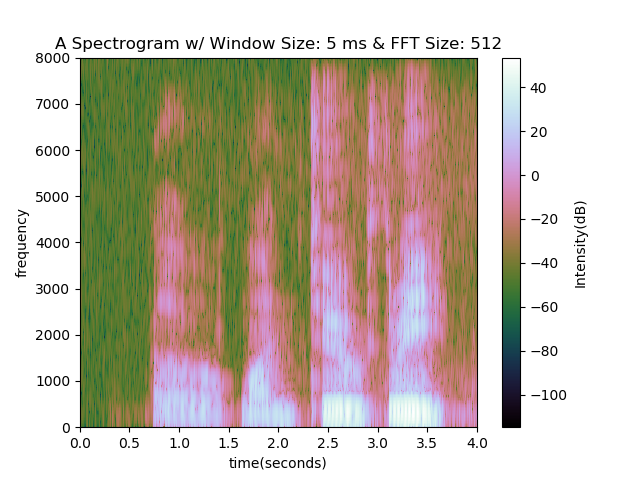
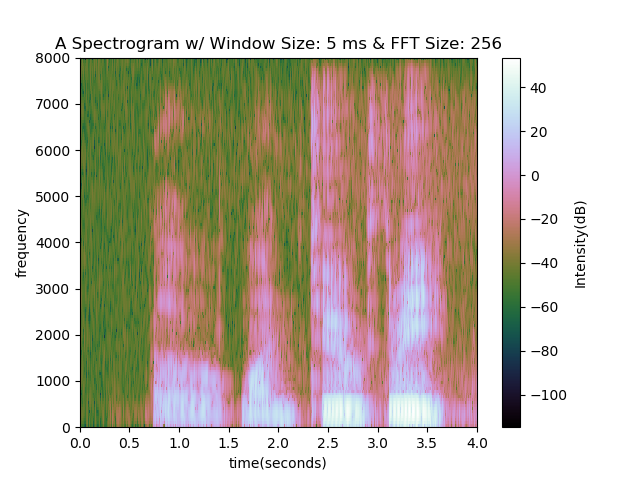
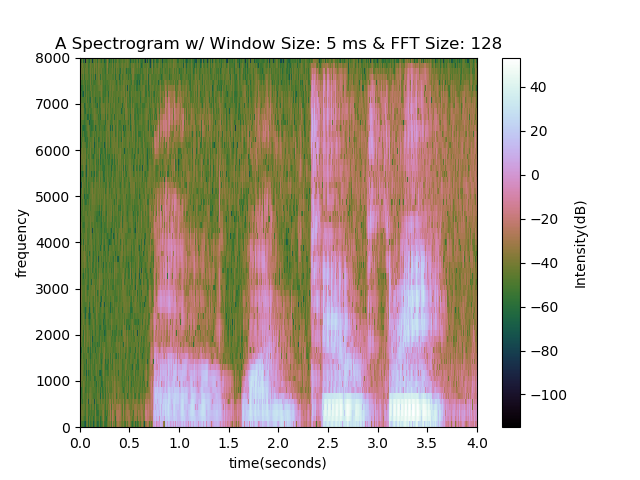
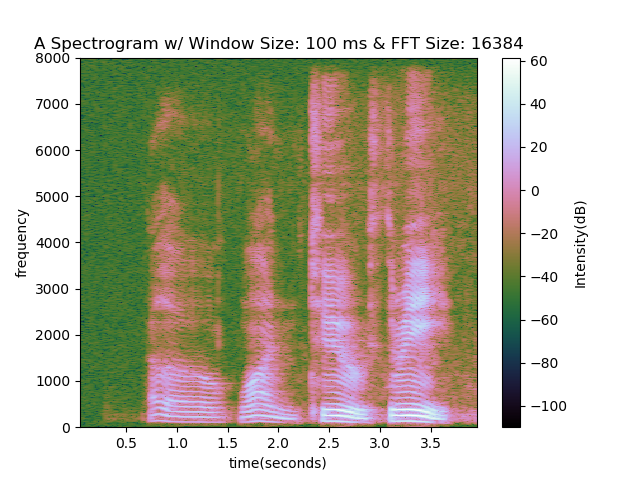
