ECE 5650/4650 Computer Project 2
Channelizer for AM Radio

This project is to be treated as a take-home exam, meaning each student or student **team of at most three** (undergraduates students only) is to do his/her own work without consulting others. The grading for this second project is merged with the first project to create the *projects grade category*. The computer projects category can count up to 20% percent of the final grade (details given below). **The project due date is 12:00 PM Wednesday, December 17, 2025 (Final Exam week)**.

From the syllabus the grading options are as follows: (1) Computer projects 20%, final exam 25%; (2) Computer project #2 15%, final exam 30%. The homework and exam percentages are constant over the two options.

Introduction

In this project you will be investigating the use of a *channelizer* filter component to process signals from a software defined radio (SDR). The main idea behind the channelizer is to break the SDR RF capture bandwidth into bandpass filtered sub-bands that isolate adjacent signals of interest. The filtering allows the isolated signal(s) to either be further channelized or moved directly to a demodulation and detection stage. In digital communications applications the sub-bands may contain data streams each associated with a sub-band modulated carrier. The desire for concurrent processing is obvious if the information in each sub-band stream is destined to say an individual user. In this project signals the SDR captures are the entire AM broadcast band running from 470 to 1670 kHz. This band contains 120 channels spaced every 10 kHz. Using a channelizer on broadcast AM seems pointless unless you have a need to concurrently recover the analog message from two or more stations. I do not listen to more than one station at a time, but with concurrency you never miss anything. With real-time word spotting you could say identify how information spreads. In this project the SDR AM radio capture is stored in a file so all the signal processing can be performed non-real-time in a Jupyter notebook. To support the project tasks a sample Jupyter notebook can be found in the accompanying ZIP package `Project2_f2025.zip`.

Background Theory

A block diagram of the function `fft_filt_bank()` is shown below in Figure 1. In the project at

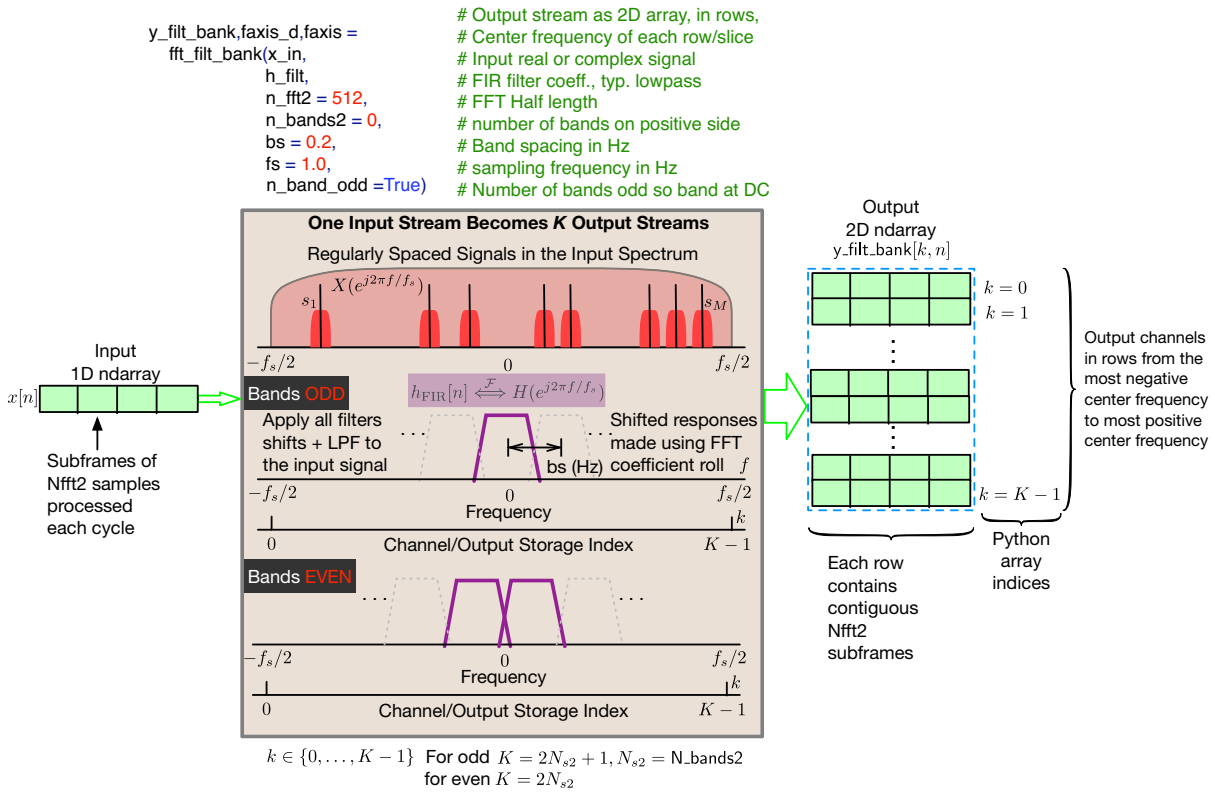

Figure 1: Filter bank function block diagram explaining the functionality for an odd number of bands or an even number of bands.

