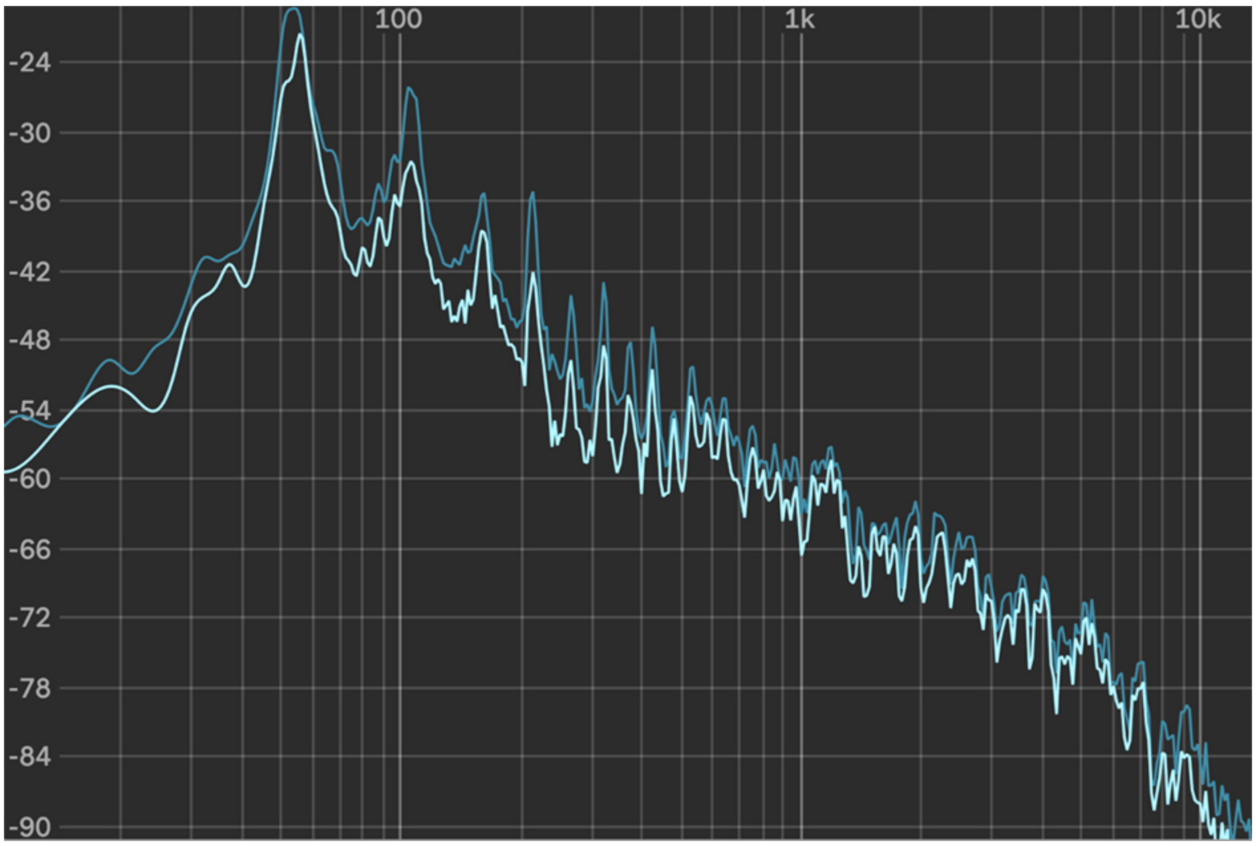
REAL–TIME AUDIO SPECTRUM ANALYZER

with MATLAB and Arduino



GROUP MEMBERS

LINDA MARY ZACHARIA (B210768EE)

MADHAV MENON (B210821EE)

NEERAJ S (B210879EE)

Abstract

This project aims to develop an audio spectrum analyzer utilizing an Arduino Uno board and MATLAB. The primary objective of this project is to design and implement an efficient system that can accurately analyze the frequency spectrum of an audio signal in real-time and visualize it in graphical format. The system achieves this by interfacing a microphone with the Arduino board and processing the audio signal using MATLAB. The output of the analysis is presented on a frequency plot that is displayed on a screen. This audio spectrum analyzer has potential applications in various fields such as sound engineering and the music industry, where it can serve as a reliable and accessible tool for analyzing audio signals.