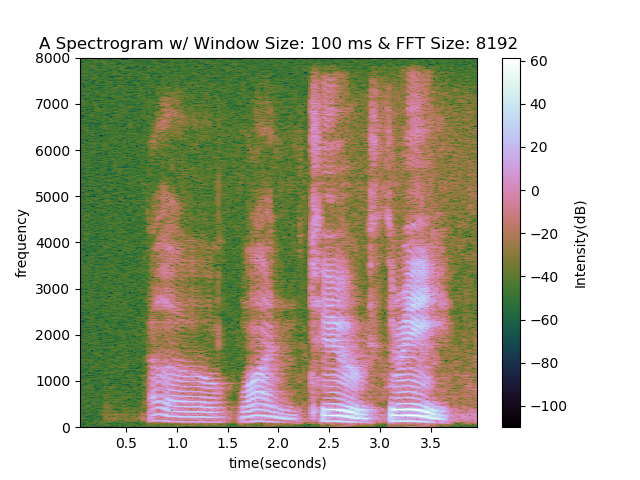
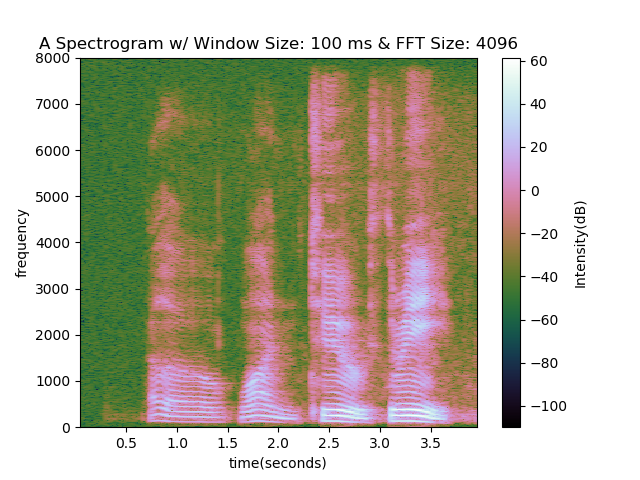
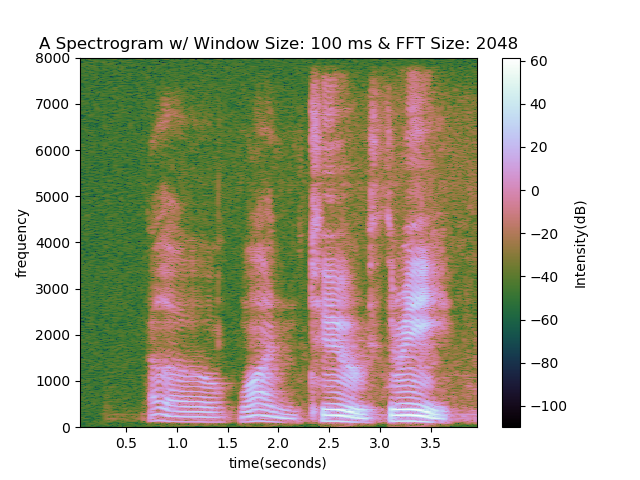
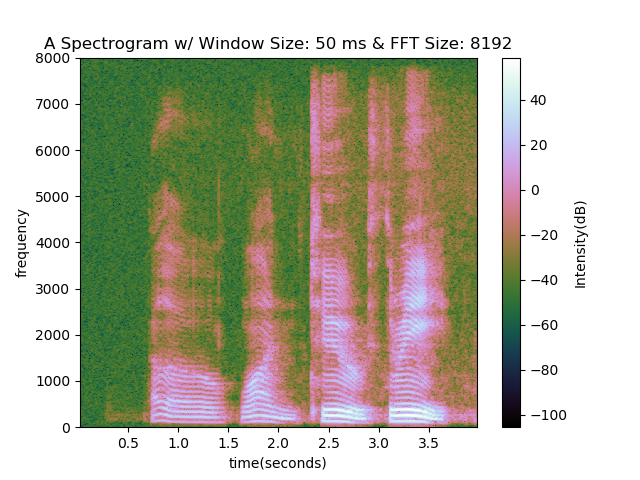
