SIGNALS AND SYSTEMS

PROJECT - FOURIER TRANSFORM OF AUDIO SIGNAL

National Institute of Technology Calicut

Submitted by:

VISHNU MOHAN E S B211274EE

ZACHARIA PRAKASH B210793EE

PRIYANANDAN K S B210818EE

Introduction

As part of the semester project for the course EE2002D (Signals & Systems), the task was assigned to design a device that enables us to capture audio signals around us and analyze the Fourier transform. This was to be done using an Arduino UNO, a microphone, and an accompanying circuit. Fast Fourier Transform (FFT) algorithm was to be used on the audio signal and an amplitude vs frequency plot and amplitude vs time plot is made instantaneously based on the audio signal.

Requirements

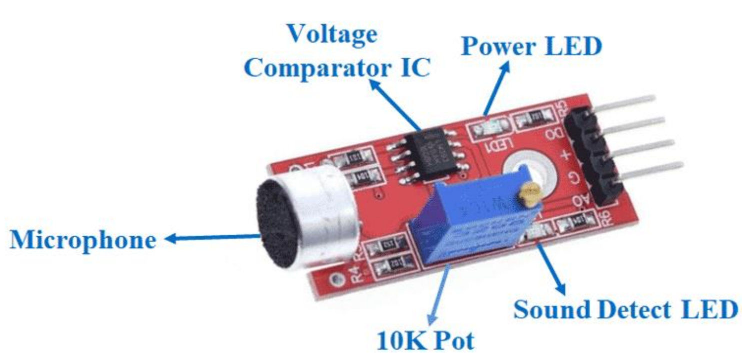
1. Sound detection sensor module -LM393

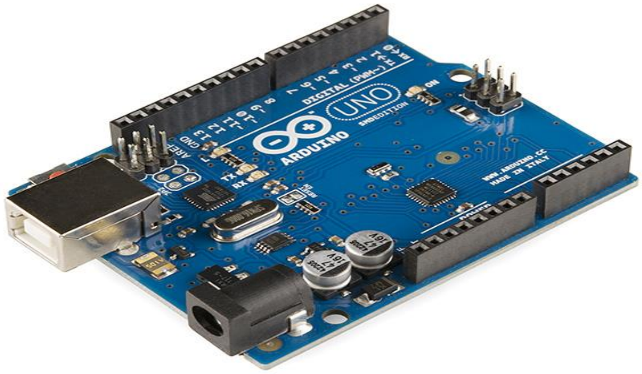
   The sound detection sensor module detects the intensity of sound where sound is detected via a microphone and fed into an LM393 op-amp. It comprises an onboard potentiometer to adjust the setpoint for sound level.

Sound Detection Sensor Module Pin Configuration

VCC	The Vcc pin powers the module, typically with +5V
GND	Power Supply Ground
DO	Digital Output Pin. Directly connected to the digital pin of the Microcontroller
AO	Analog Output Pin. Directly connected to an analog pin of the Microcontroller

