كلية الهندسة ـ جامعة الزقازيق

الفرقة: الثالثة هندسة الحاسبات والمنظومات

المقرر: دوائر الحاسب المتكاملة

الاسم: يمني محمد حافظ عطية حلاوة

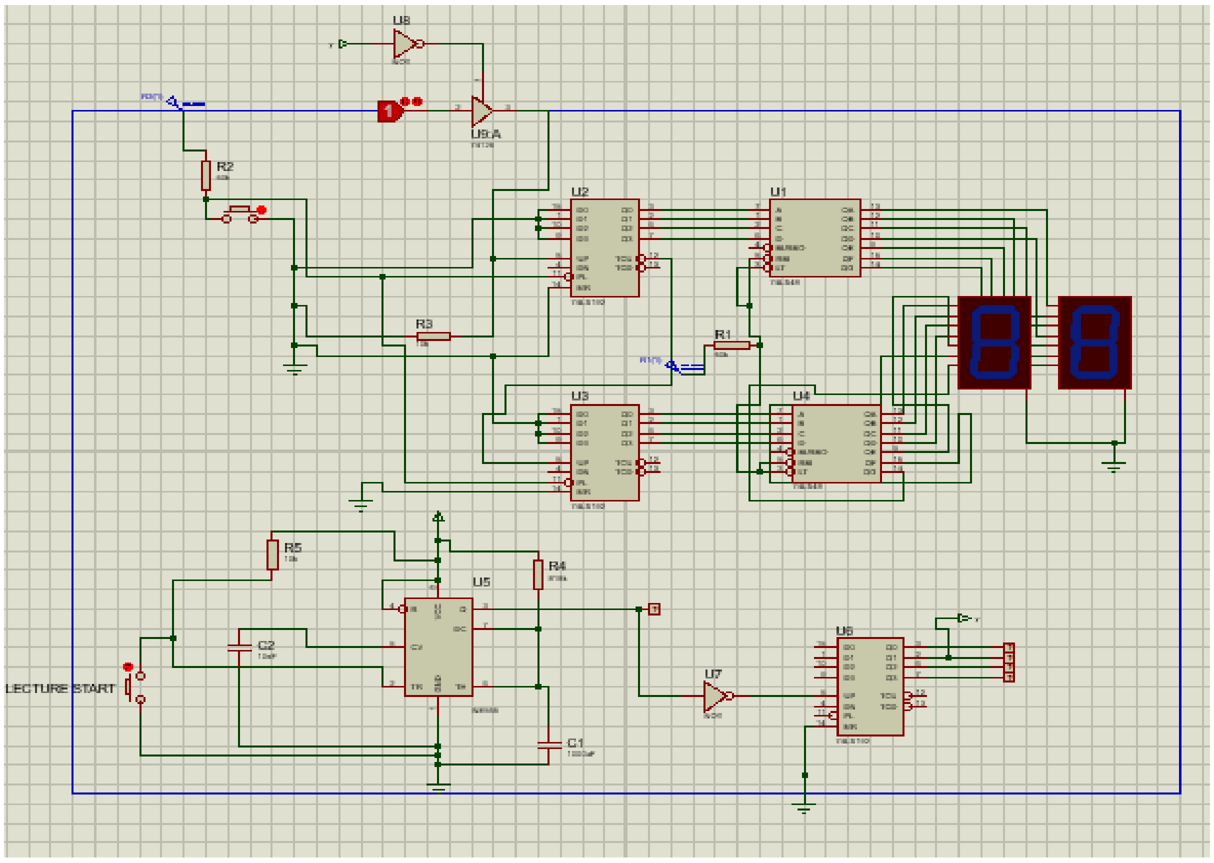
رقم الجلوس: 6951

رقم المشروع البحثي: 4

Part 1: -

In this part, classroom that counts the number of students entering the classroom, displays the count, and locks door after 15 minutes of lecture start time.... If I try to make the design as simple as possible for implementation

★ Design simulation on proteus: -



★ Explain design: -

This design consists of four parts:

- ❖ Enter the class hardness and count their number inside the class.
- ❖ Display the number of students on 7-segment.
- ❖ Timer counting for 15 minutes from the start of the lecture.
- ❖ Tie the end of the 15 minutes, close the door, and use the control on the class to lock it after the 15 minutes.

★ The components used in the classroom design:

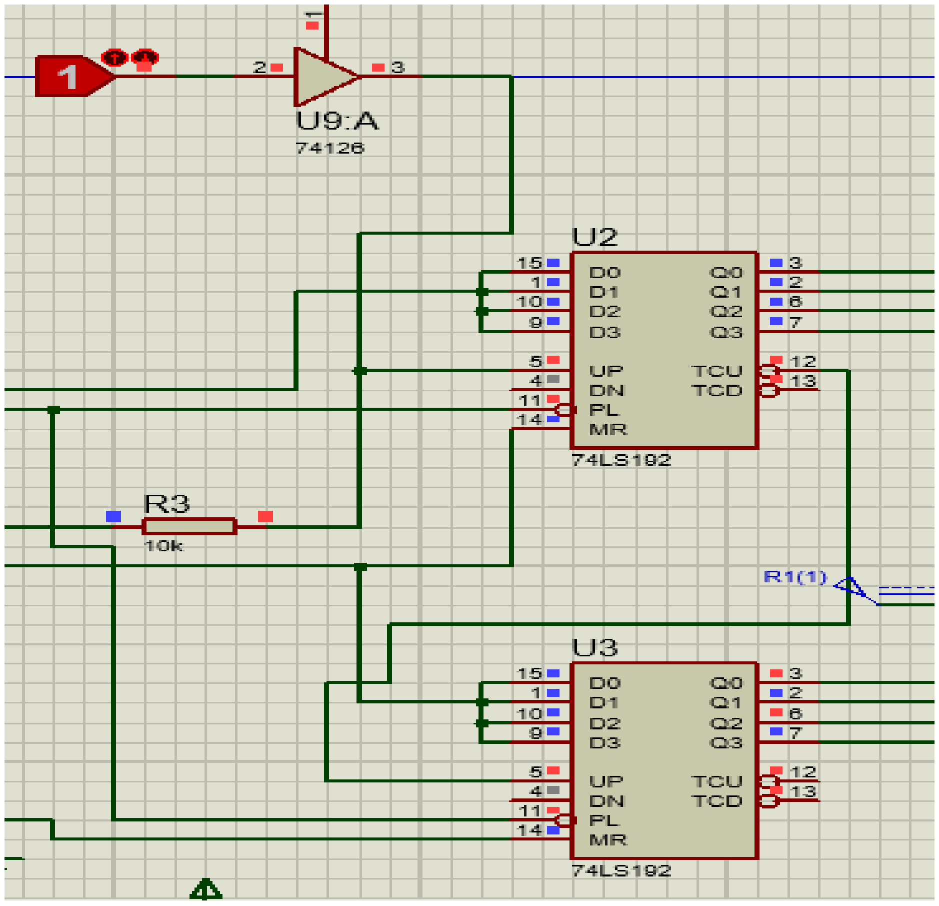
- ❖ 7SEG-COM-CAT-BLUE.
- ❖ 74126 (Quad Bus Buffer).
- ❖ 74LS48 (Bcd to 7-segment Decoder “common cathode”).
- ❖ 74LS192 (4 Bit up/down decade counter).
- ❖ NE555 (monostable “timer555”).
- ❖ Push button.
- ❖ Capacitor(1000uf)
- ❖ LOGICPROBE
- ❖ LOGIC STATE
- ❖ NOT
- ❖ RESISTORS (10k-819k-50k).

Design work:

1. Counterpart (74LS192):

I needed in this design to search for an up-counter that gives me the highest number at the end, and this number makes sense for any classroom that will accommodate this number and at the end I reached a counter that amounts to a value of up to 99 people in the classroom for that In this part I have used A synchronous 4-bit up/down decade counter symbol (74LS192) . In this design I have benefited from the up-counterpart.

Each circuit contains four master/slave flip-flops, with internal gating and steering logic to provide master reset, individual preset, count up and count down operations. Each flip-flop contains JK feedback from slave to master such that a LOW-to-HIGH transition on its T input causes the slave, and thus the Q output to change state. Synchronous switching, as opposed to ripple counting, is achieved by driving the steering gates of all stages from a common Count Up line and a common Count Down line, thereby causing all state changes to be initiated simultaneously. A LOW-to-HIGH transition on the Count Up input will advance the count by one; a similar transition on the Count Down input will decrease the count by one. While counting with one clock input, the other should be held HIGH. Otherwise, the circuit will either count by twos or not at all, depending on the state of the first flip-flop, which cannot toggle as long as either Clock input is LOW. The Terminal Count Up (TCU) is normally HIGH. When a circuit has reached the maximum count state (9 for the LS192, the next HIGH-to-LOW transition of the Count Up Clock will cause TCU to go LOW. TCU will stay LOW until CPU goes HIGH again, thus effectively repeating the Count Up Clock, but delayed by two gate delays. Since the TC outputs repeat the clock waveforms, they can be used as the clock input signals to the next higher order circuit in a multistage counter. Each circuit has an asynchronous parallel load capability permitting the counter to be preset. When the Parallel Load (PL) and the Master Reset (MR) inputs are LOW, information present on the Parallel Data inputs (P0, P3) is loaded into the counter and appears on the outputs regardless of the conditions of the clock inputs. A HIGH signal on the Master Reset input will disable the preset gates, override both Clock inputs, and latch each Q output in the LOW state. If one of the Clock inputs is LOW during and after a reset or load operation, the next LOW-to-HIGH transition of that Clock will be interpreted as a legitimate signal and will be counted.



