Anna Hsu A18104081
Elisabeth Hsu A17338370
Aileen Phuong A18086030

ECE 45 Synthesizer Project

Github Link: https://github.com/ahsuanna/ece45synthesizer

1) Discuss the potential applications of your synthesizer. This is important because an ambulance siren, for example, might need a different range of frequencies when compared to a fire alarm system.

The basic premise of our synthesizer is to create robot speech sounds. Our inspiration mainly stems from video games or movies with characters' speech replaced with a series of sounds or beeps instead of actual words, sometimes called beep speech . Examples of this include speech of characters in Animal Crossing or in RPG games when NPCs have dialogue. Our synthesizer generates these types of sounds by applying a pseudo random sample and hold LFO (low frequency oscillations) on a sustained input sound. The LFO applied to the frequency of the sustained note then works to change the frequency in a random chaotic manner at a high frequency which in turn creates the chaotic robot-like beep noises that imitates the ups and downs in frequency of a person speaking. The applications of this synthesizer are very specific to generating these types of sound and thus its main applications are most likely for video game developers and animation producers who are looking to generate robotic-like speech for their characters in video games and animations. In general, the sounds produced from this synthesizer act to replace speech.

2) Describe in detail all of the synthesizer controls and what they mean. Label all the axes and mark all the value ranges (if any) appropriately.

How to Open the Synthesizer:

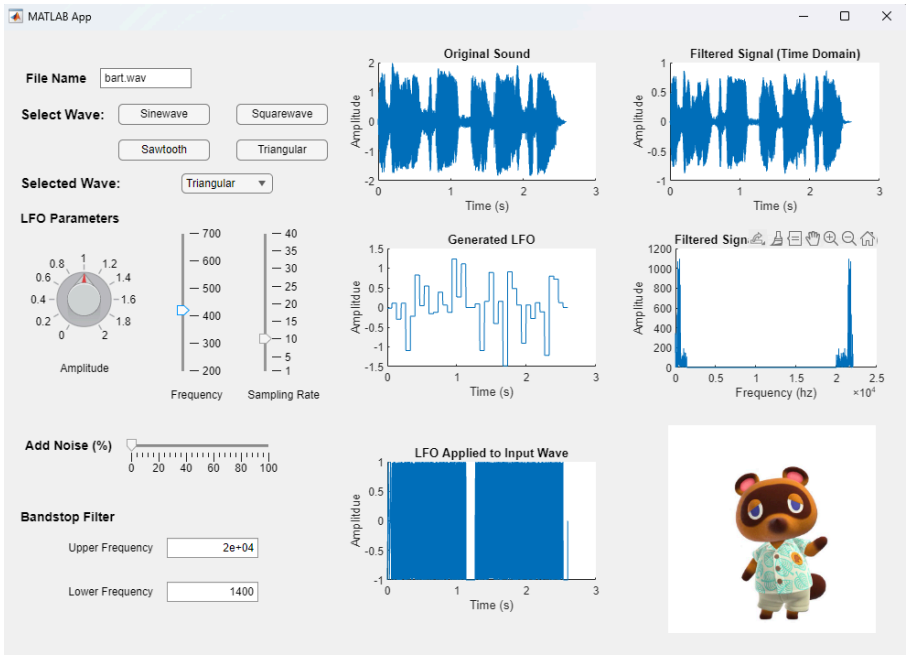
- Download all files from git repo
- Open Matlab
- Open SynthApp.mlapp
- Click Run at the top of the screen

What Each Function on the Synthesizer Does:

- File Name: insert chosen .wav file name, ex. bart.wav, into textbox, ensure that the .wav file is in the same path
- Select Wave: select the input wave that you want, creates different sound effects for output
    - Selected Wave: shows the wave selected (dropdown only shows what wave is selected and does not change wave, use buttons to change wave)
- LFO Parameters:

