Table of Contents


```
%RX_FM
%
% 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', 60, ...
    '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: 60
          EnableBurstMode: false
```

Designing a low-pass filter and determining the decimator factor to extract 15 KHz bandwidth of FM transmission Manually Inputting the decimation factor as we know that we should get down the sampling freq from 240k to 40~44k

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

Task & Explanation: Describe briefly the algorithm used in the filter design function The algorithm used is 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 The FM receiver code is made in such a way, the signal processing blocks are firstly, it performs demodultion, followed by filtering and finally decimating it.

```
FLOW = fir2(48, [0 15e3/nyq 17e3/nyq 1], [1 1 0 0]);%filter esign
%FLOW = fir1(1, (FESR+ 15e3)/(2*FESR));
%freqz(FLOW,1);
h=fvtool(FLOW,1);
```

```
% 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)
    filter_memory = zeros(1, length(FLOW)-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);

        % Your FM receiver here
        % FLOW = low-pass filter
        % NDEC = decimator factor
        % Last argument is the type of the demodulator.
        filterdelay = filter([0 1],1,rxSig);
        %filterdelay = delayseq(rxSig,1);
        fmSig = angle(filterdelay .* conj(rxSig));
        [lpSig,filter_memory] = filter(FLOW,1,fmSig,filter_memory);
        % Signal can be decimated before or after (but not both) the
 detector.
        % The code contains both alternatives
        % Which one is better?
        %modSig = lpSig(1:NDEC:end);
        %fmSig = rw_fmrx(lpSig,1);
        %fmSig = rw_fmrx(modSig,1);


        %filterdelay = delayseq(modSig,1);

        aSig = lpSig(1:NDEC:end); %fmSig
        % Underrun may occure in the loop
        nUnderrun = hAudio(aSig);
        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 12977 samples.
Audio player queue underrun by 1366 samples.
Clock receive time 25.147476 [s]
```

Release all System objects

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

Published with MATLAB® R2018a