DIGITAL SIGNAL PROCESSING – PROJECT BASED LEARNING

TITLE: MUSIC NOTE EXTRACTION

Performed By:
3844: OJASVI TUMMALA
3851: PREETI RAUT
3853: APARAJITA SARKAR
3860: SMRIDHI DHINGRA

1. INTRODUCTION

The ability to derive the relevant musical information from a live or recorded performance is relatively easy for a trained listener, but highly non-trivial for a learner and computer. For a number of practical applications, it would be desirable to obtain this information in a quick, error-free, automated fashion. This project discusses the design of a software system that accepts as input a digitized waveform representing an acoustical music signal, and that attempts to derive the notes from the signal so that a musical score could be produced. This signal processing algorithm involved include event detection, or precisely where within the signal the various notes actually begin and end, and pitch extraction, or the identification of the pitches being played in each interval. The event detection is carried out using the time domain analysis of the signal, where the problem arises with different speed. The pitch detection is (nothing but frequency identification) is more complicated because of a situation we call harmonic ambiguity; this occurs when one pitch whose fundamental frequency is an integer multiple of another pitch. The problem is solved by the careful signal processing in boththe time domain signals and frequency domain signals.

The main objective of this project is to create an aid tool for learning for Musicians, Producers, Composers, DJs, Remixer, Teachers and Music Students. This project can be treated as a box, where you give any song as input and get the features of the song out. The aim of this project is to propose methods to analyze and describe a signal, from where the musical parameters can be easily and objectively obtained, in a sensible manner.

2. LITERATURE SURVEY

2.1 Sound

A sound can be characterized by the following three quantities: (i) Pitch.(ii) Quality.(iii) Loudness.
Pitch is the frequency of a sound as perceived by human ear. A high frequency gives rise to a high pitch note and a low frequency produces a low pitch note. A pure tone is the sound of only one frequency, such as that given by a tuning fork or electronic signal generator. The fundamental note has the greatest amplitude and is heard predominantly because it has a larger intensity. The other frequencies such as 2fo, 3fo, 4fo,... are called overtones or harmonics and they determine the quality of the sound. Loudness is a physiological sensation. It depends mainly on sound pressure but also on the spectrum of the harmonics and the physicalduration.

2.2 Musical Notes

Human can hear signal frequency ranging from 20-20 kHz. From this wide range some part is associated with piano. Different pianos are having different ranges. Each tone of piano is having one particular fundamental frequency and represented by a note like C, D, ...etc. as shown in fig 1 .The later C is 12 half steps away the previous one and having double the fundamental frequency. Hence this portion (from one C immediate next C) is called one octave. Different octaves are differentiated by C_1,C_2, etc.

Fig 1. An octave of a piano

2.2.1 Equation For the Frequency Table

The basic formula for the frequency of the notes of the equal tempered scale is given by $f_n = f_0 * (a)^n$where f_0= the frequency of one fixed note which must be defined. A common choice is setting the A above middle C(A_4) at f_0 =440 Hz, n = number of half steps away from the fixed note you are, f_n= the frequency of the notes n half steps away, $a = (2)^{1/12}$

2.2.2 Frequencies for Equal Tempered Scale at A_4 = 440 Hz

Table 1. Notation to Frequency Mapping [Middle C is C_4]

n	Note	Fundamental Frequency(Hz)
-4	F_3	174.61
-3	$F_3^\#$	185
-2	G_3	196
-1	$G_3^\#$	207.65
0	A_4	220
1	$A_4^\#$	233.08
2	B_4	246.94
3	C_4	261.63 #
4	C_4	277.18
5	D_4	293.66
6	$D_4^\#$	311.13
7	E_4	329.63

2.3 Digital Signal Processing for music

2.3.1 Sampling

When a sound wave is created by your voice (or a musical instrument), it is an analog wave of changing air pressure. However, in order for a computer to store a sound wave, it needs to record discrete values at discrete time intervals. The process of recording discrete time values is called sampling, and the process of recording discrete pressures is called quantizing. Recording studios use a standard sampling frequency of 48 kHz, while CDs use the rate of 44.1 kHz.
Signals should be sampled at twice the highest frequency present in the signal.
Humans can hear frequencies from approximately 20-20,000 Hz, which explains why common sampling frequencies are in the 40 kHz range.