hand, two stages of filtering using this function will be implemented to break the broadcast AM radio band into individual channels of width ~10 kHz that can then be demodulated in isolation to obtain the underlying message signal if it is present. The overall system block diagram is presented a little later. We first explore a single filter bank using white noise testing using both an even number of bands and an odd number. The odd number case is utilized in this project, both forms are of general interest. To be clear the odd number case always places one filter band centered at $f = 0$ Hz.

With the filter bank/channelizer function, contained in the `sigsys` module, we repeatedly filter the input signal with frequency shifted versions of the input FIR filter, `h_filt`. The frequency shift values insure that a filter bank is formed with band center spacing of `bs` Hz. The design of the filter itself controls the amount of frequency response overlap between adjacent filter bank channels.

This function is based on transform domain filtering as described ECE5650 notes Chapter 7. The relevant figure is shown below in Figure 2. Shifting the filter center frequency is easy once we are in the transform domain as all we need to do is use `numpy` to `roll` the coefficients left of right by a certain number of indices of the FFT of the filter coefficients. A limitation of using coefficient rolling to shift the filter center frequency is that the smallest frequency step is limited to

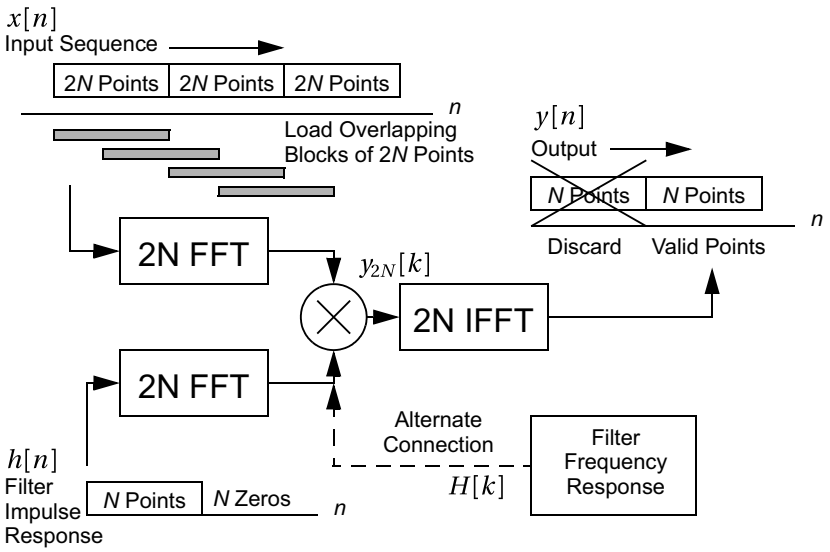

Figure 2: *Overlap-and-save* transform domain filtering.

$$\Delta f = \frac{f_s}{2 \cdot \text{Nfft2}} \text{ Hz} \tag{1}$$

As a specific example suppose $f_s = 1000$ Hz. We design an equal-ripple lowpass filter with passband of 40 Hz and stopband edge of 48 Hz, thus in the this minimizing the overlap between channels as the spacing parameter `bs = 100` Hz and `n_band_odd = True`. We set `n_bands2 = 2`, which creates a total five bands, two above 0 Hz, two below 0 Hz , and one centered on 0 Hz. To test the design we drive the system with 100000 samples of white Gaussian noise of variance $\sigma_w^2 = 1.0$. The output array `w_bank` contains five rows. We loop through all five channels computing the power spectral density (PSD) using the ss.psd() function with the `scale_mode` set to True reflect the white noise density of 1.0 in the passband.

```
# Lowpass filter design for basic example
fs = 1000 # Hz
b_lpf_chan = fir_h.fir_remez_lpf(40,48,0.1,70,fs)
len(b_lpf_chan)

S_Hz = 100 # Hz
Nfft2 = 512
N_bands2 = 2
N_bands_tot = 2*N_bands2 + 1
w = random.randn(100000)
w_bank,fax,fax_des = ss.fft_filt_bank(w,b_lpf_chan+0j,n_fft2=Nfft2,n_bands2=N_bands2,
                                      bs=BS_Hz,fs = fs,n_band_odd=True)
figure(figsize=(6,4))
for k in range(N_bands_tot):
    P_w,f_w = ss.psd(w_bank[k,:],2**10,fs,scale_noise=True)
    plot(f_w,10*log10(P_w))
title(r'Filter Bank Characterization using White Noise with $\sigma_w^2 = 1$')
ylabel(r'Gain (dB)')
xlabel(r'Frequency (Hz)')
grid();
# savefig('five_band_example.pdf')
```

