Group: B-1

Roll No.: 1909031, 1909032, 1909033, 1909034, 1909035, 1909036

Block Diagram of the Implemented Design:

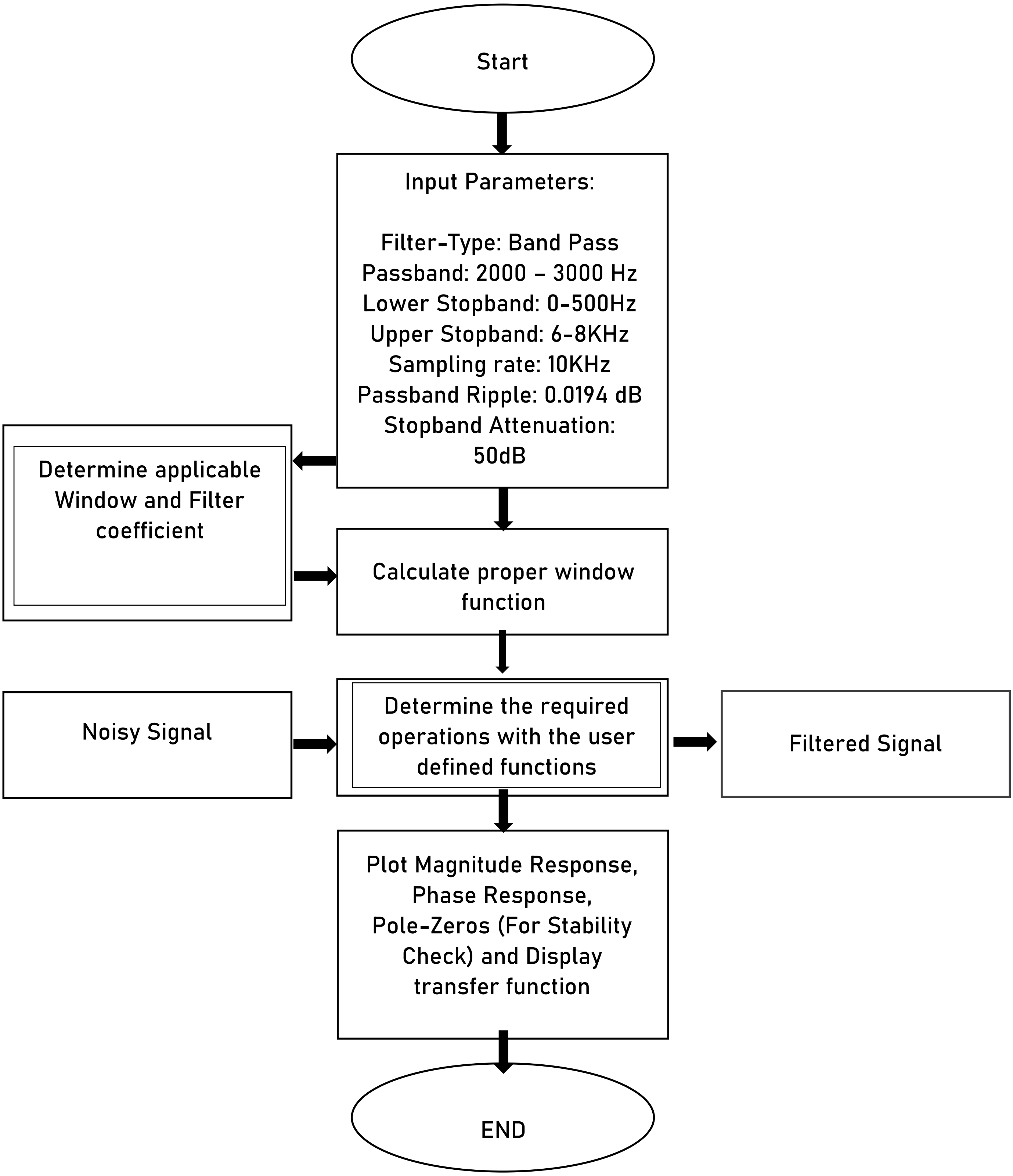

Figure 3.1: Block Diagram of designed Band Pass FIR filter

Code:

main.m:

```
[b]=select_filter_type('bpf',10000,0,0,500,6000,2000,3000
,25);
[window_type,m]=select_window(0.0194,50)
[num,h_win,w_win]=filter_coefficients(b,window_type,m);

num

figure(1)
subplot(311)
plot(w_win,20*log10(abs(h_win)))
grid;
xlabel('Frequency (Hz)');
ylabel('Magnitude Response (dB)');
subplot(312)
plot(w_win, 180*angle(h_win)/pi);grid;
xlabel('Frequency (Hz)');
ylabel('Phase (degrees)');
subplot(313),
plot(w_win,180*unwrap(angle(h_win))/pi);grid;
xlabel('Frequency (Hz)');
ylabel('Phase (degrees)');


figure(2)
subplot(1,1,1);
zplane(num,1);
title('Pole Zero plot(FIR Filter)');
grid on;




% Generate the input signal
fs=10000;
t = 0:1/fs:0.1;  % Time vector from 0 to 0.1 seconds
input_signal = exp(0.2*t) + sqrt(2)*cos(2*pi*70*t +40);
B=reshape(h_win.',1,[]); %change matrix to vector
% Filter the noisy signal using the FIR filter for the
signal
noisy_signal =
input_signal+sqrt(0.1)*randn(size(input_signal));
filtered_signal = filter(h_win,1,noisy_signal);
```

```
% Plot the original signal, noisy signal, and filtered
signal for the signal

figure(3);
subplot(3, 1, 1);
plot(t, input_signal);
title('Original Signal');
xlabel('Time (s)');
ylabel('Amplitude');
subplot(3, 1, 2);
plot(t, noisy_signal);
title('Noisy Signal');
xlabel('Time (s)');
ylabel('Amplitude');
subplot(3, 1, 3);
plot(t, filtered_signal);
title('Filtered Signal');
xlabel('Time (s)');
ylabel('Amplitude');
```

select_filter_type.m:

```
function
[num]=select_filter_type(filter_type,fs,lower_cutoff,uppe
r_cutoff,stop_freq1,stop_freq2,pass_freq1,pass_freq2,n)

switch filter_type
    case 'lpf'
        M=(n-1)/2;
        fl=lower_cutoff;
        wl = 2*pi*fl/fs;
        num = [];
        for k = 0:1:M
        pos = k+M+1;
        if k == 0
        num(pos) = wl/pi;
        else
        num(pos) = (sin(k*wl)/(k*pi));
        new_pos = pos-(2*k);
        num(new_pos) = num(pos);
        end
        end
        %[hz,w]=freqz(num,[1],512,fs);


     case 'hpf'
        M=(n-1)/2;
        fh=upper_cutoff;
        wh = 2*pi*fh/fs;
```

```
    num = [];
    for k = 0:1:M
    pos = k+M+1;
    if k == 0
    num(pos) = (pi-wh)/pi;
    else
    num(pos) = -(sin(k*wh)/(k*pi));
    new_pos = pos-(2*k);
    num(new_pos) = num(pos);
    end
    end
    %[hz,w]=freqz(num,[1],512,fs);


 case 'bsf'
     M=(n-1)/2;
     fl=lower_cutoff;
     fh=upper_cutoff;
     wl=2*pi*fl/fs;
     wh = 2*pi*fh/fs;
     num = []
     for k = 0:1:M
     pos = k+M+1;
     if k == 0
     num(pos) = pi-wh+wl/pi;
     else
     num(pos) = -(sin(k*wh)/(k*pi))+sin(k*wl)/(k*pi)
     new_pos = pos-(2*k)
     num(new_pos) = num(pos)
     end
     end
     %[hz,w]=freqz(num,[1],512,fs);

case 'bpf'
    M=(n-1)/2;
    fl=(stop_freq1+pass_freq1)/2;
    fh=(stop_freq2+pass_freq2)/2;
    wl=2*pi*fl/fs;
    wh = 2*pi*fh/fs;
    num = [];
    for k = 0:1:M
    pos = k+M+1;
    if k == 0
    num(pos) = (wh-wl)/pi;
    else
    num(pos) = (sin(k*wh)/(k*pi))-sin(k*wl)/(k*pi);
    new_pos = pos-(2*k);
```

```
        num(new_pos) = num(pos);
        end
        end
        %[hz,w]=freqz(num,[1],512,fs);
     end
end
```