2.3.2 Frequency and Fourier Transforms

A Fourier transform provides the means to break up a complicated signal, like a musical tone, into its constituent sinusoids. This method involves many integrals and a continuous signal. We want to perform a Fourier transform on a sampled (rather than continuous) signal, so we have to use the Discrete Fourier Transform instead.

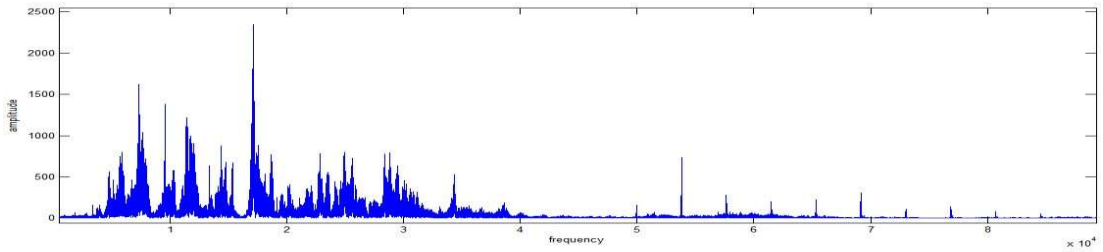


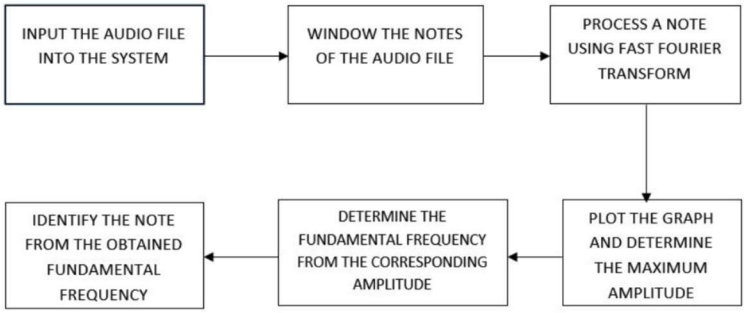
Fig 2. FFT of a Musical Signal

From the fig 2 we can see that the index corresponding to the maximum amplitude represents the most significant frequency component, that can be found using the formula

$$f = \frac{i}{T} * fs \qquad (1)$$

Where i = index at which maximum amplitude exists , T = Total samples in the fft at a time.

3. IMPLEMENTATION METHODOLOGY



The process used for this project can be described in the given stages.

1) The first step involves the input of a sequence of desired musical note whose frequency is to be found. The sequence of sounds are digitally recorded.

2) In the second step, the signal is analysed using MATLAB. In MATLAB, the series of notes are initially plotted in the time domain.

3) The third step requires more of analysis and programming. We designed a windowing technique that determines the window of each note in the sequence. This technique determines when a note begins and ends. This program takes in a double array specified by the data and outputs a vector of division points that correspond to separate and distinct notes.

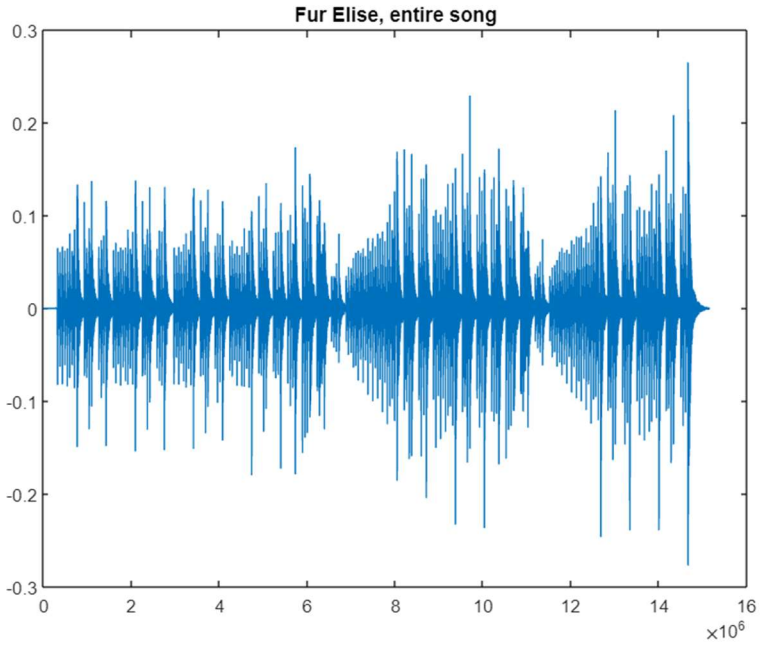
4) Next, we down sample 20 times using an average filter to take out some of the high frequency content as well as to decrease the number of points in the sample to speed up calculations specially with the upcoming FFT's.