Overlay noise spectrum plots show the collection of five spectra that take on the space of the low-pass channelizing filter. See Figure 3. From this plot we see the expected five filters passbands as

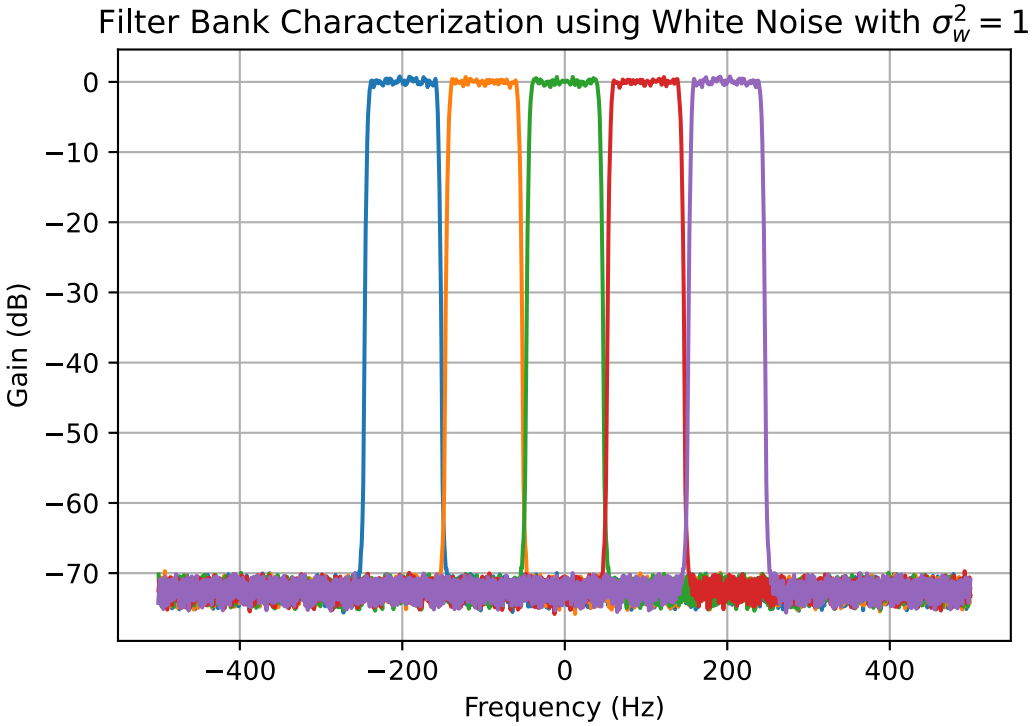

Figure 3: Spectrum of the five band channelizer output when the input is driven with white noise of variance $\sigma_w^2 = 1.0$.

a result of passing white noise. The nominal spacing between bands was set to be 100 Hz. The frequency error means that a 100 Hz center frequency becomes 100/(fs/(2*Nfft2))

$$f_{\text{actual}} = \left[\text{round}\left(\frac{100}{f_s}\cdot(2\cdot\text{Nfft2})\right)\cdot\frac{f_s}{2\cdot\text{Nfft2}}\right]\Bigg|_{\text{Nfft2}\,=\,512} \tag{2}$$

$$= 102\cdot 0.9765625 = 99.609375\ \text{Hz}$$

In addition to the 2D signal sample array `y_filt_bank`, the function also returns the actual frequency centers in `fax` and the desired frequency centers in `fax_des`:

```
print(fax) = [-199.21875   -99.609375    0.    99.609375  199.21875 ]
print(fax_des) = [-200. -100.  0.  100.  200.]
```

If we move away from power of two FFT lengths we should be able to drive the frequency error to zero. You will explore this in Task 5 of the project.

As a second example we modify the design above to use an even number of bands.

```
fs = 1000 # Hz
BS_Hz = 100 # Hz
Nfft2 = 512
N_bands2 = 3
N_bands_tot = 2*N_bands2 + 0
w = random.randn(100000)
```

```
w_bank,fax,fax_des = ss.fft_filt_bank(w,b_lpf_chan+0j,n_fft2=Nfft2,n_bands2=N_bands2,
                                      bs=BS_Hz,fs = fs,n_band_odd=False)
figure(figsize=(6,4))
for k in range(N_bands_tot):
    P_w,f_w = ss.psd(w_bank[k,:],2**10,fs)
    plot(f_w,10*log10(P_w))
title(r'Filter Bank Characterization using White Noise with $\sigma_w^2 = 1$')
ylabel(r'Gain (dB)')
xlabel(r'Frequency (Hz)')
grid();
# savefig('six_band__even_example.pdf')
```