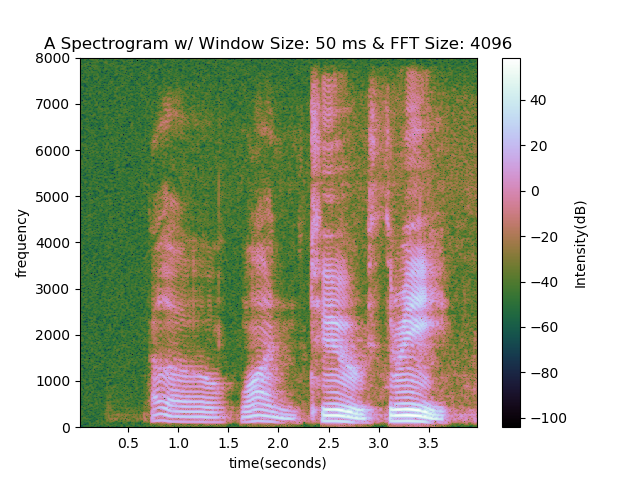
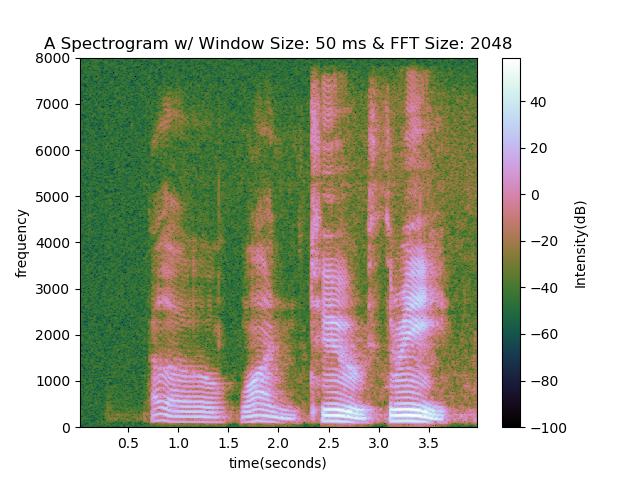
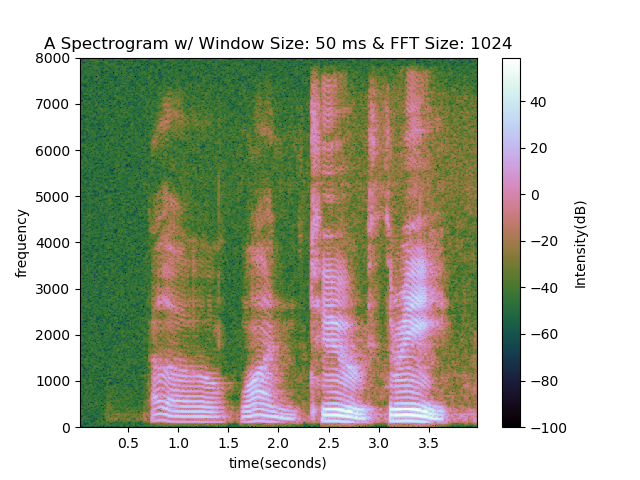
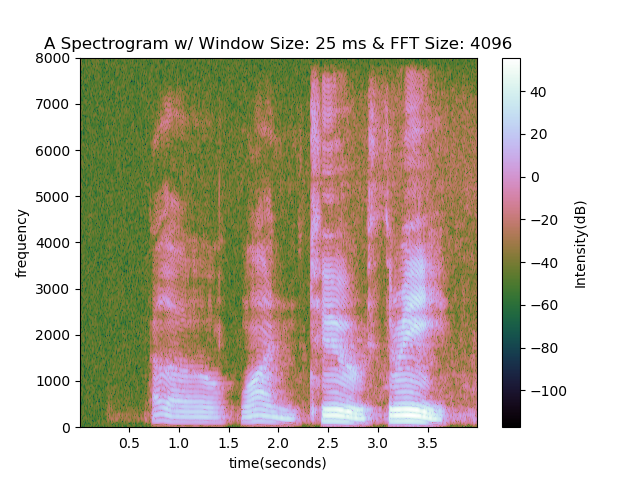