5) Next step is to threshold the notes which is done using moving average threshold filter.This is done as some of the notes are significantly louder than others and we can detect the tall peaks while also preserving the smaller ones

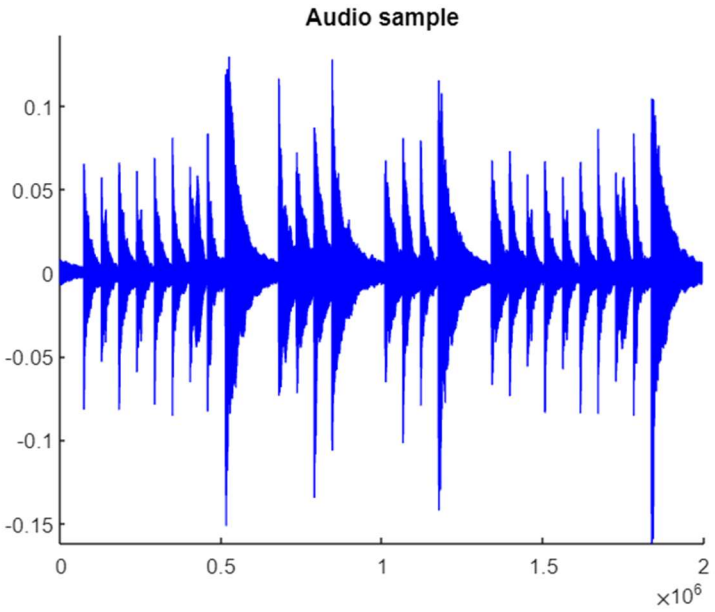
6) The sixth step involves the analysis of the signal in the frequency domain. Now as the windows of each note have been determined in the step above mentioned, each individual note is converted to the frequency domain. Hence, here comes the application of fast fourier series to get the frequency domain of our input signal.

7) The last step involves determining the fundamental frequency of each note. A standard musical note played by standard instruments is described by a fundamental frequency, which is the maximum frequency, along with other smaller harmonics. Therefore, to determine the fundamental frequency, find the maximum frequency for each signal. This maximum frequency is mapped onto the known fundamental frequency of note to find the actual note.