Introduction

Audio spectrum analyzers are used in the audio industry to analyze the frequency content of audio signals. In this project, we have utilized the Fourier transform to perform the audio signal processing. The Fourier transform is a mathematical technique that transforms a time-domain signal into its frequency-domain representation. To accomplish this for the project, we have employed the FFT (Fast Fourier Transform) algorithm. This is a highly efficient method for computing the discrete Fourier transform of a signal.

The system is designed to capture the audio signal using a microphone and process it using Arduino IDE and MATLAB. The FFT algorithm is used to generate a frequency plot to display the frequency spectrum with real-time data.

The audio spectrum analyzer has wide applications in the audio industry. It allows the accurate and efficient analysis of audio signals and provides the information required to produce audio recordings of high quality, among other uses.

Methodology

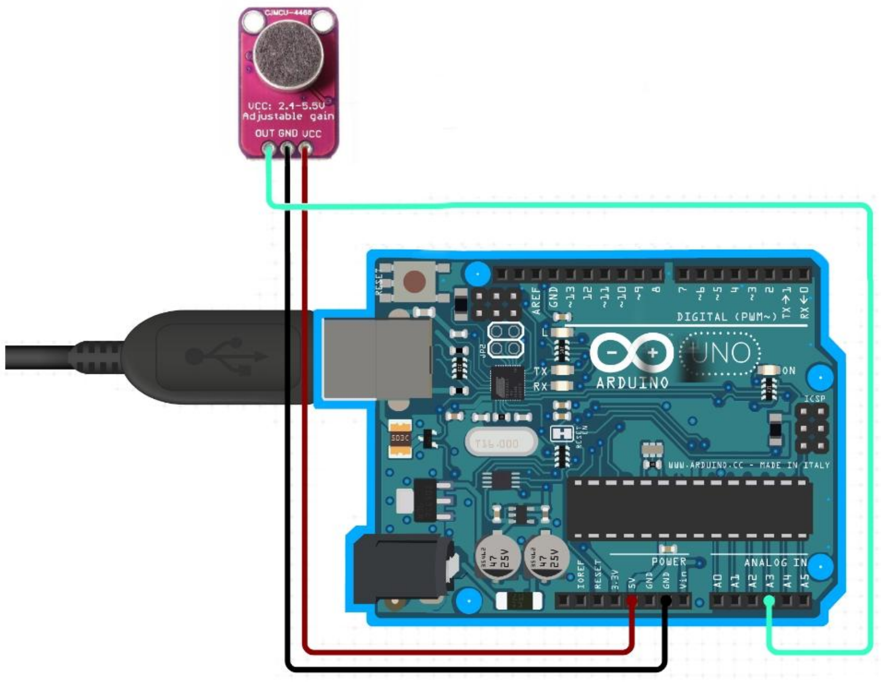
In this study, we employed MATLAB and Arduino IDE to analyze the audio frequency. Our approach involved audio data acquisition, signal processing and spectrum generation,

We utilized the Arduino IDE to establish serial communication to transfer audio data from Arduino to MATLAB. This allowed us to perform basic processing on the audio data and reduce the load on the MATLAB program.

To collect the audio data, we used a high-quality microphone connected to an Arduino board. The audio data was then transferred to MATLAB, where we performed signal processing and feature extraction. Specifically, we used the Fast Fourier Transform (FFT) to convert the audio signal into the frequency domain.

After computing the FFT, we plotted an amplitude-frequency curve to visualize the frequency content of the signal. This curve shows the magnitude of the signal at each frequency, allowing us to identify dominant frequency components and analyze the overall frequency content of the signal.

After acquiring and processing this data, we also analyzed the frequency response using MATLAB alone utilizing the highly efficient on-board processor and microphone that allows for the computation of frequency analysis using the Fast Fourier Transform (FFT) algorithm. This was done to compare the accuracy of our results.



