ECE Capstone Project
Sound-Sifter

Samuel Bhushan and Hailey Parkin
bhushan.samuel@gmail.com
hrparkin@gmail.com

Friday, December 18, 2015

Table of Contents

Executive Summary

This document details the Sound-Sifter project. The sound-sifter is a self-calibrating audio equalizer. First, calibration is done by calculating the frequency response using a swept sine signal. Using the frequency response, the Sound-Sifter identifies areas in the response that need adjustment. Once calibration is completed, filters are created that address the major issues. The project is implemented on the BeagleBone, a single-board computer. All calibration and processing is done on the BeagleBone chip.

1. Introduction

We are two ECE students who love listening to music and want to demonstrate our skills in engineering through signal processing. We see a need to increase the availability of high quality audio and believe we can contribute to this area with our engineering abilities. For this reason, we have chosen a senior project on audio equalization to correct problems with loudspeakers and room acoustics. This project has the potential to affect anyone who enjoys listening to music.

This document describes how our device, the “Sound-Sifter”, resolves equalization, one of the major problems associated with audio quality. The purpose of the device is to improve speaker performance in a given room through digital signal processing so that physical room corrections are not needed. The device is inexpensive enough to appeal to the general consumer audience.

Equalization refers to making the level of every frequency in the sound equal. Our device accomplishes by sampling the room and applying a customized real-time filter to the audio input before sending it to the loudspeakers. A swept sine , also known as a “chirp” was used to perform the room analysis because of the excellent signal to noise ratio it provides. Problem areas in the frequency response are identified by breaking the response into smaller bands, finding the average of each band, and then raising or lowering the bands that deviate the most. This method was chosen because it proved to be the simplest effective way to identify the problem areas. Finally, the filtering is done in real time using Parametric Equalization filters. We have also automated the entire process, making it possible for someone with no audio knowledge to use the device. This solves the problem of making audio equalization available to anyone at a low cost.

2. Methods

In the first phase of our methods, we measure the frequency response of speakers a room. In order to adequately sample the room, we chose to use a logarithmic sinusoidal sweep starting at 20 Hertz and rising to 20 Kilohertz. The primary reason for this is because of the high signal-to-noise ratio it provides. Since the room sampling will occur in ordinary rooms and not acoustic testing facilities, it is important to have the energy level measurably above the noise floor to pick up an accurate response.

A sine sweep contains more energy than an impulse type test sound, which makes it higher than the noise floor and therefore a better choice. As an additional advantage, it seems to startle people less who might be in hearing range during testing. There are two types of sine sweeps, linear and logarithmic. We found logarithmic superior because human hearing is logarithmic (in other words, we can better distinguish frequencies at the base end of the spectrum than we can at the treble end.)

Generating a room response from a logarithmic sweep requires some extra data manipulation. The measurements need to be linear in order to compute the Fourier Transform of the signal. The recorded data is modulated to compensate for logarithmic sweep according to the equation below.

$$n(t) = \sqrt{\frac{\omega(t)}{\omega_1}} = \sqrt{\frac{\omega(t)}{2\pi \cdot f_1}}$$

This equation makes the data linear and ready to be transformed into a frequency response.

The Fast Fourier Transform (FFT) used by our program is fftw. This code allows us to take an n point fft with n as high as the number of samples. With our twenty second sweep, we end up with 1920000 samples. A million point fft is very high resolution and creates a precise response.

The next phase of the project was correcting the measured frequency response. The problem of identifying the areas with the worst response was a difficult one to solve. There were two approaches we considered. The first one was identifying individual bumps in the frequency response and creating an exactly matching correction filter. The second was to divide the response into equal bands and take the average of each band and then apply a filter to the whole band. We went with the latter for many reasons. Probably the most important reason was the simplicity of the implementation to the band approach versus the incredible complexity of to finding individual bumps. The complex approach involves find and sorting local maximums and minimums, using derivatives to identify where one starts and another begins, and also must take into account not just the height of the bump but the area under the curve as well. There are also many special cases that could produce strange results. Averaging may pinpoint the problems less precisely, however the overall effect of the corrections is right.

