National Cheng Kung University, Taiwan

111-1 Medical Computer

醫用電腦

FINAL PROJECT:

Tinnitus suppression APP for homecare

Student Name	中文名字	Student ID
Krit Rudeejaroonrung	曾德財	P86117202
Karen	林阮銀河	P86107207
Kamonparn	王佩佩	P86107061

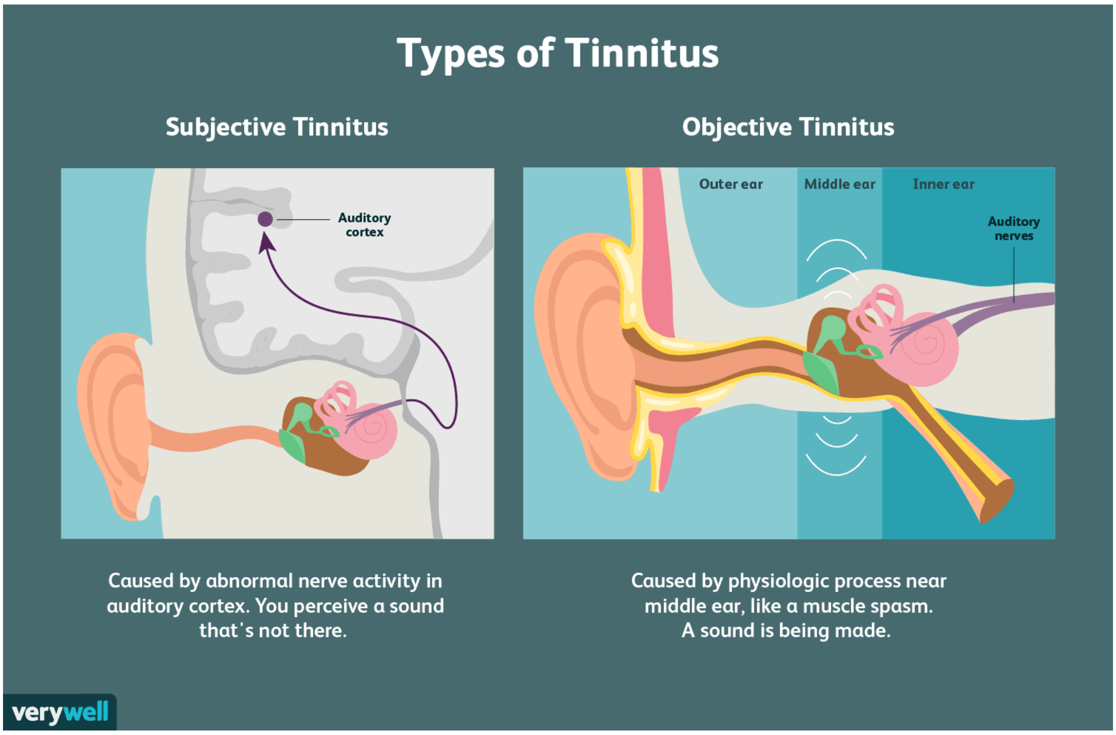
1. Disease fundamental

Tinnitus is when you experience ringing or other noises in one or both of your ears. An external sound doesn't cause the noise you hear when you have tinnitus; other people usually can't hear it.

Tinnitus is a common problem. It affects about 15% to 20% of people and is especially common in older adults.

An underlying condition, such as age-related hearing loss, an ear injury, or a problem with the circulatory system, usually causes tinnitus. Tinnitus is often described as a ringing in the ears, even though no external sound is present. However, tinnitus can also cause other types of phantom noises in your ears. [1]

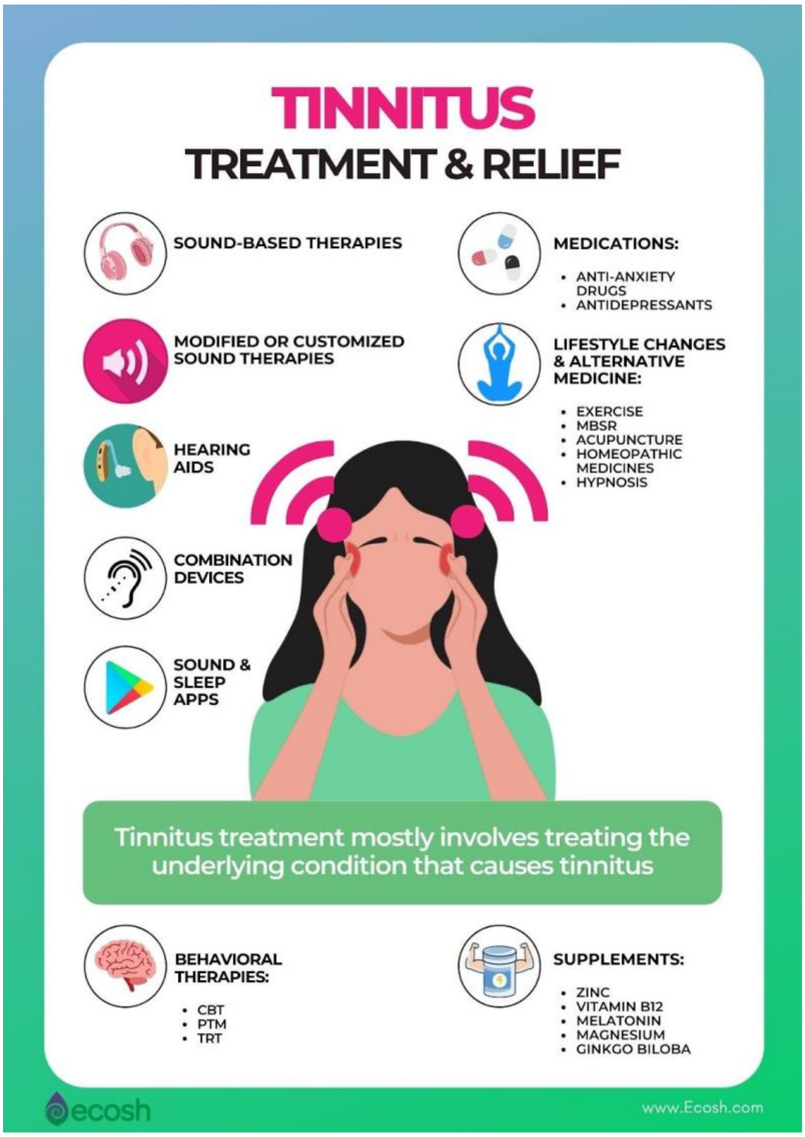
There are two types of tinnitus:



Subjective tinnitus is an ear ringing that the sufferer can only hear. This is the most common type of ear ringing and can be due to problems with the outer, middle, and inner ear. These are signals entering the hearing center of the brain, which the brain interprets as ringing.

Objective tinnitus may also be audible to the attending physician using special equipment. This relatively rare type can be caused by problems with the blood vessels, the condition of the ear bones in the inner ear, or an involuntary contraction of the muscles. [2]

2. Existing solutions

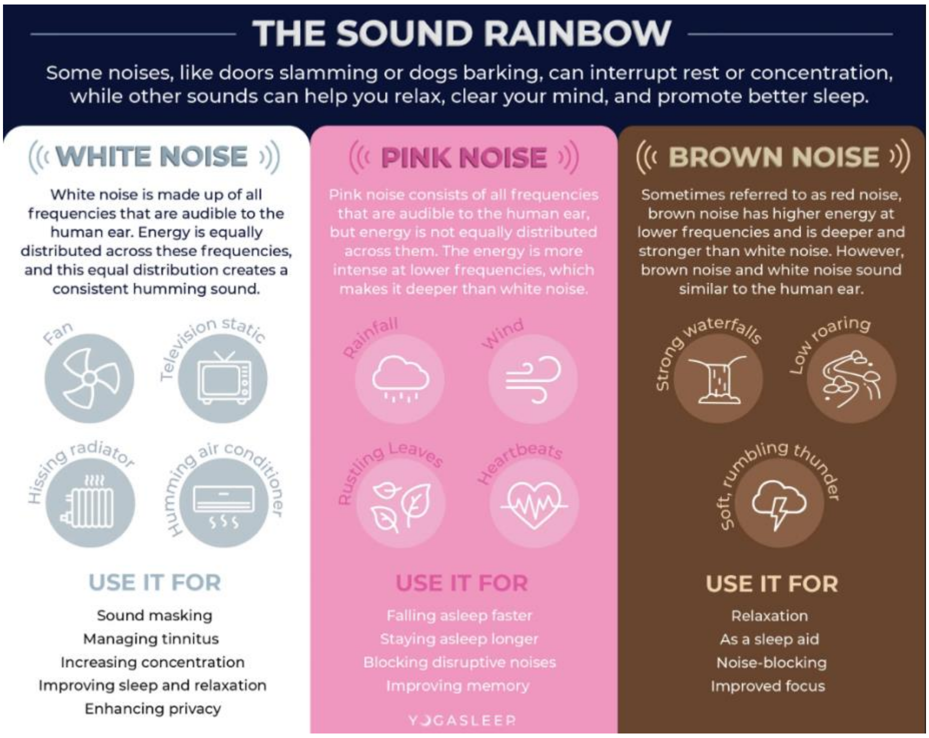


Sound-Based Therapies For Tinnitus

Sound-based therapies work on four general mechanisms of action:

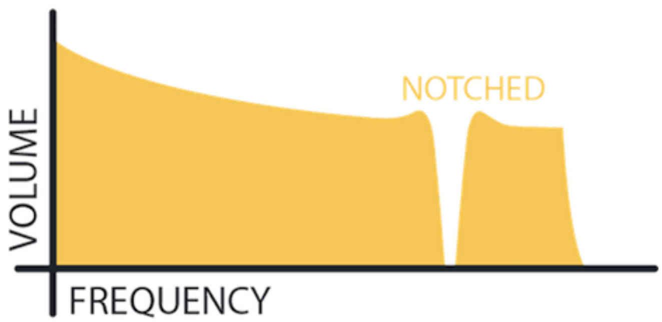
- Masking: provide external pleasurable sound that can help partially or completely cover the noise of ringing in the ears.
- Distraction: provide external sound to redirect persons' focus from tinnitus noise.
- Habituation: help the brain recategorize tinnitus as insignificant noise and deliberately ignore it.
- Neuromodulation: provide particular sound to reduce neural hyperactivity, which is believed to be the underlying cause.