The noise spectral estimate results are shown Figure 4. The center frequency accuracy is given by:

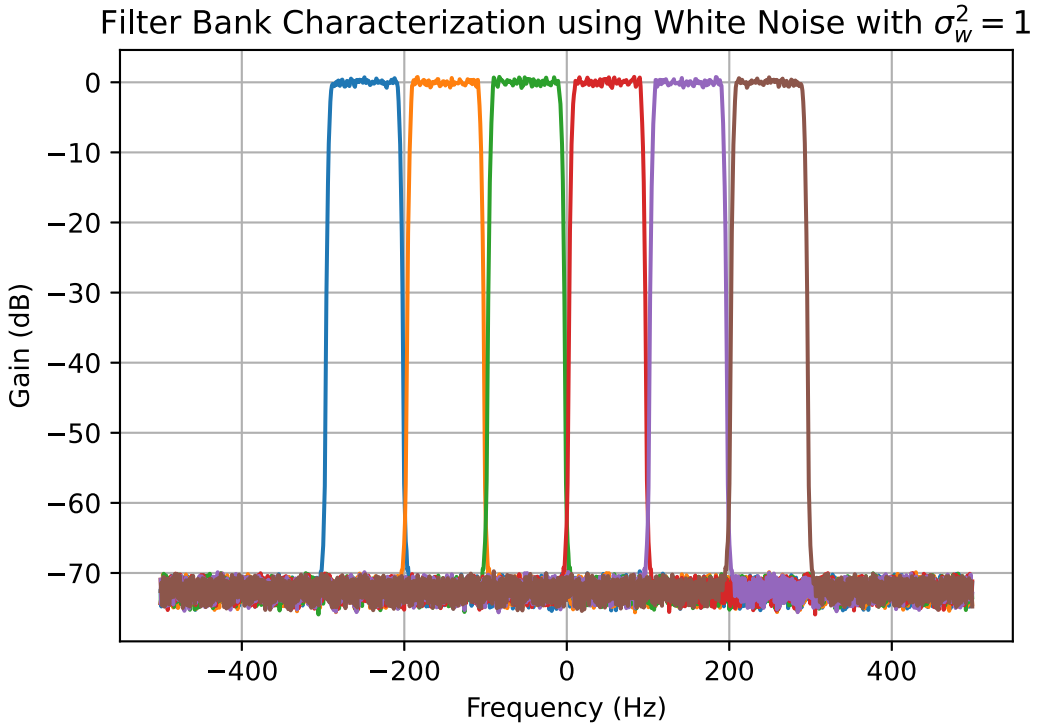

Figure 4: Spectrum of the six band (even) channelizer output when the input is driven with white noise of variance $\sigma_w^2 = 1.0$.

The actual and desired frequency centers are:

```
fax_actual:  [-249.023 -149.414  -49.805   49.805  149.414  249.023]
fax_desired:  [-250. -150.  -50.   50.  150.  250.]
```

We finally move on to study the two channelizer design used in this project.

Create a Channelizer Design Composed of Two Stages

The system block diagram is shown in Figure 5. Before getting into the design a few comments