select_window.m:

```
function [window_type,M ]= select_window(passband_ripple,
stopband_attenuation)
f_stop=500;
f_pass=2000;
fs=10000;
    if (passband_ripple >= 0.7416) &&
(stopband_attenuation > 0 && stopband_attenuation <= 21)
        % Rectangular window
        N = 0.9*fs/ abs(f_stop-f_pass);
        N=check_digit(0,N);
        M = (N - 1) / 2;
        window_type=1;
    elseif (passband_ripple >= 0.0546) &&
(stopband_attenuation > 21 && stopband_attenuation <= 44)
        % Hanning window
        N = 3.1*fs/ abs(f_stop-f_pass);
        N=check_digit(0,N);
        M = (N - 1) / 2;
        window_type=2;
    elseif (passband_ripple >= 0.0194) &&
(stopband_attenuation > 44 && stopband_attenuation <= 53)
        % Hamming window
        N = 3.3*fs/ abs(f_stop-f_pass);
        N=check_digit(0,N);
        M = (N - 1) / 2;
        window_type=3;
    elseif (passband_ripple >= 0.0017) &&
(stopband_attenuation > 53 && stopband_attenuation <= 74)
        % Blackman window
       N = 5.5*fs/ abs(f_stop-f_pass)
       N=check_digit(0,N);
        M = (N - 1) / 2;
        window_type=4;
    else
         disp('error');
```

```
    end
end

function x = check_digit(m,n)
k = round(max(m,n));
if mod(k,2)==0
x = k+1;
else
x = k;
end
end
```

filter_coefficients.m:

```
function [num,h_win, w_win] =
filter_coefficients(h,window_type,M)
n=25;
fs=10000;
    if window_type==1

        % Rectangular window
        w=[ones(size(h))];
        num=h.*w;
        [h_win,w_win]=freqz(num,[1],512,fs);

    elseif window_type==2
        % Hanning window
        w=[ones(size(h))];
        for k = 0:1:M
        pos = k+M+1;
        w(pos) = 0.5 + 0.5*cos(k*pi/M);
        new_pos = pos-(2*k);
        w(new_pos) = w(pos);
        end
        num= h.*w;
        [h_win, w_win] = freqz(num, [1], 512, fs);

    elseif window_type==3
        % Hamming window
        w=[ones(size(h))];
        for k=0:1:M
        pos=k+M+1;
        w(pos)=0.54+0.46*cos(k*pi/M);
        new_pos=pos-(2*k);
```

```
        w(new_pos)=w(pos);
        end
        num=h.*w;
        [h_win,w_win]=freqz(num,[1],512,fs);
    else
        % Blackman window
        w=[ones(size(h))];
        for k = 0:1:M
        pos = k+M+1;
        w(pos) = 0.42 + 0.5*cos(k*pi/M) +
0.08*cos(2*k*pi/M);
        new_pos = pos-(2*k);
        w(new_pos) = w(pos);
        end
        num = h.*w;
        [h_win, w_win] = freqz(num,[1],512,fs);


    end

    %w_win = linspace(0, fs/2, length(h_win));
end
```

Obtained figures:

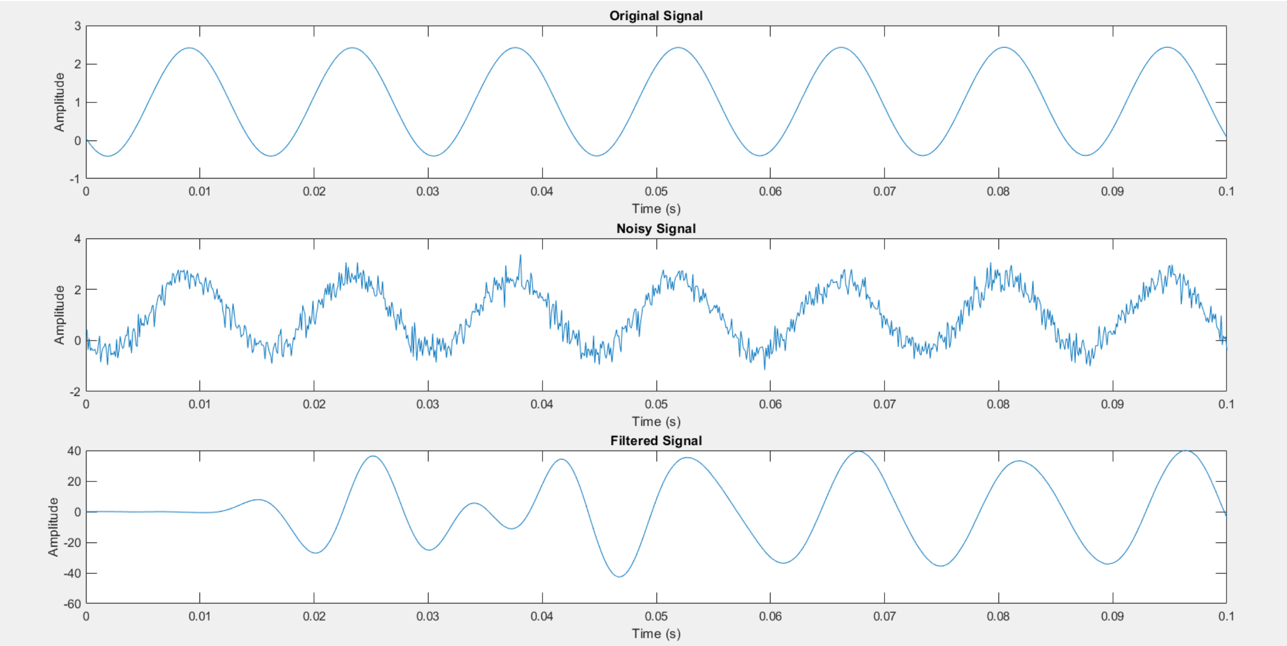

Figure 3.2: Waveform of the original signal ,noise signal and filtered signal

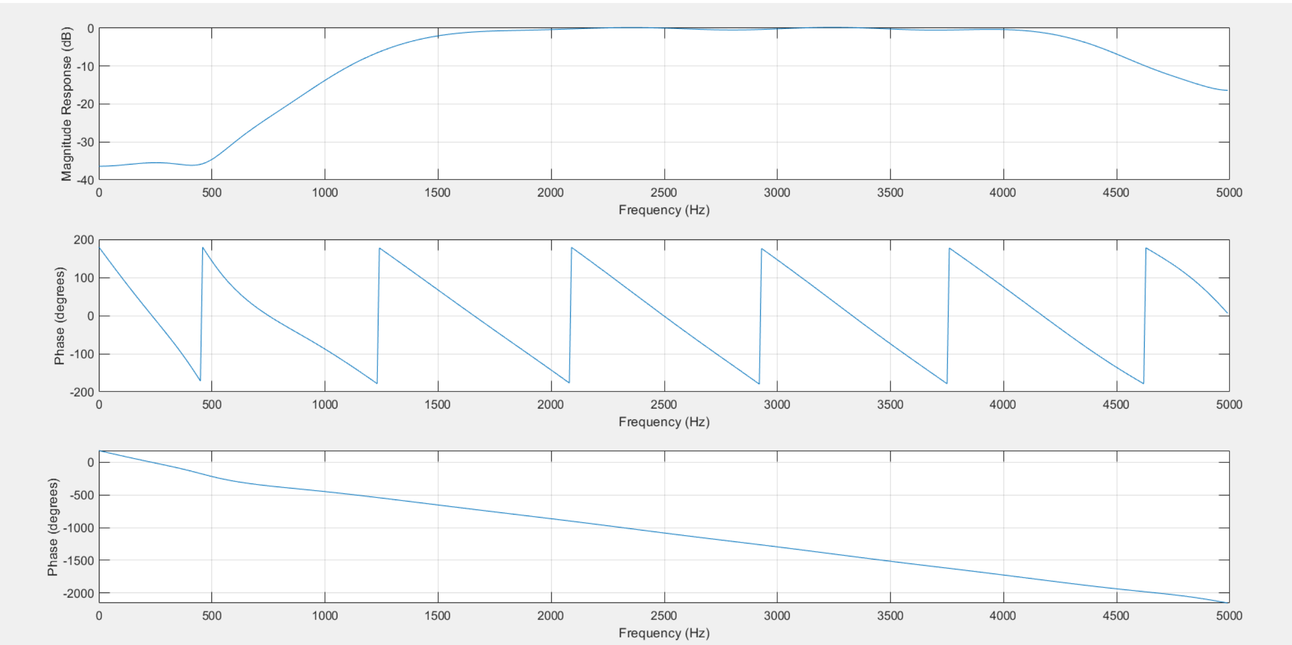