Sound therapy was introduced on the principle of distraction: if a level of noise, usually 'white noise,' is introduced, it can reduce the contrast between the tinnitus signal and background activity in the auditory system, with a decrease in the patient's perception of their tinnitus.



White noise with audio notching

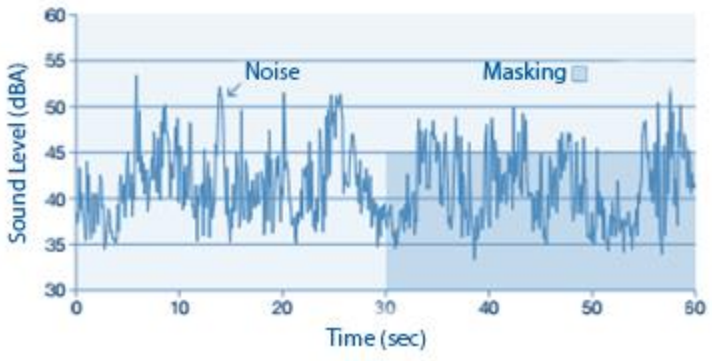
- Is accomplished by completely removing the exact tinnitus from the sound.
- When a notch white noise is played, the tinnitus is the only sound heard at that particular frequency.
- When the notched sound is turned off, it tricks the brain into turning off the tinnitus following the therapy.
- Dramatically decreases the overall intensity and loudness of the tinnitus over time.
- Audio notching is a therapy technique to help prevent tinnitus comeback after the white noise is turned off.



White noise without audio notching

- Tinnitus masking was introduced on the principle of distraction - if sound, usually 'white noise' is played, it may be sufficient to distract a patient from hearing the noises produced by their tinnitus; the new sound will mask out the patient's tinnitus sounds.
- Have a good result immediately.
- Easy for implementation.

- The tinnitus typically persists when the white noise masking is turned off.



Modified Or Customized Sound Therapies

Medical-grade sound devices can provide customized sounds shaped precisely to your tinnitus. Contrary to typical sound machines, modified-sound devices are just worn sometimes, and you should be capable of witnessing relief in symptoms also when the machine is turned off. So, over some period, you may see lasting improvement in the loudness of ear ringing.

Hearing Aids

In most times, tinnitus evolves as a sign of hearing loss because the brain changes how it processes sounds. In this case, people may notice that the better they hear external sounds, the less they notice ear ringing. A hearing aid is a tiny gadget that increases the volume of sounds outside and can help the brain learn new methods to interpret noises.

Combination Devices (Hearing Aid Together With Sound Therapy)

There are also hearing aids that are combined with sound-making technology that ongoingly provides white noise or other customized noises. These types of gadgets unite the advantages of a hearing aid with other sound therapies and maybe, therefore especially suitable for people with tinnitus and measurable hearing loss. Also, because of the portable nature of these devices, they can provide semi-continuous use and more consistent benefit throughout the day.

Free Sound And Sleep Apps

There are so many sound and sleep apps that you may find it hard to figure out the most suitable ones which might precisely relieve your symptoms or help you to sleep better. The selection of sound and sleep apps (available in Apple’s App Store and Google Play) compiled by an audiologist with tinnitus includes apps.

Behavioral Therapies

Sometimes, doctors do not find a cure or relief for your ear ringing. In these cases, the treatment focuses on acceptance and coping. Meaning that you`ll just have to learn to live with this condition as best as you can. Luckily some therapies can assist you in finding ways to lessen the distress your tinnitus is causing you.

Cognitive Behavioral Therapy (CBT) is a kind of talk therapy that helps to identify and change negative thought patterns and may also help people with tinnitus learn to cope with the condition.

Medications

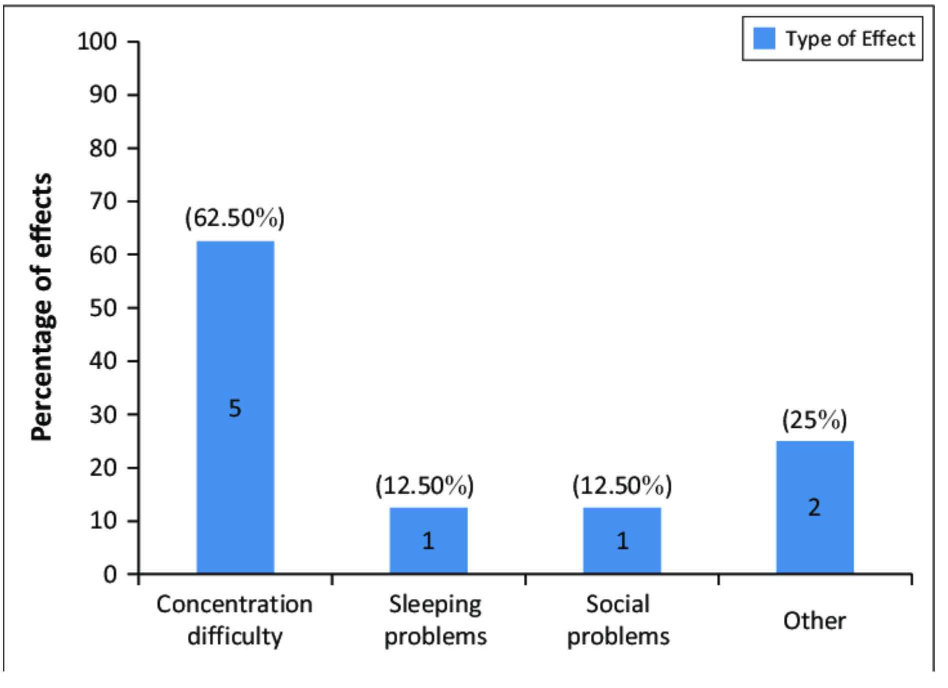
There are no medicines that could cure tinnitus directly, only those that may help make the noise in your ears more bearable. These medications include, for example:

- Anti-anxiety drugs.
- Antidepressants.

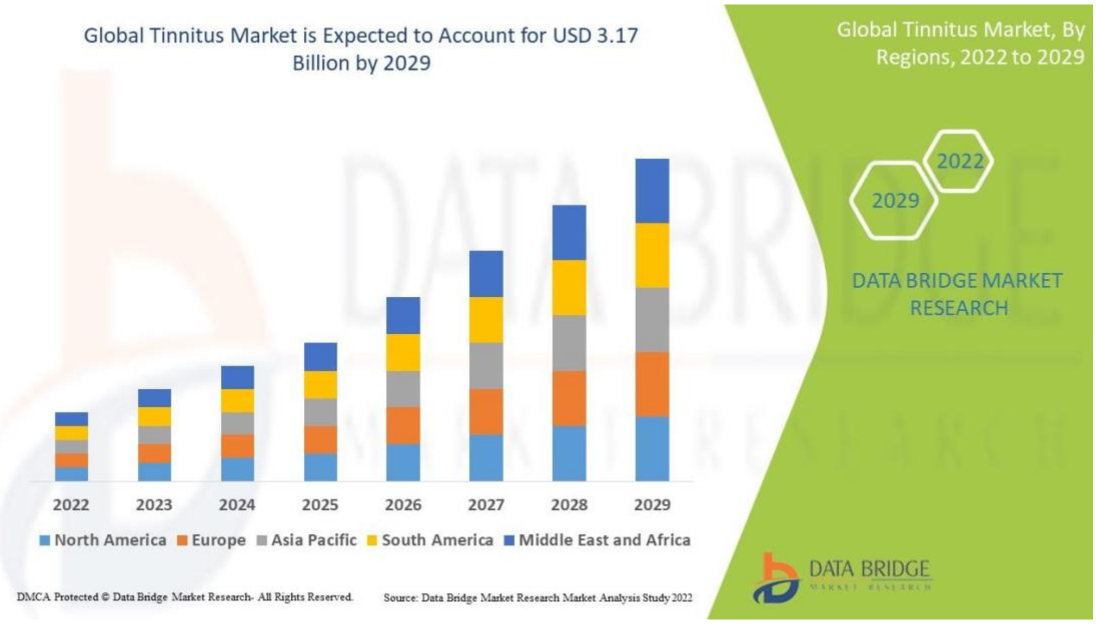
Lifestyle Changes, Self Help, Home Remedies, and Alternative Medicine For Tinnitus
When it comes to lessening tinnitus symptoms by yourself, there are some lifestyle changes that you can start practicing already today, and thus take a step closer to a more fulfilled and noise-free life.

- Exercise. As anxiety, depression, stress, illness, and lack of sleep all aggravate ringing in the ears, systematic and consistent exercise might help.
- Mindfulness-based stress reduction (MBSR).
- Alternative medicine, complementary therapies, and remedies

3. Motivation



Tinnitus impacts the patient's life in many varied ways. Some of the troubles include poor concentration, difficulty in relaxation, irritability, discomfort in quiet, sleep problems, feelings of depression, interference with work or social activities, and more. Over two-thirds of our patients came in with multiple problems in these areas. Remember that these patients are devastated psychologically, emotionally, and often socially. They are in bad shape when they show up at the clinic. [3]



The Global tinnitus market is expected to account for USD 3.17 Billion by 2029, with the CAR around 2.4%.