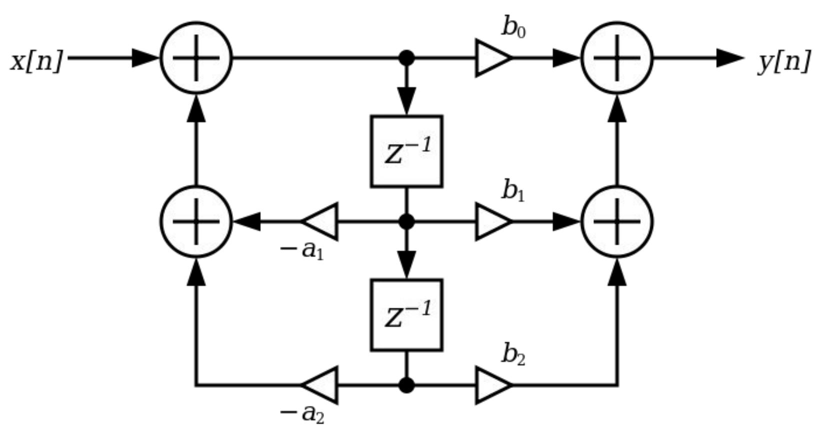
The averaging corrections are computed as follows. The the response is divided into twenty bands logarithmically. The frequency ranges are computed according to

$$f_{next} = f*10^{3/\#bands}$$

The frequencies within each band are averaged to get the average for the band. The overall average is also calculated. Each band is compared to the average and filters are applied to the bands that deviate most from the average.

The filters chose to use are parametric equalization filters. These are multi-band filters that allow control of amplitude at a given center frequency and over a given bandwidth. This is accomplished through equations for the filter coefficients that make the filters flexible. The filters are implemented as a Direct Form II biquad.

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}$$



$b_0 = (1+a+K-Ka)\frac{1}{2}$
$b_1 = (b+ba)$
$b_2 = (1+a-K+Ka)\frac{1}{2}$
$a_1 = b1$
$a_2 = a$

$b = -\cos(2\pi F_0/Fs)$
$a = \dfrac{1-\tan(\pi B/Fs)}{1+\tan(\pi B/Fs)}$ $where\ B = F_0/Q$
$K = 10^{G/20}$ $where\ G\ is\ gain\ at\ F_0\ in\ dB$

The Sound-Sifter currently implements six of these filters, which means the six worst bands can be corrected. There could be room to expand the number of filters, however we stopped at six due to concerns about the BeagleBone's capacity to compute more computations in real time. It was discovered that the problems were not due to the filtering computations, they were due to the settings of the audio interface being incorrect. At that point the was no more time to test increasing the number of filters.

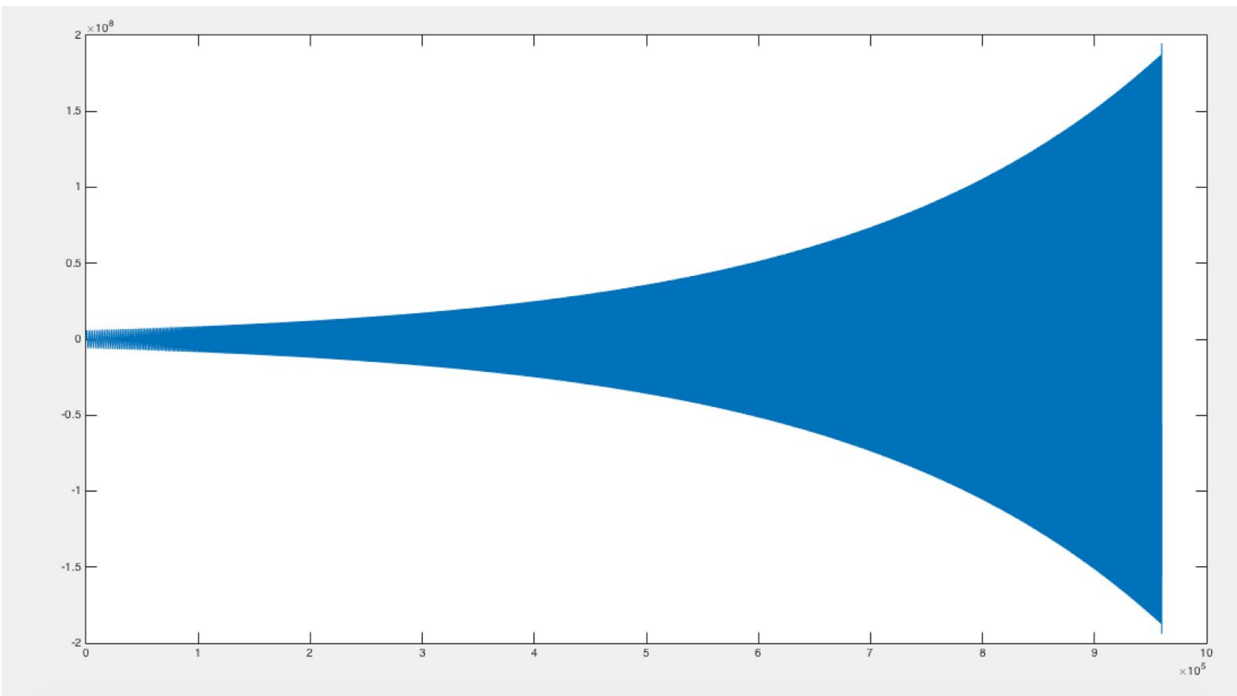
In order to implement the filters in real-time Port Audio, an audio input/output library, was used. This software allows transparency between audio programs and the platform they are running on. Port audio chooses an input and an output, and provides a callback function. The callback function provides a place to process audio before it is sent out to speakers.