2. Arduino Uno



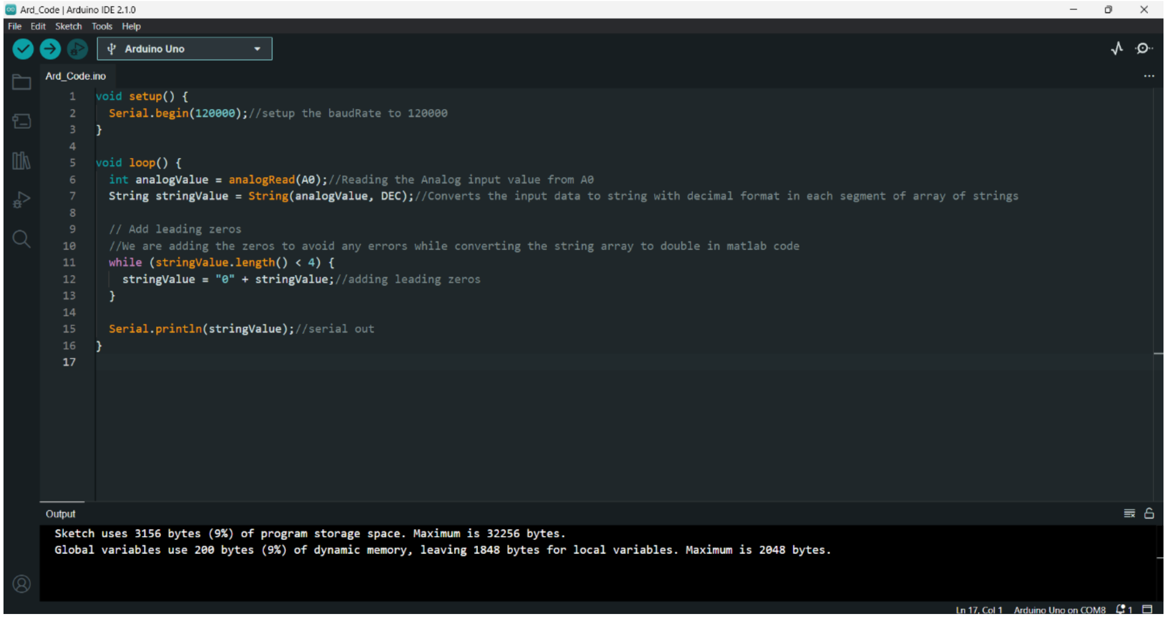


Arduino UNO is a microcontroller board based on the **ATmega328P**

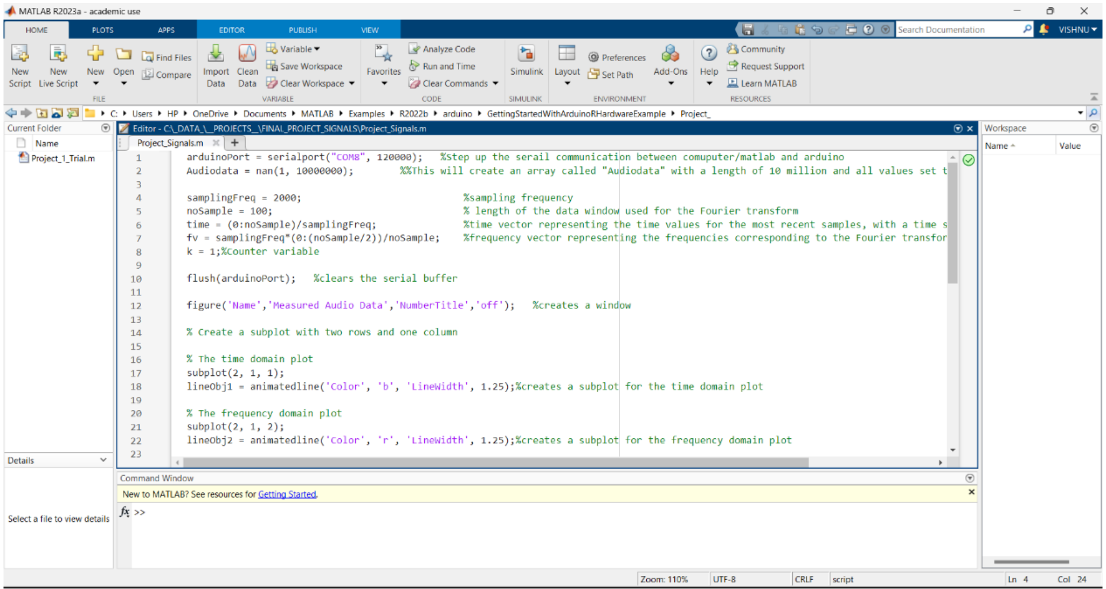
3. Jumper Wires
4. Arduino Cable

Tools

- Arduino IDE



- Matlab R2023a



Code

Arduino

```
void setup() {
  Serial.begin(120000);//setup the baudRate to 120000
}

void loop() {
  int analogValue = analogRead(A0);//Reading the Analog input value from A0
  String stringValue = String(analogValue, DEC);//Converts the input data to string with decimal format in each segment of array of strings

  // Add leading zeros
  //We are adding the zeros to avoid any errors while converting the string array to double in matlab code
  while (stringValue.length() < 4) {
    stringValue = "0" + stringValue;//adding leading zeros
  }

  Serial.println(stringValue);//serial out
}
```