Code (using MATLAB alone)

(This code was used to obtain the real-time frequency spectrum using MATLAB alone. The audio signal was captured using the laptop microphone. No external devices were used.)

```
% Create an audio recorder object
recorder = audioDeviceReader;

% Define sampling rate and number of samples per frame
fs = recorder.SampleRate;
samples_per_frame = 1024;

% Create a frequency vector for plotting
w = linspace(0 , fs/2 , samples_per_frame/2+1);

% Create a figure for plotting the spectrum
figure;
h = plot( w , zeros(samples_per_frame/2+1 , 1) , '-b ' ,  'LineWidth' , 2 );
title( 'Real-time Spectrum Analyzer' );
xlabel( 'Frequency (Hz)' );
ylabel( 'Magnitude' );

% Loop to continuously read and plot audio data
while true
   % Read a frame of audio data
   audio_data = recorder();

   % Calculate the one-sided spectrum using the FFT algorithm
   spectrum = abs( fft( audio_data(: , 1) , samples_per_frame));
   spectrum = spectrum(1: samples_per_frame/2+1);

   % Update the plot with the new spectrum data
   set(h , 'YData' , spectrum);
   ylim([0 , max(spectrum)+10]);
   drawnow;
end

% Release the audio recorder object
release(recorder);
```

Code (integrating Arduino IDE and MATLAB)

(These codes were used in Arduino IDE and MATLAB to obtain the real-time frequency spectrum with an Arduino UNO)

In Arduino IDE

```
 void setup()
{
        // Set up the microphone pin
        const int micPin = A3;

        // Set up the serial communication
        int baudRate = 240000; // Set the desired baud rate
        Serial.begin(240000);
}
// Define a string variable to hold the analog voltage reading
String s;

void loop()
{
         // Read the analog voltage from pin A0 and convert it to a string
        s = String(analogRead(A3));

        // Add leading zeros to the string if necessary to make it 4 characters long
                if (s.length() < 2)                                    //If string length is less than 2, the 3 if loops
                                                             ensure that 3 leading zeros are added and length becomes 4
        {
                 Serial.print(0);
        }

                    if (s.length() < 3)                                //If string length is less than 3, the 2 if loops
                                                             ensure that 2 leading zeros are added and length becomes 4
        {
                 Serial.print(0);
        }

                if (s.length() < 4)                          //If string length is less than 4, the 1 if loop
                                                  ensure that 1 leading zero is added and length becomes 4

        {
                Serial.print(0);
        }

        // Send the analog voltage reading via serial communication
        Serial.println(s);
}
```

<u>In MATLAB</u>

```
% Open serial communication with the Arduino on port COM5 and a baud rate of 240000
Port = serialport("COM5", 240000);

% Create an array of large number of zeros to store the data received from the Arduino
array = zeros(1, 10e6);

% Initialize the index i to 1
i = 1;

% Set up the FFT parameters
Fs = 4000;
N = 100;

% Create a time vector t with 100 elements, representing the time values of the signal
t = (0:N)/Fs;
```

```
% Create a frequency vector w with 51 elements so as to satisfy Nyquist Criterion, representing
the frequencies of the FFT output
w = Fs*(0:(N/2))/N;

% Flush the serial port buffer to ensure clean communication
flush(Port);

% Create a new figure window
fig = figure;

% Get the current axes of the figure
% gca is an inbuilt function that will return the handle of the current axes
ax = gca;

% Initialize the plot with a vector of zeros, with the same size as w
% plotHandle is a variable that holds the plot handle created by the plot function
plotHandle = plot(w, zeros(size(w)));

% Set the x-axis label of the plot to "Frequency(Hz)" and the y-axis label of the plot to
“Amplitude”
xlabel("Frequency(Hz)");
ylabel("Amplitude");

% Automatically set the y-axis limits based on the data
ylim auto;

% Main loop
while i < 10e6

  % Read a line of data from the Arduino
  in = readline(Port);

  % Convert the received data from string to double and scale it by a factor of 0.0049
  % Division by 1024 (number of possible values of 10-bit ADC of Arduino) to obtain voltage
values
  array(i) = str2double(in)*0.0049;        % (1/1024)*5 = 0.0049
                                           % Therefore array(i) will contain voltage values
```

```
    % If enough samples have been received, and the index is a multiple of 100, update the plot
    % The if loop is executed only when i is a multiple of 100

    if i > N && mod(i, 100) == 0
        % Apply the FFT to the last l samples of data
        x = fft(array(i-N+1:i));

        % Calculate the two-sided spectrum P2
        P2 = abs(x/N);

        % Calculate the single-sided spectrum P1
        P1 = P2(1:N/2+1);
        P1(2:end-1) = 2*P1(2:end-1);

        % Update the y-data of the plot with the new spectrum values
        set(plotHandle, 'YData', P1);

        % Set y-axis limits based on maximum value of plot
        set(ax, 'YLim', [0 max(P1)*1.1]);

        % Update the figure window
        drawnow;
    end

    % Increment the index
    i = i+1;
end
```

