ABSTRACT

Digital Audio Workstations (DAW) are software program used for composing, producing, recording, mixing, and editing audios like music, speech, radio, podcasts and nearly any other situation where complex recorded audio is needed and MIDI. In this project, we try to build a basic version of DAW to acquire sounds from real-time and process them using LabVIEW which is a system-design platform and development environment for a visual programming language.

This project requires the use of an audio interface to record live audio. An audio interface (sound card) is an Analog to Digital converter used to convert Analog sounds into a digital computer in a digital form. Sounds in real-time are Analog in nature, which computers cannot process and store. To store the data in computers, it must be in digital format. We use an Audient Evo 4 audio interface and an Electric Violin in this project. Sounds from the Violin are fed to the audio interface which converts it to digital format which are fed into the computer and stored. We record multiple sounds and try to mix them into one single audio file. We merge two WAV files and write it into a single WAV.

.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

Digital Audio Workstations (DAW) are software program used for composing, producing, recording, mixing, and editing audios like music, speech, radio, podcasts and nearly any other situation where complex recorded audio is needed and MIDI. Some of popularly used DAWs are Logic Pro, FL Studio, Garage Band etc. From a short Instagram video to a large-scale movie production, DAW is used everywhere to record and edit music. Mixing and mastering the recorded samples are also done on DAWs to produce the best audio output. Many techniques of Signal Processing like Source separation, audio reversal, increasing the audio speed etc are used in DAWs practically.

This project requires the use of an audio interface to record live audio. An audio interface (sound card) is an Analog to Digital converter used to store Analog sounds into a digital computer in a digital form. Sounds in real-time are Analog in nature, which computers cannot process and store. To store the data in computers, it must be in digital format. We use an Audient Evo 4 audio interface and an Electric Violin in this project. Sounds from the Violin are fed to the audio interface which converts it to digital format which are fed into the computer and stored. We record multiple sounds and try to mix them into one single audio file. We merge two WAV files and write it into a single WAV.

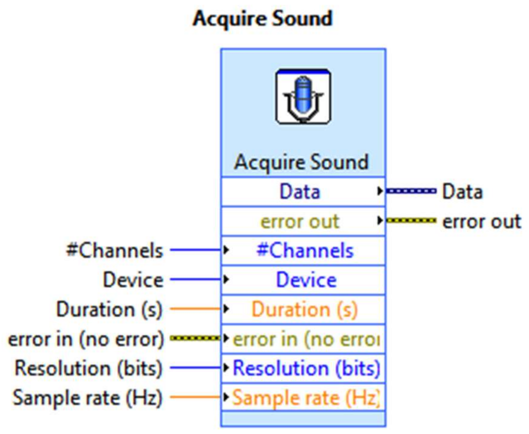
The audio format used is WAV. WAV format is considered to have the highest quality of audio. Uncompressed WAV files are large, so file sharing of WAV files over the Internet is uncommon except among video, music, and audio professionals where the uncompressed form has become the most popular of all audio formats and, for most, high speed large bandwidth web connections are commonplace. Many audios and music software manufacturers now favour it as their default file format though others are often supported. The high resolution of the format makes it suitable for retaining first generation archived files of high quality, for use on a system where disk space is not a constraint, or in applications such as audio editing where the time involved in compressing and uncompressing data, and the losses in quality of such conversions are a concern.

CHAPTER 2 : METHODOLOGY

In this simpler version of a DAW, we just record and mix two audios. One audio is recorded live from the Electric Violin and the other audio sample is an audio file already stored on the system. The audio format used is WAV. WAV format is considered to have the highest quality of audio. In LabVIEW there are many VIs which can do many interesting processes. Some of them are for sounds. Under Sounds pallette in the Graphics and Sounds section, we can find an Express VI called Acquire Sound. This VI directly takes in sound from an input. The input audio is then plotted on a Waveform Chart to visualise the audio. The appropriate input is selected reference to the Audio Interface we use. The duration is manually set in this project to 17 secs. The path for the mixed audio to store is given manually. The audio file we use as the secondary audio is **D#.wav**. This audio file is perfectly mixed with the recorded audio sample from the Electric Violin and is stored in the path provided as **test1.wav** on Desktop. The basic use of a DAW is to mix several audio files into a single audio file.

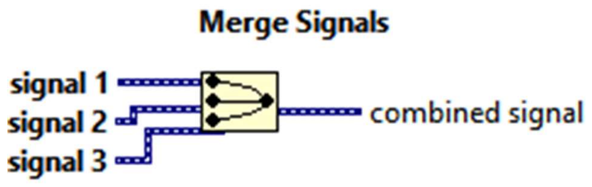
2.1 Acquire Sound VI

Acquires data from a sound device. This Express VI automatically configures an input task, acquires the data, and clears the task after the acquisition completes.



2.2 Merge Signals Function

Merges two or more supported signals, such as scalar numeric, 1D or 2D arrays of numeric, scalar Booleans, 1D or 2D arrays of Booleans, waveforms, or 1D arrays of waveforms, into a single output. Resize the function to add inputs. This function automatically appears on the block diagram when you wire a signal output to the wire branch of another signal. This function outputs a single wire with a dynamic data type. Internally, an array of waveforms composes this wire. When you merge signals, each input signal becomes one or more elements in the waveform array.