4. Project goal

To design an application that can mock the volume and frequency of ringing sounds that patients undergo through a user interface and customize white noise sounds that resemble natural sounds using the notching filter concept

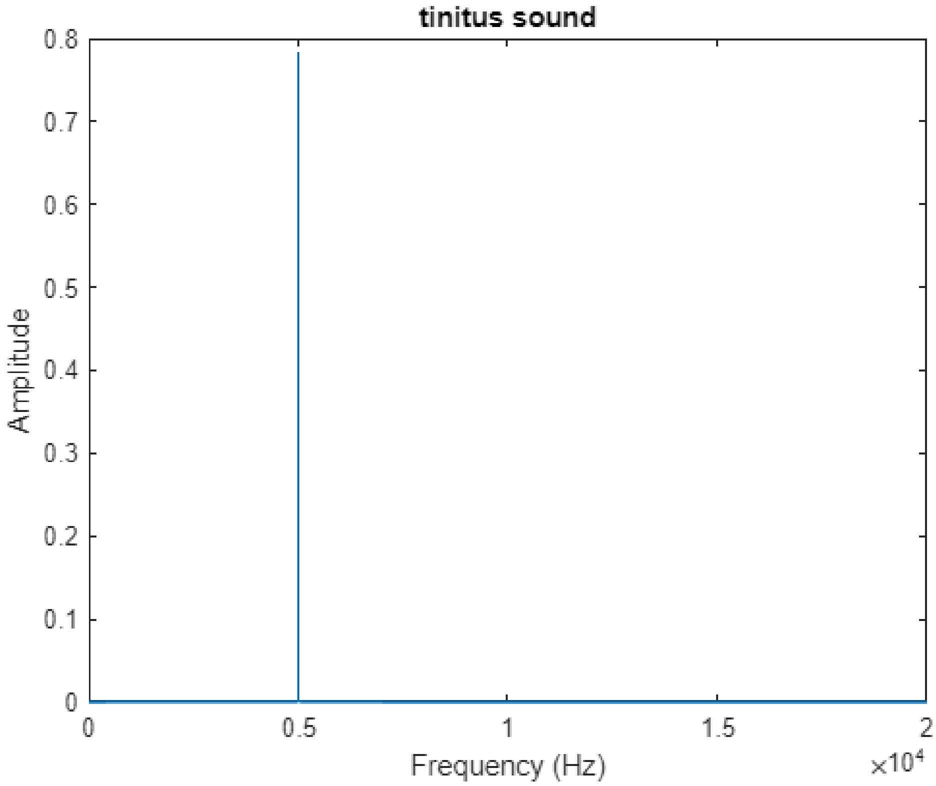
5. Process workflow

To design the application for homecare tinnitus, the design process can be divided into five parts. The first part generates the tinnitus wave to mimic the phantom sound inside the tinnitus patient's ear. The second part is white noise; we generate the white noise by using the MatLab function. The third part is natural sound; we combine the natural sound with the white noise because the white noise alone is boring, which makes the patients not willing to listen to it. The fourth part is designing notching filters. Due to our research, we found that white noise with audio notching can help prevent tinnitus comeback after the white noise is turned off. Accordingly, we decide to apply this principle to our application too. The final part is designing applications. We design the real application UI for the user and consider what the users need to adjust to customize their own playlists.

➢ **Tinnitus sound**

There are several types of tinnitus sound, such as ringing, hissing, roaring, crickets, screeching, sirens, pulsing, etc. However, we select only the ringing sound for this study.

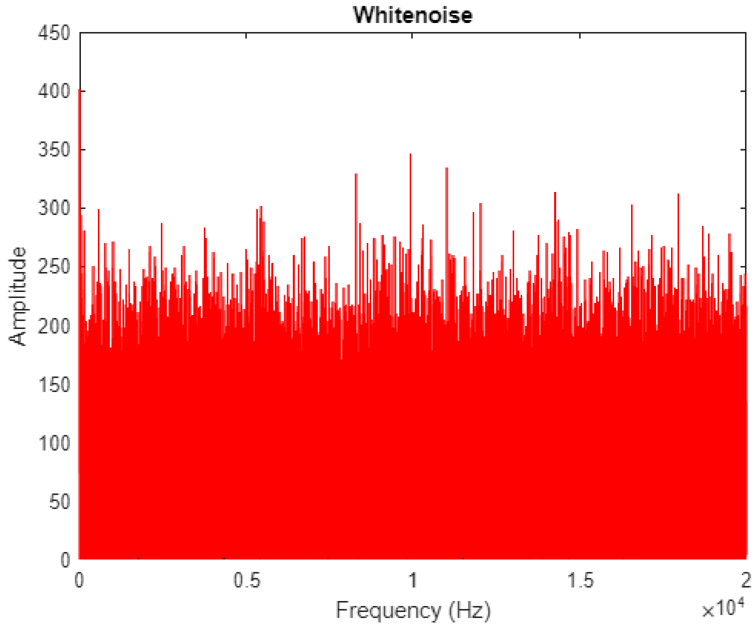
The characteristic of ringing sound has only one specific frequency. Furthermore, the different wave shapes can lead to different sound outputs as well. For example, if we compare the 1kHz sinusoidal wave with the 1kHz sawtooth shape wave, even though the pitch of both of them is the same, the sound still not be exactly the same. For this study, we select only sinusoidal waves to illustrate the application feature. Accordingly, one sinusoidal wave is created and represented in the frequency domain as in the picture below:



In this study, the frequency of this wave is fixed at 5 kHz. However, the frequency will be adjustable after revising the code again in the MATLAB APP designer. The sampling frequency is 44.1 kHz, which is the standard audio sampling frequency. We limit the x-axis to 20 kHz because otherwise, another spectrum will appear at 30 kHz.

White noise

The white noise is the random frequency wave to suppress the phantom sound inside the ear. In this study, we select white Gaussian noise to generate the white noise and represent the wave in the frequency domain by using Fourier transform function in Matlab.

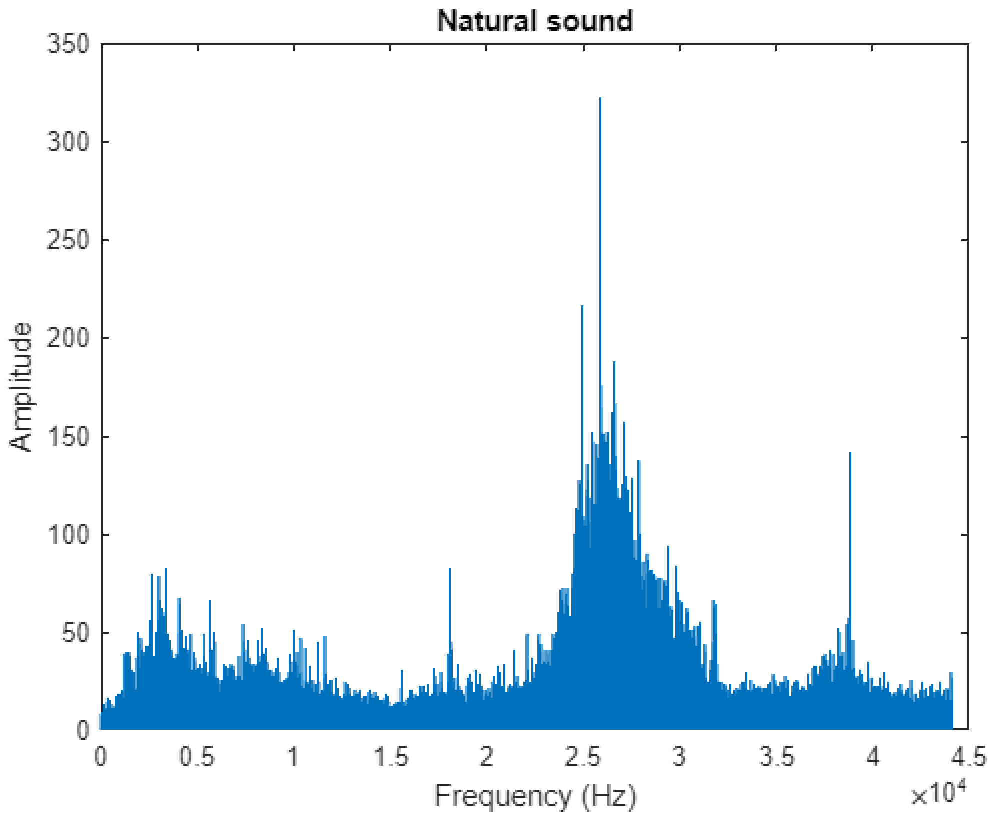


Although this white noise can be used to suppress the tinnitus sound, however, we still need to apply a notching filter to this signal to achieve better tinnitus treatment.

Natural sounds

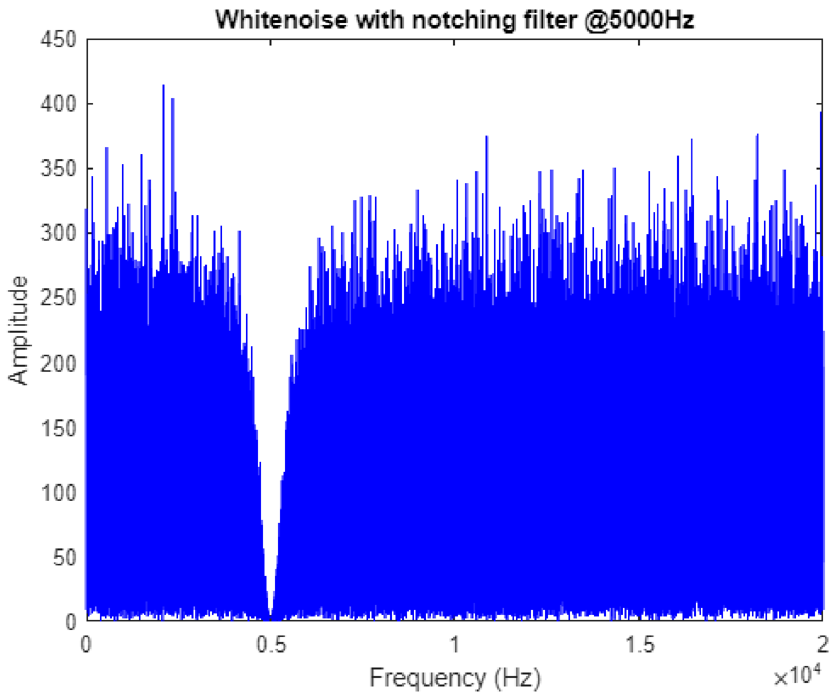
Are normally combined with white noise to make it an easy-to-listen sound. There are several natural sounds in this world, such as the ocean, raindrops, waterfalls, etc. In this study, we select the ocean sound as a preliminary study.

We import the ocean sound from the free source that is available on the internet into Matlab. After that, we convert it into the matrix form and generate the signal representing its sound. The signal is shown as a frequency domain similar to tinnitus sound and white noise.