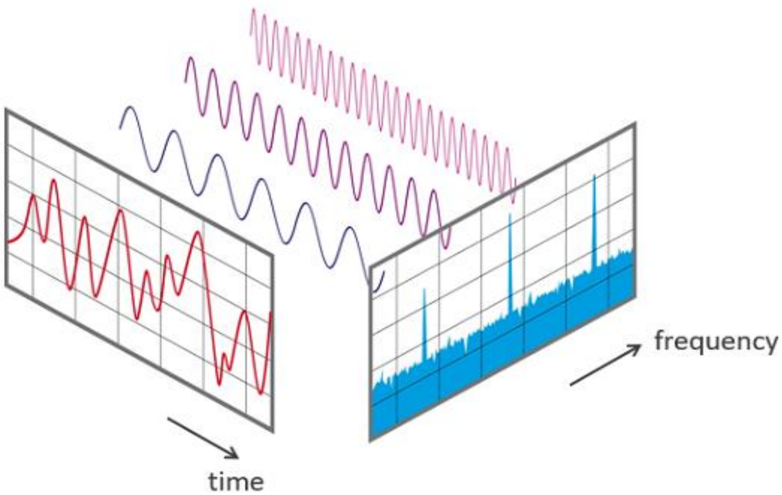
MATLAB

```
arduinoPort = serialport("COM8", 120000);   %Step up the serail communication between
comuputer/matlab and arduino
Audiodata = nan(1, 10000000);        %%This will create an array called "Audiodata"
with a length of 10 million and all values set to NaN.

samplingFreq = 2000;                           %sampling frequency
noSample = 100;                                % length of the data window used for
the Fourier transform
time = (0:noSample)/samplingFreq;              %time vector representing the time
values for the most recent samples, with a time step of 1/samplingFrequency
fv = samplingFreq*(0:(noSample/2))/noSample;    %frequency vector representing the
frequencies corresponding to the Fourier transform output
k = 1;%Counter variable

flush(arduinoPort);   %clears the serial buffer

figure('Name','Measured Audio Data','NumberTitle','off');   %creates a window

% Create a subplot with two rows and one column

% The time domain plot
subplot(2, 1, 1);
lineObj1 = animatedline('Color', 'b', 'LineWidth', 1.25);%creates a subplot for the
time domain plot

% The frequency domain plot
subplot(2, 1, 2);
lineObj2 = animatedline('Color', 'r', 'LineWidth', 1.25);%creates a subplot for the
frequency domain plot

duration = inf;% scalar representation of positive infinity.
tic;% records the current time
while toc < duration%check for elapsed time
    in = readline(arduinoPort); %reading one line of data from the serial port
    Audiodata(k) = str2double(in)*0.0049; %Converts the string data to a double and
stores it in the Audiodata vector, and scales it by a factor of 0.0049.
    if k > noSample && mod(k, 100) == 0%Checks in if enough the current sample data
        x = fft(Audiodata(k-noSample+1:k));%computes the FFT of the sample data
collected
        P2 = abs(x/noSample);%computes the two-sided amplitude spectrum of the FFT
        P1 = P2(1:noSample/2+1);%extracts the positive frequencies from the two-sided
spectrum.
        P1(2:end-1) = 2*P1(2:end-1);%doubles the amplitudes of the positive
frequencies

        % Update the time domain plot
        subplot(2, 1, 1);
        clearpoints(lineObj1);%clears all points from the animated line specified by
an for plot 1
        addpoints(lineObj1, time, Audiodata(k-noSample:k));%adds points defined by x
and y to the animated line specified by an for plot 2
        xlabel("Time (s)");%naming the x axis
        ylabel("Amplitude");%naming the y axis
```