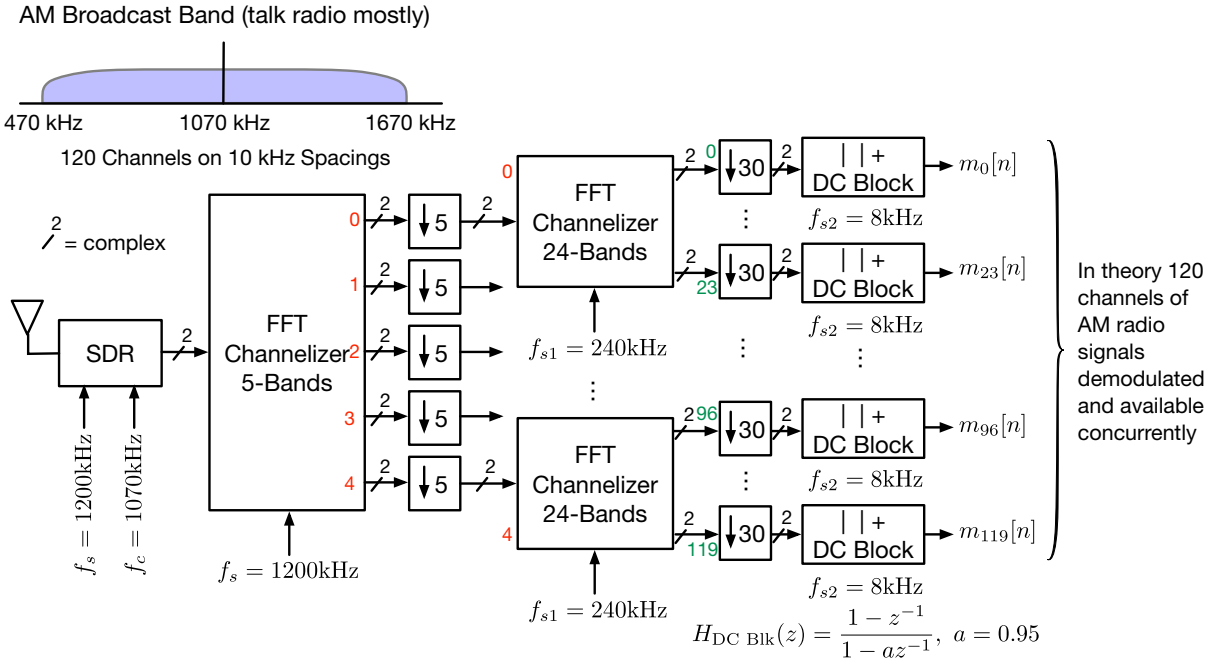

Figure 5: System block diagram.

about AM radio in Colorado Springs.

The Colorado Springs AM Radio Market

We have 3 three signals that I was able to easily receive during the day time. However, after sun-down the ionosphere moves closer the surface of the earth making it possible for AM stations to skip from one end the continental US to the other. The FCC mandates that most local AM stations reduce their transmit power to avoid interference with the strong *clear-channel* stations so they can be heard across the country (https://en.wikipedia.org/wiki/Clear-channel_station). The stations of interest from Colorado Springs in the daytime are:

- KVOR at 740 kHz
- KRDO at 1240 kHz
- KCSF at 1300 kHz
- Others are supposedly at 1460 (KZNT), 1530 (KOSC), and 1580 (KFCS) KHz

The spectrum capture, obtained at 10pm, using a sampling rate of $f_s = 1200$ kHz is shown in Figure 6. The purpose of the red dashed line is explain in Problem 1. There are few strong signal in spite of my using a long wire antenna inside a wood-frame house.

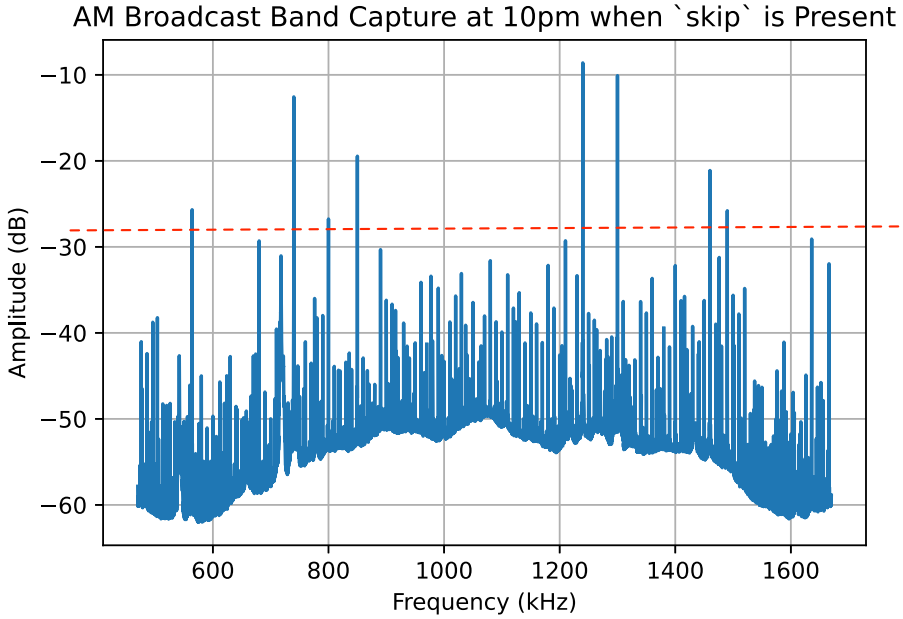

Figure 6: Full spectrum plot of the SDR capture.

Design Details

The lowpass filters are designed as follows.

```
b_lpf_stage1 = fir_h.fir_remez_lpf(105,115,0.1,60,fs_Hz/1e3)
b_lpf_stage2 = fir_h.fir_remez_lpf(4.0,5.2,0.25,50,fs1_Hz/1e3,n_bump=5)
(len(b_lpf_stage1),len(b_lpf_stage2))
(305, 389)
```

Note the tap lengths are less that 512 in both cases, so the minimum power of 2 length `Nfft2 = 512` or higher is suitable.