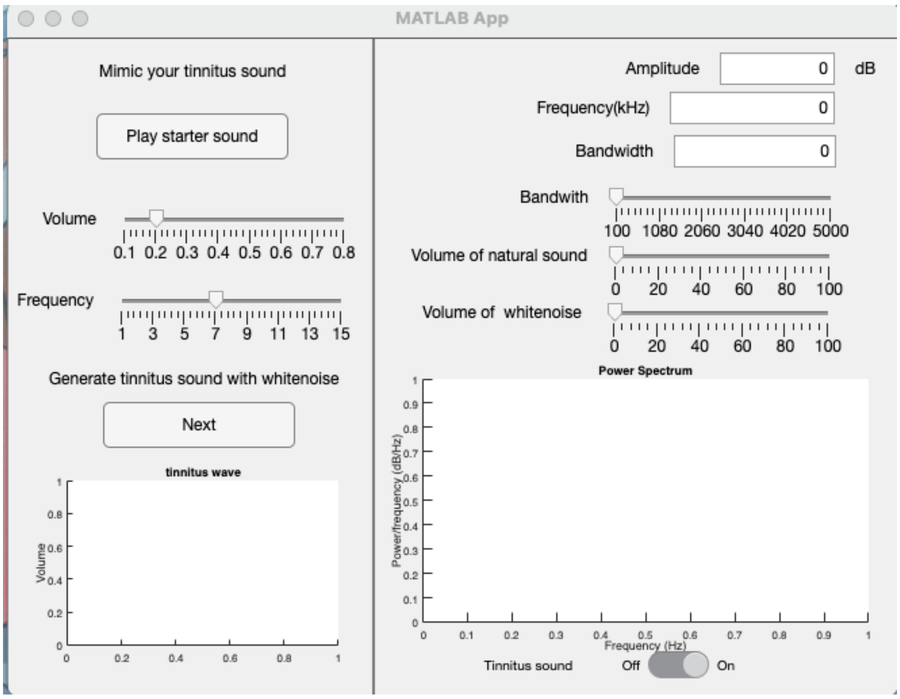


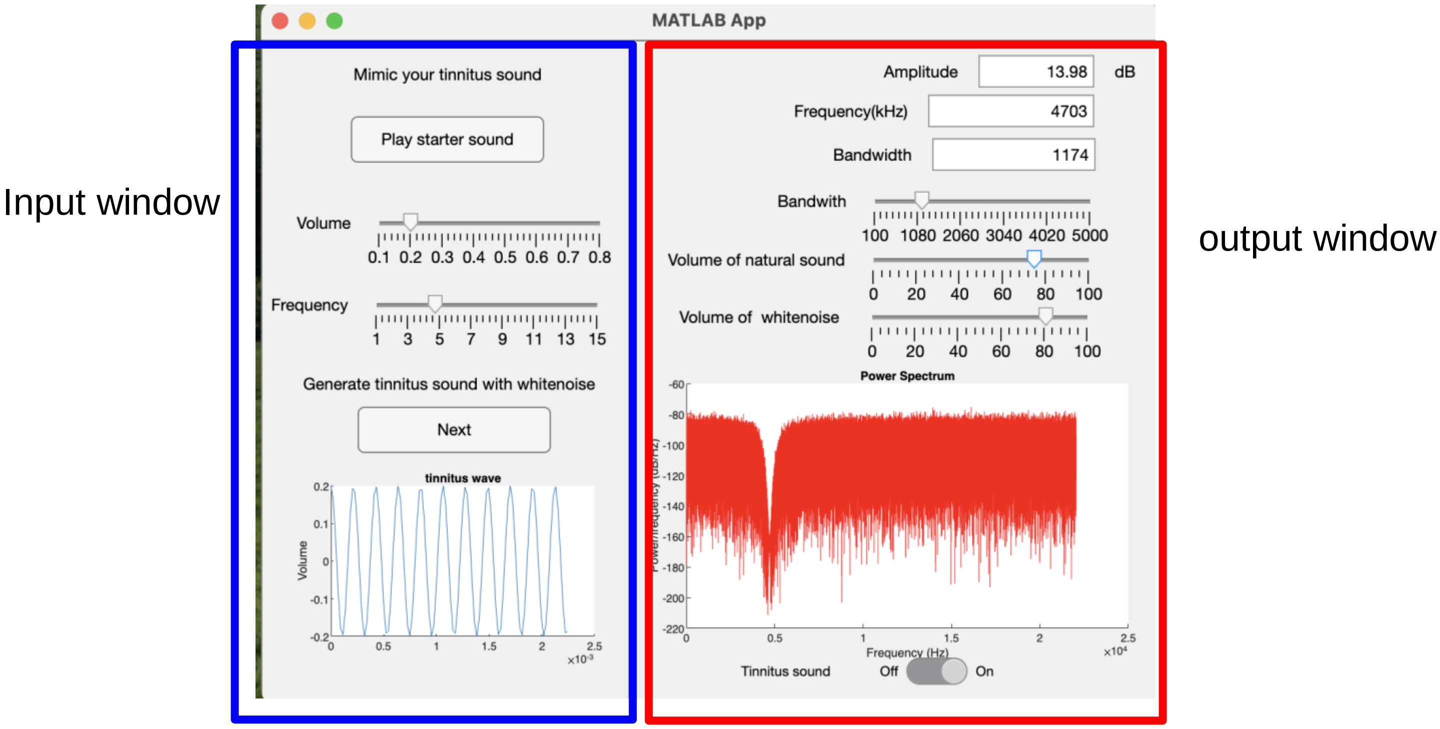
Notching filter

After all signals have been created, we design the notching filter by using the ‘irr-notch’ function, which has similar characteristics as the band stop filter because we would like to decide the white noise that is going to be replaced by missing frequency from the tinnitus sound. The frequency center of this notching is similar to the tinnitus sound frequency, which is 5 kHz. The notching bandwidth is designed to be 800 Hz. However, similar to the frequency center, we will adjust the code again after putting it into the MATLAB APP designer. The picture below shows the white noise after being filtered by a notching filter at 5000 Hz with 800 Hz of bandwidth which can compare to the original white noise plot above. This concept was also applied to natural sound.



➢ **MATLAB APP designer**





User Interface design

The user interface of this APP is designed to be user-friendly and consists of two windows(input and output). The first window is for inputting data and displaying the waveform of the tinnitus sound. It includes a frequency slider for adjusting the sound and an amplitude slider for adjusting the volume. The second window allows the user to customize and adjust the output to be more comfortable. It contains sliders for adjusting the bandwidth, the volume of natural sound, and white noise, as well as a switch for turning off the tinnitus sound when a suitable waveform is found. A plot, called an "axe," is provided to show the notch filter's results, allowing the user to see the effects of adjusting the bandwidth. The user interface makes it easy for the user to customize and adjust the tinnitus sound waveform to their liking.

APP Process:

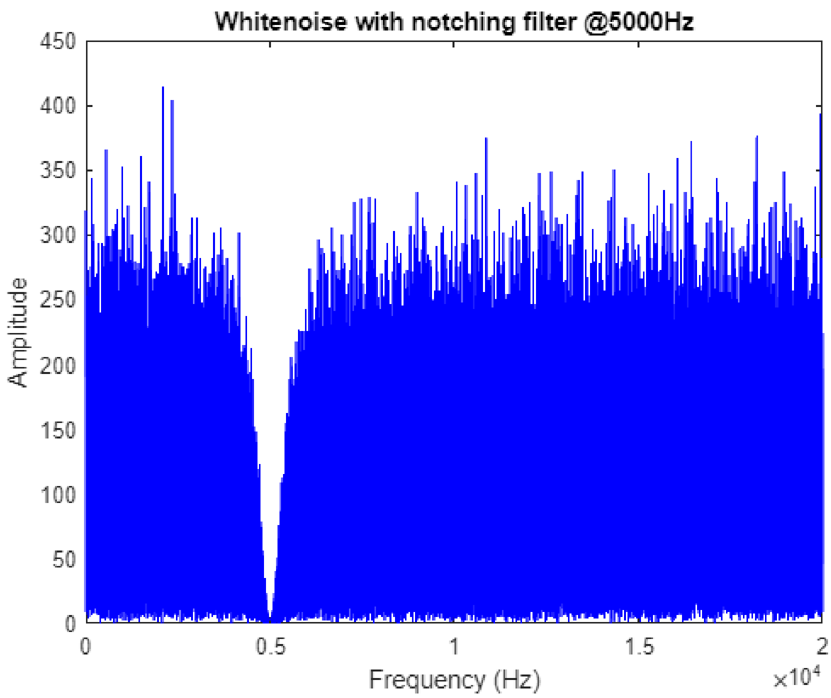
1. The user can adjust the volume and frequency of the sine wave(tinnitus sound) using sliders.
2. The user can press a button to play the tinnitus sound based on the current settings.
3. Then, generates a new tinnitus sound with the specified amplitude and frequency
4. Next, play the tinnitus sound and plot it on a graph.
5. The user can press a button to generate white noise and combine it with the tinnitus sound to create an audio output with added noise and natural sound with a notch filter.
6. The user can adjust the volume and frequency of the output sound based on the slider settings of bandwidth and also the volume of white noise and natural sound.
7. Plays the audio output with customization based on the user's tinnitus characteristic.

Limitation

- Can be applied with only 1 constant frequency of tinnitus sound.
- The output sound isn't continued play for long times we need to
- provide duration time.
- The project need more clinical trial to evaluate the efficiency of the final product
-

6. Results

According to this study, the irr-notch function is suitable for audio notching. purpose. The signal at a certain frequency has been taken away because of the filter.



We observe the voice suppression capability by playing the audio output together with the tinnitus sound. The tinnitus sound frequency was adjusted to 5000, 7500, 10000, and 12500 Hz, respectively. We found that the output audio can increase our tolerance to listen to the tinnitus sound after playing the output audio for all frequency sounds. Accordingly, we conclude that our audio output can suppress the tinnitus ringing sound.

7. Discussion

This is the preliminary study of a tinnitus suppression APP for homecare by employing the audio notching concept. Accordingly, the application is limited to only tinnitus patients with a ringing sound inside the ear. This application still needs further development to apply to the other type of tinnitus sounds, such as pulsing sounds.

This application has only one natural sound, which is ocean sound. For subsequent development, it's possible to add more natural sound into the library so that users can have more options to select their own favorite natural sound with optimized white noise.

To measure the effectiveness of sound quality, we need to study and collaborate further with audiologists to validate the effectiveness of this playlist and then improve further.