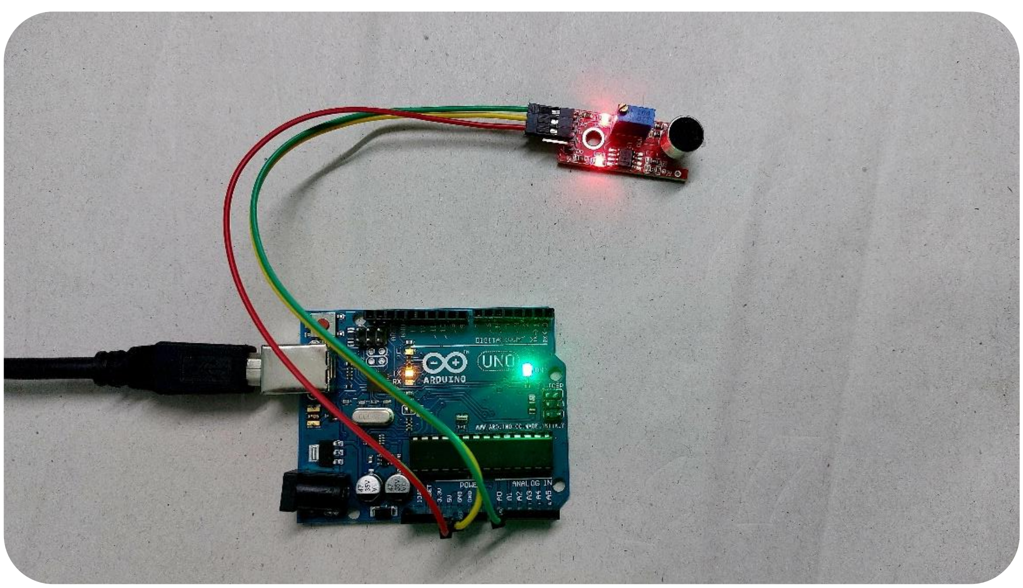
```
        % Update the frequency domain plot
        subplot(2, 1, 2);
        clearpoints(lineObj2);%clears all points from the animated line specified by
an for plot 2
        addpoints(lineObj2, fv, P1);%adds points defined by x and y to the animated
line specified by an for plot 2
        xlabel("Frequency (Hz)");%naming the x axis
        ylabel("Amplitude");%naming the y axis

        drawnow limitrate;%update the plot window
    end
    k = k + 1;%counter updation
end
```

I. Theory behind the implementation

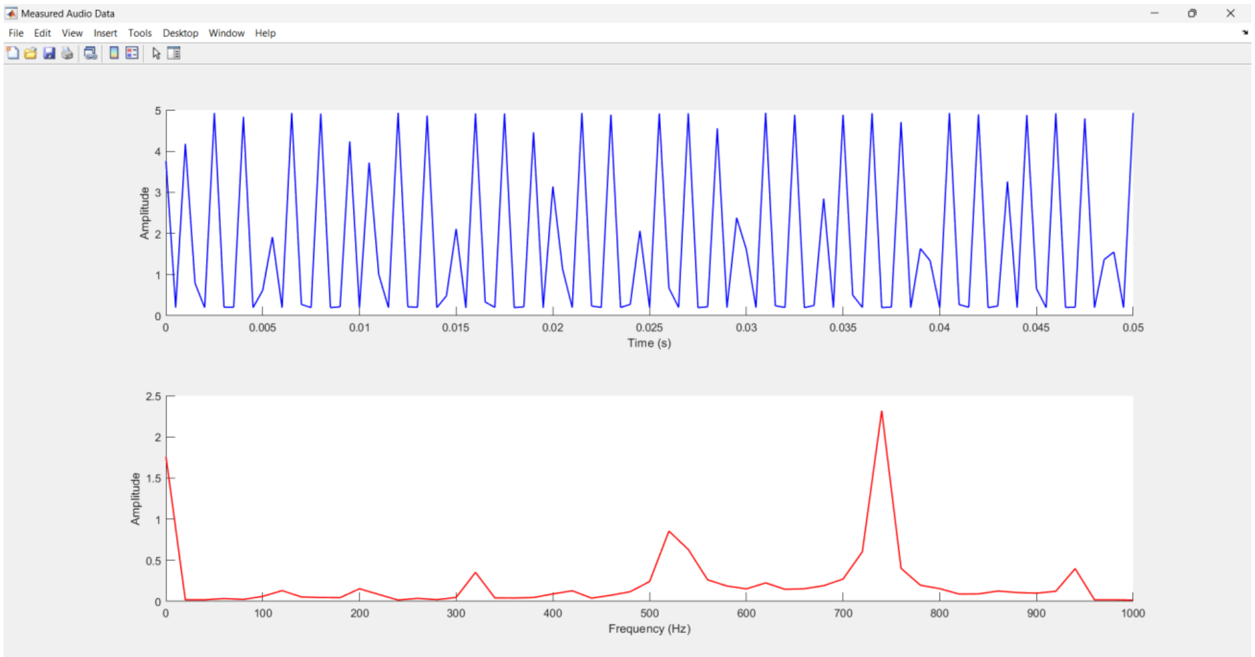
The amplitude vs frequency plot that we observe was made based on the theory of Fast Fourier Transform (FFT). It converts a signal into individual spectral components and thereby provides frequency information about the signal. The FFT is an optimized algorithm for the implementation of the Discrete Fourier Transformation (DFT). A signal is sampled over a period of time and divided into its frequency components. These components are single sinusoidal oscillations at distinct frequencies each with their own amplitude and phase.