The entire process was automated using a bash script within Linux on the BeagleBone. This ran each of our programs from the three stages at the appropriate time. In addition, it had a small routine that ensured that the speaker volume was high enough for the sine sweep to be recorded clearly.

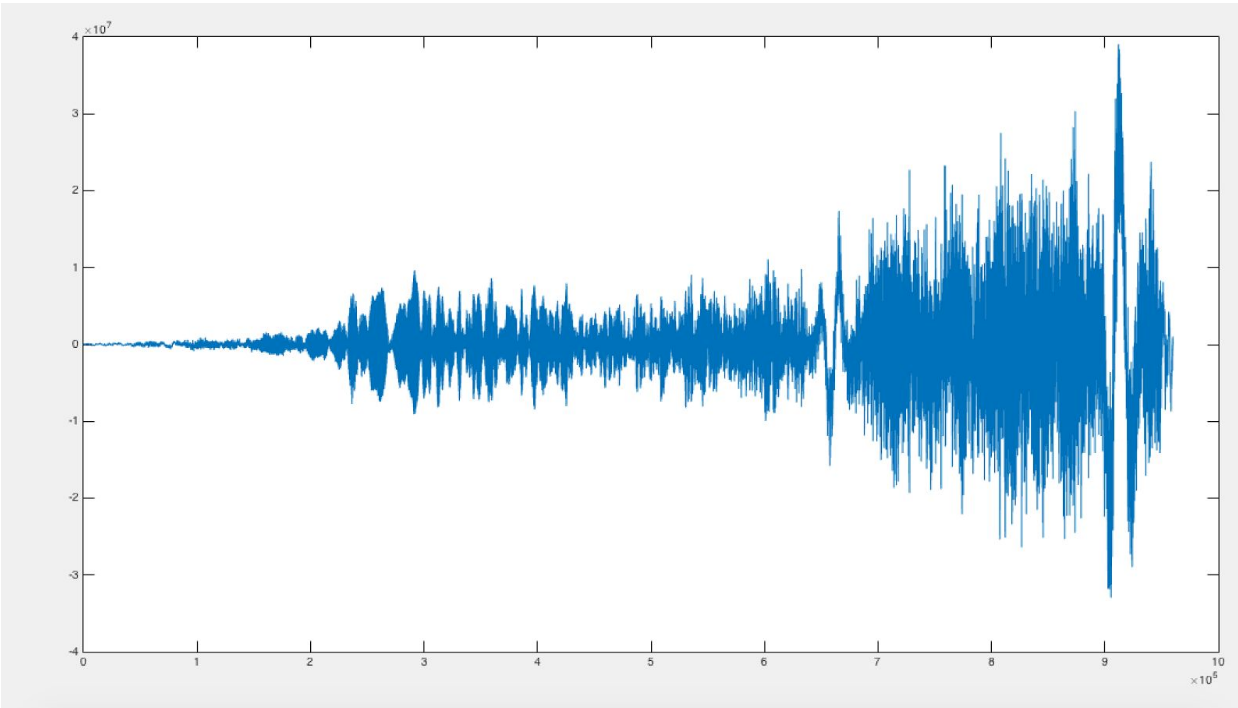
3. Results

We were able to successfully record a sine sweep from speakers, and the FFT showed the frequency response. In order to test for correctness, we manually adjusted the frequency response of the speakers by changing the 'bass' and 'treble' settings. We then looked for a corresponding change in the frequency banks of the filter. This test showed that the microphone was correctly recording the room.

Room Sweep Response Example:
The two pictures shown below compare the measured sine sweep in the room with the ideal sine sweep signal. These are modulated signals, causing the amplitude to increase throughout the sweep.