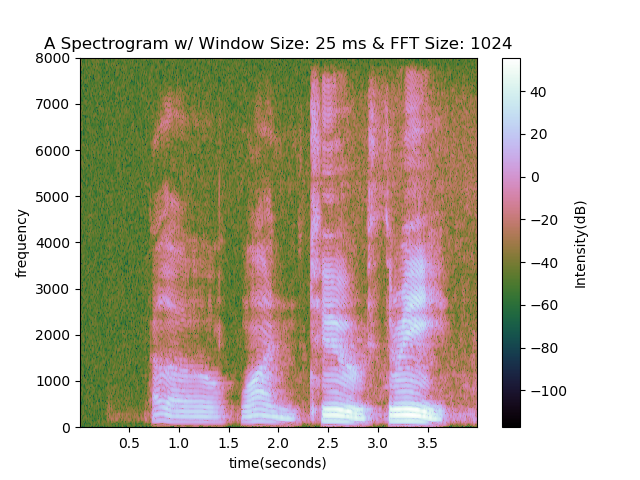
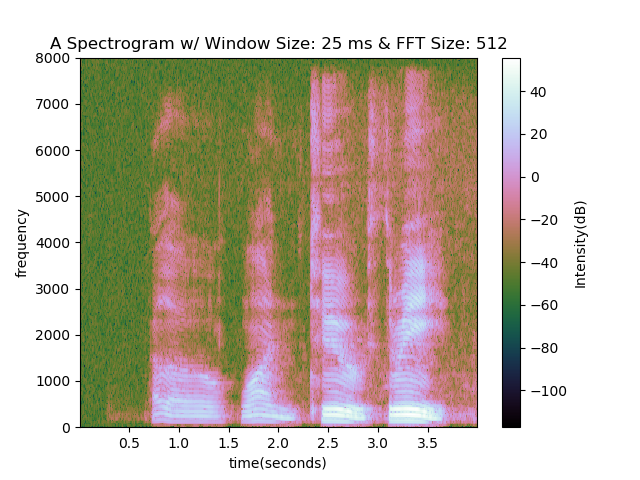
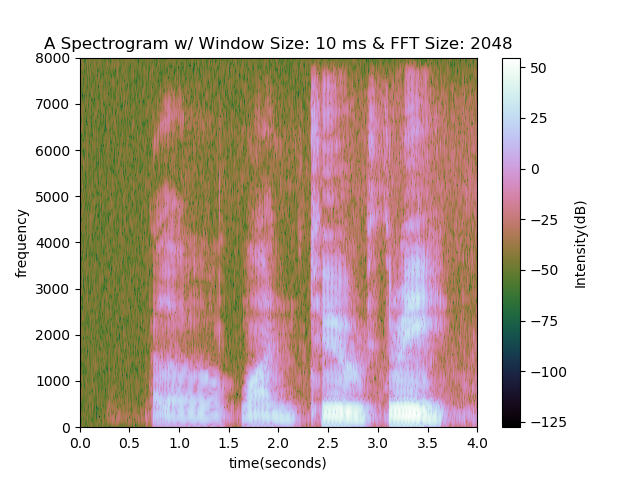