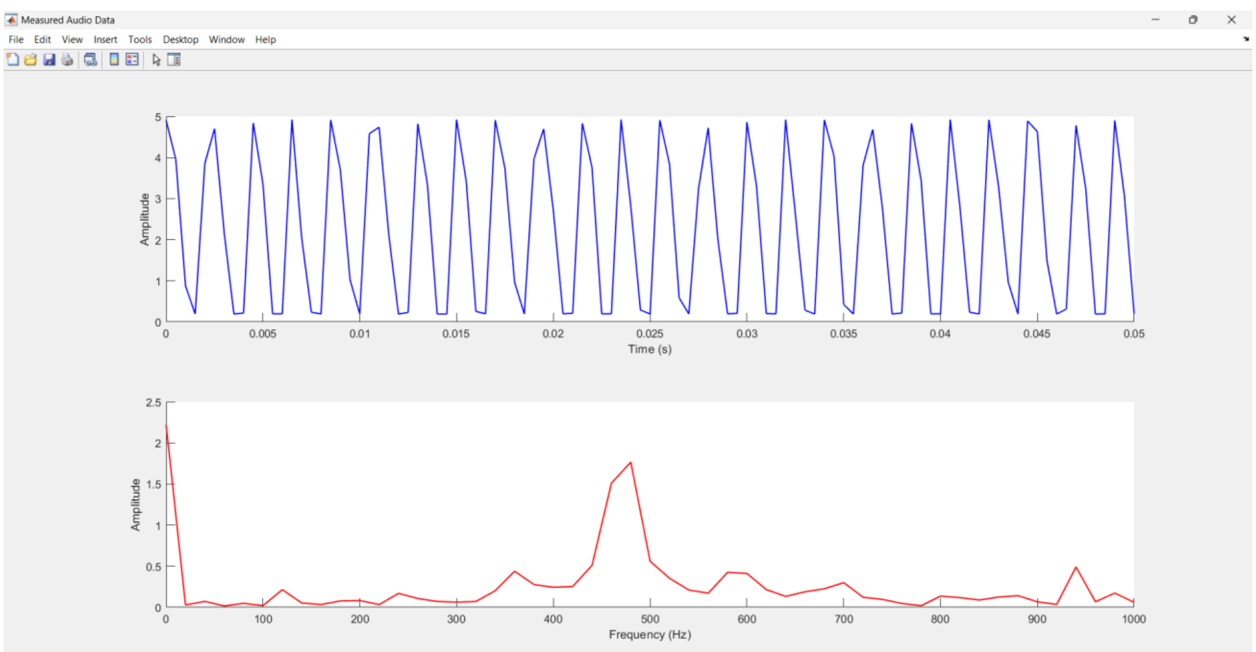
Circuit



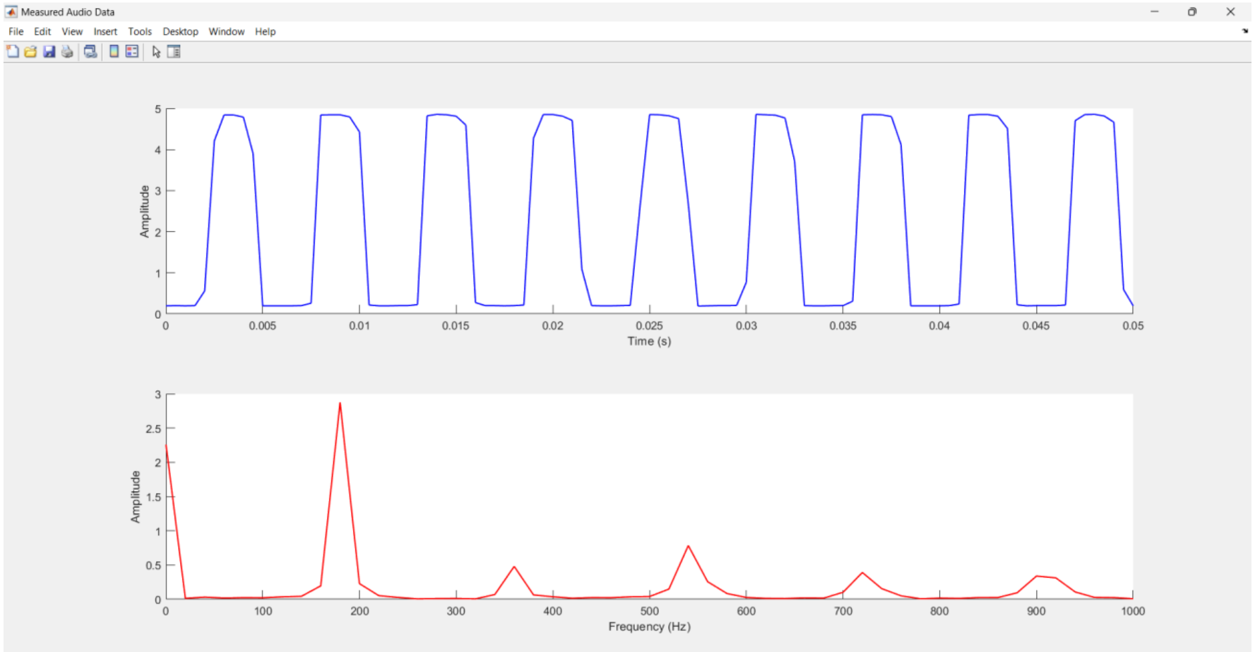
Results



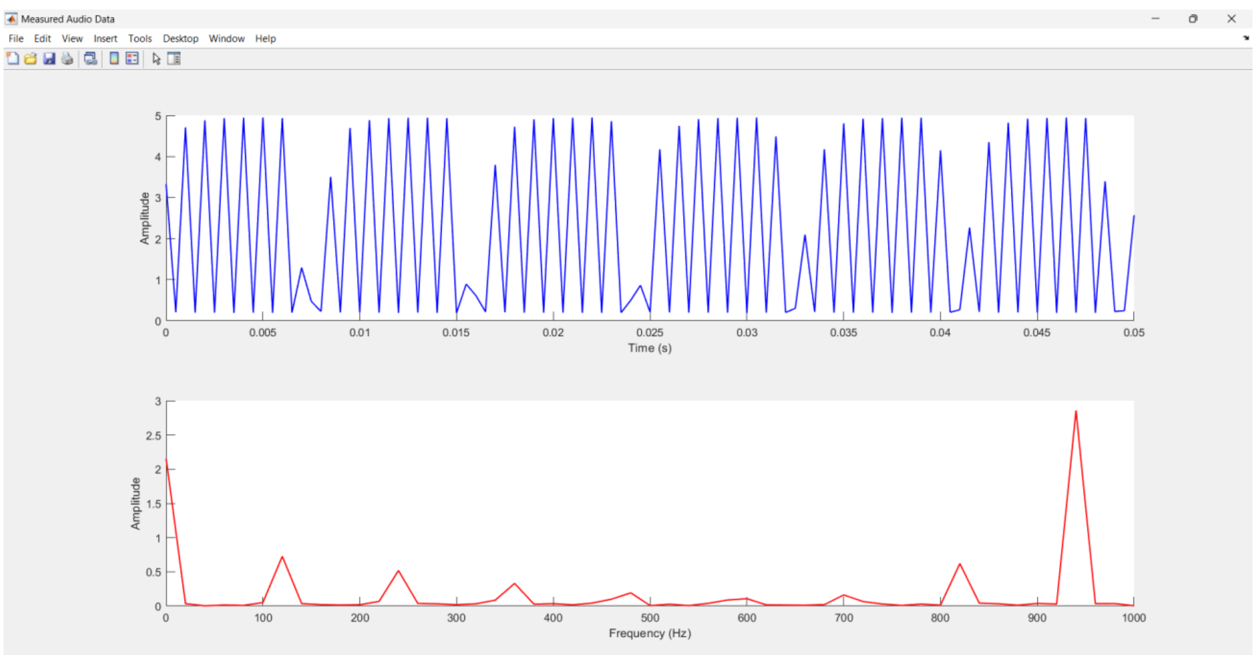
The input signal of Frequency 720Hz



Random audio



The input signal of Frequency 175Hz



The input signal of Frequency 920Hz

Conclusion

A device was developed that can capture audio signals making use of an Arduino board and a microphone. The Fourier transform of the audio signal was obtained using the code written in MATLAB. This output gives us the real-time amplitude vs. frequency graph. The quality of the recorded audio data could be improved by using more advanced microphones or by optimizing the electronic circuit used to connect the microphone to the Arduino board.