Result

The audio spectrum analyzer was successfully implemented using the Arduino UNO and MATLAB. The system was capable of analyzing the audio signal and plotting the frequency spectrum quite accurately in real time. The signal was sampled at a rate of 4000 samples per second as specified in the code.

However, **greater accuracy** can be obtained by various means including reducing the cable length and improving the cable quality; but the most important among them being **reducing the baud rate (from the present value of 240000 to a value around or less than 115200)** and also **maintaining the relation given by**

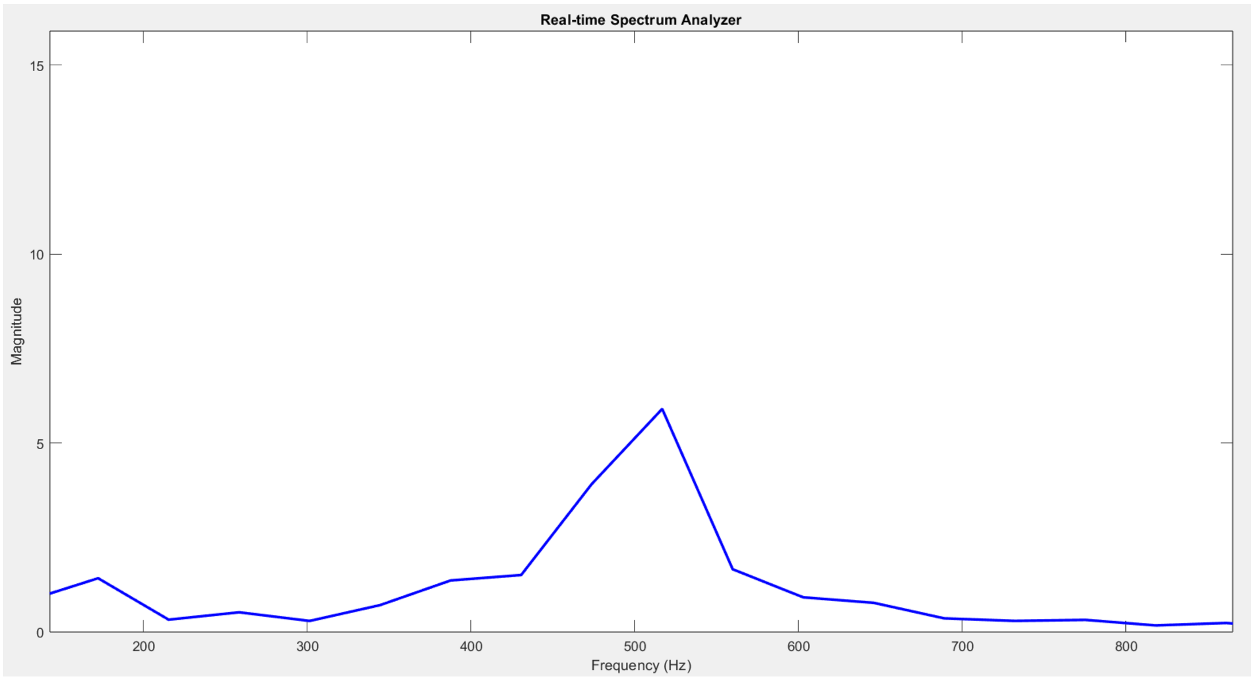
(Number of bits in a string of data) * (Sampling Frequency) = Baud Rate of Serial Communication —(a)

Here, we are sending **60 bits of data in a string** via serial communication.

The maximum permissible baud rate for Arduino Uno for its effective communication is 115200. On increasing the baud rate, we observe errors in serial communication due to the shorter time available to transmit each bit. Accounting a baud rate of 240000 resulted in slight errors in the frequency spectrum.