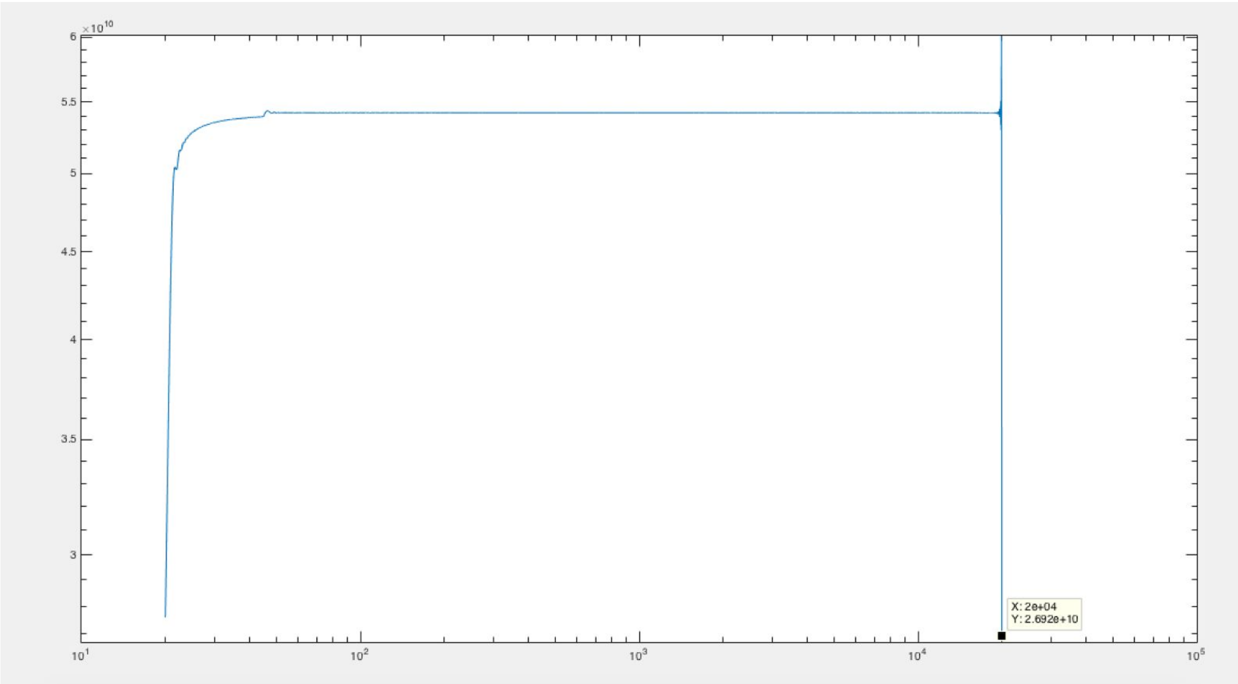


Ideal Sine Sweep Signal from 20 Hertz to 20 Kilohertz in time. (Magnitude / Sample Number)