(1)

As shown in Figure 1, I have used two of 74LS192 to count from 0 to 99 people entering the classroom. The process of increasing the number is in logic state, which will be used by who will enter the class The change from zero to 1 that we produce the logic state will be the clock entering the up-counter.

With every change in the Logic State (i.e. with the members of the classroom entering), the clock changes up-counter, until the 9th count returns to the Reset state. In the case of the first up-counter count, the TCU pin in the High State will switch to low, and the up-counter count will start.

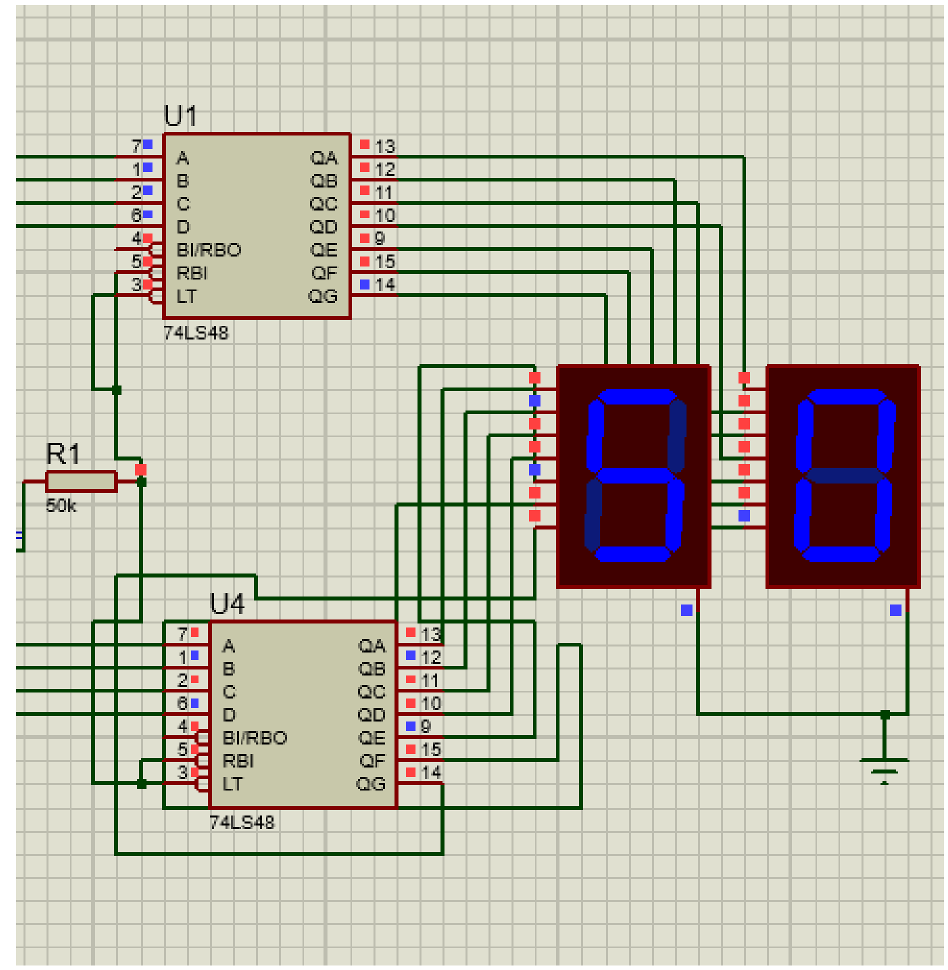
MODE SELECT TABLE:

MR	PL`	CPU	MODE
1	d	d	Reset
0	0	d	preset
0	1	1	No change
0	1	Clock o-->1	Count up

Truth table of 74ls192: -

Input pulses	Q1	Q2	Q3	Q4
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

2-displaying part on two 7-segment: -



(2)

As shown in Figure (2), I have used it to display students who want to enter the classroom two of 7-segment and two of 74LS48 (Bcd to 7-segment decoder).

- The output of up counter will enter to 74LS48 (Bcd to 7-segment decoder) as input.

Truth table of 74LS48 (Bcd to 7-segment decoder): -

Q1	Q2	Q3	Q4	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

- Equations from Karnaugh map: -

For a = Let d = don’t care = 1 (a) = Q3 +Q1 +Q2Q4+Q2`Q4`

	Q1`	Q1`	Q1	Q1	
Q3`	1	0	d	1	Q4`
Q3`	0	1	d	1	Q4
Q3	1	1	d	d	Q4
Q3	1	1	d	d	Q4`
	Q2	Q2`	Q2`	Q2	

For b = Q3`Q4` +Q3Q4 +Q2` +Q1

	Q1`	Q1`	Q1	Q1	
Q3`	1	1	d	1	Q4`
Q3`	1	0	d	1	Q4
Q3	1	1	d	d	Q4
Q3	1	0	d	d	Q4`
	Q2	Q2`	Q2`	Q2	

For c = Q3` +Q4 +Q2 +Q1

	Q1`	Q1`	Q1	Q1	
Q3`	1	1	1	1	Q4`
Q3`	1	1	1	1	Q4
Q3	1	1	1	1	Q4
Q3	0	1	1	1	Q4`
	Q2	Q2`	Q2`	Q2	

For d = Q1` +Q3Q4` +Q2`Q3 +Q2`Q4` +Q2Q3`Q4

	Q1`	Q1`	Q1	Q1	
Q3`	1	0	1	1	Q4`
Q3`	0	1	1	1	Q4
Q3	1	0	1	1	Q4
Q3	1	1	1	1	Q4`
	Q2	Q2`	Q2`	Q2	

For e = Q2`Q4` +Q3Q4` +Q1Q2 +Q1Q3

	Q1`	Q1`	Q1	Q1	
Q3`	1	0	1	0	Q4`
Q3`	0	0	1	1	Q4
Q3	0	0	1	1	Q4
Q3	1	1	1	1	Q4`
	Q2	Q2`	Q2`	Q2	

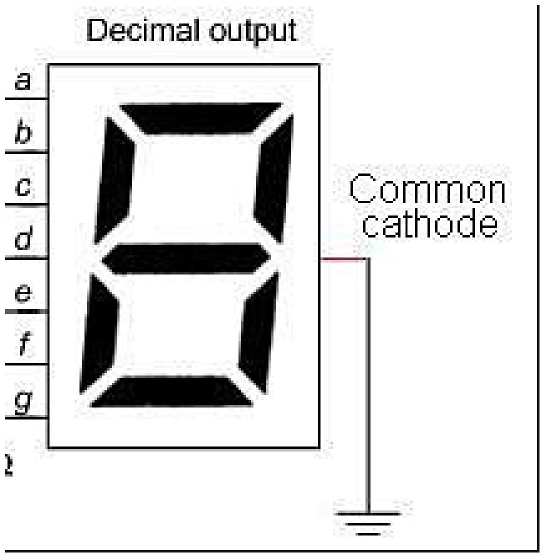
For f = Q3`Q4` +Q2Q3` +Q1 +Q2Q4`

	Q1`	Q1`	Q1	Q1	
Q3`	1	1	1	1	Q4`
Q3`	0	1	1	1	Q4
Q3	0	0	1	1	Q4
Q3	0	1	1	1	Q4`
	Q2	Q2`	Q2`	Q2	

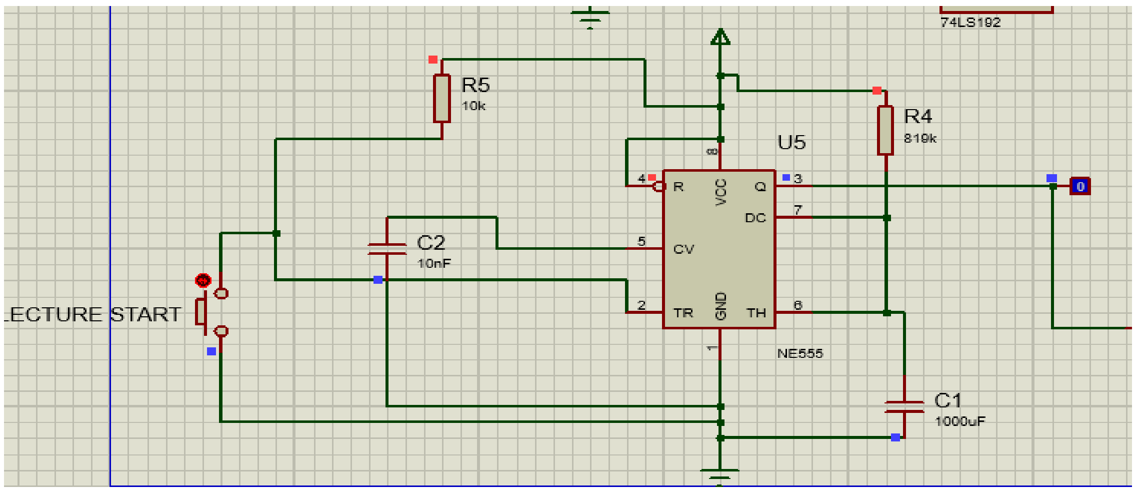
For g = Q1 + Q2Q3` +Q2`Q3 +Q2Q4`

	Q1`	Q1`	Q1	Q1	
Q3`	0	1	1	1	Q4`
Q3`	0	1	1	1	Q4
Q3	1	0	1	1	Q4
Q3	1	1	1	1	Q4`
	Q2	Q2`	Q2`	Q2	

The output of 74LS48 (Bcd to 7-segment decoder) will be input of 7-segment and display the count.

