Nolan Anderson

CPE 381

Emil Jovanov

16 March 2021

Real-time Signal Processing

Project 1 Phase 1

Section 1 : Report Questions

1. Short description of the problem and proposed solution.

The problem described for this assignment is to implement real-time audio signal processing in C and C++. The WAV file must first be modified using matlab to replace the last 4 seconds with a sin wave. Then, using C/C++ we need to process the matlab modified file sample-by-sample to add noise to the original wav file.

The solution to this problem can be found in appendix 1 and appendix 2, and I will briefly describe them here. Appendix 1 shows the matlab portion of this assignment and simply takes in a 22050 Hz, 14 second wav file. We then create timing based on the data found in the function "audioinfo". The variables x and y are the 2 sin waves that we will be replacing the last 4 seconds. Finally, the for loop at the end replaces the length from 10-end of the audio file with x and y. This will produce a "beeping" sound. The next portion, shown in appendix 2, shows the C++ real-time implementation of the solution. The wavHeader was found on google and is commonplace in C++. We then take in the file and create variables, note that the max Amplitude is 65535 for 16 bits. We then loop through to the end of the file in the while loop and store it into the data variable using fread. We calculate and overwrite the absolute maximum value and create 2 variables Max and random_noise. These 2 are a part of the noise signal. We then update the data variable, check for overflow, and then write it to the output wav file. Then we write the final data to the output text file.

2. Short description of WAV file format. What control fields do you have in the header? What is their location?

WAV files were created by IBM and Microsoft to store audio bitsream on PC's. Typically used for raw and typically uncompressed audio. Bitstream encoding is linear-pulse code modulation format. The wavHeader shown in appendix 2 shows the different control fields in the header.

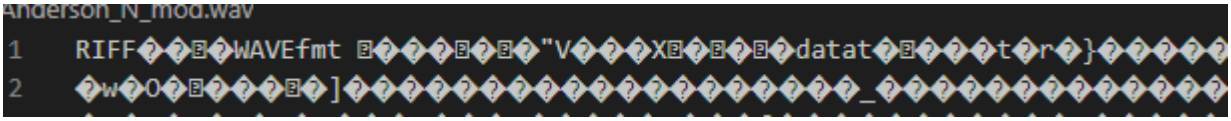

Figure 1: Wav header in Anderson_N_mod.wav

Figure 1 shows where the wave header is in the file. This will hold all the basic information of the wav file. “data” is where the data begins. The ? boxes are the data and the diamonds are .’s. Overall, the header will be from 00 -> location(data)

3. The content of the summary text file.

Anderson_N_sum.txt

```
1   Input File Name:                Anderson_N_matlab.wav
2   Output File Name:               Anderson_N_mod.wav
3   Summary File Name:              Anderson_N_sum.txt
4
5   Sampling Frequency:             22050 hz
6   Number of Channels:             2
7   Number of Bits In Each Sample:  16
8   Length:                         14s
9   Maximum Value of Sample:        5548
10  Execution Time:                 31ms
```

4. Real-time operation. Discuss if your program can work in real-time.

If you were to change input file names to argv[] arguments to the c++, you could do this in real time. It would essentially take input files and then spit out a summary and new wav file. You would essentially call ./PhaseOneExecutable input.wav output.wav summary.txt and you would get output continually. So yes and no, it can work in real time if you did some modifications and wrote publishing and subscribing methods, but in its current implementation it’s technically not real time.

5. Confirmation on my end that the program compiles and runs correctly.

```
[npa0002@blackhawk Project 1]$ make
g++ -g -std=c++11 Anderson_N_Phase1.cpp -o PhaseOneExecutable
[npa0002@blackhawk Project 1]$ ./PhaseOneExecutable
Please enter a file name: Anderson_N_original.wav
[npa0002@blackhawk Project 1]$
```

