Table of Contents

Simple FM radio receiver

```
%RTL_FM
%
% No stereo, no de-emphasis filter
% By R.W.

clear all, close all

% Script to define some essential parameters and functions you need to
% figure yourself. The script is included as a p-code file so that the
% template works. You should implement your own version, though
%
% You can comment out the script and write your own functions. In this
% exercise they are 1) the low-pass filter used before down-sampling and
% 2) the discriminator implementing the FM receiver.
%rw_ini_fm
```

Radio parameters

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

% Front-end sampling frequency and decimation factor. Can be changed at
% will like the other parameters below.
% Usually, the larger the sampling rate, the better.
% The following may depend on the type of the dongle and its tuner:
% SampleRate R must conform to the following conditions: 225e3 < R <= 300e3
% or 900e3 < R <= 3200e3
FESR = 240e3;
% Down-sampling/decimation factor
NDEC = 8;

% #Samples to read at one round
nSample = 4096*4;
% Number of  vectors/frames to read
```

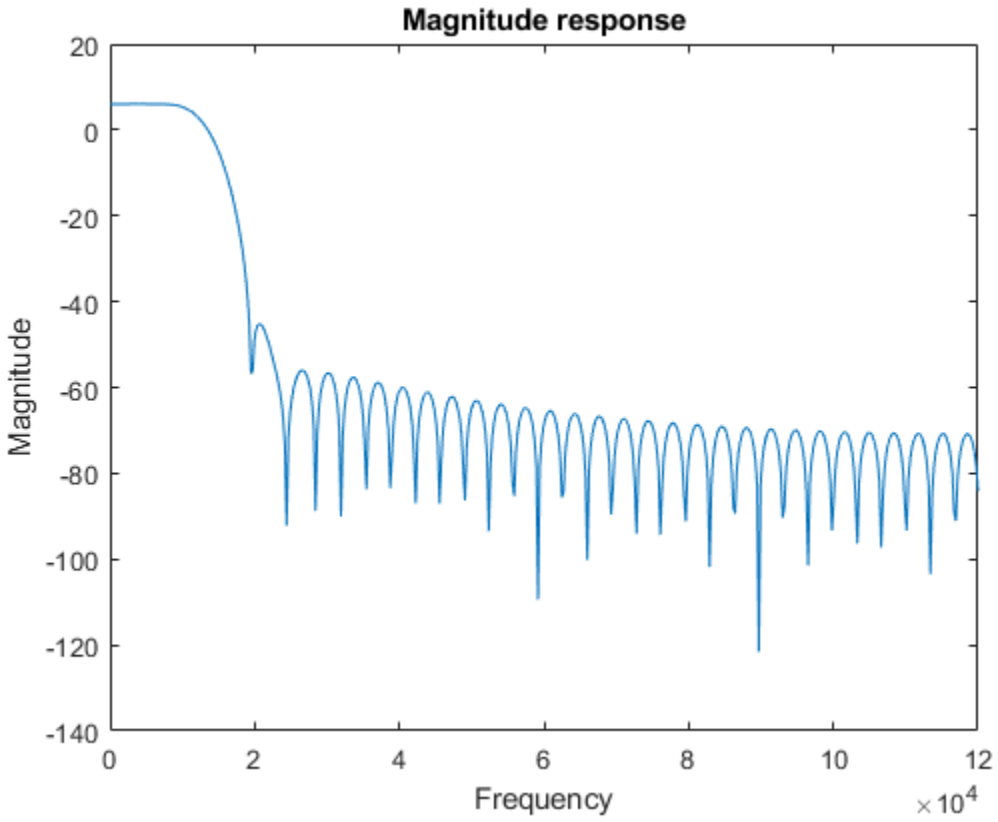
```
nFrame = 2e3;

% Your dongle's frequency offset correction here
% Parts per million = frequency_offset/carrier_frequency * million
% Must be integer
PPM = 0;

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

% If your Matlab version does not have figposition() just comment it out
hSpectrumAnalyzer = dsp.SpectrumAnalyzer(...
    'Name',             'FM signal',...
    'Title',            'FM signal', ...
    'SpectrumType',     'Power density',...
    'FrequencySpan',    'Full', ...
    'SampleRate',       FESR, ...
    'YLimits',          [-50,0],...
    'SpectralAverages', 10, ...
    'FrequencySpan',    'Start and stop frequencies', ...
    'StartFrequency',   -50e3, ...
    'StopFrequency',    50e3,...
    'Position',         figposition([50 30 30 40]));


% Audio object to listen to the radio
% Max. sampling freuqency depends on the hardware
% Determine a suitable decimation factor NDEC and front-end sampling ratio
% FESR
% FESR/NDEC had to be integer, but not any more in new Matlab versions
hAudio = audioDeviceWriter(FESR/NDEC,'BufferSize',ceil(nSample*2/NDEC));
% List available audio outputs
getAudioDevices(hAudio);
```

```
hSDRrRx =

  comm.SDRRTLReceiver with properties:

            RadioAddress: '0'
        CenterFrequency: 87900000
         EnableTunerAGC: true
             SampleRate: 240000
         OutputDataType: 'double'
        SamplesPerFrame: 16384
    FrequencyCorrection: 0
        EnableBurstMode: false
```

Display the low-pass filter. The DC gain of the filter is normalized to 1 in

linear scale and 0 in dB scale rw_window is a function that returns the low-pass filter coefficients.