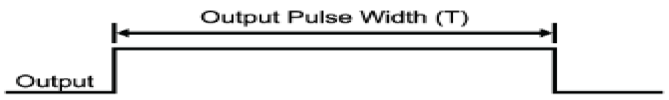


3- timer for 15 minute (NE555): -



(3)

In this part as shown in Figure (3) at the beginning to enter the lecture in order for the timer to start after 15 minutes, he has to press the connected button on pin 2. the output voltage becomes high for a set duration (T) when a falling edge is detected on pin 2 (trigger). The circuit above is also called a This calculator is designed to compute for the output pulse width of a 555-timer monostable circuit.



To make the output pulse Width(T) equal to 15 minutes I used this equation: -

$$T = 1.1 * R*C$$

Let T = 15 minutes = 15 * 60 = 900 seconds.

And C = 1000 uf.

✓ Then R will equal to 819k

When lecture start and press on button it will start count 15 minutes and logic indicator will change from 1 to 0 that means the duration end.

❖ **Truth table: -**

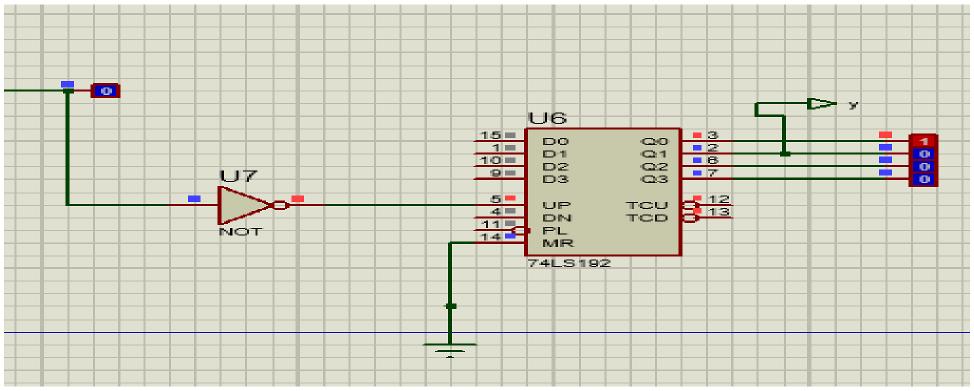
S	R	Q	Q`
0	0	Not change	Not change
0	1	0	1
1	0	1	0
1	1	Undefined	Undefined

Q = S R`

4-controle part: -

In this part, to control the closing of the classroom door 15 minutes after the start of the lecture, I used two parts.

the first part: -



1-As shown in the figure, the first part of the control depends on the output of the timer. Logic indicator present this state:

0	The lecture has not started yet
1	The lecture started and Timer began counting the 15 minutes
0	Timer finished counting the 15 minutes

2- then I take the output of Logic indicator and enter it to Not gate.

3- the output of Not gate enter to up-counter where up counter still counts 1 while the lecture has not started and count two when the lecture starts and Timer finished counting the 15 minutes.

How it counts two the Timer finished counting the 15 minutes??

I have made the up-counter count the number of impressions. 1 came out of the Not gate, if the number of ones is two, this means that the timer concluded the 15 minutes.

❖ **Truth table: -**

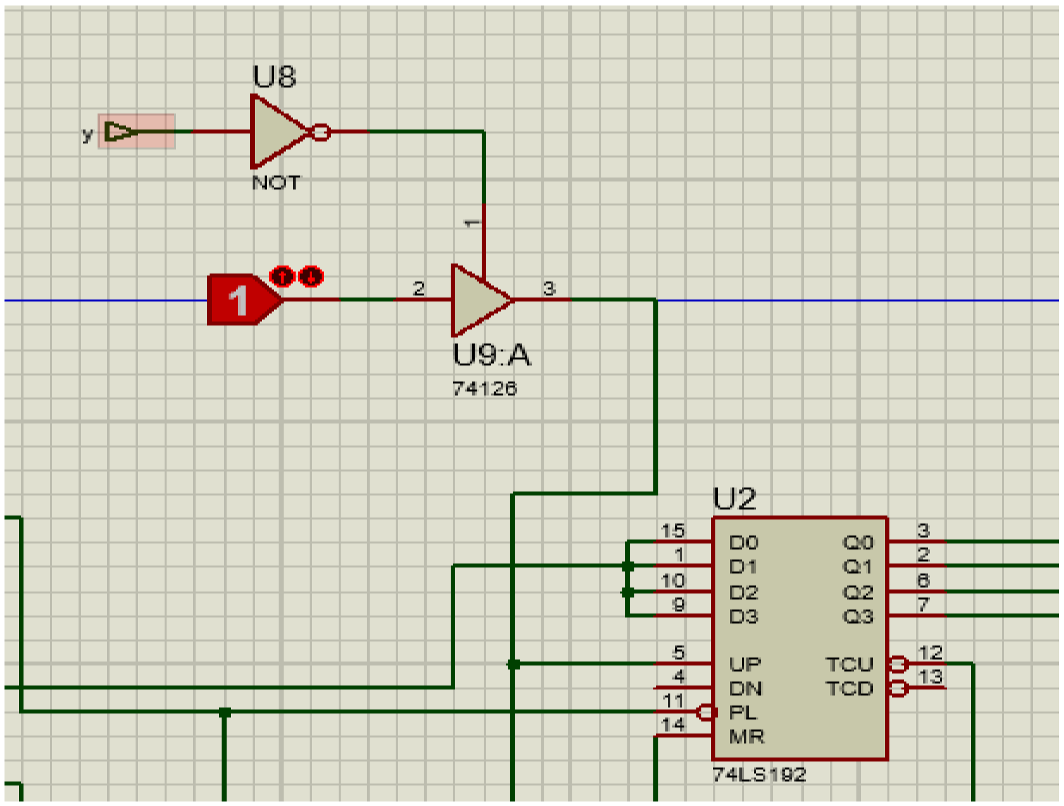
Not gate output/ up-counter input	Q0	Q1	Q2	Q3
0(1 to 0)	0	0	0	1
1(0 TO 1)	0	0	1	0
Dot care	d	d	d	d

☺ Second part of control: -

In order to control the closing of the door, i.e., no matter how the state of the logic changes, after 15 minutes have passed since the start of the lecture, the up-counter does not count the number and only displays the current number within the classroom.so I used to be 74126 as controller.

y in the figure (4), when the count is finished and is in a high state. For each stop count after 15 minutes have passed, the enable (y) of tristate buffer equal 1 so, to make the output in High impedance state so I used Not gate to make(y) equal to zero

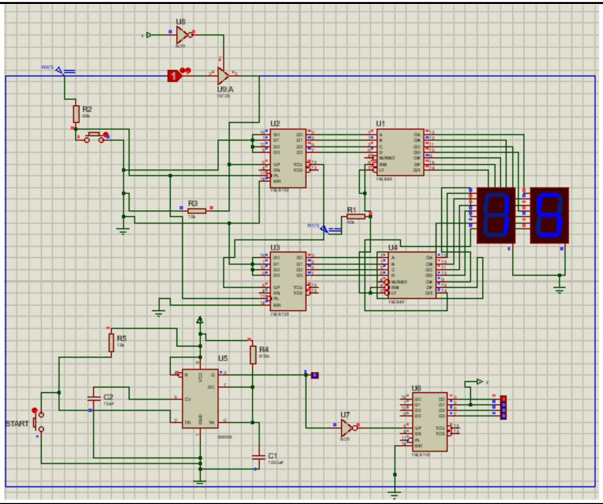
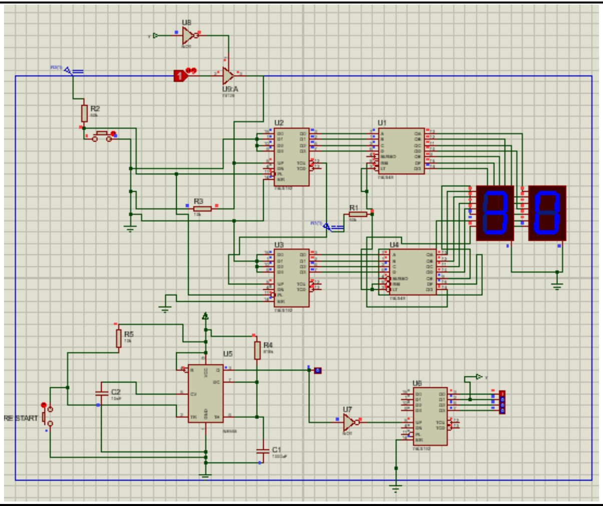
so, the up-counter stop counts and display the current number in classroom.