Choose Sampling Rate Contants

```
# Sampling rate constants used in the system design.
fs_Hz = 1200000 # Hz
fs1_Hz = 240000 # Hz
fs2_Hz = 8000   # Hz
```

Noise Test the Stage 1 Design

```
S_Hz_1 = 240000
Nfft2 = 512
N_bands2_1 = 2
N_bands_tot = 2*N_bands2_1+ 1
# Select input source: noise test or SDR signal
w = random.randn(100000)
w_bank1,fax1,fax_des1  = ss.fft_filt_bank(w,b_lpf_stage1,n_fft2=Nfft2,
                                          n_bands2=N_bands2_1,bs=BS_Hz_1,
                                          fs = fs_Hz,n_band_odd=True)
figure(figsize=(6,4))
for k in range(N_bands_tot):
```

```
    # Set scale_noise = True for noise and False for SDR signal
    # Shift the frequency to represent the actual RF frequency
    P_x_in,f_x_in = ss.psd(w_bank1[k,:],2**10,fs_Hz/1e3,scale_noise=True)
plot(f_x_in + 1070,10*log10(P_x_in))
# title(r'Filter Bank Characterization using White Noise with $\sigma_w^2 = 1$')
title(r'Filter Bank1 Characterization using White Noise with $\sigma_w^2 = 1$')
ylabel(r'Gain (dB)')
xlabel(r'Frequency (kHz)')
# ylim(-70,0)
grid();
# savefig('stage1_channelizer_noise_test.pdf')
```

The channelizer bands noise spectrum plots are shown in Figure 7 The bandwidth of the filters

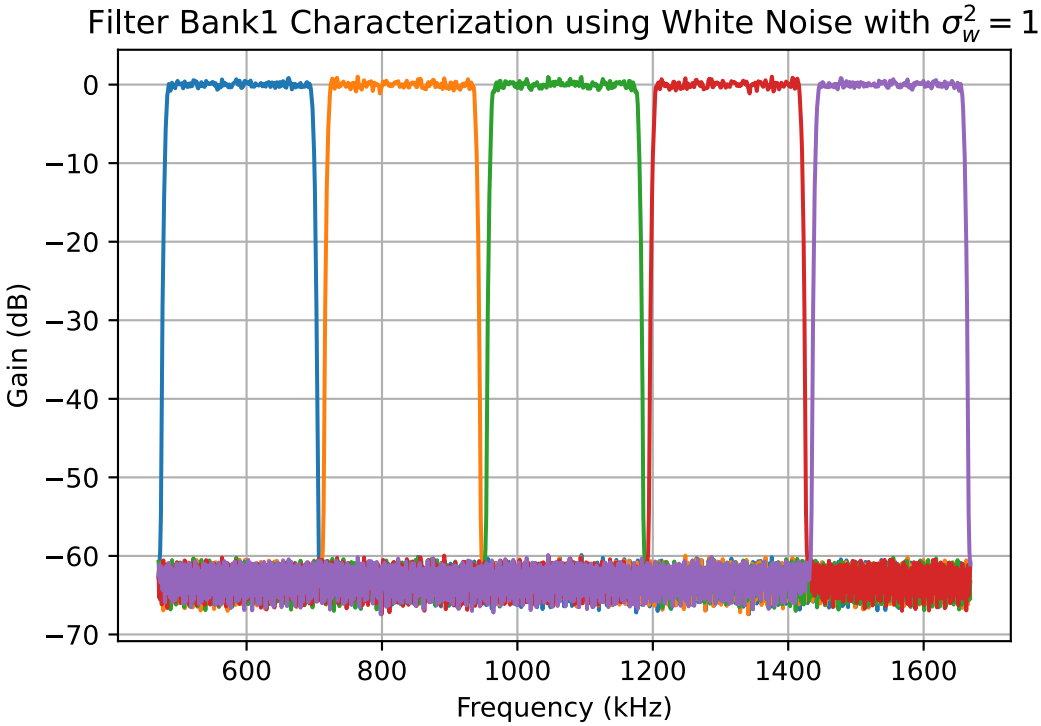

Figure 7: Stage 1 5-band channelizer noise spectrum plots.

could be wider to insure that radio signas at the band edges (high and low) do suffer attenuation. For the purposes of this project we will stay with the design, as signals of interest do not lie in these regions.