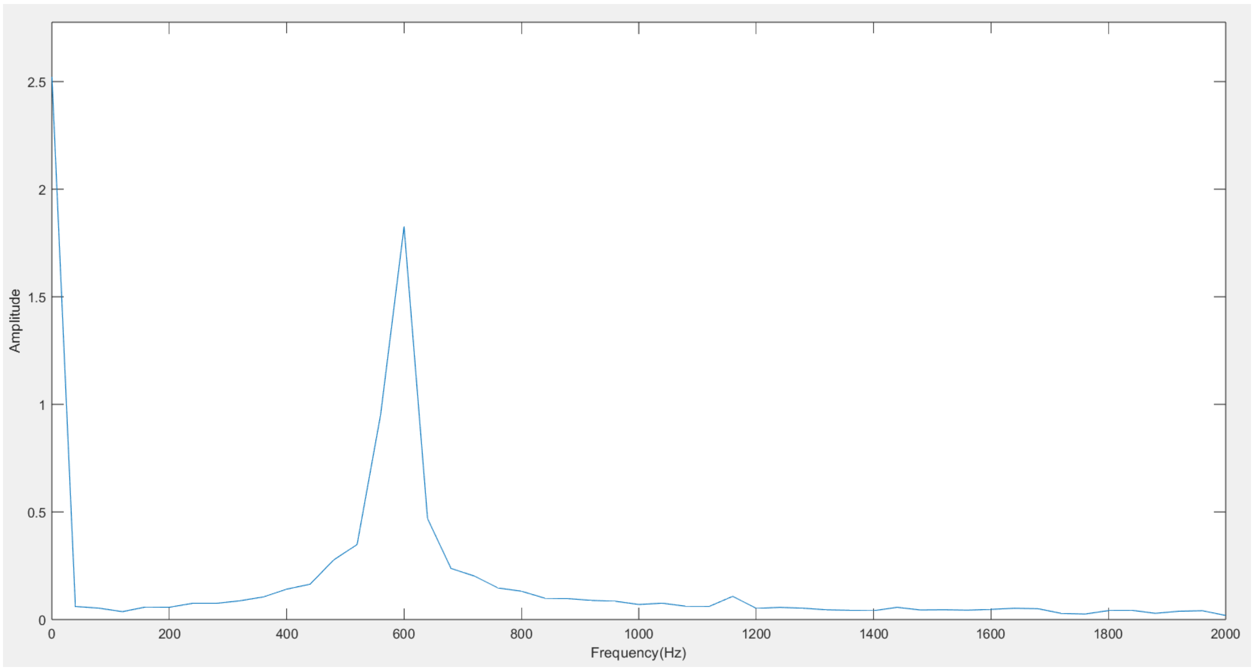
This can be explained by the following scenarios:

Frequency plot obtained using MATLAB only (*when a signal of 500 Hz was applied* ***)***

