```
clc;
clear;
close all;
Fs = 4000;
Channels = 1;
bits = 16;
r = audiorecorder(Fs, bits, Channels);
duration = 8;
disp('Recording started');
recordblocking(r, duration);
disp('Recording Stopped');
X = getaudiodata(r);

% Create an audioplayer object for the original signal
player_original = audioplayer(X, Fs, bits);

% Play the original signal
disp('Playing Original Signal...');
playblocking(player_original); % Play and wait for completion

% Add a 3-second pause
pause(3);

% Define filter parameters
cutoff_frequency = 500; % Adjust this to your desired cutoff frequency in Hz
filter_order = 4; % Filter order (adjust as needed)

% Design a high-pass Butterworth filter
[b, a] = butter(filter_order, cutoff_frequency / (Fs / 2), 'high');

% Apply the filter to the recorded audio (X)
filteredX = filter(b, a, X);

% Create an audioplayer object for the filtered signal
player_filtered = audioplayer(filteredX, Fs, bits);

% Play the filtered signal
disp('Playing Filtered Signal...');
playblocking(player_filtered); % Play and wait for completion

% Create time vectors for both original and filtered audio
t = 0:1/Fs:(length(X)-1)/Fs;
```

```
t_filtered = 0:1/Fs:(length(filteredX)-1)/Fs;

% Plot the time-domain signals
subplot(2,2,1);
plot(t, X, 'LineWidth', 1.5);
xlabel('Time (sec)');
ylabel('Amplitude');
title('Time Domain Plot of the Recorded Signal');

subplot(2,2,2);
plot(t_filtered, filteredX, 'LineWidth', 1.5);
xlabel('Time (sec)');
ylabel('Amplitude');
title('Time Domain Plot of the Filtered Signal');

% Compute frequency-domain representations
n = length(X);
F = 0:(n-1)*Fs/n;
Y = fft(X, n);

n_filtered = length(filteredX);
F_filtered = 0:(n_filtered-1)*Fs/n_filtered;
Y_filtered = fft(filteredX, n_filtered);

% Shift and compute magnitudes
F_0 = (-n/2:n/2-1)*(Fs/n);
Y_0 = fftshift(Y);
AY_0 = abs(Y_0);

F_filtered_0 = (-n_filtered/2:n_filtered/2-1)*(Fs/n_filtered);
Y_filtered_0 = fftshift(Y_filtered);
AY_filtered_0 = abs(Y_filtered_0);

% Plot the frequency-domain signals
subplot(2,2,3);
plot(F_0, AY_0, 'Linewidth', 1.5);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Domain Plot of the Original Signal');

subplot(2,2,4);
plot(F_filtered_0, AY_filtered_0, 'Linewidth', 1.5);
xlabel('Frequency (Hz)');
```

```
ylabel('Amplitude');
title('Frequency Domain Plot of the Filtered Signal');

filename = 'myvoice.wav';
audiowrite(filename, X, Fs);
```