Section 2 : Appendix

```
% Nolan Anderson
% CPE 381 Spring 2021
% Project 1 Phase 1
filename = 'Anderson_N_original.wav';
F1 = 2200; F2 = 2210;           % Left and right audio
info = audioinfo(filename);     % Info of the audio file
[y, Fs] = audioread(filename);  % Read the audio into y and Fs
Ts = 1/Fs;                      % Ts
t = 0:Ts:info.Duration;
tnew = 10:Ts:info.Duration;
t3 = 0:Ts:10;
A = max(0.1*max(y));            % Maximum amplitude

x = A.*sin((2*pi*F1*tnew) + pi/3);
z = A.*sin((2*pi*F2*tnew) + 0);

index = length(t3);
for n = 1:length(x)
    y(index, 1) = x(n);
    y(index, 2) = z(n);
    index = index +1;
end

audiowrite('Anderson_N_matlab.wav', y, Fs);
```

Appendix 1: Matlab code for adding sin wave to wav file.

```
// ********************************************
// Name: Nolan Anderson
// Course: CPE 381
// Assignment: Project 1 Phase 1
// Due date: 16 March 2021
// Compilation: UAH Blackhawk servers. Use make file to compile or:
//              g++ -g -std=c++11 Anderson_N_Phase1.cpp -o PhaseOneExecutable
//              All of this is performed in bash.
// Execution    ./PhaseOneExecutable
// Description: Reads through a .wav file and modifies it by
//              adding noise to the original values.
//              Also outputs additional information about the
//              modifications made.
// ********************************************
#include <iomanip>
#include <iostream>
#include <cmath>
#include <random>
#include <time.h>
#include <chrono>
```

```
using namespace std;
using namespace std::chrono;

struct wavHeader{
    char       chunkId[4];
    int        chunkSize;
    char       format[4];
    char       subChunk1Id[4];
    int        subChunk1Size;
    short int  audioFormat;
    short int  numChannels;
    int        sampleRate;
    int        byteRate;
    short int  blockAlign;
    short int  bitsPerSample;
    char       subChunk2Id[4];
    int        subChunk2Size;
};

int main(int argc, char* argv[])
{
    string in;
    string WavModified = "Anderson_N_mod.wav";
    string OutSummary = "Anderson_N_sum.txt";
    cout << "Please enter a file name: ";
    cin >> in;
    // in = "Anderson_N_matlab.wav";

    wavHeader audioFile;

    FILE* input = fopen(in.c_str(), "r");
    FILE* output = fopen(WavModified.c_str(), "w+");
    FILE* summary = fopen(OutSummary.c_str(), "w+");

    fread(&audioFile, sizeof(audioFile), 1, input);
    fwrite(&audioFile, sizeof(audioFile), 1, output);

    int16_t data;
    long maxAbsVal = 0;
    float t = 0.0;
    int Amax = 65535;

    auto start = high_resolution_clock::now();
    while(!feof(input))
    {
```

```
        fread(&data, (audioFile.bitsPerSample / 8), 1, input);
        t += 1.0f / audioFile.sampleRate;
        if (abs(data) > maxAbsVal)              // Maximum absolute value of the sample
        {
            maxAbsVal = abs(data);
        }
        short Max = 0.01 * Amax;                                // 0.01*Amax
        short random_noise = 0.02 * short(rand() / Amax);       // Randomnoise
        data = data + random_noise - Max;                       // Final output
        if(data > Amax/2)                                       // Overflow correction
        {
            data = Amax/2;
        }
        fwrite(&data, (sizeof(data)), 1, output);               // Write to output file
    }

    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<milliseconds>(stop - start);
    int sampleLength = (audioFile.subChunk2Size / 4) / audioFile.sampleRate;

    fprintf(summary, "Input File Name:              %s\n", in.c_str());
    fprintf(summary, "Output File Name:             %s\n", WavModified.c_str());
    fprintf(summary, "Summary File Name:            %s\n\n", OutSummary.c_str());
    fprintf(summary, "Sampling Frequency:           %d hz\n", audioFile.sampleRate);
    fprintf(summary, "Number of Channels:           %d\n", audioFile.numChannels);
    fprintf(summary, "Number of Bits In Each Sample: %d\n", audioFile.bitsPerSample);
    fprintf(summary, "Length:                       %ds\n", sampleLength);
    fprintf(summary, "Maximum Value of Sample:      %ld\n", maxAbsVal);
    fprintf(summary, "Execution Time:               %dms\n", duration);
    return 0;
}
```

Appendix 2: C++ code for adding noise to the output from appendix 1.