Audio Mixer

Kabir Guron and Ved Patel
TA: Stephen Yang

1.0 Project Summary

The Audio Mixer is a program capable of using input audio from a computer's line-out headphone jack, providing visual feedback of the audio samples (volume level and frequency spectrum). Several effects (recording, forward, reverse, slow-motion, and fast-motion playback, echo, low-pass filter, additional sound effects) can be applied, which are controlled by a PS/2 keyboard, before outputting from the speakers. For keybinds, refer to Figure 1. Note that the filter and echo strengths, as well as the echo delay can be controlled using the switches on the DE1-SoC board. Additionally, a microphone can be connected directly into the mic-in port rather than using the line-out headphone jack.

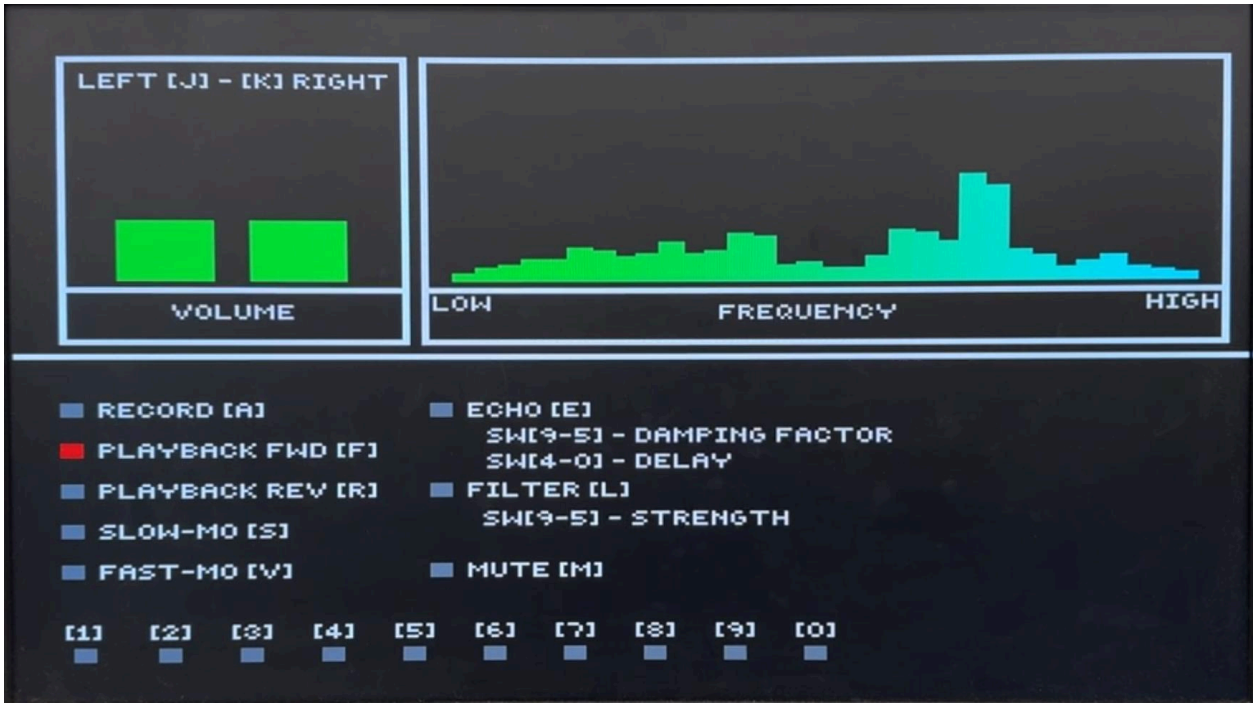

Figure 1. User interface for sound mixer with keybinds.

2.0 Detailed Description

In order for line-out audio to be converted into adequate Microphone-In, the voltage should be downconverted to avoid distortion for the speakers. The circuit provided in Figure 2 shows the circuit used to realize this problem. As an added benefit, the circuit also reduces unwanted noise to the microphone input.