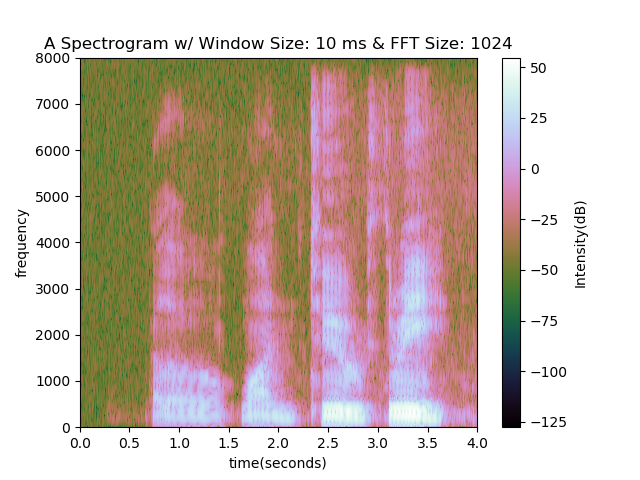
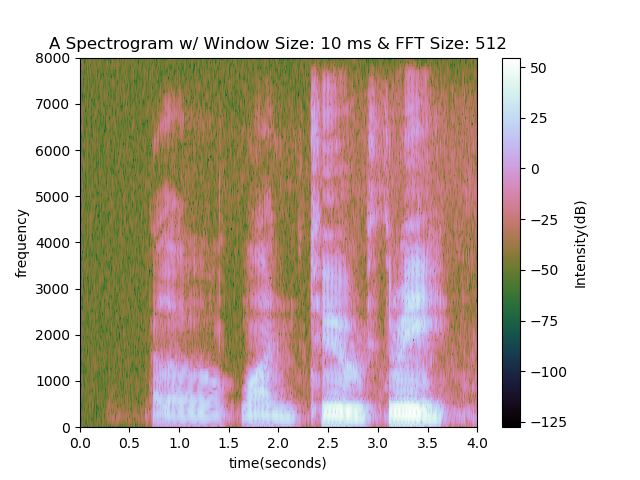
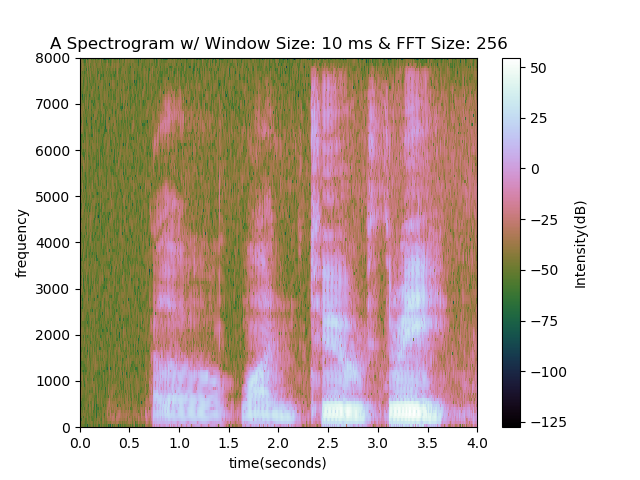
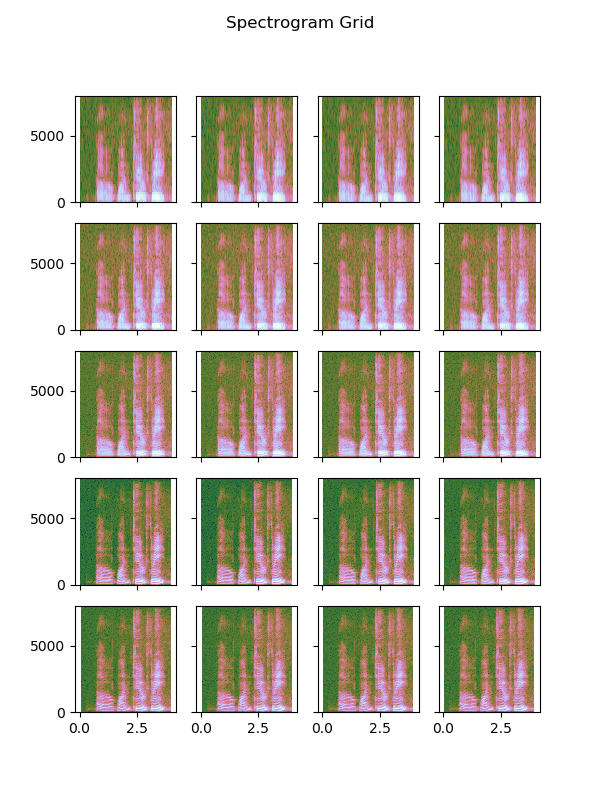
Problem 1