```
FLOW = rw_window(FESR); % input sampling rate
HH = freqz(FLOW,1);
ww = linspace(0,FESR/2,length(HH));
plot(ww,20*log10(abs(HH))), ylabel Magnitude, xlabel Frequency
title('Magnitude response')
```

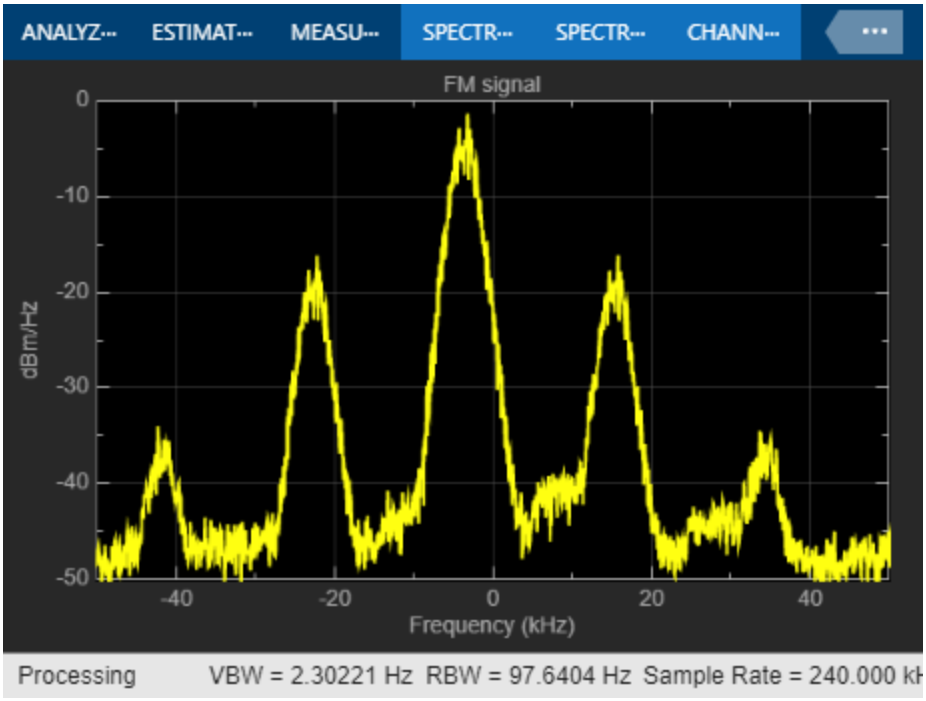


Stream Processing

```
if isempty(sdrinfo(hSDRrRx.RadioAddress))
    error(message('SDR:sysobjdemos:MainLoop'))
end

fprintf('Receive time %f [s]   \n', nSample/FESR*nFrame)

% Memory retains the state of the filter between the calls. No notable
% effect in the demodulatd signal
```

```
filter_memory = zeros(1, length(FLOW)-1);
% Timing the loop
tic;

% Run as real time as possible. Variables needn't declared bu don't
% change the size of the array within the loop
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 coefficient (vector)
    % NDEC = decimator factor
    % Signal can be decimated before or after (but not both) the detector.
    % The code contains both alternatives
    % Which one is better?

    % Decimation after discriminator
    fmSig = rw_fmrx(rxSig);
    [lpSig,filter_memory] = filter(FLOW,1,fmSig,filter_memory);
    aSig = lpSig(1:NDEC:end);

    % Discriminator before downsampling
    % [lpSig,filter_memory] = filter(FLOW,1,rxSig,filter_memory);
    % fmSig = rw_fmrx(lpSig);
    % aSig = fmSig(1:NDEC:end);
    %
    % Decimation before discriminator
    % [lpSig,filter_memory] = filter(FLOW,1,rxSig,filter_memory);
    % aSig = rw_fmrx(lpSig(1:NDEC:end));

    % Underrun may occure in the loop
    % Arbitrary scaling of the signal amplitude controls the volume
    nUnderrun = hAudio(0.5*aSig);
    if nUnderrun > 0
        fprintf('Audio player queue underrun by %d samples.\n',nUnderrun);
    end
end

fprintf('Clock receive time %f [s]\n', toc)
```

```
Receive time 136.533333 [s]
Clock receive time 140.672838 [s]
```



Release all System objects

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

Answer for the best combination

The best option is the combination: Discriminator, Low-pass Filter, and then do the down samping. It is because that the low-pass filter and the down sampling will reduce the quality of signal, making the sound quality worse. However, this method require more computational resources because the discriminator operates at the original high sampling rate.

Published with MATLAB® R2023a