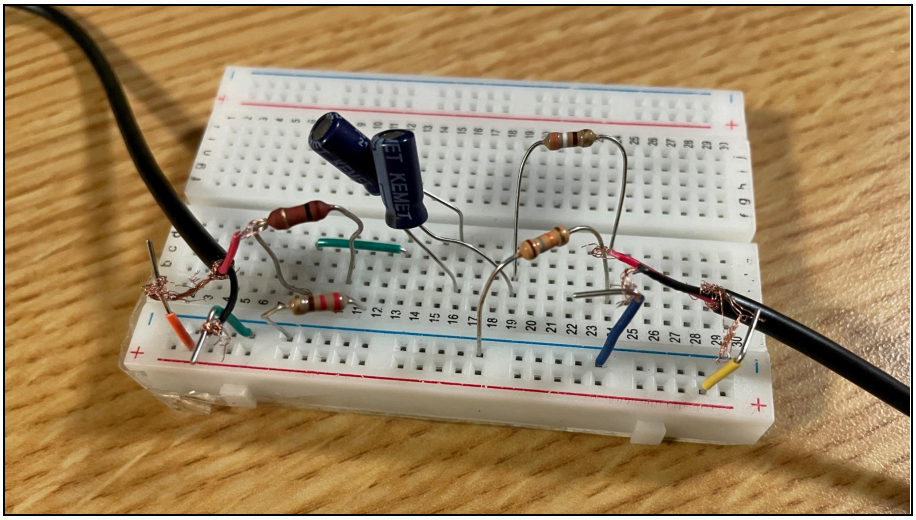

Figure 2. Breadboard circuit to down convert incoming voltage. Inspired by the Drone Workshop: https://youtu.be/aovtGu_pG4w?feature=shared&t=391.

Once samples are received by the board, they are not immediately outputted. Instead, an audio interrupt must be called before audio plays. Interrupts are only called once either 75% of the input FIFO's are full or whenever the output FIFO's are 25% empty. Then, the audio ISR is called.

Within the audio ISR, the incoming data is always read into variables left and right. Depending on the mode selected (via keyboard inputs), the audio ISR may do a variety of different actions.

For example, consider a first order FIR low pass filter given by the equation below:

$$y[n] = \alpha x[n] + (1 - \alpha) y[n - 1]$$

Where α can be chosen to proportionally adjust the cutoff frequency. Intuitively, the equation ensures that previous valued outputs are weighted to penalize sudden abrupt changes. If the global mode is set to Low Pass Filter mode, the variables left and right are computed and stored into the global variable that holds y[n−1].

Another important function is playing pre recorded sounds. Pre-recorded sounds can include live recorded audio over a four second interval or particular sound (such as an air horn) sampled prior to compilation. In either case, a large array (~32000 for a four second sample) stores the sample we want to play. Both an index and the total array size should be stored in order to keep track of the next sample to be played. To play two sounds at once, simply sum the value of the input and recorded sample.

To display a frequency spectrum, KissFFT, an external library, was used to calculate discrete-time Fourier transforms (DTFT). The input to the DTFT function is normalized versions of the most recent 64 audio samples (shifted right 24 bits), and the output is a real and imaginary component which represents the value of each frequency bin. Due to the 8kHz sampling rate on the DE1-SoC board, each frequency bin

captures 8000 / 64 = 125Hz. The heights of the frequency spectrum bars correspond to the magnitude values of each frequency bin.

KissFFT was chosen because it computes the discrete Fourier transforms using the Fast Fourier Transform (FFT) algorithm, which has a time complexity of O(NlogN), rather than manually computing DTFT using general formulas, which would be O(N^2).

3.0 Block Diagram

See Figure 3.