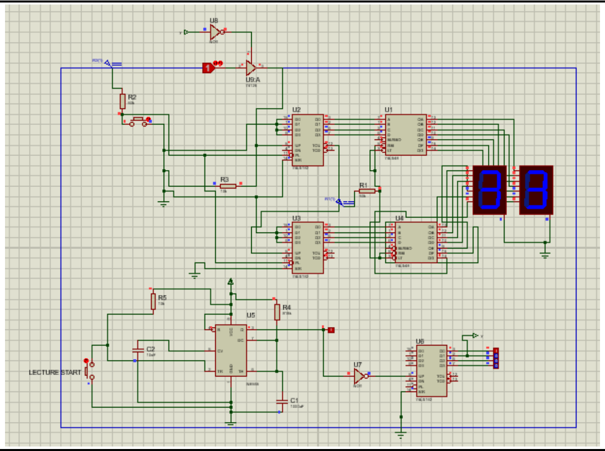
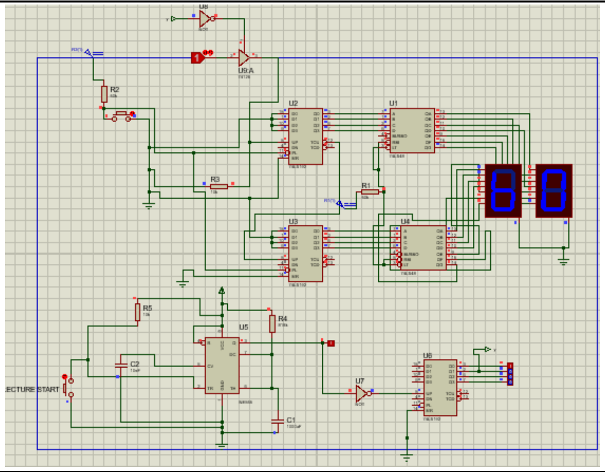
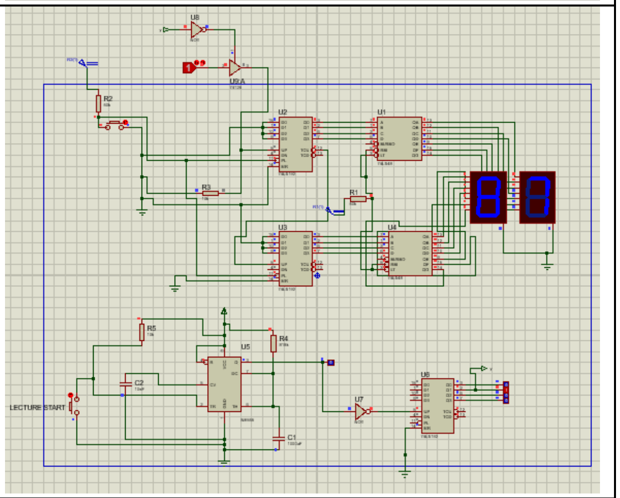


Tri-state Buffer: -

(y) /enable	Clock (logic state)	output
0	d	High impedance(Z)
0	d	High impedance(Z)
1	0 to 1	1

★ 5examples of working: -

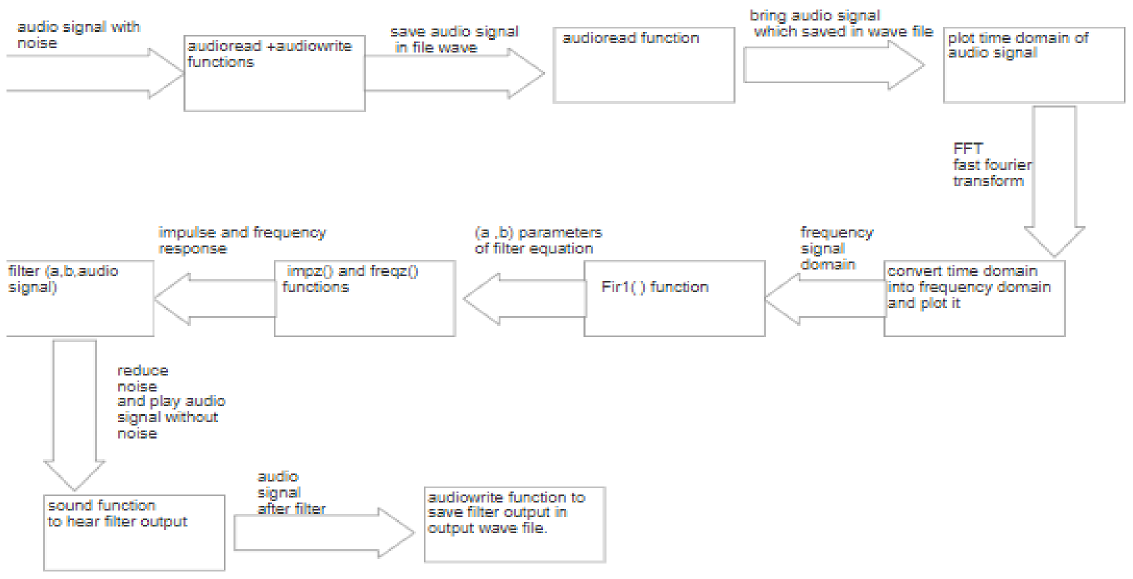
comment	picture
15 students enter the classroom before the start of the lecture.	
30students enter the classroom before the start of the lecture.	

The lecture started and Timer began to count the 15 minutes and students entered during the count	
Timer is still counting the 15 minutes from the beginning of the lecture and the number of students in the classroom has reached 60	
The 15 minutes have passed, and despite the change in the Logic State, the increase in the count on the number has been stopped within the classroom and the current number has been displayed.	

Part 2: -

In this part, record my voice with noise in background for 10 sec and designing filter to attenuate the noise and enhance the recording. And change level of noise and see filter perform.

★ Block diagram


audio signal with noise
audioread +audiowrite functions
save audio signal in file wave
audioread function
bring audio signal which saved in wave file
plot time domain of audio signal
FFT fast fourier transform
convert time domain into frequency domain and plot it
frequency signal domain
Fir1( ) function
(a ,b) parameters of filter equation
impz() and freqz() functions
impulse and frequency response
filter (a,b,audio signal)
reduce noise and play audio signal without noise
sound function to hear filter output
audio signal after filter
audiowrite function to save filter output in output wave file.

❑ **This part consists of four main parts: -**

→ Recording my voice with noise.

→ Designing filter to reduce noise.

→ Display the output of filter.

→ Test filter output by changing noise level.

B) explain my solution in details: -

★ Recording my voice with peep noise I n background for 10 sec using microphone.

Steps: -

1- sampling frequency (Fs)

Fs defines the number of samples per second taken from a continuous signal to take a discrete or digital signal. Usual values for the sampling frequency are 44100 Hz .so I choose sampling frequency equal 44100 with recording duration of 10 (sec) so it will give me 441000 samples.

2-audiorecorder and recObj

Use an `audiorecorder` object to record audio data from a microphone. he `audiorecorder` object contains properties that enable additional flexibility during recording.

RecObj = `audiorecorder` creates and returns an `audiorecorder` object with these properties:

- Sampling frequency `Fs` = 44100 hertz
- Bits per sample `bits` = 16
- Number of channels = 1

3- recordblocking and getaudiodata

`recordblocking(`*`recObj`*`, time)` records audio from a microphone for the number of seconds specified by time. The `recordblocking` method does not return control until recording completes. recObj is an `audiorecorder` object that defines the sample rate, bit depth, and other properties of the recording. getaudiodata Returns the recorded audio data to the MATLAB workspace save it name as signal.

4- play ,length,linspace and audiowrite

Play(recObj) : -Creates an audio player, plays the recorded audio data. Length(recobj) returns the length of recobj and I saved it variable (samples). It is equivalent to the command max(size(recobj)).

Linspace (0, time, samples): - Return a row vector with samples linearly spaced elements between 0 and time. And saved the return value in (t) variable.

Audiowrite(‘mytast.wav, signal, Fs): - writes a matrix of audio data, signal, with sample rate Fs to a file called(mytask.wav). The filename input also specifies the output file format. The output data type depends on the output file format and the data type of the audio data, signal. And that is wave audio signal.

Code used: -

```
% record voice from microphone with beep noise
fs = 44100;  %sampling frquency
time =10; %10 seconds duration of record
recObj = audiorecorder(fs,16,1); %object of entering data in 16 bits per samples and mono
recordblocking(recObj,time); %stop record after 10 seconds
signal = getaudiodata(recObj);%save sound data
play(recObj); % to here the record
samples =length(signal); % max size of signal
t = linspace(0,time,samples);%dicretize time(Return a row vector with samples )
audiowrite('mytask.wav',signal,fs)% save sound into file
```

★ Filter design

Steps: -

1-audioread, play signal and plot in time domain

[signal,fs]=audioread('mytask.wav'): - audioread(mytask.wav) reads data from the file named (mytask), and returns sampled data, signal, and a sample rate for that data, Fs.

sound(signal,fs,16): -plays the audio(signal) with a bit depth of 16 bits per sample.

