Table of Contents

Name: SaiKrishna Saketh #723400 Group B

```
%RTL_FM_RX_DIFF
%
%Simple FM radio receiver
%No stereo, no de-emphasis filter
%By R.W.

clear all, close all
```

Radio parameters

FM transmitter

```
%expFreq = 89.5e6;
% YLE 1
%expFreq = 87.9e6;
% YLE Puhe
expFreq = 103.7e6;
% YLE Radio Suomi
%expFreq = 94e6;

% Front-end sampling rate

FESR = 240e3;
nSample = 4096;
nFrame = 12e2;

hSDRrRx = comm.SDRRTLReceiver(...
    'RadioAddress', '0',...
    'CenterFrequency',     expFreq, ...
    'EnableTunerAGC',      true, ...
    'SampleRate',          FESR, ...
    'SamplesPerFrame',     nSample, ...
    'FrequencyCorrection', 75, ...
    'OutputDataType',      'double')
fprintf('\n')

hSpectrumAnalyzer = dsp.SpectrumAnalyzer(...
    'Name',             'Actual Frequency Offset',...
    'Title',            'Actual Frequency Offset', ...
    'SpectrumType',     'Power density',...
    'FrequencySpan',    'Full', ...
```

```
    'SampleRate',        FESR, ...
    'YLimits',           [-60,0],...
    'SpectralAverages', 10, ...
    'FrequencySpan',     'Start and stop frequencies', ...
    'StartFrequency',   -50e3, ...
    'StopFrequency',     50e3,...
    'Position',          figposition([50 30 30 40]));
```

```
hSDRrRx =

  comm.SDRRTLReceiver with properties:

             RadioAddress: '0'
          CenterFrequency: 103700000
           EnableTunerAGC: true
               SampleRate: 240000
           OutputDataType: 'double'
          SamplesPerFrame: 4096
      FrequencyCorrection: 75
          EnableBurstMode: false
```

We know that N = N1xN2

```
NDEC = 6;
NDEC1 = 3;
NDEC2 = 2;
```

Manually Inputting the decimation factor as we know that we should get down the sampling freq from 240k to 40~44k

```
fmax = 40e+03;
nyq = fmax/2;
```

The approach is two design two filters which can first downsampple the signal from 240k to 90k~~ and then use the next filter for our range of frequencies that is 15k~~ The algorithm used for both FLOW1 and FLOW2 are 48th order with frequency magnitude characterstics specified by how we set out cutoff frequency, in the second vector containing the desired magnitude response at each of the points specified in the vector where cutoff frequency is specifies. Frequency sampling-based FIR filter design is adopted where FLOW holds the values for filter coeffcients which consequently is used to filter

```
FLOW1 = fir2(48, [0 95e3/200e3 98e3/200e3 1], [1 1 0 0]);
FLOW2 = fir2(48, [0 15e3/nyq 17e3/nyq 1], [1 1 0 0]);
```

Computatational Saving

```
N_Stage1= ceil(NDEC1/2)*FESR;
N_Stage2= ceil(NDEC2/2)*95e3;
N_twostage = N_Stage1 +N_Stage2;

N = ceil(NDEC/2)*FESR;
fprintf(' Comparsion of Number of Multiplactions compared to Two
 Stage is %d, while the Number of Multiplactions for No-stage is %d
 \n',N_twostage,N)
```

```
fvtool(FLOW1);
fvtool(FLOW2);
```

Comparsion of Number of Multiplactions compared to Two Stage is 575000, while the Number of Multiplactions for No-stage is 720000

Differentator is implmented as follows by windowing, a simple windowing technique was used to implment, the diffrentiation is vector multiplication of impulse response and hamming window

```
for n = -20:1:20
    if(n == 0)
        imp(n+21) = 0;

    else
        imp(n+21) = (((-1)^n)/(n));
    end
end

% figure
% stem(h);

figure
win = hamming(length(imp));
stem(win);

FDIFF = imp.*win';
% stem(FDIFF)
% fvtool(FDIFF)
```

Audio object to listen to the radio Max. sampling freuqency for aidio is 44.1 KHz so the recieved signal must be decimated

```
hAudio = audioDeviceWriter(FESR/NDEC,'BufferSize',ceil(nSample*2/
NDEC));
% List available audio outputs
%getAudioDevices(hAudio);
```

Stream Processing

```
if ~isempty(sdrinfo(hSDRrRx.RadioAddress))

 fprintf('Receive time %f [s]    \n', nSample/FESR*nFrame)
    %rw filter_mem1 = zeros(1, length(FLOW)-1);
    filter_mem_stage1 = zeros(1, length(FLOW1)-1);
    filter_mem_stage2 = zeros(1, length(FLOW2)-1);
    filter_mem2 = zeros(1, length(FDIFF)-1);
 tic;
    % Run as real time as possible. Variables needn't declared bu
 don't
    % change the size of the array withi
```

```
    for iFrame = 1 : nFrame
        rxSig = step(hSDRrRx);
        rxSig = rxSig - mean(rxSig);  % Remove DC component

        % Display received frequency spectrum
        hSpectrumAnalyzer(rxSig);

         % FLOW = low-pass filter
        % FDIFF = differentiator
        % NDEC = decimator factor
        % Here the signal is first differentiated and then low-pass
 filtered
        % Can you swap the order? Which order is better or is there
 any
        % difference
        % Is it possible to swap the order of down-sampling and
 filtering
        % to make operation more efficient?
        % Is it possible to low-pass filter before envelope detection
        %
        % This shows one-stage decimator only. In addition you should
        % implement two-stage decimator and compare the number of
        % operations in one-stage vs. two-stage.

        %[lpSig,filter_mem1] = filter(FLOW,1,rxSig,filter_mem1);
        [diffSig, filter_mem2] = filter(FDIFF,1,rxSig,filter_mem2);
        aSig = abs(diffSig);
        %rw[lpSig,filter_mem1] = filter(FLOW,1,aSig,filter_mem1);
        [lpSig1,filter_mem_stage1] =
 filter(FLOW1,1,aSig,filter_mem_stage1);
        lpsig_3=lpSig1(1:NDEC1:end);
        [lpSig2,filter_mem_stage1] =
 filter(FLOW2,1,lpsig_3,filter_mem_stage2);

        auSig = lpSig2(1:NDEC2:end);

        % Underrun may occure in the loop
        nUnderrun = hAudio(auSig);
        if nUnderrun > 0
            fprintf('Audio player queue underrun by %d samples.
\n',nUnderrun);
        end
    end
else
    warning(message('SDR:sysobjdemos:MainLoop'))
end
fprintf('Clock receive time %f [s]\n', toc)
```

```
Receive time 20.480000 [s]
Audio player queue underrun by 32101 samples.
Audio player queue underrun by 683 samples.
Clock receive time 26.612538 [s]
```

Release all System objects

```
release(hSDRrRx);
clear hSDRrRx
release(hAudio);
```

The Signal Processing blocks used are diffrentiator,,envelope detector,LPF 1, Decimation 1,LPF 2, Decimation 2 while you hear the output after NDEC2 which is evident above after trying several combinations.

Published with MATLAB® R2018a