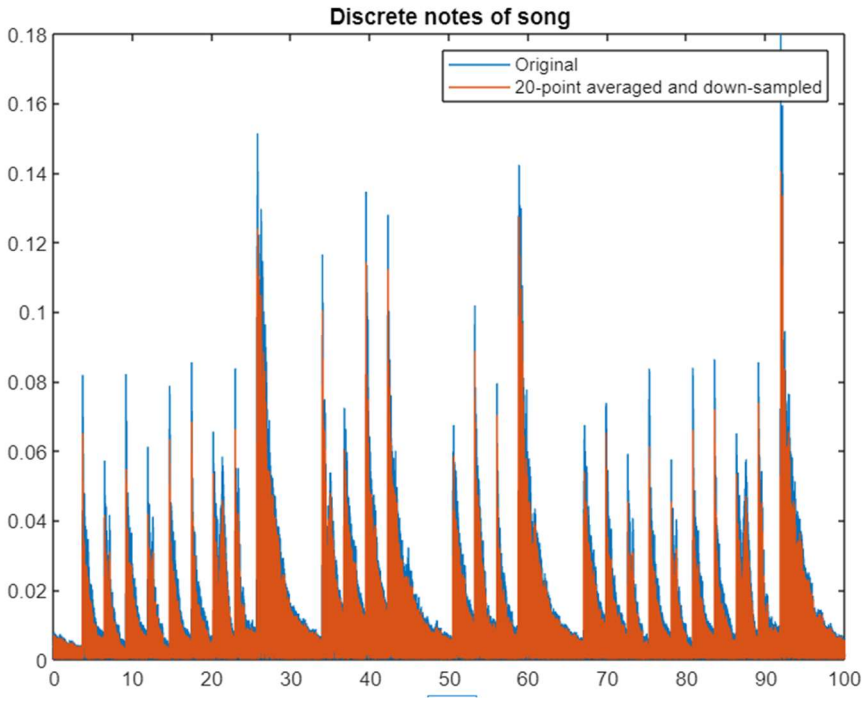
4. SIMULATION RESULTS



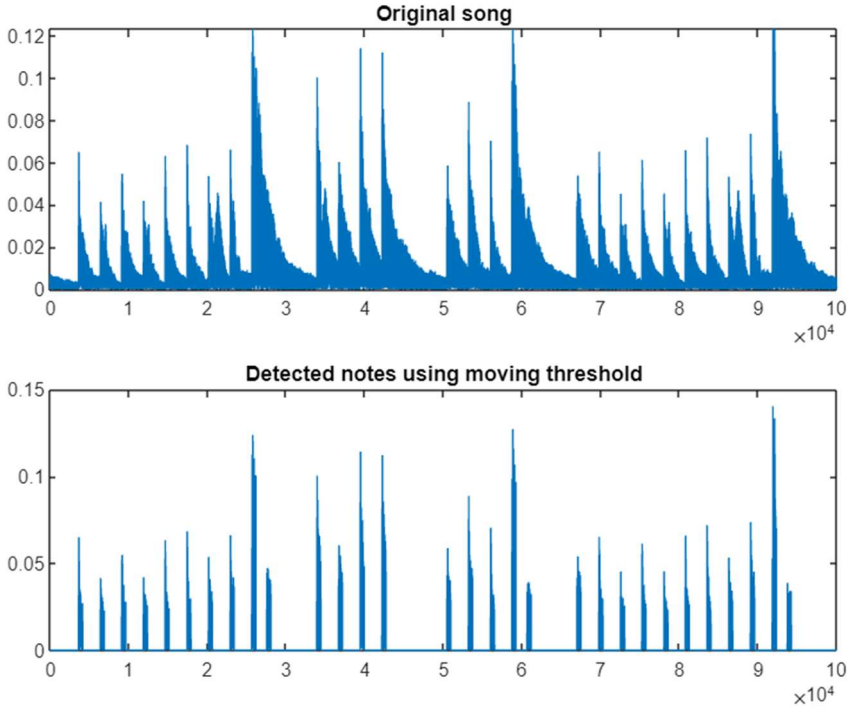
AFTER WINDOWING:



DOWNSAMPLED:



AVERAGE MOVING THRESHOLD



FUNDAMENTAL FREQUENCY OF EACH NOTE (1-30)

Figure 1

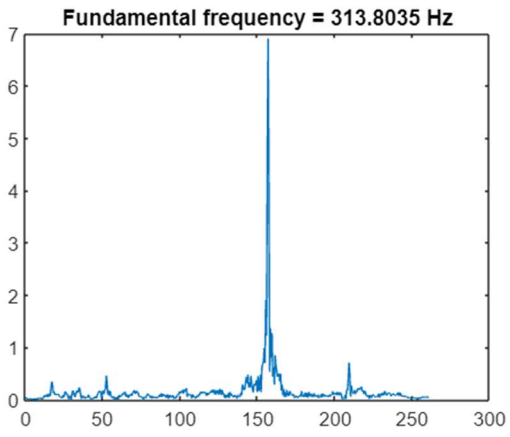
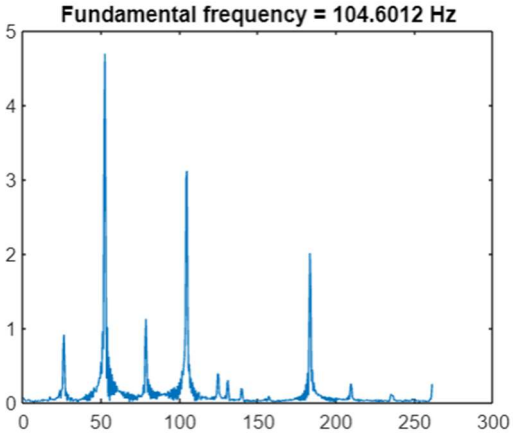