8. Reference

[1]https://www.mayoclinic.org/diseases-conditions/tinnitus/symptoms-causes/syc-20350156

[2]https://www.verywellhealth.com/what-is-tinnitus-causes-effects-and-treatment-1046499

[3]Bagwandin, Vedika & Joseph, Lavanithum. (2017). A survey exploring awareness and experience of tinnitus in young adults. South African Journal of Communication Disorders. 64. 10.4102/sajcd.v64i1.545.

9. Work Contribution

Karen

- Build up tinnitus sound, natural sound, and white noise sound
- Cover background part for presentation and report

Krit

- Design the filter and apply to the signal
- Cover method part for presentation and report

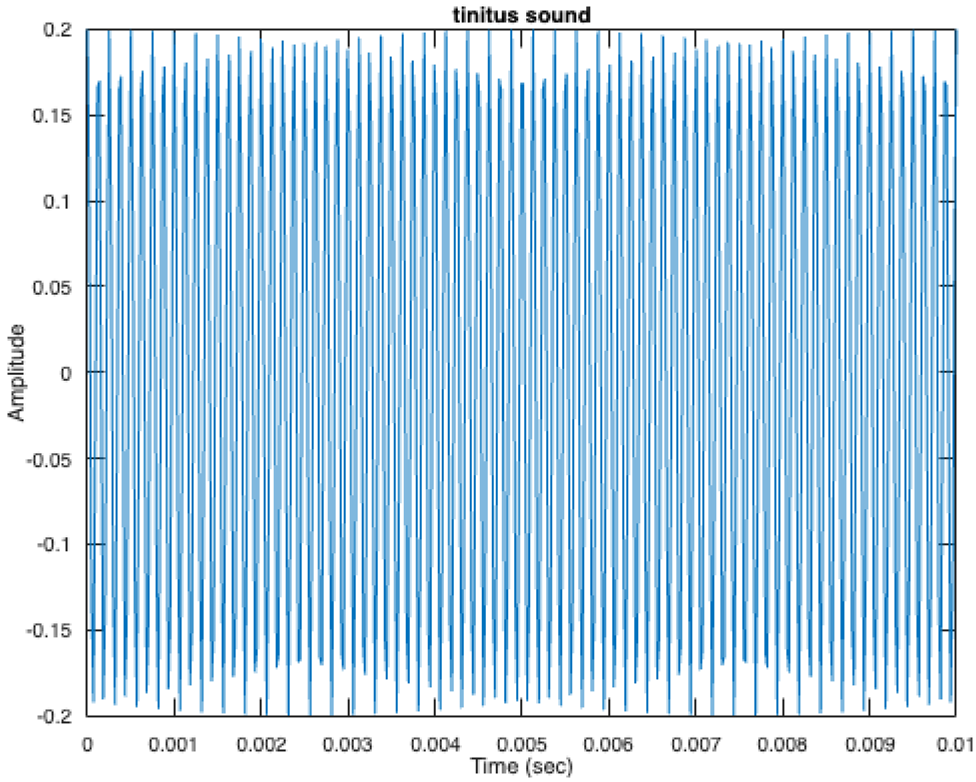
Kamonpan

- Gather the data, combine and verify the code then applied to design and build the APP
- Cover APP part for presentation and report

Code

- **Code in matlab file**

```
% configure signal settings
duration =10;                    % duration in seconds
amp = 0.2;                   % amplitude(dB)
f1 =8000;                          % frequency in Hertz
BW=800;
 %%configure output settings
fs = 44100;                        % sampling rate
T = 1/fs;                          % sampling period
t = 0:T:duration;                  % time vector
 %%create the signal
omega1 = 2*pi*f1;                  % angular frequency in radians
signal = (cos(omega1*t)*amp); % sinusoidal partial 1
% plot tinnitus with time vector
figure
plot(t,signal);xlabel('Time (sec)'); ylabel('Amplitude'); title('tinitus
sound'),xlim([0 0.01]);
```

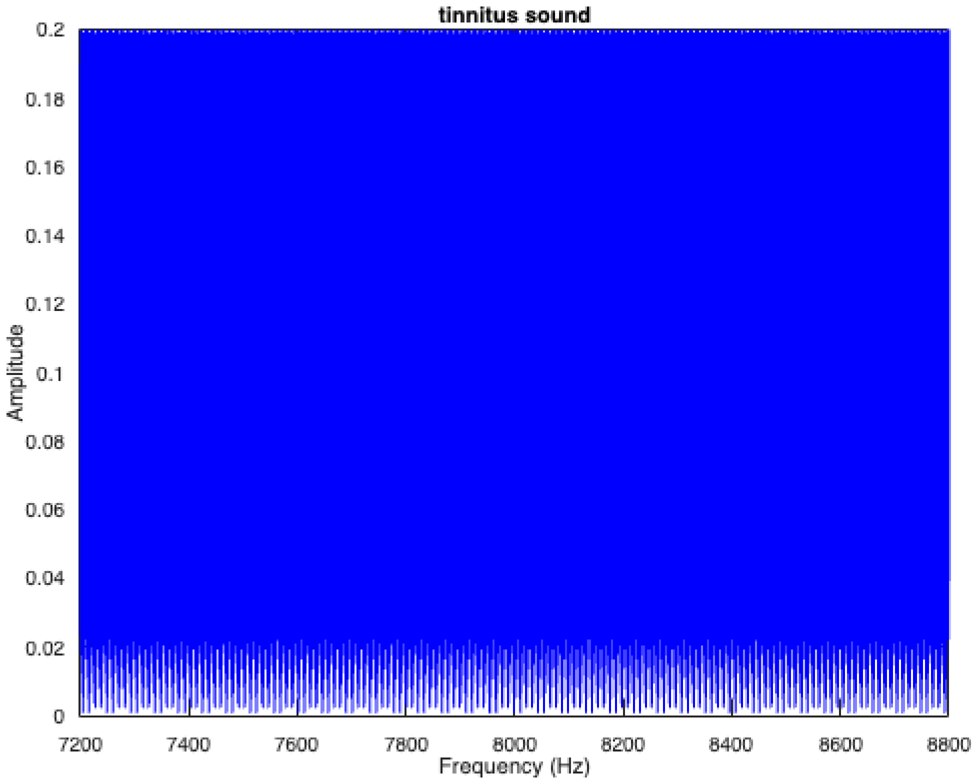



```
N=length(signal);
f=(0:N-1)*(fs/N);
S1=abs(fft(signal));
% Check tinitus sound
```

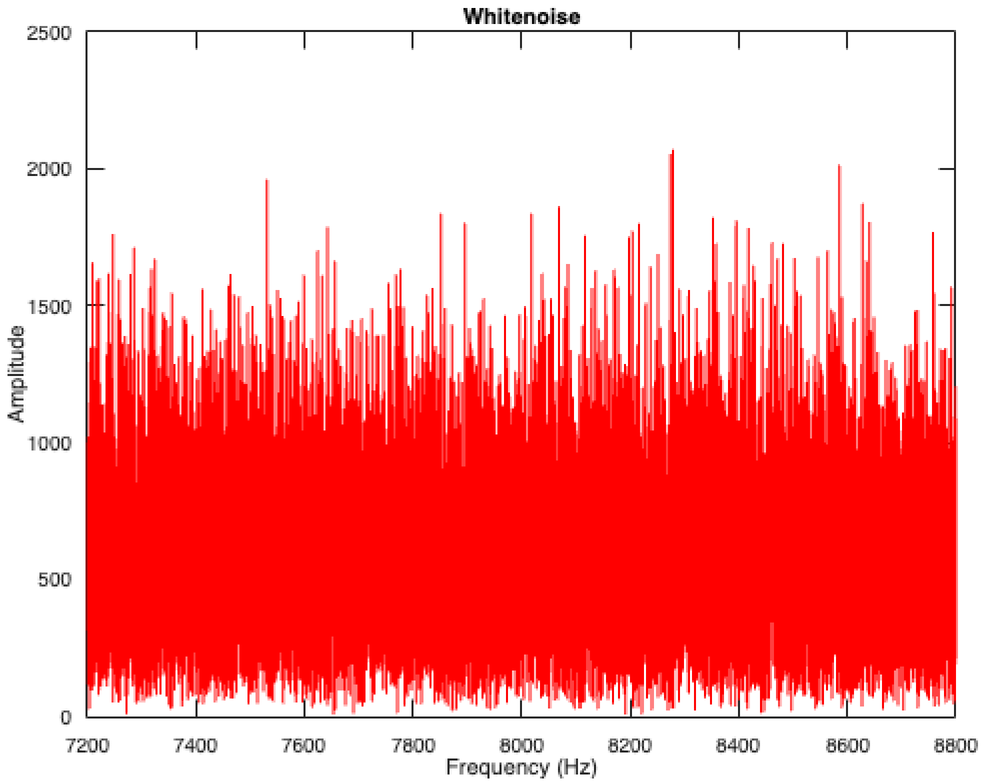
```
%  freqz((signal))
% find the mean amplitude of tinnitus sound
amplitude = mean(abs(fft(signal)))
```

amplitude = 1.0216

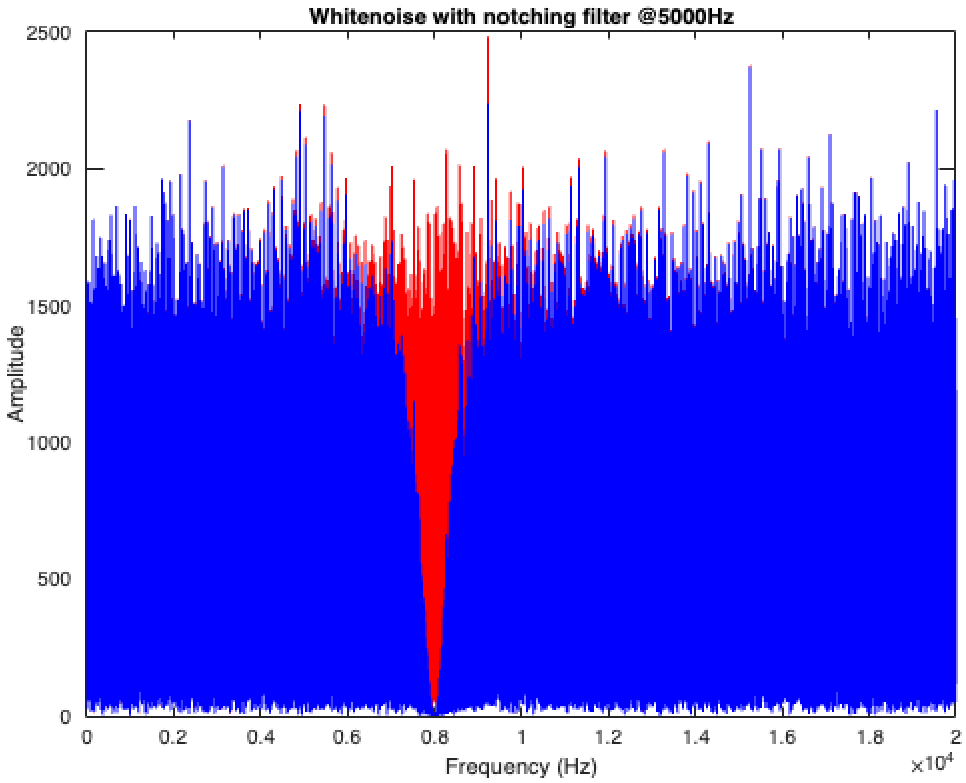
```
%%Create whitenoise wave
SNR=1;
x=gauspuls(t,f1);
znew=awgn(x,SNR,amplitude);
% set lim for show f
fup = f1+BW;
fdown=f1-BW;
figure
plot(f,abs(signal),'b');xlim([fdown fup]);xlabel('Frequency (Hz)');
ylabel('Amplitude'); title('tinnitus sound')
```