Noise Test the Stage 2 Design

```
S_Hz_2 = 10000 # Hz
Nfft2 = 512  # At one point tried 2048 and later returned to 512
 N_bands2_2 = 11
N_bands_tot = 2*N_bands2_2 + 1
# Select input source: noise test or SDR signal
w = random.randn(100000)
w_bank2,fax2,fax_des2  = ss.fft_filt_bank(w,b_lpf_stage2,n_fft2=Nfft2,
                                          n_bands2=N_bands2_2,bs=BS_Hz_2,
                                          fs = fs1_Hz,n_band_odd=True)
figure(figsize=(6,4))
for k in range(N_bands_tot):
    # Set scale_noise = True for noise and False for SDR signal
```

```
    P_x_in,f_x_in = ss.psd(w_bank2[k,:],2**10,fs1_Hz/1e3,scale_noise=True) # change to
False for SDR signal
    plot(f_x_in,10*log10(P_x_in))
title(r'Filter Bank1 Characterization using White Noise with $\sigma_w^2 = 1$')
ylabel(r'Gain (dB)')
xlabel(r'Frequency (kHz)')
grid();
# savefig('stage2_channelizer_noise_test.pdf')
```

Note `Nfft2 = 2048` was tried in an attempt to improve the center frequency accuracy. For the purposes of this project both work fine. The channelizer bands noise spectrum plots are shown in Figure 8

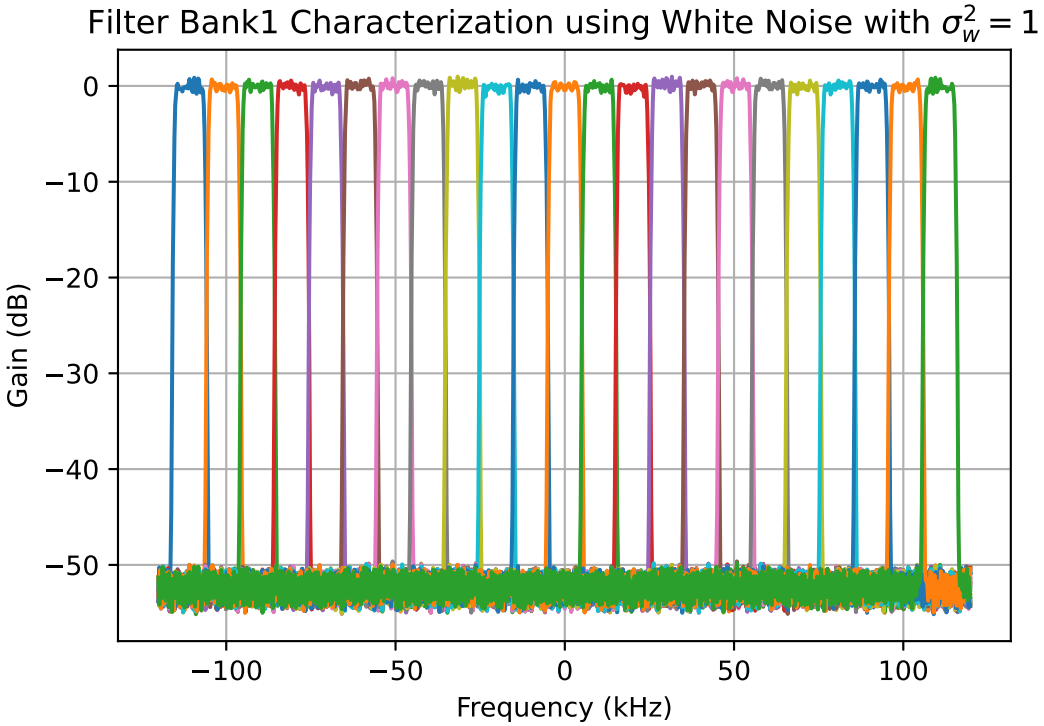

Figure 8: Stage 2 24-band channelizer noise spectrum plots.

Load The SDR Signal

```
# Check the contents of the npz file and see what 'keys' are available:
load('am3_fs1200_fc1070.npz')

output: NpzFile 'am3_fs1200_fc1070.npz' with keys: am3_cpx

# Using the key 'am3_cpx' we copy the stream into an array
am_sdr_capture3 = load('am3_fs1200_fc1070.npz')
am_fs1200_fc1070 = am_sdr_capture3['am3_cpx']
# Check the sample length
len(am_fs1200_fc1070)

output: 35635198
```