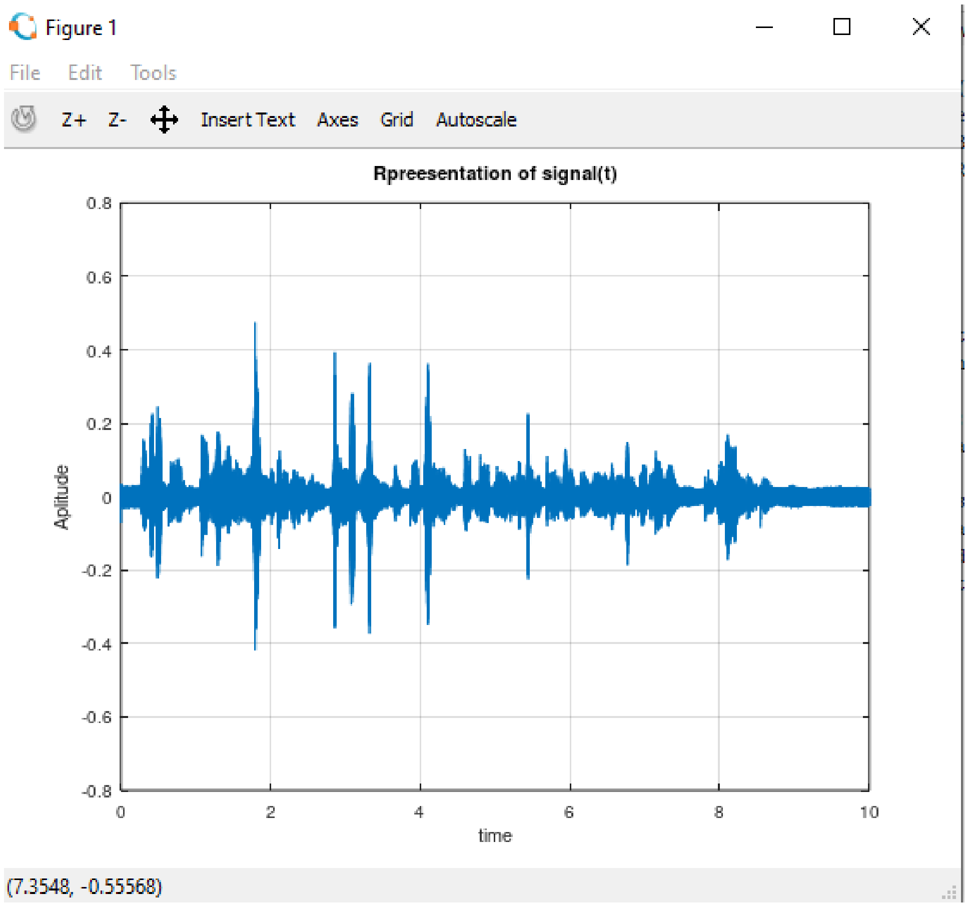
signal = signal (:1): - signal displaying increasing by one rate along axis.

samples =length(signal): - determine the max size of audio signal.

t = (0: samples-1)/fs: - determine the number of discrete samples which start from 0 to last number of samples –1.

Plot in time domain using function plot and give it the length of samples in signal and signal (signal).

Signal plot in time domain: -



In this graph its present distribution of my voice with noise in time domain (my recording by amplitude and time).

2- convert from time domain to frequency domain and plot frequency domain: -

This step to see noise frequencies that I want to remove and frequencies to move from filter.

Convert time domain to frequency domain using Fourier transform: -

The Fourier transform is a mathematical formula that relates a signal sampled in time or space to the same signal sampled in frequency. In signal processing, the Fourier transform can reveal important characteristics of a signal, namely, its frequency components.

The Fourier transform is defined for a vector x (signal) with n (samples) uniformly sampled points by

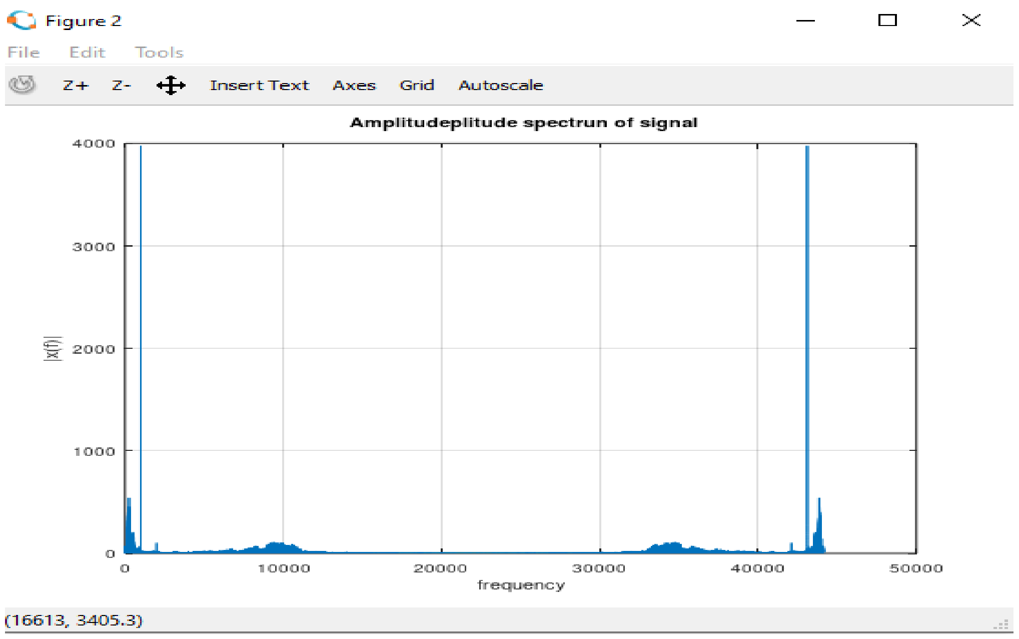
$$y_{k+1} = \sum_{j=0}^{n-1} \omega^{jk} x_{j+1}.$$

W = $e^{((-2\pi i)/n))}$ is one of n complex roots of unity where i is the imaginary unit. For x and y, the indices j and k range from 0 to n−1.

The fft function in MATLAB uses a fast Fourier transform algorithm to compute the Fourier transform of data.

signal_k = abs(fft(signal)) : - in this part I took Fourier transform for my recording then the output off fft() function is complex-valued, so i want to plot the magnitudes so I used (abs()) function and saved return value in (signal_k).

Plot frequency domain: -



As shown in the figure bellow: - the magnitude of the signal as a function of frequency, the spikes in magnitude correspond to the signal's frequency components of 944.7 (hz).so I will design filter to remove these spikes. The transform also produces a mirror copy of the spikes, which correspond to the signal's negative frequencies.

Code used: -

```
%reading frequencies from the plot
signal_k = abs(fft(signal)); %to calculate absolute value of fft
f = linspace(0,fs,samples); %discretize frequency
% plot Amplitudeplitude spectrum of signal
figure(2);plot(f,signal_k)
grid on
xlabel('frequency')
ylabel('|x(f)|')
title(' Amplitudeplitude spectrun of signal')
```

3- filter design: -

According of figure (2), the filter I designed is FIR Bandstop filter to reduce noise and pass my voice. FIR (finite impulse response or non-recursion) filter has all (a_k) expect a0 equal to zero, it is weighted sum of inputs samples. Forming a weighted sum of the past input and output samples: -

$$a_0 y(n) = \sum_{k=0}^{N} b_k x(n-k) - \sum_{k=1}^{M} a_k y(n-k)$$

Where a represent X variables and b represent Y variables. Any filter depends on values of a and b.

❖ Order: - Order of the filter means the maximum number of delay elements used in the filter circuit. This can be easily observed by observing the difference equation and finding out at which sample we are having our maximum delay. represent the filter order .as filter order bigger as filter performed enhance so I put the filter order of my design equal to 100.

❖ Frequencies: - lowpass frequency, high pass frequency and cutoff frequency

Depending on figure (2),

I put FL=200 (HZ), FH =300(HZ) so FC = $\sqrt{F_1 F_H}$ =866(HZ).

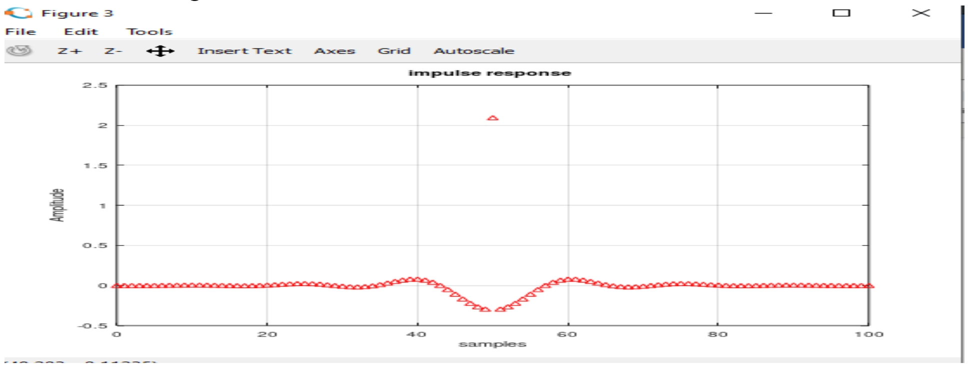
❖ A and b parameters: - depending on FIR we have Y parameter in one side so (a) equal to 1. X Parameter (b) to find(b) values of filter I used fir1() function.

b = fir1(order, [200 3000]/(fs/2), 'stop'): - this code calculate best b coefficient for FIR bandstop filter. It will take filter order, and [200 3000]/(fs/2) which is argument expresses the *normalized* frequencies for the stopbands. That means that for a discrete filter, these are already on the interval (0, pi), where pi is defined as the Nyquist frequency (half the sampling frequency). So that information is inherent in the argument. At end it will take the type of filter which is stop.