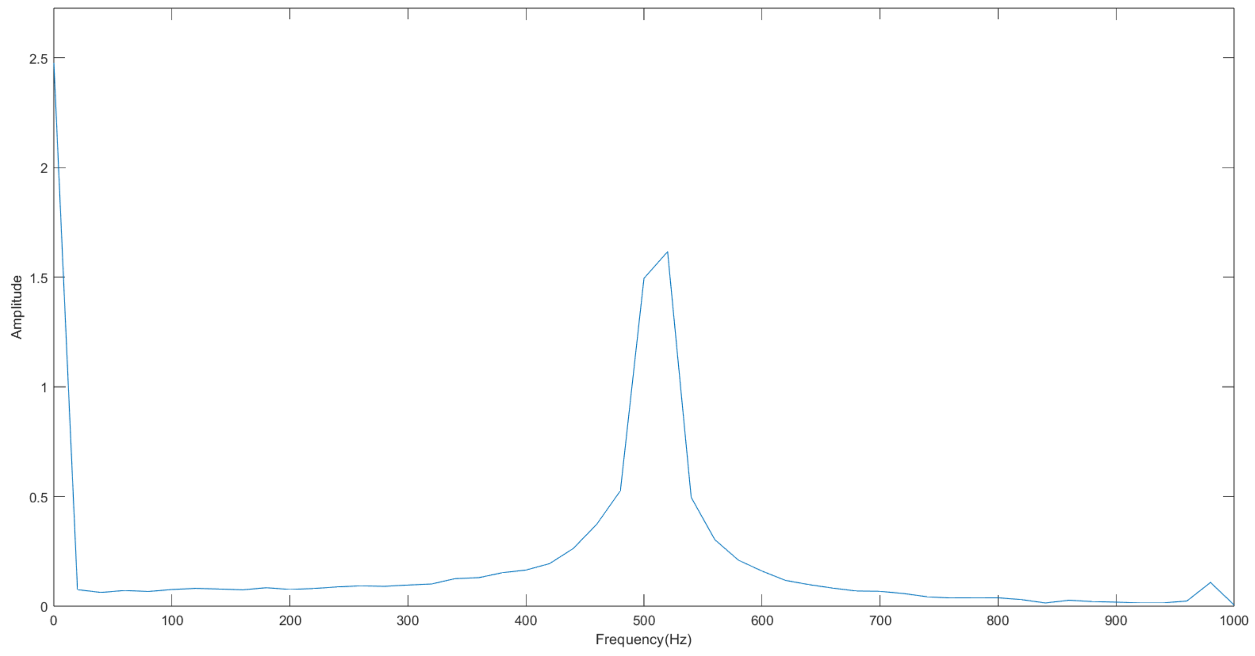


Frequency plot obtained using MATLAB and Arduino

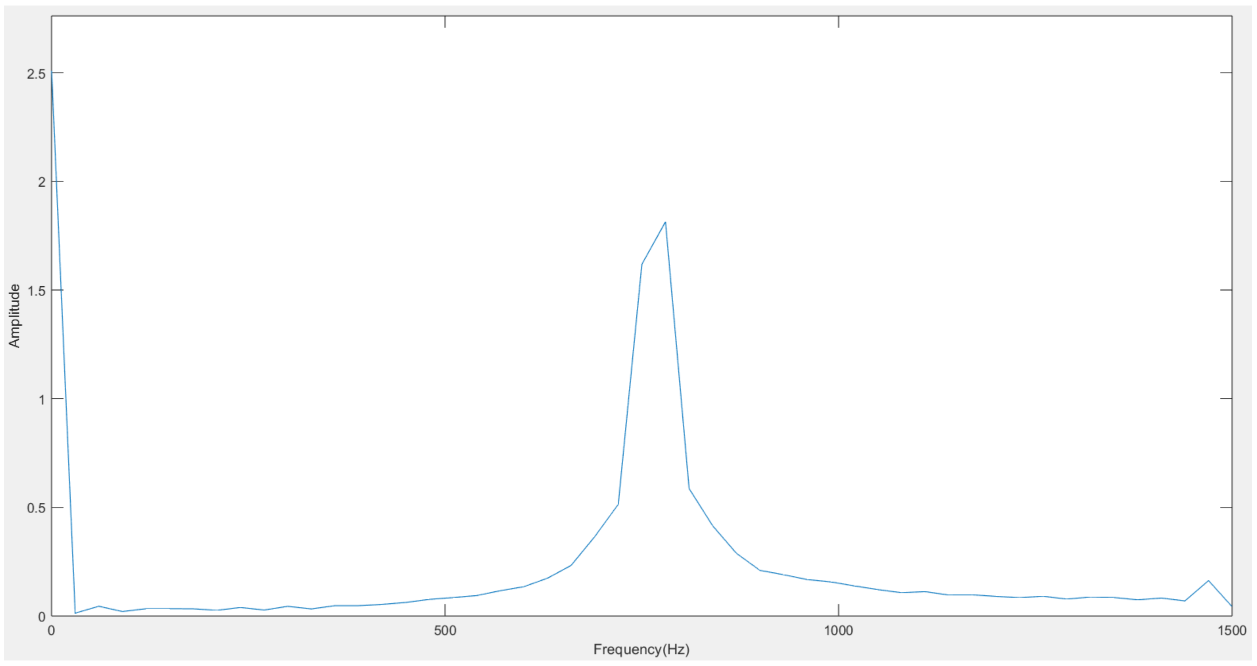
01. *When a signal of 500 Hz with Fs = 4000 Hz and Baud Rate = 240000 was applied* ***)***