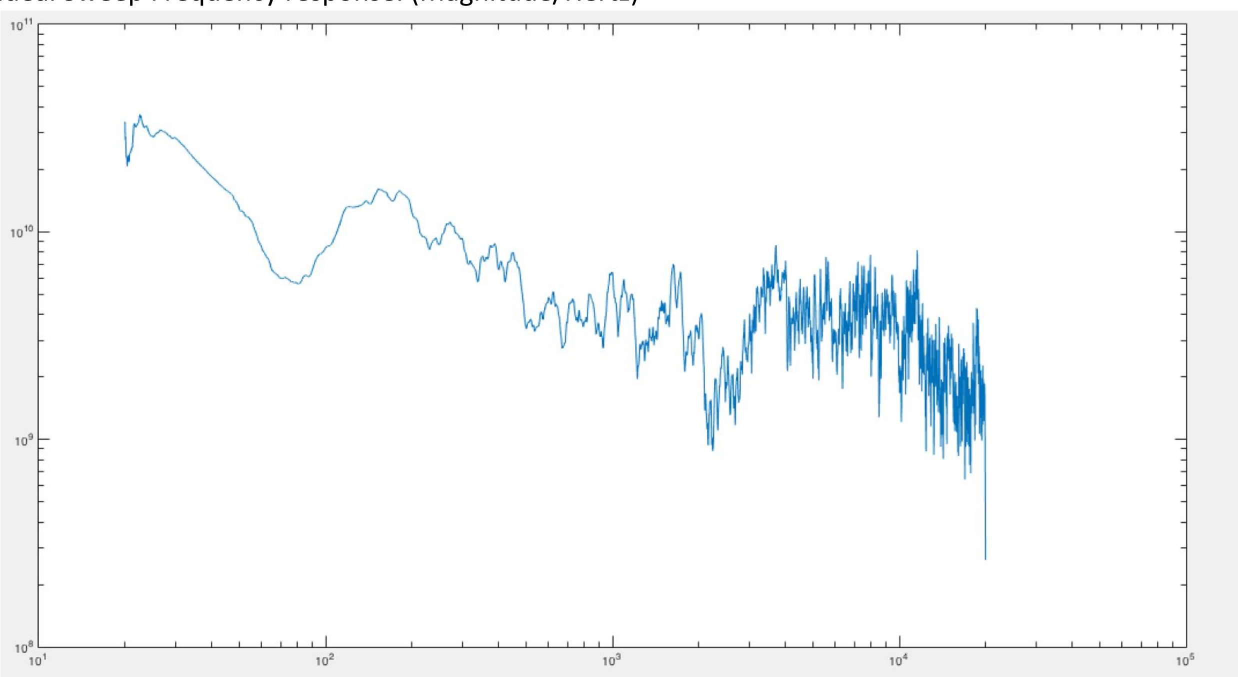


Measured Sine Sweep Signal in time. (Magnitude / Sample Number)

The next two graphs show the frequency response calculated through via the FFT.



Ideal Sweep Frequency response. (Magnitude/Hertz)



Measured Room Frequency response. (Magnitude/Hertz)

The parametric filtration was successfully implemented within the callback function. For each filter, a new instance of a filter class was run. We were unable to get stereo successfully passed through the filter due to an audio driver issue. In order to show that the filters were created correctly, we manually adjusted the audio. We then ran calibration and recorded the filters that were created. The first line is the number of filters and the subsequent lines have the format: <center frequency(Hz)> <width(Hz)> <gain(dB)>

```
6
66.9931 23.2538 -4
94.6303 32.8468 -5
188.812 65.538  -3
266.704 92.5749 -2
751.675 260.912 -2
1499.79 520.587 -3
```

Filters Created with no speaker adjustments.

```
6
33.5761 -5      376.73
47.4275 -3      532.145
66.9931 -5      751.675
94.6303 -4      1499.79
133.669 46.3974 -8
188.812 65.538  -9
```

Filters Created with Bass Boosted and Treble Reduced

```
33.5761 11.6545 3
47.4275 16.4624 3
133.669 46.3974 6
188.812 65.538  3
376.73  130.766 3
532.145 184.711 3
```

Filters Created with Treble Boosted and Bass Reduced

```
6
66.9931 23.2538 -3
94.6303 32.8468 -3
266.704 92.5749 -3
751.675 260.912 -4
1499.79 520.587 -4
16827.9 5841.08 5
```

Filters Created with Treble Reduced and Bass Reduced

Initially, the real time functionality was inconsistent, sometimes audio processing stopped and no sound could be heard. But, once we removed processing in the left channel, which was not functional, the audio processing succeeded. In order to get sound out of each speaker we duplicated the right channel to both channels.

4. Discussion

In this section, we will discuss the results of the project and potential developments.

In the calibration stage we were successfully able to measure the frequency response of the room. Initially, we were measuring the logarithmic signal and obtaining the frequency response with a linear FFT. This caused the FFT to incorrectly indicate high levels in the lower frequencies. After researching this issue, we found a method of addressing this problem via modulation. After this treatment, the response was correctly flattened as shown in the results section.

When looking at the frequency response of most measured rooms, we did find an abnormal boost in the lower frequencies even after modulation. This may have been caused by standing waves in the room or other phenomena. Because of this, filters were usually applied to reduce the lower frequencies.

In the higher frequencies, the filters appeared to be created more correctly. In the example where the bass was reduced and the treble was increased, filters were added that raised the bass and mid range up to match the levels in the treble range. In the second example, bass was reduced and treble was reduced. Here filters were created, dropping the mid range.

These preliminary results seem to point that the project works to some extent, but further optimizations are necessary to polish the equalization. Applying existing methods for equalization could further improve the project. Overall, we achieved enough to begin fixing some of the deeper problems.

5. Conclusion

This project has been exciting and a major learning experience. We were able to apply significant portions of our education towards this project. Many difficult challenges were overcome in all phases of development. We now have a strong understanding of digital audio processing. The prototype we developed for this project has demonstrated a working proof of concept, and could be expanded into a commercial venture.