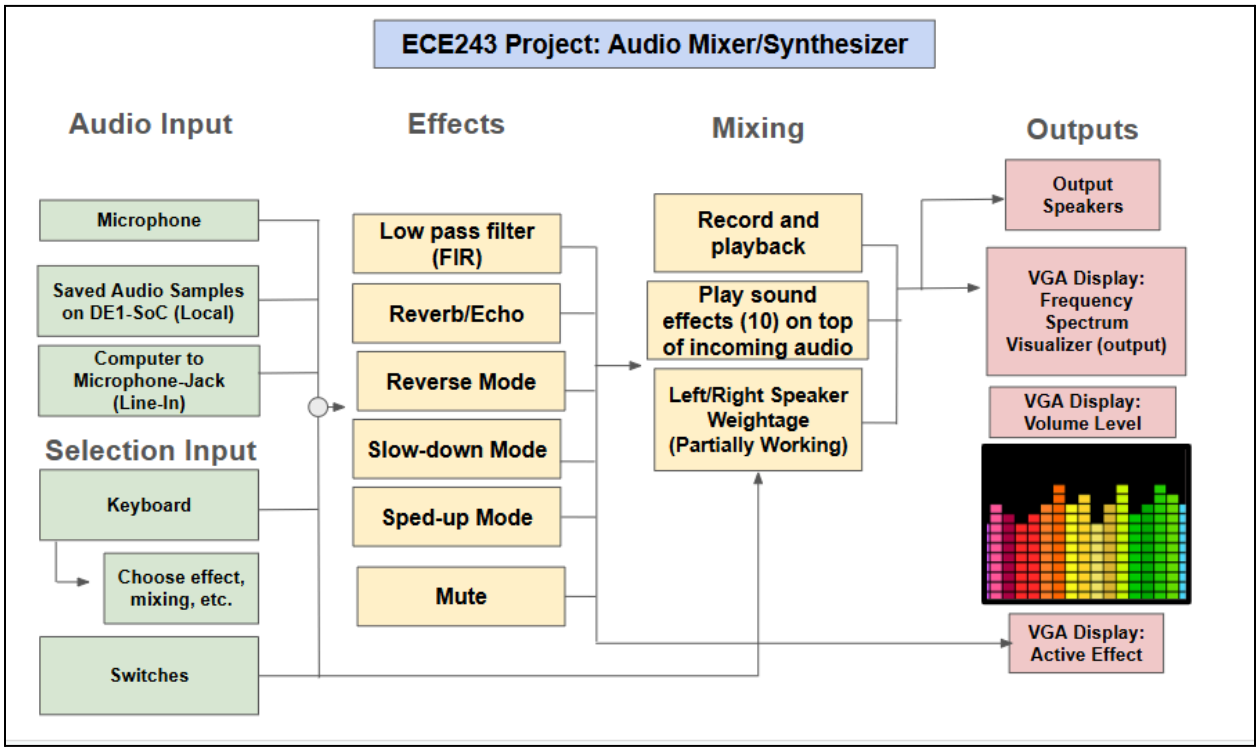

Figure 3. Updated block diagram containing all working features.

4.0 Attribution Table

Table 1 describes the general contributions made by both team members Ved and Kabir respectively. Note, the collaborative column describes features that were implemented simultaneously by both partners (i.e. both partners working at one computer implementing code beside each other). This was necessary for seamless integration. Both partners are satisfied with work distribution.

Table 1. Attribution table.

Kabir	Highly Collaborative	Ved
• FIR low pass filter • Recording, reversing,	• Audio interrupts • Echo	• Incorporating the FFT library by tuning

play-forward, slow-mo, fast-mo • Sound effect imports • Circuit to down-convert voltage and reduce noise • Keyboard interrupts	• Mute • Resolving audio bugs, adjusting general code structure	parameters, resolving compatibility conflicts, and researching • VGA display including creating background, drawing frequency spectrum, volume boxes, and mode-indicators • Feature integration from various commits

Signatures:
Kabir G.
Ved P.

Additional accreditations:

- KISS FFT was the library used to compute the discrete-time Fourier Transform
- A base interrupt template (Listing 6) was used originally and built-upon to add keyboard and audio interrupts
- Vishwas Puri's site was used to convert audio (MP3) and PNGs to C arrays
- Line-Out to Microphone-In inspiration: https://youtu.be/aovtGu_pG4w?feature=shared&t=391