Figure 3.3: Magnitude Response & Phase Response of the designed FIR filter

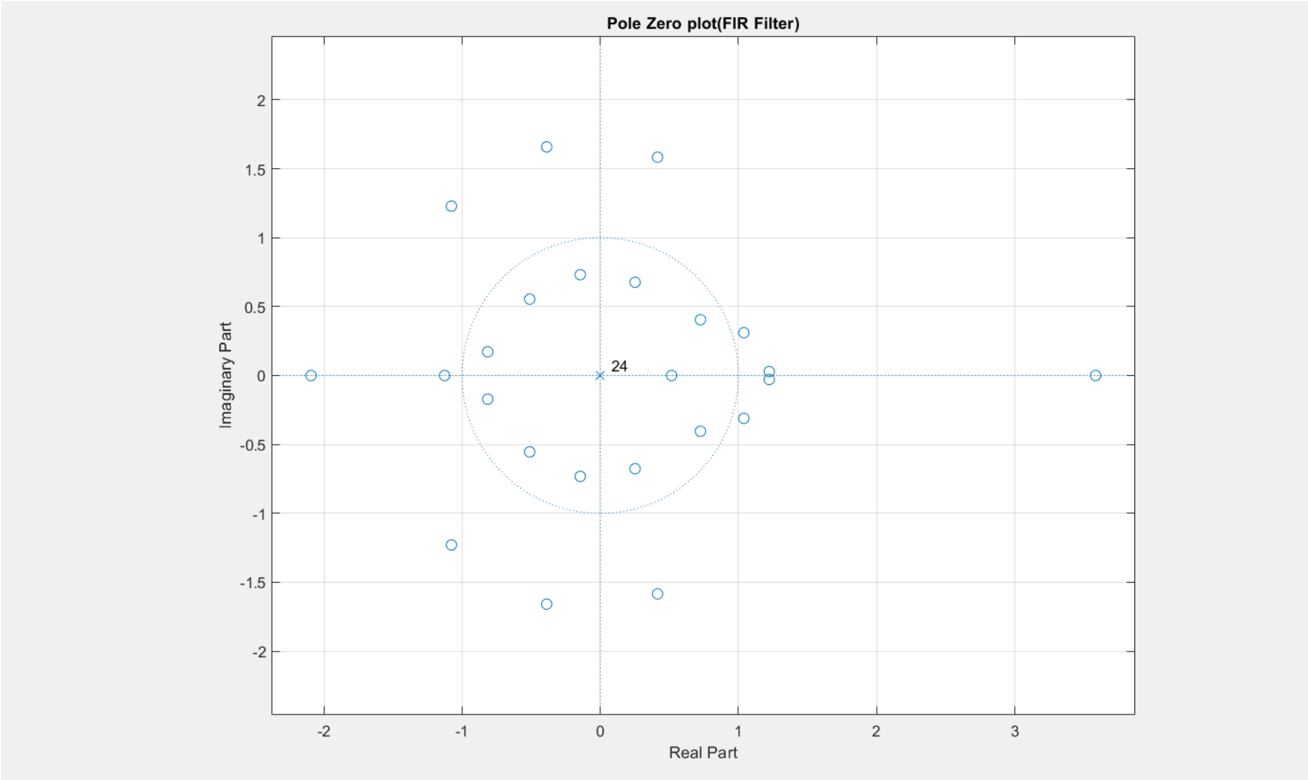

Figure 3.4: Pole-Zeros Plot of the FIR Filter (Stability Check)