❖ Plot impulse response of (a,b) filter parameter: -

impz(b,a) :- with no output arguments plots the impulse response of the filter.

As shown in figure: -



The range of pass frequency from almost 20 to 80 HZ and other frequency almost equal to 0

❖ Plot frequency response of fir1(a, b): -

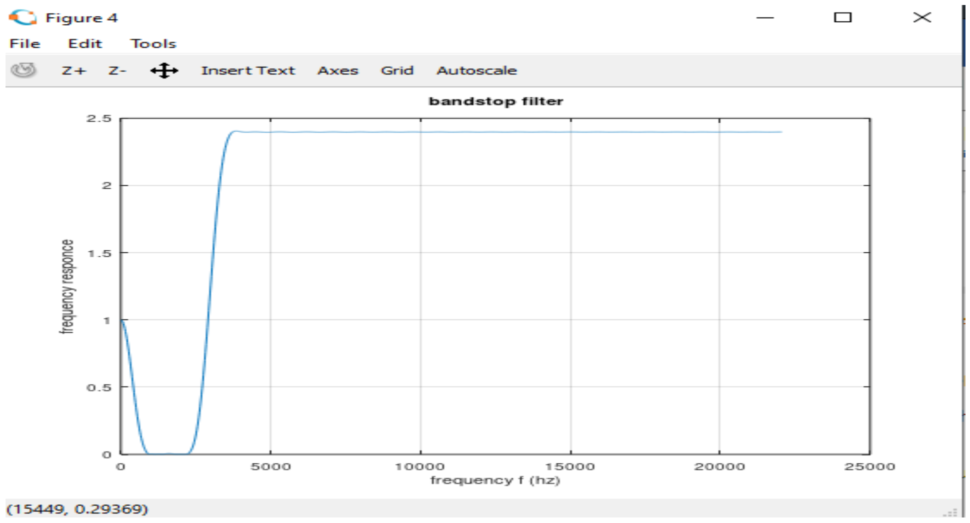
H = freqz(b,a,f,fs): - freqz is used to get or plot the frequency response of a digital system.

And in this code, I used it to get the frequency response of the filter.

To design the filter i have to first create the transfer function of the filter. Then pass the numerator and de-numerator to freqz().

Because i have the transfer function create polynomials of numerator and denominator and apply them to freqz().

This figure show frequency response: -



Filter reject the frequencies from 194 to 2877 (hz) by this it reduces the noise frequency.

- At end using filter function: - The `filter` function filters a data sequence using a digital filter which works for both real and complex inputs.

output =filter(b,a,signal):- returns the final conditions, `zf`, of the filter delays.at end I used sound function to hear output recording.

Used audiowrite function to save output voice into wave file.

Code used: -

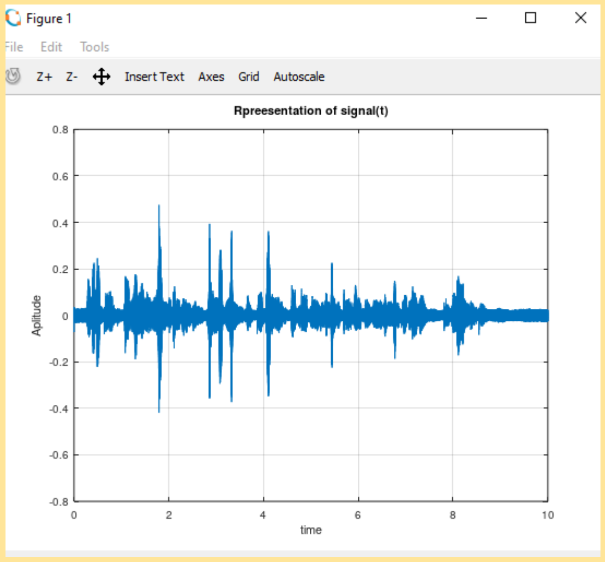
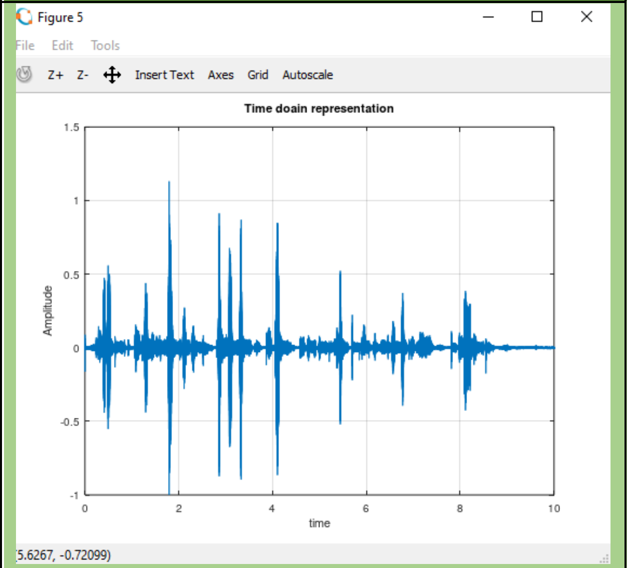
```
% designing bandstop FIR fiter
order =100; % fiter order
fc = 866;  %cutoff frquency of filter
a = 1;      %Y factor in the filter equation
b = fir1(order, [200  3000]/(fs/2), 'stop');%X factor in the filter equation

% plot impulse response
figure(3);impz(b,a),grid; % get samples and plot it
xlabel('samples')
ylabel('Amplitude')
title('impulse response ')
% compute and display frequency response
f = (0:.001:1)*fs/2;
H = freqz(b,a,f,fs);
figure(4);plot(f,abs(H)),grid;
```

```
xlabel('frequency f (hz)')
ylabel('frequency responce')
title('bandstop filter')
output =filter(b,a,signal);%function that uses for filtring the recording
sound(output,fs) %converting output data to sound

%saving the output in wave files
audiowrite('out1.wav',output,fs)% save sound into file
```

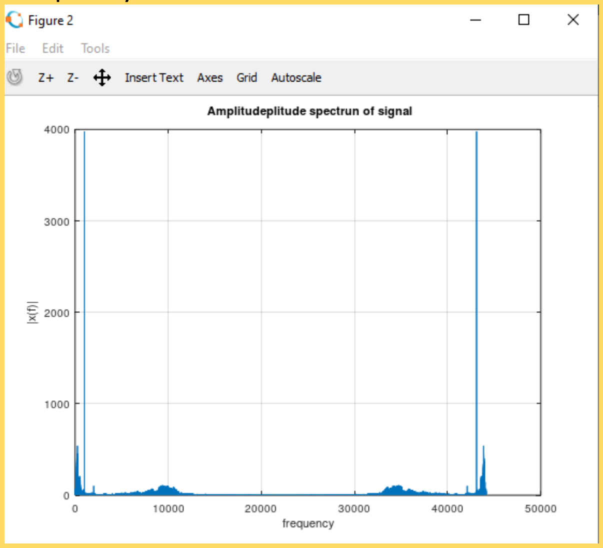
★ Display the result

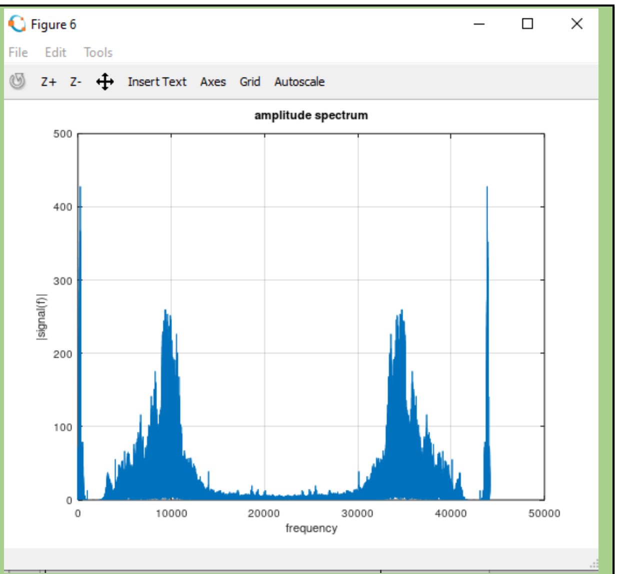
Code used: -

```
%plot Time doain representation
figure(5);plot(t,output),grid;
xlabel('time')
ylabel('Amplitude')
title('Time doain representation')
output_k = abs(fft(output));%to calculate absolute value of fft
f =linspace(0,fs,samples);%discretize frquency
%plot amplitude spectrum
figure(6);plot(f,output_k),grid;
title('amplitude spectrum')
xlabel('frequency')
ylabel('|signal(f)|')
```