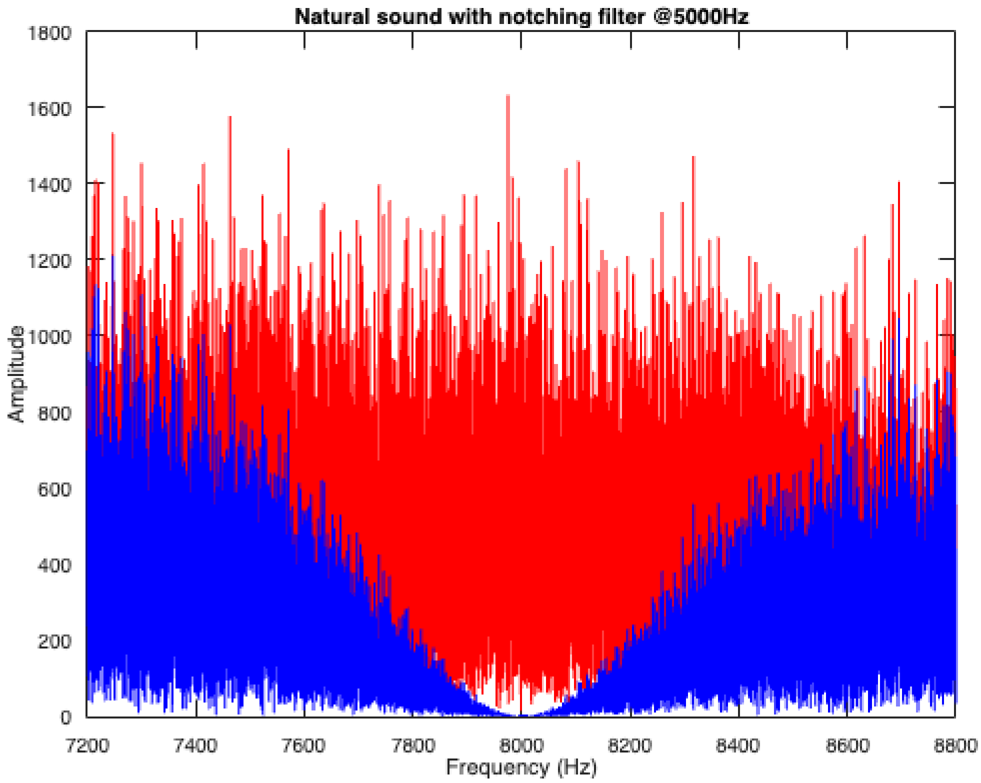

```
figure
plot(f,abs(fft(znew)),'r');xlim([fdown fup]);xlabel('Frequency (Hz)');
ylabel('Amplitude'); title('Whitenoise')
```



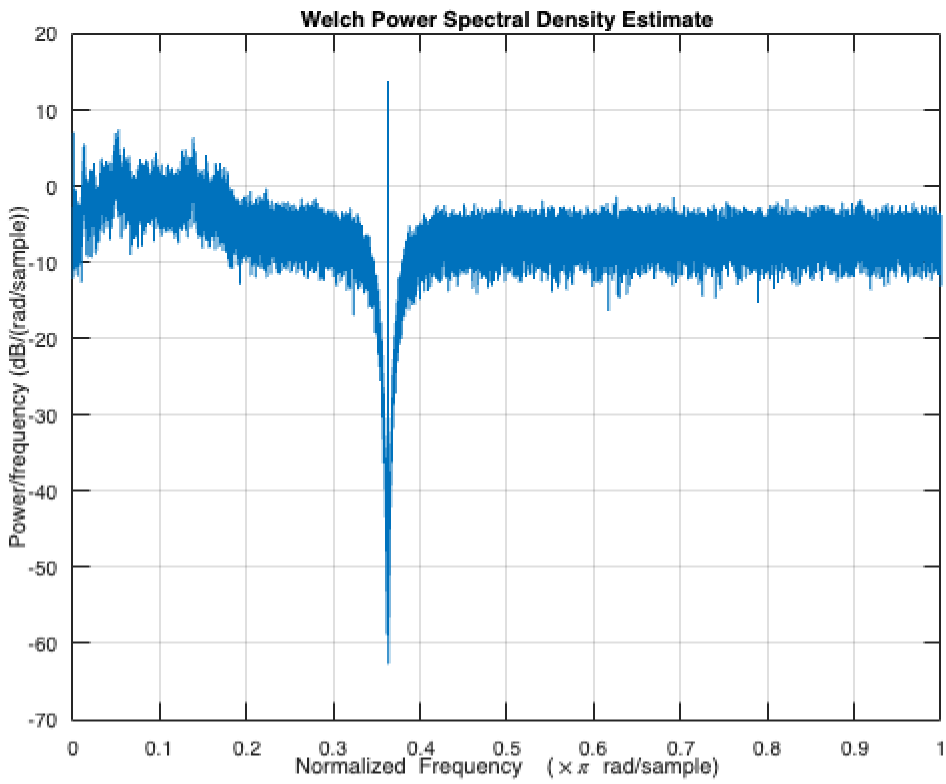

```
 % apply the notch filter to whitenoise
 % design notch
 [b, a] = iirnotch(f1/(fs/2), BW/(fs/2));
 New = filtfilt(b,a,znew);
 amp_white=abs(fft(New));
 whitenoise=New*2.5;

 %plot compare result to check notch filter
 plot(f,abs(fft(znew)),'r');xlim([0 20000]);xlabel('Frequency (Hz)');
ylabel('Amplitude'); title('Whitenoise')
 hold on
 plot(f,amp_white,'b'); xlim([0 20000]); xlabel('Frequency (Hz)');
ylabel('Amplitude'); title('Whitenoise with notching filter @5000Hz')
 %freqz(i)
 hold off
```




```
% load natural sound
[c,Fsn]=audioread('natural sound.mp3');
ns=c(10000:9999+length(New))*30*(amplitude);
%plot natural sound
naturalsound = filtfilt(b,a,ns);
amp_natural=abs(fft(naturalsound));
%plot compare result to check notch filter
plot(f,abs(fft(ns)),'r');xlim([fdown fup]);xlabel('Frequency (Hz)');
ylabel('Amplitude'); title('Natural sound')
hold on
plot(f,amp_natural,'b'); xlim([fdown fup]); xlabel('Frequency (Hz)');
ylabel('Amplitude'); title('Natural sound with notching filter @5000Hz');
hold off
```




```
audioout=(signal+whitenoise+naturalsound)/3;
pwelch(audioout)
```



```
% play sound
sound(audioout,fs)
% freqz(audioout)
```