2.3 Sound File Read Simple

Reads data from a .wav file into an array of waveforms. This VI automatically opens, reads, and closes the .wav file.

number of samples/ch specifies the number of samples per channel to read from the file. -1 specifies all samples.

path specifies the absolute path to the wave file. If the path is empty or invalid, the VI returns an error. The default is <Not A Path>.

position mode, together with position offset, specifies where the read operation begins. **Absolute** starts the operation at the beginning of the file plus position offset, so the offset is relative to the beginning of the file. Relative starts the operation at the current location of the file mark plus position offset. The default is Relative.

position offset specifies how far from the location specified by position mode to start reading. You express position offset in units of samples. The default is 0.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

path out identifies the wave file passed in path.

data reads any sound data from the file. For multi-channel sound data, data is an array of waveforms where each element of the array is a single channel. t0 is the start time for the first sample read. LabVIEW approximates the initial time that the first sample was read because the sound file does not contain this data.

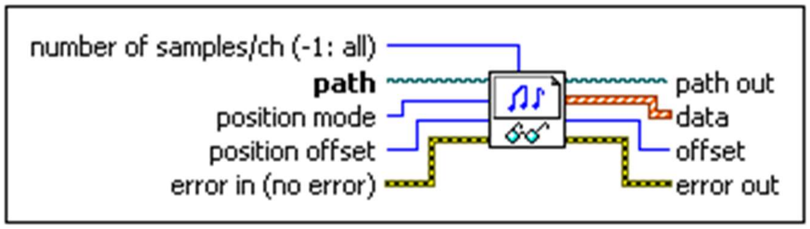
dt is the sampling interval of the wave file data.

Y is the sound data. If the array data type is a floating-point numeric, Y ranges from -1.0 to 1.0.

The specified data type determines the range of values for the sound data.

offset indicates the new location of the file mark relative to the beginning of the file, in units of samples. The default is 0.

error out contains error information. This output provides standard error out functionality.



2.4 Sound File Write Simple

Path specifies the absolute path to the wave file. If the path is empty or invalid, the VI returns an error. The default is <Not A Path>.

Data writes any sound data to the internal buffers. For multi-channel sound data, data is an array of waveforms where each element of the array is a single channel. t0 is ignored.

dt is the sampling interval of the wave file data.

Y is the sound data. If the array data type is a floating-point numeric, Y can range from -1.0 to 1.0.

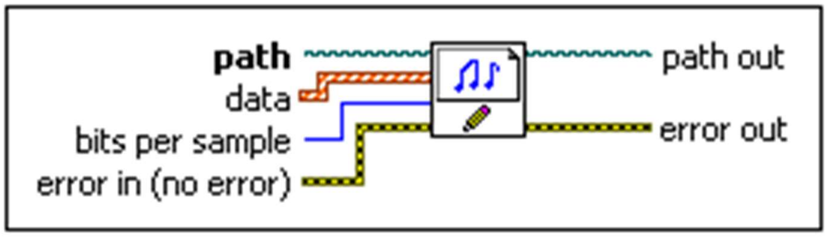
The specified data type determines the range of values for the sound data.

bits per sample specifies the quality of each sample in bits. Common resolutions are 16 bits and 8 bits. The default is 16 bits.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

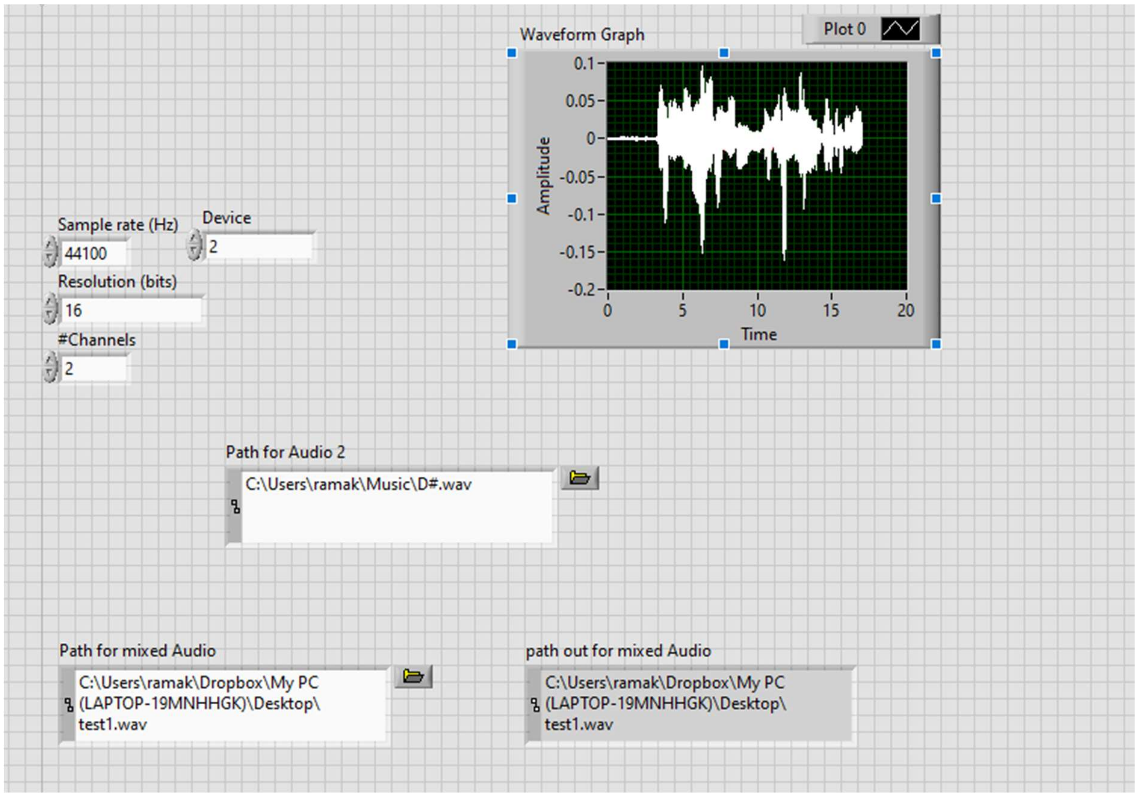
path out identifies the wave file passed in path.