1. Each of the following 20 spectrograms represents an audio clip for a recording of me saying “testing one two three”. The spectrograms can be separated into groups each with a different window size increasing from 5ms, to 10ms, 25ms, 50ms, and 100ms. Within each group, four different FFT sizes were used by padding the windowed data to match a data size matching an increasing power of two.





1. The following is a grid of all the spectrograms in a single figure. Each row represents a different window size with the top row using a 5ms window and the bottom row using a 100ms window. Each column represents a different FFT size with the left most column using the smallest power of two that is also bigger than the windowed sample size.



1. When using small window sizes, the frequency resolution seems to be lower. Many of the harmonic features that can be seen with the larger window sizes can’t be seen with the smaller window size. The features seem to blur together.
2. The differences between the graphs that used the same window size but different FFT sizes seemed to be very small. There were small areas occasionally that seemed to have some slight differences in how the intensity varies, but there were no major differences that would prevent me from visually differentiating one spectrogram from another as long as the window sizes were the same.

Problem 2

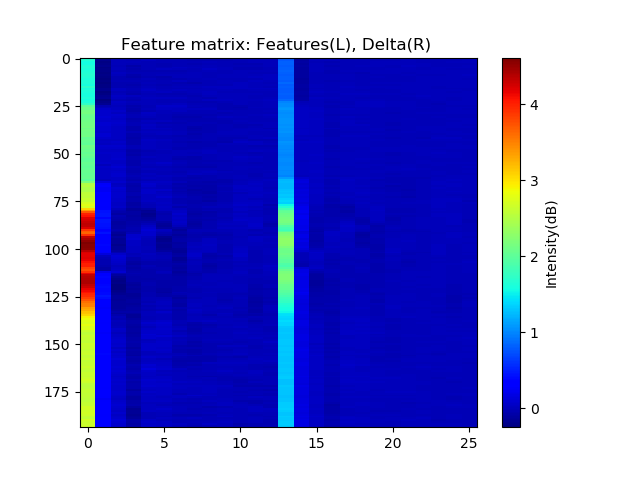
1. A
2. B
3. C

Problem 3

1. For the speech windowing function, I used a hamming window function:

Where N = the width of the window which in this case is 25ms or 400 sample points. With the windowed audio sample, I then ran it through the MFCC warping and filtering algorithms.

1. For each phrase, I stored 20 separate recordings as a .wav audio file.
2. The following is an image representation of the feature matrix created out of a recording of myself saying “Odessa”. The left half of the matrix represents the original features while the right half represents the delta features. Time is shown vertically while different columns represent a different features or delta feature.



1. The full matrix is of size 26 x (T – 2M) because of the parameters that were chosen. When creating the MFCC, the first 13 coefficients were chosen to be kept for further analysis. This means when the delta features are computed, there are 13 sets of features to analyze, one for each coefficient. Also, the length of the matrix is determined by the size of the recorded audio data and the window size used when calculating the delta features. With a window size of 2, there are two data points at the beginning and end of the data that the algorithm can’t compute since the algorithm would require reaching out of bounds in those areas. That leaves the overall length of the feature matrix to be T – 2M.