- ## Code in APP

```
classdef tinnitus_v3 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                   matlab.ui.Figure
        GridLayout                 matlab.ui.container.GridLayout
        LeftPanel                  matlab.ui.container.Panel
        NextButton                 matlab.ui.control.Button
        GeneratetinnitussoundwithwhitenoiseLabel  matlab.ui.control.Label
        FrequencySlider            matlab.ui.control.Slider
        FrequencySliderLabel       matlab.ui.control.Label
        VolumeSlider               matlab.ui.control.Slider
        VolumeSliderLabel          matlab.ui.control.Label
        MimicyourtinnitussoundLabel   matlab.ui.control.Label
        PlaystartersoundButton     matlab.ui.control.Button
        UIAxes                     matlab.ui.control.UIAxes
        RightPanel                 matlab.ui.container.Panel
        kHzLabel                   matlab.ui.control.Label
        dBLabel                    matlab.ui.control.Label
        BandwithSlider_2           matlab.ui.control.Slider
        BandwithSlider_2Label      matlab.ui.control.Label
        TinnitussoundSwitch        matlab.ui.control.Switch
        TinnitussoundSwitchLabel   matlab.ui.control.Label
        VolumeofwhitenoiseSlider   matlab.ui.control.Slider
        VolumeofwhitenoiseSliderLabel  matlab.ui.control.Label
        VolumeofnaturalsoundSlider   matlab.ui.control.Slider
        VolumeofnaturalsoundSliderLabel  matlab.ui.control.Label
        BandwidthEditField         matlab.ui.control.NumericEditField
        BandwidthEditFieldLabel    matlab.ui.control.Label
        FrequencyEditField         matlab.ui.control.NumericEditField
```

```
    FrequencyEditFieldLabel       matlab.ui.control.Label
    AmplitudeEditField          matlab.ui.control.NumericEditField
    AmplitudeEditFieldLabel       matlab.ui.control.Label
    UIAxes2                  matlab.ui.control.UIAxes
end

% Properties that correspond to apps with auto-reflow
properties (Access = private)
    onePanelWidth = 576;
end



properties (Access = public)
% Signal settings
durationin = 5;     % duration in seconds
durationout = 10;   % duration in seconds
amplitude;         % amplitude(dB)
f1;                % frequency in Hertz
phi = pi;          % phase offset
T;                 % sampling period
t;                 % time vector of tinitus sound
% Output settings
fs = 44100;        % sampling rate
player;            % audio player object
signal;            % sine wave signal
SNR = 1;           % signal-to-noise ratio
audioout;          % audio with added noise
playernoise;       % noise player object
BW;                % bandwidth
N;                 % number of samples
amp;               % signal amplitude
vn;                % noise variance
vw;                % signal+noise variance
switchvalue;       % switch value
audioout2;         % audio with added noise
playernoise2;      % noise player object
```

```
    dataOut;           % output data
    naturalnoise;       % natural noise
    tout;              % time of sound out
    c;                 % natural sound
    end

    properties (Access = private)
    end



    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
%           load natural sound file
            [app.c,~]=audioread('natural sound.mp3');

        end

        % Button pushed function: PlaystartersoundButton
        function PlaystartersoundButtonPushed(app, event)
% play sine wave
 app.AmplitudeEditField.Value=abs(20*log10(app.VolumeSlider.Value));
 app.FrequencyEditField.Value=app.FrequencySlider.Value*1000;
app.amplitude=app.VolumeSlider.Value;              % amplitude(dB)
 app.f1= app.FrequencySlider.Value*1000;                    % frequency in Hertz
 %%create the signal
 app.T = 1/app.fs; % sampling period
 app.t = 0:app.T:app.durationin; % time vector
 omega1 = 2*pi*app.f1;              % angular frequency in radians
app.signal = (cos(omega1*app.t)*app.amplitude); % sinusoidal partial 1
app.N=length(app.signal);
app.amp = mean(abs(fft(app.signal)));
 %%plot the signal
app.player=audioplayer(app.signal,app.fs);
```

```
play(app.player)
plot(app.UIAxes,app.t(1:10000),app.signal(1:10000))
        end

        % Value changing function: VolumeSlider
        function VolumeSliderValueChanging(app, event)

        end

        % Value changed function: VolumeSlider
        function VolumeSliderValueChanged(app, event)
        app.amplitude=app.VolumeSlider.Value;
        app.AmplitudeEditField.Value = app.VolumeSlider.Value*100;
             stop(app.player)
 app.f1= app.FrequencySlider.Value*1000;                    % frequency in Hertz
 %%create the signal
 app.T = 1/app.fs;                     % sampling period
 app.t = 0:app.T:app.durationin;            % time vector
 omega1 = 2*pi*app.f1;             % angular frequency in radians
app.signal = (cos(omega1*app.t)*app.amplitude); % sinusoidal partial 1
app.N=length(app.signal);
% set new amp
app.amp = mean(abs(fft(app.signal)));
 %%plot the signal
app.player=audioplayer(app.signal,app.fs);
play(app.player)
plot(app.UIAxes,app.t(1:100),app.signal(1:100))

        end

        % Value changed function: FrequencySlider
        function FrequencySliderValueChanged(app, event)
%           get data
            app.f1= app.FrequencySlider.Value*1000;
           app.amplitude=app.VolumeSlider.Value;
           app.FrequencyEditField.Value = app.FrequencySlider.Value*1000;
```

```
%           stop old player
            stop(app.player)
 %%create the signal
 app.T = 1/app.fs;                    % sampling period
 app.t = 0:app.T:app.durationin;             % time vector
 omega1 = 2*pi*app.f1;             % angular frequency in radians
app.signal = (cos(omega1*app.t)*app.amplitude); % sinusoidal partial 1
app.N=length(app.signal);
app.amp = mean(abs(fft(app.signal)));
 %%plot the signal
app.player=audioplayer(app.signal,app.fs);
% play sound
play(app.player)
plot(app.UIAxes,app.t(1:100),app.signal(1:100))
        end

        % Button pushed function: NextButton
        function NextButtonPushed(app, event)
             %%Create whitenoise wave
% stop plying the old one
stop(app.player)
app.BW=app.BandwithSlider_2.Value;
app.tout = 0:app.T:app.durationout;
x=gauspuls(app.t,app.f1);
z=awgn(x,app.SNR,app.amp);
wo = app.f1/(app.fs/2);
[b, a] = iirnotch(wo,app.BW/(app.fs/2));
app.dataOut = filtfilt(b,a,z);
% get the natural sound and filter
ns=app.c(10000:9999+length(app.dataOut))*(app.amp*30);
app.naturalnoise = filtfilt(b,a,ns);
app.audioout=(app.signal+app.dataOut+app.naturalnoise)/3;
app.playernoise=audioplayer(app.audioout,app.fs);
% plot the after notch filter
[~,~] = periodogram(z,[],[],app.fs);
[pnew,fnew] = periodogram(app.dataOut,[],[],app.fs);
```

```
plot(app.UIAxes2,fnew,20*log10(abs(pnew)),'r')
% play the sound after add mofidied white noise and nateral sound
play(app.playernoise)
        end

        % Value changed function: BandwithSlider_2
        function BandwithSlider_2ValueChanged(app, event)
  app.BandwidthEditField.Value = app.BandwithSlider_2.Value;
  % stop plying the old one
            stop(app.playernoise)
app.BW=app.BandwithSlider_2.Value;
app.tout = 0:app.T:app.durationout;

x=gauspuls(app.t,app.f1);
z=awgn(x,app.SNR,app.amp);
wo = app.f1/(app.fs/2);
[b, a] = iirnotch(wo,app.BW/(app.fs/2));
app.dataOut = filtfilt(b,a,z);
% get the natural sound and filter
ns=app.c(10000:9999+length(app.dataOut))*(app.amp*30);
app.naturalnoise = filtfilt(b,a,ns);
app.audioout=(app.signal+app.dataOut+app.naturalnoise)/3;
% plot the after notch filter and play
app.playernoise=audioplayer(app.audioout,app.fs);
[~,~] = periodogram(z,[],[],app.fs);
[pnew,fnew] = periodogram(app.dataOut,[],[],app.fs);
plot(app.UIAxes2,fnew,20*log10(abs(pnew)),'r')
play(app.playernoise)
        end

        % Value changing function: BandwithSlider_2
        function BandwithSlider_2ValueChanging(app, event)

        end

        % Value changing function: FrequencySlider
```

```
        function FrequencySliderValueChanging(app, event)


        end

        % Value changed function: VolumeofnaturalsoundSlider
        function VolumeofnaturalsoundSliderValueChanged(app, event)
  app.vw = app.VolumeofwhitenoiseSlider.Value;
            app.vn = app.VolumeofnaturalsoundSlider.Value;

stop(app.playernoise)
app.BW=app.BandwithSlider_2.Value;
app.tout = 0:app.T:app.durationout;
x=gauspuls(app.t,app.f1);
z=awgn(x,app.SNR,app.amp);
wo = app.f1/(app.fs/2);
[b, a] = iirnotch(wo,app.BW/(app.fs/2));
app.dataOut = filtfilt(b,a,z);
% forrest
ns=app.c(10000:9999+length(app.dataOut))*(app.amp*30);
app.naturalnoise = filtfilt(b,a,ns);
app.audioout=(app.signal+(app.dataOut.*app.vw/100)+(app.naturalnoise.*app.vn/100))/3;
app.playernoise=audioplayer(app.audioout,app.fs);
play(app.playernoise)

        end

        % Value changed function: VolumeofwhitenoiseSlider
        function VolumeofwhitenoiseSliderValueChanged(app, event)
            app.vw = app.VolumeofwhitenoiseSlider.Value;
            app.vn = app.VolumeofnaturalsoundSlider.Value;
            stop(app.playernoise)
app.BW=app.BandwithSlider_2.Value;
app.tout = 0:app.T:app.durationout;
x=gauspuls(app.t,app.f1);
z=awgn(x,app.SNR,app.amp);
```

```
wo = app.f1/(app.fs/2);
[b, a] = iirnotch(wo,app.BW/(app.fs/2));
app.dataOut = filtfilt(b,a,z);
% forrest
ns=app.c(10000:9999+length(app.dataOut))*(app.amp*30);
app.naturalnoise = filtfilt(b,a,ns);
app.audioout=(app.signal+(app.dataOut.*app.vw/100)+(app.naturalnoise.*app.vn/100))/3;
app.playernoise=audioplayer(app.audioout,app.fs);
play(app.playernoise)
        end

        % Value changed function: TinnitussoundSwitch
        function TinnitussoundSwitchValueChanged(app, event)
app.switchvalue = app.TinnitussoundSwitch.Value;
app.vw = app.VolumeofwhitenoiseSlider.Value;
app.vn = app.VolumeofnaturalsoundSlider.Value;
 switch  app.switchvalue
     case 'Off'
           stop(app.playernoise)
           app.audioout2=((app.dataOut.*app.vw/100)+(app.naturalnoise.*app.vn/100))/2;
           app.playernoise2=audioplayer(app.audioout2,app.fs);