Figure 30



Command Window

```
"Eb4"
"D4"
"Eb4"
"D4"
"Eb4"
"Bb3"
"Db4"
"B3"
"Ab3"
"Ab2"
"B2"
"Eb3"
"Ab3"
"Bb3"
"Eb3"
"G3"
"Bb3"
"B3"
"Ab2"
"Eb3"
"Eb4"
"D4"
"Eb4"
"D4"
"Eb4"
"Bb3"
"Db4"
"B3"
"Ab3"
"Ab2"
```

5. FUTURE SCOPE

There is still much room for future development that would enhance the system and increase its usage value. The following items are some suggestions:

Non Periodic Signal analysis: The process is relatively simple if the signal were sinusoidal or periodic. But the real life musical notes or vocals are approximately periodic and the frequency itself changes with time because a sample may contain more than one note and that is how music is played. While Fourier Analysis is a nice solution to this problem, it is not sufficient. Theoretically it may be sufficient but its high level implementation is not as there is resolution and run time limit. There is scope to overcome latter by designing algorithms especially for the purpose of frequency estimation and not focusing on phase detection.

Multiple Notes at a time: Our project assumes that only a single note is played at a time. But that is not it. We can develop it further by using Fourier Analysis again. There are existing algorithms which can isolate multiple notes. After splitting the audio sample into individual notes, we can apply our own techniques to find the frequency.

Combine with multiple feature recognition: Different instruments can be isolated and features of human voice can be studied to develop an advanced system that can be implemented on all songs and live audio recordings. Scoring can be integrated in this system which can have real life applications such as music training and judging events.

6. BIBLIOGRAPHY

1] https://www.sciencedirect.com/science/article/pii/S1877050915020281
2] https://en.wikipedia.org/wiki/Music_information_retrieval
3] https://www.scitepress.org/papers/2007/21393/21393.pdf
4] https://github.com/ananya2407/Music-Note-Recognition

MATLAB PROGRAM

NoteExtraction.m main function

```
% main program to extract notes from audio file
% input file: FurElise_Slow.mp3

%% set up song
clear; clc; clf; close all

mute = false; % set this to false to hear audio throughout program
              % useful for debugging

[song,Fs] = audioread('FurElise_Slow.mp3');
Fs = Fs*4;   % speed up song (original audio file is very slow)
figure, plot(song(:,1)), title('Fur Elise, entire song')

%% set parameters (change based on song)
t1 = 2.9e6; t2 = 4.9e6;

% analyze a window of the song
y = song(t1:t2);
[~,n] = size(y);
t = linspace(t1,t2,n);
if ~mute, plotsound(y,Fs); end
audiowrite('fur_elise_window.wav',y,Fs);

%% downsample by m
clc
m = 20;
Fsm = round(Fs/m);
p = floor(n/m);
y_avg = zeros(1,p);
for i = 1:p
    y_avg(i) = mean(y(m*(i-1)+1:m*i));
end
figure, plot(linspace(0,100,n),abs(y)), hold on
        plot(linspace(0,100,p),abs(y_avg))
        title('Discrete notes of song')
        legend('Original', '20-point averaged and down-sampled')
if ~mute, sound(y_avg,Fsm); end


%% threshold to find notes
close all
y_thresh = zeros(1,p);
i = 1;
while (i <= p)
    thresh = 5*median(abs(y_avg(max(1,i-5000):i)));
    if (abs(y_avg(i)) > thresh)
        for j = 0:500
            if (i + j <= p)
                y_thresh(i) = y_avg(i);
                i = i + 1;
            end
        end
        i = i + 1400;
```

```
    end
    i = i + 1;
end

figure, subplot(2,1,1), plot(abs(y_avg)), title('Original song'), ylim([0
1.1*max(y_avg)])
        subplot(2,1,2), plot(abs(y_thresh)), title('Detected notes using moving
threshold')

if ~mute, sound(y_thresh,round(Fsm)); end

%% find frequencies of each note
clc; close all

i = 1;
i_note = 0;
while i < p
    j = 1;
    end_note = 0;
    while (((y_thresh(i) ~= 0) || (end_note > 0)) && (i < p))
        note(j) = y_thresh(i);
        i = i + 1;
        j = j + 1;
        if (y_thresh(i) ~= 0)
            end_note = 20;
        else
            end_note = end_note - 1;
        end
        if (end_note == 0)
           if (j > 25)
               note_padded = [note zeros(1,j)]; % pad note with zeros to double size (N -
-> 2*N-1)
               Note = fft(note_padded);
               Ns = length(note);
               f = linspace(0,(1+Ns/2),Ns);
               [~,index] = max(abs(Note(1:length(f))));
               if (f(index) > 20)
                   i_note = i_note + 1;
                   fundamentals(i_note) = f(index)*2;
                   figure, plot(f,abs(Note(1:length(f))))
                           title(['Fundamental frequency =
',num2str(fundamentals(i_note)),' Hz'])
                           %plot(note_padded)
               end
               i = i + 50;
           end
           clear note;
           break
        end

    end
    i = i + 1;
end



%% play back notes
```

```
amp = 1;
fs = 20500;  % sampling frequency
duration = .5;
recreate_song = zeros(1,duration*fs*length(fundamentals));
for i = 1:length(fundamentals)
    [letter(i,1),freq(i)]= FreqToNote(fundamentals(i));
    values = 0:1/fs:duration;
    a = amp*sin(2*pi*freq(i)*values*2);
    recreate_song((i-1)*fs*duration+1:i*fs*duration+1) = a;
    if ~mute, sound(a,fs); pause(.5); end
end
letter
audiowrite('fur_elise_recreated.wav',recreate_song,fs);
```