error out contains error information. This output provides standard error out functionality.



CHAPTER 3: RESULTS

3.1 Front Panel



Sample rate (Hz) 44100

Device 2

Resolution (bits) 16

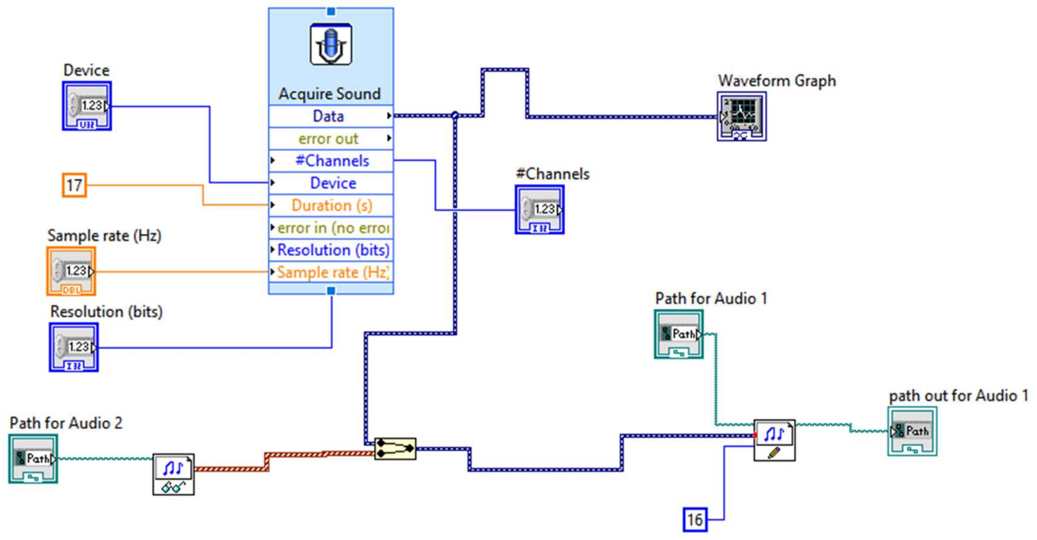
#Channels 2

Path for Audio 2
C:\Users\ramak\Music\D#.wav

Path for mixed Audio
C:\Users\ramak\Dropbox\My PC (LAPTOP-19MNHHGK)\Desktop\test1.wav

path out for mixed Audio
C:\Users\ramak\Dropbox\My PC (LAPTOP-19MNHHGK)\Desktop\test1.wav

3.2 Block Diagram



CHAPTER 4: CONCLUSION

4.1 CONCLUSION:

The recorded audio from the violin is perfectly mixed with the audio stored in the system. The audio file we use as the secondary audio is D#.wav. This audio file is perfectly mixed with the recorded audio sample from the Electric Violin and is stored in the path provided as test1.wav on Desktop. The basic use of a DAW is to mix several audio files into a single audio file. Professional DAWs have volume controls for each track so that the mixed signal is in the right mix. DAWs also have Equalizers which are used for adjusting the LOWS (lower frequencies), MIDS (middle frequencies), HIGHS (higher frequencies) of the audio samples. The equalizer is a very important application of Filtering from Signal Processing. In the equalising process, the frequencies are adjusted according to the artist by adjusting Bass (LOWS) and Treble (HIGHS).

Volume adjustment for tracks, Equalizer, Start Stop Record options are the main additional features found in professional DAWs. High end DAWs even provide plugins (effects) to enhance the beauty of the audio. Those plugins are usually based on convolution and Impulse responses. A good DAW is one which is very-artist friendly and with a good user-interface.