Recording with noise	Filtering recording
Time domain 	 Some of amplitude values in this figure decreased its amplitude value for example at time = .74 A= .1 after filter its value become = .07

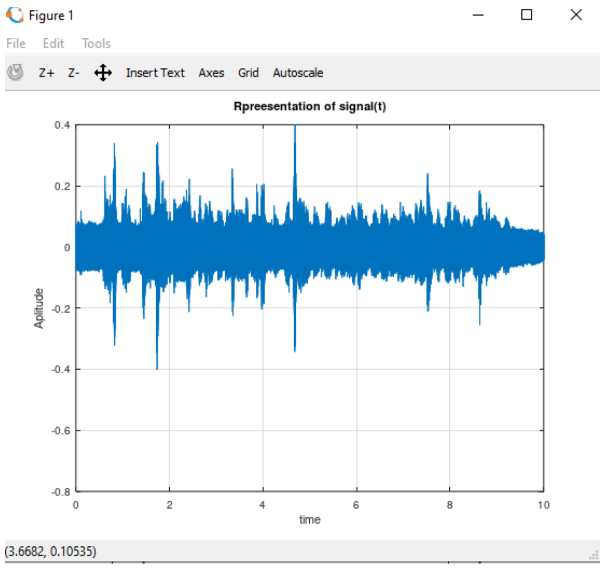
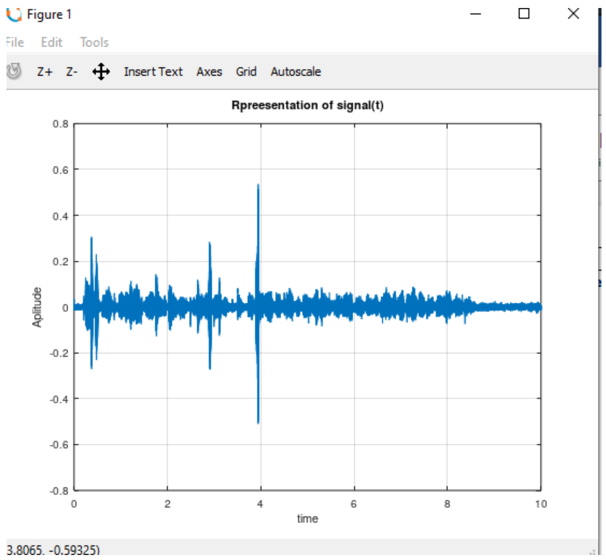
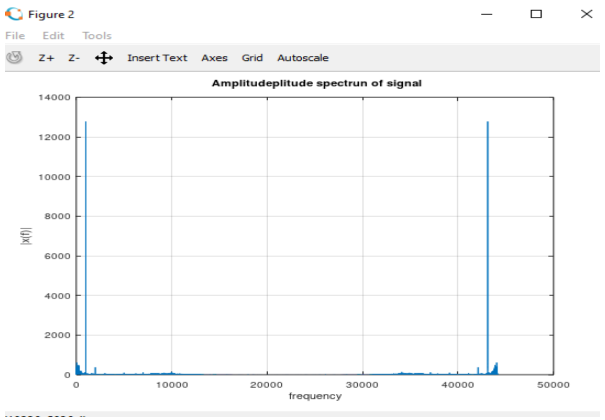
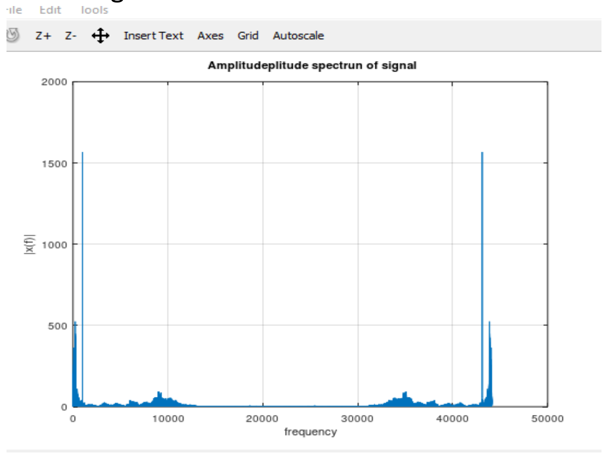
Frequency domain



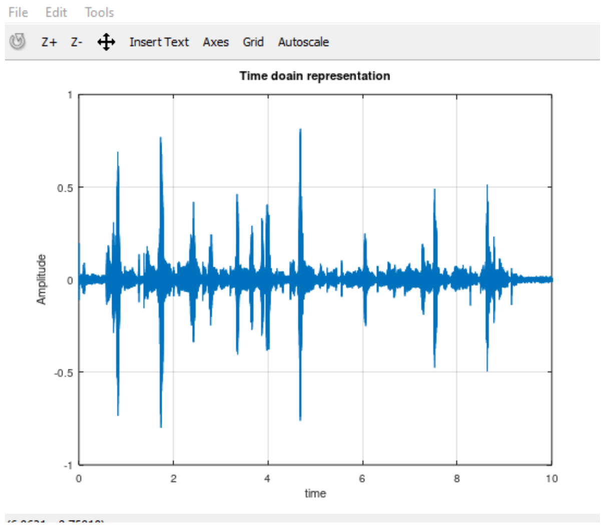


Reduce the noise at 944 frequency point from3980 to 14.7 so it reduces the noise frequency.

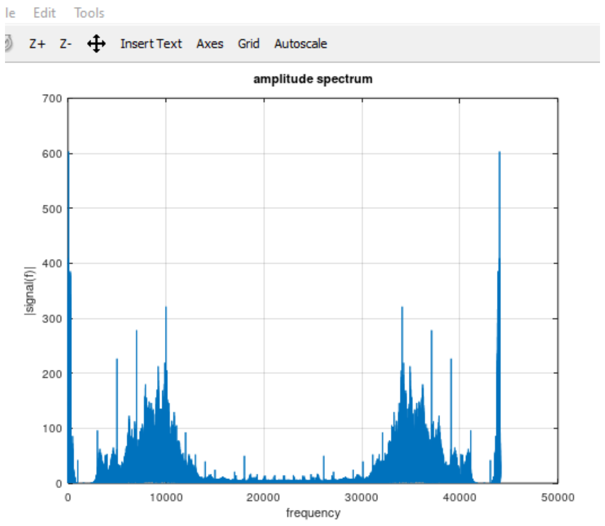
★ Test and change noise level

Increase noise	Decrease noise
Time domain/frequency domain without filter	**Time domain/frequency domain without filter**
	
Increasing in noise amplitude than original recording	Decreasing in noise amplitude than original recording
	
Noise at 944.7 frequency point retched to 12782 HZ greater than 3980	Noise at 944.7 frequency point retched to 1573.5 HZ less than 3980

Time domain/frequency domain filter

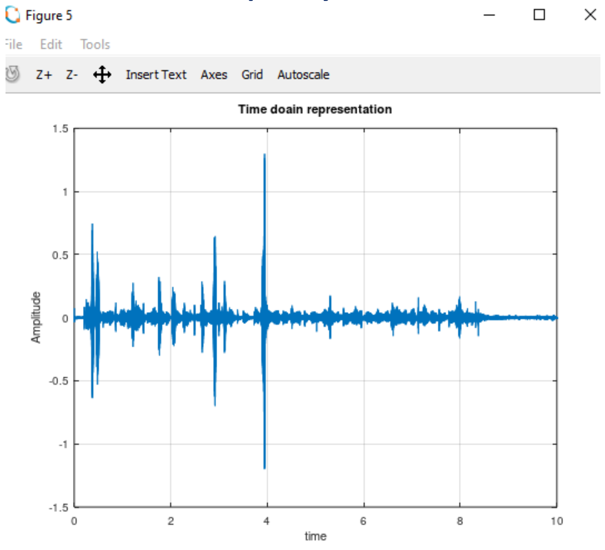


Amplitude in time .48 sec decrease from .08 to .02 after filtering.

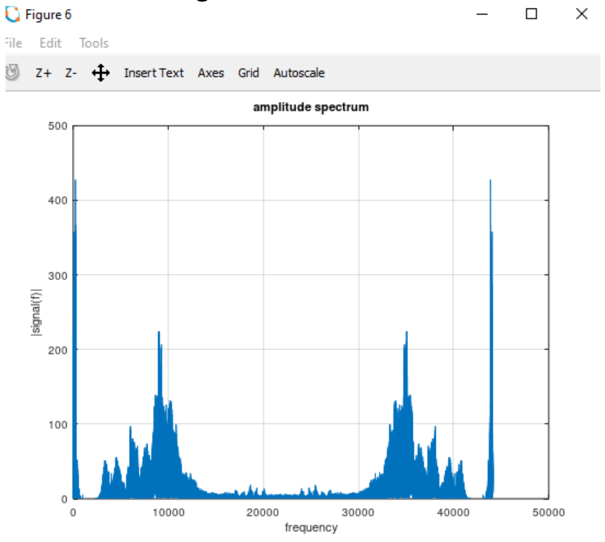


Noise at 944.7 frequency point decrease from 12782 to 39.246 HZ after filtering.

