Project Phase I Report

1) Problem Description

This phase consisted of manipulating a recorded signal using both MATLAB and C++. First, we recorded a 14 second audio with sampling frequency 2050Hz. In my case I said my name and added a song I like. With that audio we wrote a MATLAB script that imported the original WAV file and modify it like this:

a) The first 10 seconds stay the same.
b) From second 10 to second 14 we had to add two sinusoids (one per channel) with specified frequencies of 2000 Hz and 2200 Hz and an amplitude of 10% max amplitude (1.00).

The resulting wave was then exported to another WAV file that would be again modified using C++. In this case the objective was to add random noise following the function out(t) = in(t) + 0.02*rn(t) – 0.01*A_{max}, where rn(t) is a random integer between 0 and $A_{max.}$ Figure 1 shows the MATLAB script.

```
load handel.mat
filename = 'phase1.wav'
[y,Fs] = audioread(filename);
t = 10:1/Fs:14;
rightSine = 0.1*sin(2*pi*2210.*t);
leftSine = 0.1*sin(2*pi*2200.*t+(pi/3));
y(Fs*10:Fs*14,2) = rightSine;
y(Fs*10:Fs*14,1) = leftSine;
audiowrite('Tarrat_Juan_orig.wav',y,Fs);
```

Figure 1: MATLAB script

2) WAV file format

Figure 2 shows the format of a WAV file. As we can see, it has a header with control fields that specify how the data is organized.

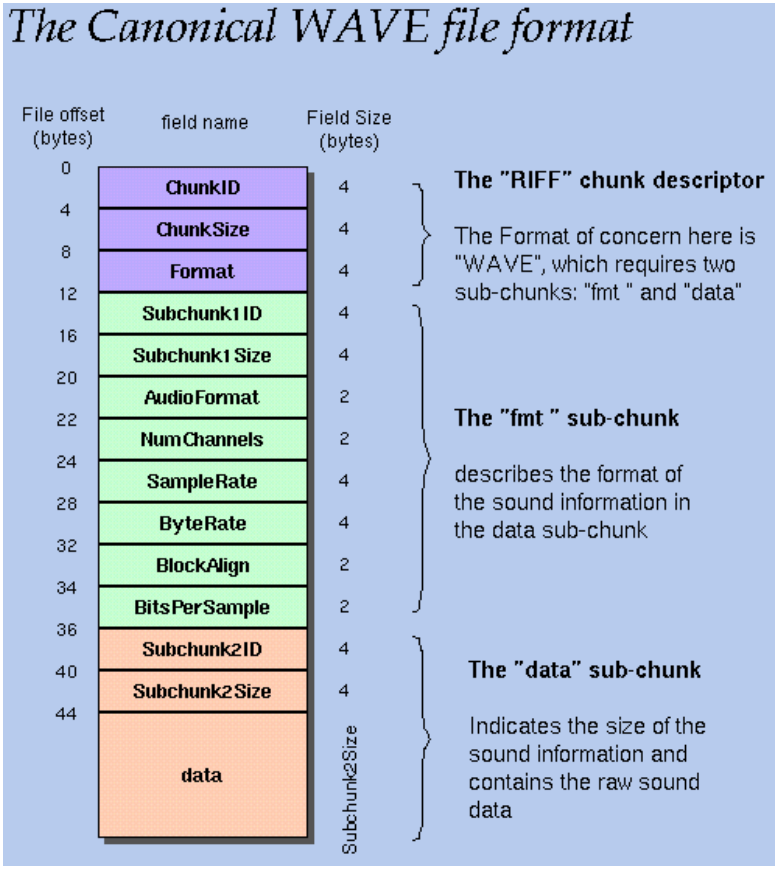

Figure 2: WAV file outline

Some control fields that are very useful when manipulating a signal are Sample Rate, Number of Channels, Bits Per Sample. All these fields us provide us with the necessary data to understand the information in the file.

3) Summary File

Below is a sample output of the output file.

Summary

Filename: Tarrat_Juan_orig.wav

Sampling Freq: 22050

Number of channels: 2

Bits per Sample: 16

File duration: 14s

Maximum value (absolute value): 12010

Channel: 2

Program duration: 206ms

The values on the second section are obtained straight from the information in the header. The third section is obtained while reading the data from the file.

4) Real Time

As we can observe from the contents of the summary file above, the execution time for the last run we did was 206ms, which is way below the duration of the audio (14s). This means that this is suitable for a real time application since each second of audio takes 14.71ms, which is a very small delay. It is important to point out that execution time is not always constant, but always smaller than 1s.