plotsound.m function

```
function plotsound(y,Fs)
waitTime = .1;
samples = ceil(Fs*waitTime);
figure, hold on, xlim([0 length(y)]), ylim([.9*min(y) 1.1*max(y)])
        title('Audio sample')
sound(y,Fs)
    for i = 1:samples:length(y)-samples
        plot(i:i+samples,y(i:i+samples),'b')
        pause(waitTime)
    end
```

frequencytonote.m function

```
function [note,frequency] = FreqToNote(f)
    index = round(17.31232*log(f) - 47.37620);
    frequencies = [16.35 17.32 18.35 19.45 20.6 21.83 23.12 24.5 25.96 27.5 29.14 30.87
32.7 34.65 36.71 38.89 41.2 43.65 46.25 49 51.91 55 58.27 61.74 65.41 69.3 73.42 77.78
82.41 87.31 92.5 98 103.83 110 116.54 123.47 130.81 138.59 146.83 155.56 164.81 174.61
185 196 207.65 220 233.08 246.94 261.63 277.18 293.66 311.13 329.63 349.23 369.99 392
415.3 440 466.16 493.88 523.25 554.37 587.33 622.25 659.25 698.46 739.99 783.99 830.61
880 932.33 987.77 1046.5 1108.73 1174.66 1244.51 1318.51 1396.91 1479.98 1567.98 1661.22
1760 1864.66 1975.53 2093 2217.46 2349.32 2489.02 2637.02 2793.83 2959.96 3135.96 3322.44
3520 3729.31 3951.07 4186.01 4434.92 4698.63 4978.03 5274.04 5587.65 5919.91 6271.93
6644.88 7040 7458.62 7902.13];
    notes = ["C0" "Db0" "D0" "Eb0" "E0" "F0" "Gb0" "G0" "Ab0" "A0" "Bb0" "B0" "C1" "Db1"
"D1" "Eb1" "E1" "F1" "Gb1" "G1" "Ab1" "A1" "Bb1" "B1" "C2" "Db2" "D2" "Eb2" "E2" "F2"
"Gb2" "G2" "Ab2" "A2" "Bb2" "B2" "C3" "Db3" "D3" "Eb3" "E3" "F3" "Gb3" "G3" "Ab3" "A3"
"Bb3" "B3" "C4" "Db4" "D4" "Eb4" "E4" "F4" "Gb4" "G4" "Ab4" "A4" "Bb4" "B4" "C5" "Db5"
"D5" "Eb5" "E5" "F5" "Gb5" "G5" "Ab5" "A5" "Bb5" "B5" "C6" "Db6" "D6" "Eb6" "E6" "F6"
"Gb6" "G6" "Ab6" "A6" "Bb6" "B6" "C7" "Db7" "D7" "Eb7" "E7" "F7" "Gb7" "G7" "Ab7" "A7"
"Bb7" "B7" "C8" "Db8" "D8" "Eb8" "E8" "F8" "Gb8" "G8" "Ab8" "A8" "Bb8" "B8"];
    note = notes(index);
    frequency = frequencies(index);
end
```