Problem 4

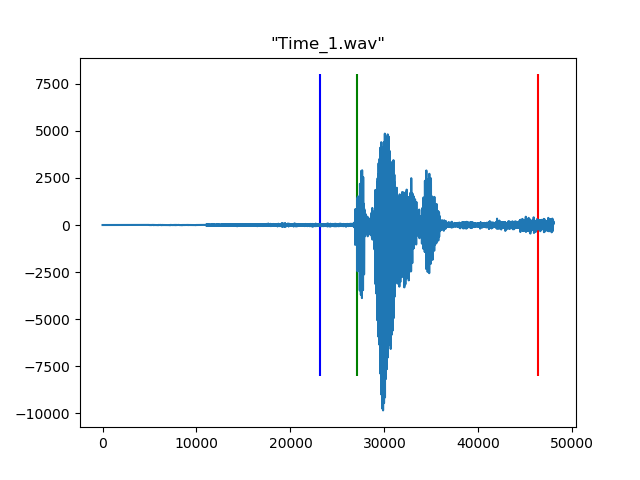
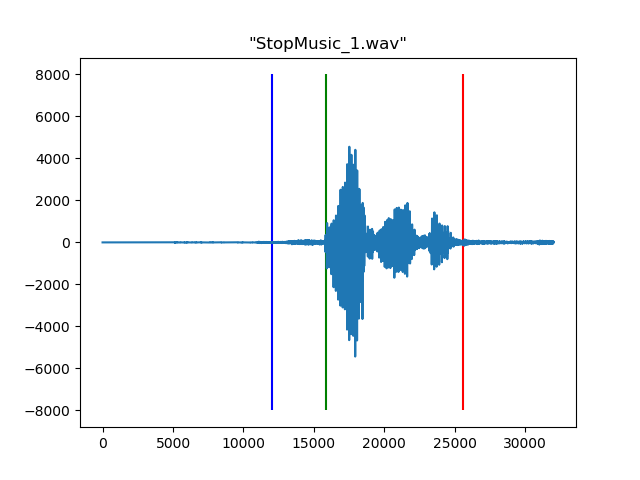
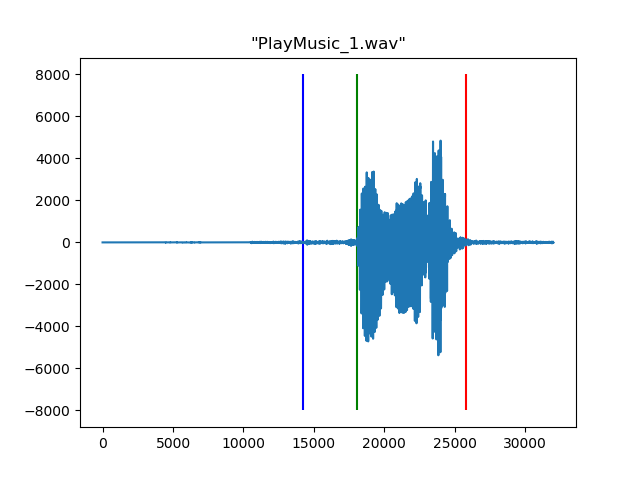
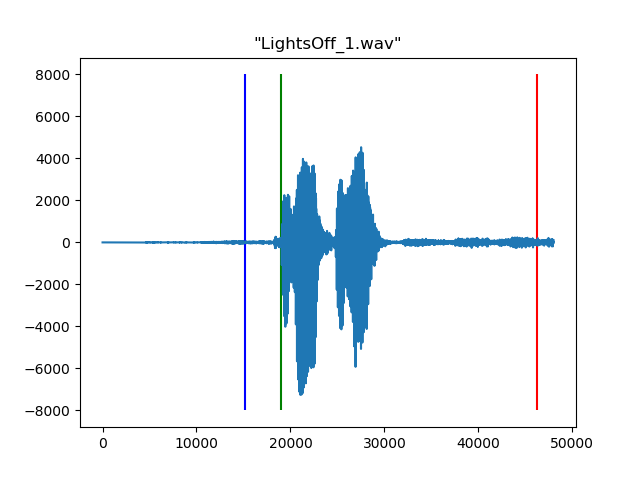
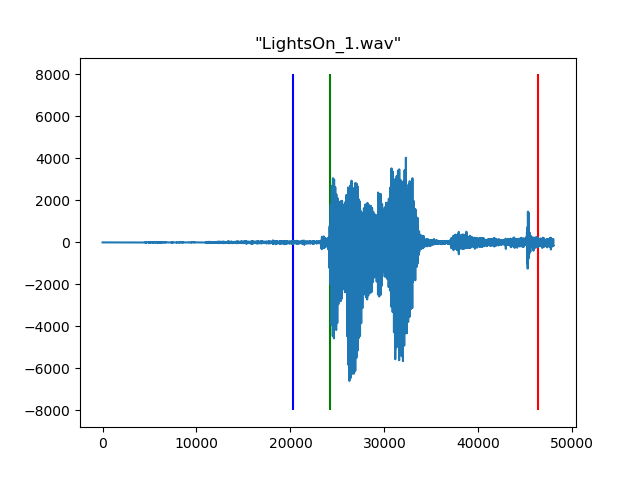
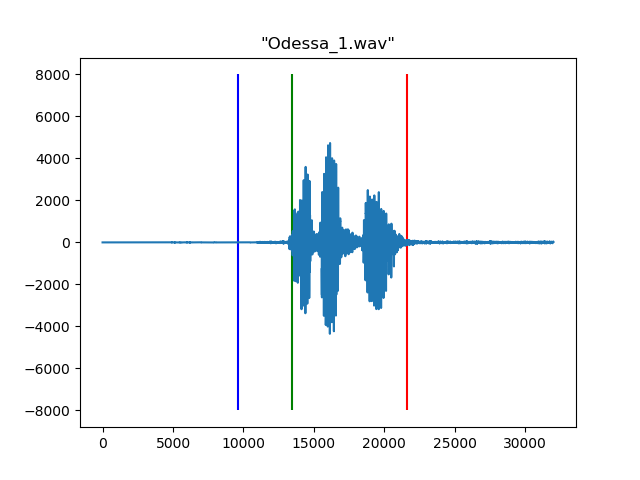
1. A
2. When performing silence detection, I initially started by analyzing a pre-recorded audio file. This allows me to get a consistent measure of how the algorithm is performing. I started by creating the Rabiner/Sambur algorithm. While it seems like that algorithm isn’t as robust as the later described methods, it seems like it would be a good starting point since it would at least reduce the size of the data that needs to be analyzed without too high of a risk to miss important data.

As described in the homework description, the steps include sectioning the audio file into 10ms segments. Within each segment, the values of the audio file are summed together to represent the in that area of the file. The audio file can then be represented by an array shorter than the audio file depending on the width of the segments. In this case, a 10ms window means the array length is:

Where fs is the sampling frequency which is 16kHz in this example.

The first thing I tried to do was copy and paste the functions and algorithms into the code that was slightly modified to handle streamed data instead of a pre-recorded .wav file. However, that caused the system to believe that it was never silent. The reason for this is because for each block of data being analyzed, a new zero crossing threshold is created. When a full audio file was being analyzed, the threshold was established using only the beginning section of the audio file where it was assumed that the audio is silent. With a stream, it is not guaranteed that the beginning of the data is silent since a new block is analyzed several times per second.  
  
For some reason, responds really well to clicks

1. The following are signal plots of the 6 phrases that will eventually be used for the final project.



1. The following are signal plots of the 6 phrases that will eventually be used for the final project.