            play(app.playernoise2)
     case 'On'
    stop(app.playernoise2)
     play(app.playernoise)
end

        end

        % Changes arrangement of the app based on UIFigure width
        function updateAppLayout(app, event)
            currentFigureWidth = app.UIFigure.Position(3);
            if(currentFigureWidth <= app.onePanelWidth)
                % Change to a 2x1 grid
                app.GridLayout.RowHeight = {487, 487};
                app.GridLayout.ColumnWidth = {'1x'};
```

```
            app.RightPanel.Layout.Row = 2;
            app.RightPanel.Layout.Column = 1;
        else
            % Change to a 1x2 grid
            app.GridLayout.RowHeight = {'1x'};
            app.GridLayout.ColumnWidth = {271, '1x'};
            app.RightPanel.Layout.Row = 1;
            app.RightPanel.Layout.Column = 2;
        end
    end
end

% Component initialization
methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)

        % Create UIFigure and hide until all components are created
        app.UIFigure = uifigure('Visible', 'off');
        app.UIFigure.AutoResizeChildren = 'off';
        app.UIFigure.Position = [100 100 659 487];
        app.UIFigure.Name = 'MATLAB App';
        app.UIFigure.SizeChangedFcn = createCallbackFcn(app, @updateAppLayout, true);

        % Create GridLayout
        app.GridLayout = uigridlayout(app.UIFigure);
        app.GridLayout.ColumnWidth = {271, '1x'};
        app.GridLayout.RowHeight = {'1x'};
        app.GridLayout.ColumnSpacing = 0;
        app.GridLayout.RowSpacing = 0;
        app.GridLayout.Padding = [0 0 0 0];
        app.GridLayout.Scrollable = 'on';

        % Create LeftPanel
        app.LeftPanel = uipanel(app.GridLayout);
```

```
        app.LeftPanel.Layout.Row = 1;
        app.LeftPanel.Layout.Column = 1;

        % Create UIAxes
        app.UIAxes = uiaxes(app.LeftPanel);
        title(app.UIAxes, 'tinnitus wave')
        ylabel(app.UIAxes, 'Volume')
        zlabel(app.UIAxes, 'Z')
        app.UIAxes.FontSize = 8;
        app.UIAxes.Position = [19 27 232 146];

        % Create PlaystartersoundButton
        app.PlaystartersoundButton = uibutton(app.LeftPanel, 'push');
        app.PlaystartersoundButton.ButtonPushedFcn = createCallbackFcn(app,
@PlaystartersoundButtonPushed, true);
        app.PlaystartersoundButton.Position = [66 397 142 34];
        app.PlaystartersoundButton.Text = 'Play starter sound';

        % Create MimicyourtinnitussoundLabel
        app.MimicyourtinnitussoundLabel = uilabel(app.LeftPanel);
        app.MimicyourtinnitussoundLabel.Position = [68 444 144 36];
        app.MimicyourtinnitussoundLabel.Text = 'Mimic your tinnitus sound';

        % Create VolumeSliderLabel
        app.VolumeSliderLabel = uilabel(app.LeftPanel);
        app.VolumeSliderLabel.HorizontalAlignment = 'right';
        app.VolumeSliderLabel.Position = [20 342 46 22];
        app.VolumeSliderLabel.Text = 'Volume';

        % Create VolumeSlider
        app.VolumeSlider = uislider(app.LeftPanel);
        app.VolumeSlider.Limits = [0.1 0.8];
        app.VolumeSlider.MajorTicks = [0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8];
        app.VolumeSlider.ValueChangedFcn = createCallbackFcn(app, @VolumeSliderValueChanged,
true);
```

```
app.VolumeSlider.ValueChangingFcn = createCallbackFcn(app, @VolumeSliderValueChanging,
true);
app.VolumeSlider.MinorTicks = [0.1 0.12 0.14 0.16 0.18 0.2 0.22 0.24 0.26 0.28 0.3 0.32 0.34
0.36 0.38 0.4 0.42 0.44 0.46 0.48 0.5 0.52 0.54 0.56 0.58 0.6 0.62 0.64 0.66 0.68 0.7 0.72 0.74 0.76 0.78
0.8];
app.VolumeSlider.Position = [87 351 151 7];
app.VolumeSlider.Value = 0.1;

% Create FrequencySliderLabel
app.FrequencySliderLabel = uilabel(app.LeftPanel);
app.FrequencySliderLabel.HorizontalAlignment = 'right';
app.FrequencySliderLabel.Position = [2 282 62 22];
app.FrequencySliderLabel.Text = 'Frequency';

% Create FrequencySlider
app.FrequencySlider = uislider(app.LeftPanel);
app.FrequencySlider.Limits = [1 15];
app.FrequencySlider.ValueChangedFcn = createCallbackFcn(app,
@FrequencySliderValueChanged, true);
app.FrequencySlider.ValueChangingFcn = createCallbackFcn(app,
@FrequencySliderValueChanging, true);
app.FrequencySlider.Position = [85 291 151 7];
app.FrequencySlider.Value = 1;

% Create GeneratetinnitussoundwithwhitenoiseLabel
app.GeneratetinnitussoundwithwhitenoiseLabel = uilabel(app.LeftPanel);
app.GeneratetinnitussoundwithwhitenoiseLabel.Position = [31 217 220 36];
app.GeneratetinnitussoundwithwhitenoiseLabel.Text = 'Generate tinnitus sound with whitenoise';

% Create NextButton
app.NextButton = uibutton(app.LeftPanel, 'push');
app.NextButton.ButtonPushedFcn = createCallbackFcn(app, @NextButtonPushed, true);
app.NextButton.Position = [71 184 142 34];
app.NextButton.Text = 'Next';

% Create RightPanel
```

```matlab
app.RightPanel = uipanel(app.GridLayout);
app.RightPanel.Layout.Row = 1;
app.RightPanel.Layout.Column = 2;

% Create UIAxes2
app.UIAxes2 = uiaxes(app.RightPanel);
title(app.UIAxes2, 'Power Spectrum')
xlabel(app.UIAxes2, 'Frequency (Hz)')
ylabel(app.UIAxes2, 'Power/frequency (dB/Hz)')
zlabel(app.UIAxes2, 'Z')
app.UIAxes2.ZTickLabelRotation = 0;
app.UIAxes2.FontSize = 8;
app.UIAxes2.Position = [11 33 361 215];

% Create AmplitudeEditFieldLabel
app.AmplitudeEditFieldLabel = uilabel(app.RightPanel);
app.AmplitudeEditFieldLabel.HorizontalAlignment = 'right';
app.AmplitudeEditFieldLabel.Position = [148 449 93 22];
app.AmplitudeEditFieldLabel.Text = 'Amplitude';

% Create AmplitudeEditField
app.AmplitudeEditField = uieditfield(app.RightPanel, 'numeric');
app.AmplitudeEditField.Position = [256 448 85 24];

% Create FrequencyEditFieldLabel
app.FrequencyEditFieldLabel = uilabel(app.RightPanel);
app.FrequencyEditFieldLabel.HorizontalAlignment = 'right';
app.FrequencyEditFieldLabel.Position = [142 420 62 22];
app.FrequencyEditFieldLabel.Text = 'Frequency';

% Create FrequencyEditField
app.FrequencyEditField = uieditfield(app.RightPanel, 'numeric');
app.FrequencyEditField.ValueDisplayFormat = '%.2f';
app.FrequencyEditField.Position = [219 419 122 24];

% Create BandwidthEditFieldLabel
```

```
        app.BandwidthEditFieldLabel = uilabel(app.RightPanel);
        app.BandwidthEditFieldLabel.HorizontalAlignment = 'right';
        app.BandwidthEditFieldLabel.Position = [144 388 63 22];
        app.BandwidthEditFieldLabel.Text = 'Bandwidth';

        % Create BandwidthEditField
        app.BandwidthEditField = uieditfield(app.RightPanel, 'numeric');
        app.BandwidthEditField.Position = [222 387 120 24];
        app.BandwidthEditField.Value = 100;

        % Create VolumeofnaturalsoundSliderLabel
        app.VolumeofnaturalsoundSliderLabel = uilabel(app.RightPanel);
        app.VolumeofnaturalsoundSliderLabel.HorizontalAlignment = 'right';
        app.VolumeofnaturalsoundSliderLabel.Position = [23 311 135 22];
        app.VolumeofnaturalsoundSliderLabel.Text = 'Volume of natural sound';

        % Create VolumeofnaturalsoundSlider
        app.VolumeofnaturalsoundSlider = uislider(app.RightPanel);
        app.VolumeofnaturalsoundSlider.ValueChangedFcn = createCallbackFcn(app,
@VolumeofnaturalsoundSliderValueChanged, true);
        app.VolumeofnaturalsoundSlider.Position = [179 320 158 7];
        app.VolumeofnaturalsoundSlider.Value = 100;

        % Create VolumeofwhitenoiseSliderLabel
        app.VolumeofwhitenoiseSliderLabel = uilabel(app.RightPanel);
        app.VolumeofwhitenoiseSliderLabel.HorizontalAlignment = 'right';
        app.VolumeofwhitenoiseSliderLabel.Position = [31 269 126 22];
        app.VolumeofwhitenoiseSliderLabel.Text = 'Volume of  whitenoise ';

        % Create VolumeofwhitenoiseSlider
        app.VolumeofwhitenoiseSlider = uislider(app.RightPanel);
        app.VolumeofwhitenoiseSlider.ValueChangedFcn = createCallbackFcn(app,
@VolumeofwhitenoiseSliderValueChanged, true);
        app.VolumeofwhitenoiseSlider.Position = [178 278 158 7];
        app.VolumeofwhitenoiseSlider.Value = 100;
```

```
            % Create TinnitussoundSwitchLabel
            app.TinnitussoundSwitchLabel = uilabel(app.RightPanel);
            app.TinnitussoundSwitchLabel.HorizontalAlignment = 'center';
            app.TinnitussoundSwitchLabel.FontSize = 10;
            app.TinnitussoundSwitchLabel.Position = [79 6 71 22];
            app.TinnitussoundSwitchLabel.Text = 'Tinnitus sound';

            % Create TinnitussoundSwitch
            app.TinnitussoundSwitch = uiswitch(app.RightPanel, 'slider');
            app.TinnitussoundSwitch.ValueChangedFcn = createCallbackFcn(app,
@TinnitussoundSwitchValueChanged, true);
            app.TinnitussoundSwitch.FontSize = 10;
            app.TinnitussoundSwitch.Position = [203 7 45 20];
            app.TinnitussoundSwitch.Value = 'On';

            % Create BandwithSlider_2Label
            app.BandwithSlider_2Label = uilabel(app.RightPanel);
            app.BandwithSlider_2Label.HorizontalAlignment = 'right';
            app.BandwithSlider_2Label.Position = [103 354 56 22];
            app.BandwithSlider_2Label.Text = 'Bandwith';

            % Create BandwithSlider_2
            app.BandwithSlider_2 = uislider(app.RightPanel);
            app.BandwithSlider_2.Limits = [100 5000];
            app.BandwithSlider_2.ValueChangedFcn = createCallbackFcn(app,
@BandwithSlider_2ValueChanged, true);
            app.BandwithSlider_2.ValueChangingFcn = createCallbackFcn(app,
@BandwithSlider_2ValueChanging, true);
            app.BandwithSlider_2.Position = [180 363 158 7];
            app.BandwithSlider_2.Value = 100;

            % Create dBLabel
            app.dBLabel = uilabel(app.RightPanel);
            app.dBLabel.Position = [356 449 25 22];
            app.dBLabel.Text = 'dB';
```

```
            % Create kHzLabel
            app.kHzLabel = uilabel(app.RightPanel);
            app.kHzLabel.Position = [357 420 26 22];
            app.kHzLabel.Text = 'kHz';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = tinnitus_v3

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
```

```
    end
end
```