Time domain/frequency domain filter

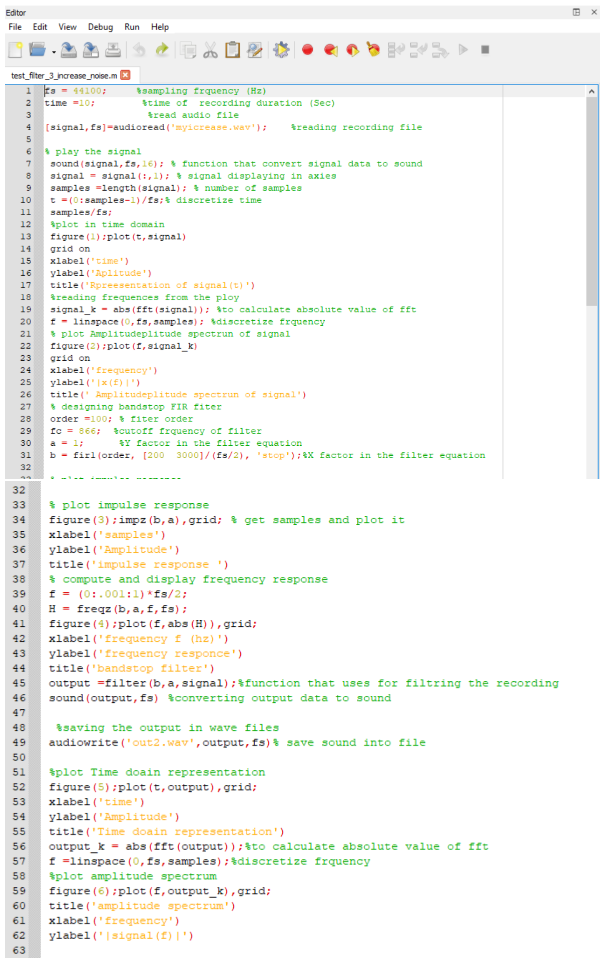


Amplitude in time .71 sec decrease from .07 to .05 after filtering.



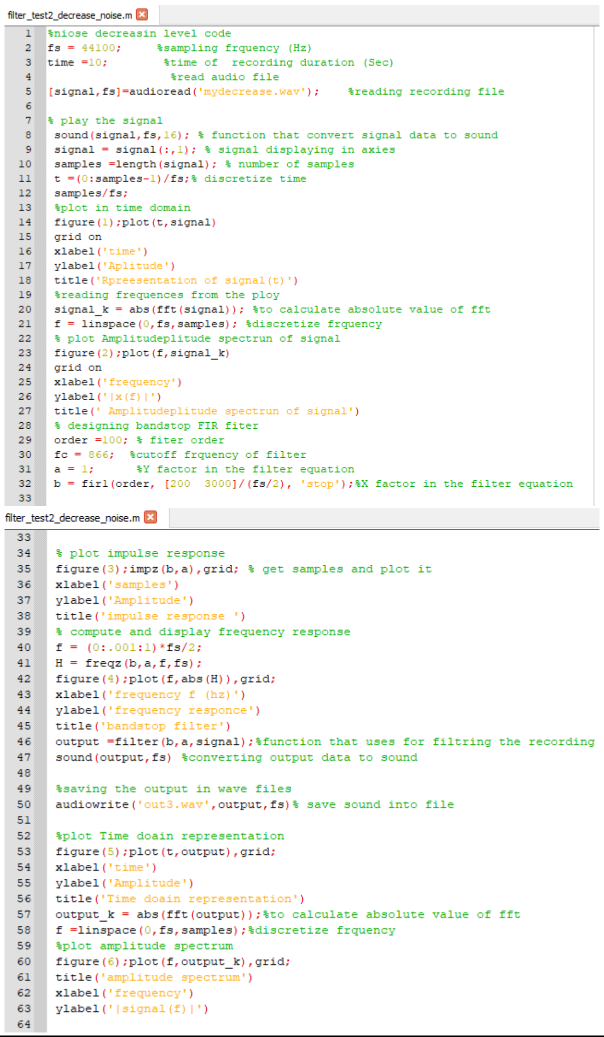
Noise at 944.7 frequency point decrease from 1573.5 to 5.9HZ after filtering.

code



```
fs = 44100;       %sampling frquency (Hz)
time =10;         %time of  recording duration (Sec)
                   %read audio file
[signal,fs]=audioread('myicrease.wav');    %reading recording file

% play the signal
 sound(signal,fs,16); % function that convert signal data to sound
 signal = signal(:,1); % signal displaying in axies
 samples =length(signal); % number of samples
 t =(0:samples-1)/fs;% discretize time
 samples/fs;
 %plot in time domain
 figure(1);plot(t,signal)
 grid on
 xlabel('time')
 ylabel('Aplitude')
 title('Rpreesentation of signal(t)')
 %reading frequences from the ploy
 signal_k = abs(fft(signal)); %to calculate absolute value of fft
 f = linspace(0,fs,samples); %discretize frquency
 % plot Amplitudeplitude spectrun of signal
 figure(2);plot(f,signal_k)
 grid on
 xlabel('frequency')
 ylabel('|x(f)|')
 title(' Amplitudeplitude spectrun of signal')
 % designing bandstop FIR fiter
 order =100; % fiter order
 fc = 866;  %cutoff frquency of filter
 a = 1;      %Y factor in the filter equation
 b = firl(order, [200  3000]/(fs/2), 'stop');%X factor in the filter equation

 % plot impulse response
 figure(3);impz(b,a),grid; % get samples and plot it
 xlabel('samples')
 ylabel('Amplitude')
 title('impulse response ')
 % compute and display frequency response
 f = (0:.001:1)*fs/2;
 H = freqz(b,a,f,fs);
 figure(4);plot(f,abs(H)),grid;
 xlabel('frequency f (hz)')
 ylabel('frequency responce')
 title('bandstop filter')
 output =filter(b,a,signal);%function that uses for filtring the recording
 sound(output,fs) %converting output data to sound

  %saving the output in wave files
 audiowrite('out2.wav',output,fs)% save sound into file

 %plot Time doain representation
 figure(5);plot(t,output),grid;
 xlabel('time')
 ylabel('Amplitude')
 title('Time doain representation')
 output_k = abs(fft(output));%to calculate absolute value of fft
 f =linspace(0,fs,samples);%discretize frquency
 %plot amplitude spectrum
 figure(6);plot(f,output_k),grid;
 title('amplitude spectrum')
 xlabel('frequency')
 ylabel('|signal(f)|')
```

code



```
%niose decreasin level code
fs = 44100;      %sampling frequency (Hz)
time =10;        %time of  recording duration (Sec)
                  %read audio file
[signal,fs]=audioread('mydecrease.wav');    %reading recording file

% play the signal
 sound(signal,fs,16); % function that convert signal data to sound
 signal = signal(:,1); % signal displaying in axies
 samples =length(signal); % number of samples
 t =(0:samples-1)/fs;% discretize time
 samples/fs;
 %plot in time domain
 figure(1);plot(t,signal)
 grid on
 xlabel('time')
 ylabel('Aplitude')
 title('Rpreesentation of signal(t)')
 %reading frequences from the ploy
 signal_k = abs(fft(signal)); %to calculate absolute value of fft
 f = linspace(0,fs,samples); %discretize frquency
 % plot Amplitudeplitude spectrun of signal
 figure(2);plot(f,signal_k)
 grid on
 xlabel('frequency')
 ylabel('|x(f)|')
 title(' Amplitudeplitude spectrun of signal')
 % designing bandstop FIR fiter
 order =100; % fiter order
 fc = 866;  %cutoff frquency of filter
 a = 1;      %Y factor in the filter equation
 b = firl(order, [200  3000]/(fs/2), 'stop');%X factor in the filter equation


 % plot impulse response
 figure(3);impz(b,a),grid; % get samples and plot it
 xlabel('samples')
 ylabel('Amplitude')
 title('impulse response ')
 % compute and display frequency response
 f = (0:.001:1)*fs/2;
 H = freqz(b,a,f,fs);
 figure(4);plot(f,abs(H)),grid;
 xlabel('frequency f (hz)')
 ylabel('frequency responce')
 title('bandstop filter')
 output =filter(b,a,signal);%function that uses for filtring the recording
 sound(output,fs) %converting output data to sound

 %saving the output in wave files
 audiowrite('out3.wav',output,fs)% save sound into file

 %plot Time doain representation
 figure(5);plot(t,output),grid;
 xlabel('time')
 ylabel('Amplitude')
 title('Time doain representation')
 output_k = abs(fft(output));%to calculate absolute value of fft
 f =linspace(0,fs,samples);%discretize frquency
 %plot amplitude spectrum
 figure(6);plot(f,output_k),grid;
 title('amplitude spectrum')
 xlabel('frequency')
 ylabel('|signal(f)|')
```

☺ References: -

- ➢ Filter Design for Signal Processing Using MATLAB and Mathematica 1st Edition
- ➢ https://www.mathworks.com/solutions/signal-processing.html?s_tid=srchtitle
- ➢ https://en.wikipedia.org/wiki/Audio_filter
- ➢ https://en.wikipedia.org/wiki/Audio_signal_processing
- ➢ https://www.howtoforge.com/tutorial/octave-audio-signal-processing-ubuntu/