    - Amplitude: changes the amplitude of the square wave used to sample and generate the LFO, increasing amplitude increases contrast between low and high notes of output sound
    - Frequency: changes the frequency of the sine wave input multiplied by the LFO, increasing frequency increases the pitch of the output sound
    - Sampling rate: changes the rate at which the LFO wave is sampled at, increasing sampling rate increases the frequency of the beeps
    - Drop down menu: changes the base wave input that is multiplied by the LFO
- Add noise: adds noise to the original sound file, allowing for more random generation of the sample and hold LFO
- Bandstop Filter:
    - Upper frequency: enter upper limit of frequencies that need to be cut out
    - Lower frequency: enter the low frequency minimum of the frequencies that need to be cut out
- Original Sound: shows the original sound wave in the time domain
- Generated LFO: shows the generated LFO in the time domain
- LFO Applied to Input Wave: shows the output of the sound wave with LFO applied to input wave in the time domain
- Filtered Signal (Time Domain): Shows the filtered signal in the time domain
- Filtered Signal (Frequency Domain): Shows the filtered signal in the frequency domain

Example of Using the Synthesizer:



Example Output:
https://drive.google.com/file/d/1yNrLOXgzaRg47s_cRYht2eftOiFnCUAG/view?usp=sharing

3) In the process of designing your synthesizer, you must apply at least one of the concepts learned in class. Clearly explain how you applied the chosen concepts in your report.

In the process of designing this synthesizer we mainly used two class concepts: sampling and fourier transforms. To create the LFO, we sampled a .wav file at a given sampling rate and held that value until the next sample to create a pseudo random step wave LFO by multiplying the sample value to a square wave and sampling at the next edge of the square wave to generate the sample and hold LFO. The LFO serves to oscillate the frequency of a given input sound which we set to a sine wave from frequencies ranging from 200 to 700 Hz. When testing the code, for example, we used a sine wave of 440 hertz which produces a sustained sound of note A. Applying the LFO to this input sine wave, we oscillate the frequency to create the shifting of notes at high frequencies which corresponds to the LFO wave which results in the seemingly random and chaotic beeps that mimic robotic speech.

Due to possible discrepancies between the square wave and the input .wav file we used to generate the LFO, the resulting sound does have some static. We used Fourier transforms and inverse Fourier transforms to visualize our output sound wave in the frequency domain, cut out certain frequencies, then revert it back to the time domain to play the filtered sound. In our case, our synthesizer uses a bandstop filter as the frequency of the notes desired are generally at the low and high ends in the frequency domain while the middle frequencies are just static. While the static seems to persist throughout the frequency domain, cutting out the middle frequencies lowers the amount of static in general. Completely cutting out all the static would probably take more processes than just a Fourier transform and could be a possible next step for our synthesizer.

4) Cite all of the codes and functions (or predefined modules) that you borrow from other programmers across the web.

- https://learningsynths.ableton.com/en/recipes/old-fashioned-computer
- https://www.mathworks.com/help/signal/ug/simulate-a-sample-and-hold-system.html
- https://www.gaussianwaves.com/2014/07/sampling-a-signal-in-matlab/
- https://www.mathworks.com/help/matlab/ref/fft.html
- https://www.mathworks.com/help/matlab/ref/ifft.html
- ECE 45 HW 2
- ECE 45 Final Project Example: LFO codes

5) Contributions

Anna Hsu

- Wrote code for sampling from .wav file and applying square wave to generate LFO
- Wrote code for applying LFO to sine wave to generate output sound
- Assisted with writing code for bandstop filter

Elisabeth Hsu

- Compiled and implemented code to create GUI
    - Implemented graphs of Original Sound, Generated LFO, LFO applied to Input Wave, and Filtered Signal to be viewable on app
    - Implemented file selection, changing LFO parameters, adding noise, and bandstop filter on GUI

Aileen Phuong

- Wrote code for bandstop filter to filter out noise
- Wrote code for adding noise to original signal
- Assisted with implementation of GUI
    - Added option to select wave