview of the audio file

III. BLOCK DIAGRAM OF VI

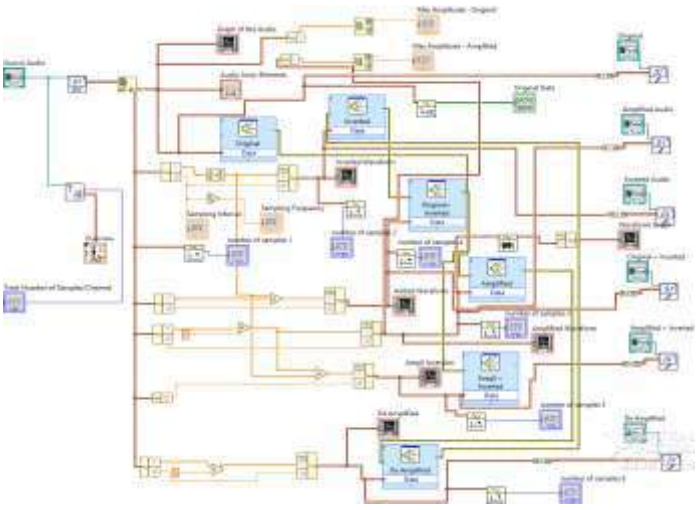


Fig 16. Block diagram of the top level VI

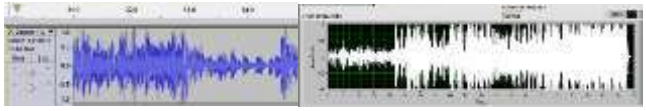
ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 1 | January 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 5.614||ISI Value: 1.188

IV. ADVANTAGES

When compared with other audio editing applications, only one process can be achieved at a time. When LabVIEW is used, multiple processes can be done at a single click.

Since, LabVIEW is a graphical programming language, everyone can easily learn to use it and code in it. The amplification and de-amplification factors can be changed by user too, once the person learns programming in LabVIEW.

Other audio editing applications have the value of audio loudness only from -1 dB to +1 dB. Using LabVIEW will display the original amplitude as such without any clipping which most applications do, which will eventually depreciate the audio quality.



a. Original files in Audacity and LabVIEW

b. Amplified files in Audacity and LabVIEW

Fig 17. Comparison of Audacity with LabVIEW

The digital form of the audio will not be displayed by any other audio editing softwares other than LabVIEW.

V. RESULT & ANALYSIS BY APPLICATION

As LabVIEW was used for audio signal processing and analysis, the amplified version of the .wav audio file had a maximum amplitude of 4.99 dB which is a proof for audio not being clipped at the peaks as seen in Fig 17(a) & Fig 17(b).

Since all the processes are done using only one VI (Fig 16), once the 'RUN' button is clicked, the audio is processed and is stored in the output paths at an instant.

Evidently, from Fig 10,11,12,13&15 the analysis is done and is displayed to the user.

Applications

This software program can be used in Entertainment industry where the audio is recorded or saved as RAW format (.wav) which is large in size and is time consuming to edit in other applications.

It can be used display all the details of an audio even when it is unknown and then can be processed accordingly.

VI. CONCLUSION

Audio signal processing and analysis was implemented using LabVIEW where an audio was given as an input and all of it's details were analysed and displayed followed by processing it with a few operations. This paper overcomes the existing disadvantage of the existing audio editing applications which are more time consuming and hard to learn.

Future Enhancement

In the coming future, this program can be enhanced to it's full potential which can reduce noise of the given audio signal, manipulate it's length, append audios, introduce the playback speed according to user's wish.

The obtained digital data too can be manipulated using LabVIEW to add/remove noise in the audio file.

VII. REFERENCES

1. *Jffery Travis, "LabVIEW for Everyone", Prentice Hall, 2002.*
2. *Audacity software for creating .wav files available at www.audacity.com.*
3. *NI LabVIEW2014 from www.ni.com.*
4. *Rick Bitter, TaqiMohiuddin, Matt Nawrocki, "LabView: Advanced Programming Techniques, Second Edition", CRC Press, 2006.*
5. *Peter A. Blume, "The LabVIEW Style Book", Pearson Education, 2007.*
6. *John Essick, "Hands-On Introduction to LabVIEW for Scientists and Engineers", Oxford University Press, 2015.*
7. *Gary W. Johnson, "LabVIEW Graphical Programming: Practical Applications in Instrumentation and Control", McGraw-Hill, 1997.*
8. *Nasser Kehtarnavaz, Namjin Kim, " Digital Signal Processing System-level Design Using LabVIEW", Newnes, 2005.*
9. *Cory Clark, "LabVIEW Digital Signal Processing: and Digital Communications", McGraw Hill Professional, 2005.*
10. *"PLAYING .WAV FILES IN LABVIEW" [Website]*
11. *Available:http://digital.ni.com/public.nsf/allkb/400FA87E2B9B46C68625651D00483B18*
12. *"PLAYING WAV FILES USING AUDACITY TOOLKIT" [Website]*
13. *Available:http://esatjournals.net/ijret/2015v04/i02/IJRET20150402062.pdf*
14. *"READING AND WRITING .WAV FILES IN LABVIEW" [Website]*
15. *Available:https://archive.cnx.org/contents/007678e8-a65f-4f03-8637-8505b0bb30c7@6/reading-and-writing-audio-files-in-labview*