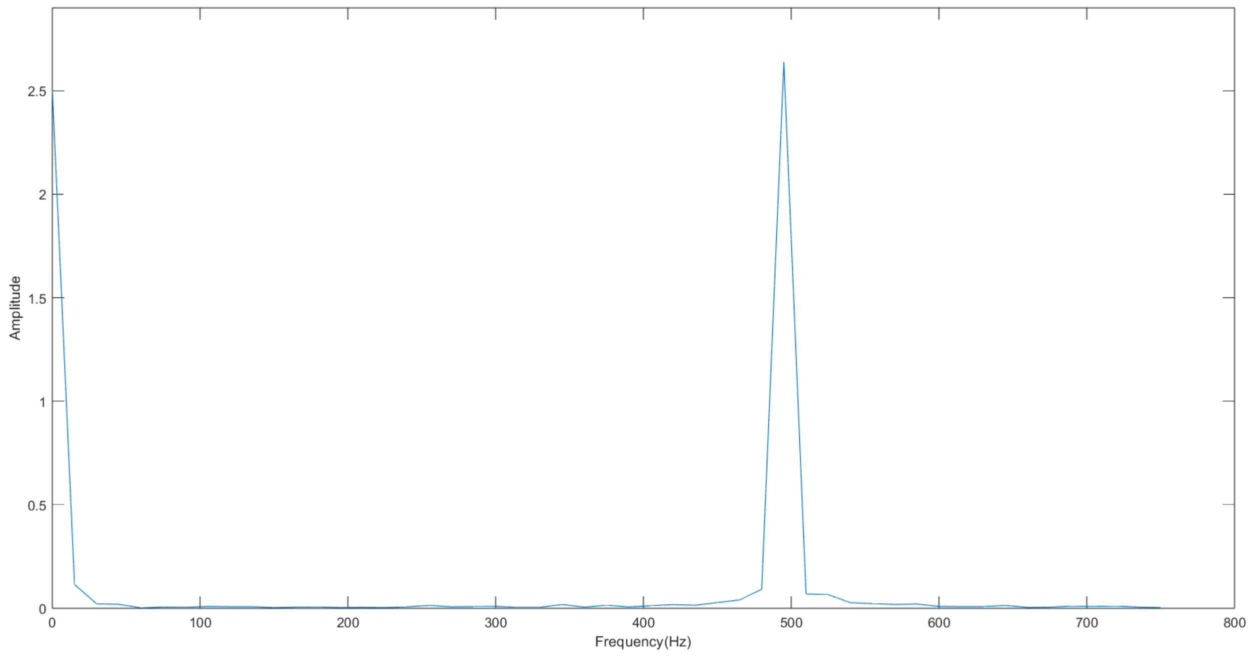
02.** When a signal of 500 Hz with Fs = 2000 Hz and Baud Rate = 120000 was applied **)



03.** When a signal of 500 Hz with Fs = 3000 Hz and Baud Rate = 120000 was applied **)



04.** When a signal of 500 Hz with Fs = 1500 Hz and Baud Rate = 90000 was applied **)



From the graphs, it can be observed that

1) Graph ***02*** gives a more accurate reading than Graph ***01*** because the value of Baud Rate used for plotting Graph ***02*** is closer to the permissible value.
2) Graph ***04*** gives a more accurate reading than Graph ***03*** because the relation given by **(a)** is satisfied in the case of Graph ***04*** whereas it is not satisfied in Graph ***03***.
3) The Graphs obtained using MATLAB and Arduino is verified by comparing it with the Graph obtained using MATLAB alone utilizing the onboard processor and microphone.

Conclusion

The Arduino-based frequency spectrum analyzer was successfully designed and implemented using the Arduino UNO board and MATLAB software. The frequency spectrum for the audio signal was obtained in real time and represented in a plot using the Fast Fourier Transform (FFT) algorithm. We observed that the signal primarily consisted of frequencies ranging from 0 Hz to 4000 Hz.

It is important to note that the accuracy of the FFT analysis can be affected by factors such as the sampling rate (Fs), window function (N), and baud rate of serial communication. Further studies could explore the impact of these factors on the accuracy of frequency spectrum analysis.

Overall, this project demonstrates the potential of using Arduino in conjunction with softwares like MATLAB to develop efficient audio signal processing systems at a low cost and has significant implications for the field of signal processing and enlightened us with the importance of signal processing techniques in analyzing complex signals.