Follow Stage1 Channel k=3 into a Stage2 Channelizer

We start by passing the SDR signal `x_in` into stage1, look at the output spectrum, then move the

k=3 signal into stage2.

```
BS_Hz_1 = 240000
Nfft2 = 512
N_bands2_1 = 2
N_bands_tot = 2*N_bands2_1+ 1
# Select input source: noise test or SDR signal
# x_in = random.randn(100000)
x_in = am_fs1200_fc1070
x_bank1,fax1,fax_des1  = ss.fft_filt_bank(x_in,b_lpf_stage1,n_fft2=Nfft2,
                                          n_bands2=N_bands2_1,bs=BS_Hz_1,
                                          fs = fs_Hz,n_band_odd=True)

figure(figsize=(6,4))
for k in range(N_bands_tot):
    # Set scale_noise = True for noise and False for SDR signal
    P_x_in,f_x_in = ss.psd(x_bank1[k,:],2**10,fs_Hz/1e3,scale_noise=False) # change to
False for SDR signal
    plot(f_x_in + 1070,10*log10(P_x_in),label='k= %d' % (k,))
# title(r'Filter Bank Characterization using White Noise with $\sigma_w^2 = 1$')
title(r'PSD of Captured SDR Signals Passed Through 5 Filters')
ylabel(r'PSD (dB)')
xlabel(r'Frequency (kHz)')
legend()
ylim(-70,0)
grid();
# savefig('stage1_channelizer_output_with_SDR_input.pdf')
```

The stage1 channels output spectrum plot is shown in Figure 9

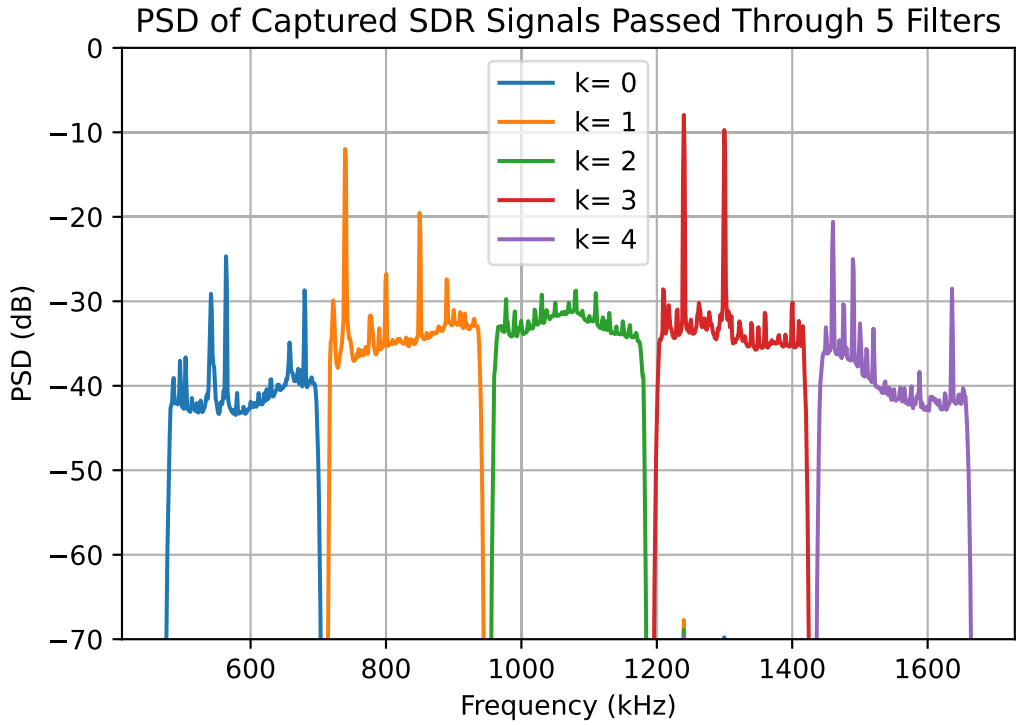

Figure 9: Stage 1 SDR spectrum spread over the five bands.

We will next follow the k=3 signal into the stage2 24 band channelizer.

```
BS_Hz_2 = 10000 # Hz
```

```
Nfft2 = 512
N_bands2_2 = 11
N_bands_tot = 2*N_bands2_2 + 1

x_in2 = ss.downsample(x_bank1[3,:],5) # Process the k = 3 band decimated by 5
x_bank2,fax2,fax_des2  = ss.fft_filt_bank(x_in2,b_lpf_stage2,n_fft2=Nfft2,
                                          n_bands2=N_bands2_2,bs=BS_Hz_2,
                                          fs = fs1_Hz,n_band_odd=True)

figure(figsize=(6,4))
for k in range(N_bands_tot):
    # Set scale_noise = True for noise and False for SDR signal
   P_x_in,f_x_in = ss.psd(x_bank2[k,:],2**10,fs1_Hz/1e3,scale_noise=False) # change to
False for SDR signal
    plot(f_x_in + 1070 + 240,10*log10(P_x_in))
title(r'Filter Bank Characterization using White Noise with $\sigma_w^2 = 1$')
ylabel(r'PSD (dB)')
xlabel(r'Frequency (kHz)')
ylim(-70,0)
grid();
# savefig('stage2_channelizer_output_stage1_k3_input.pdf')
```

The stage2 channels output spectrum plot, for `k=3`, is shown in Figure 10 To obtain demodulated

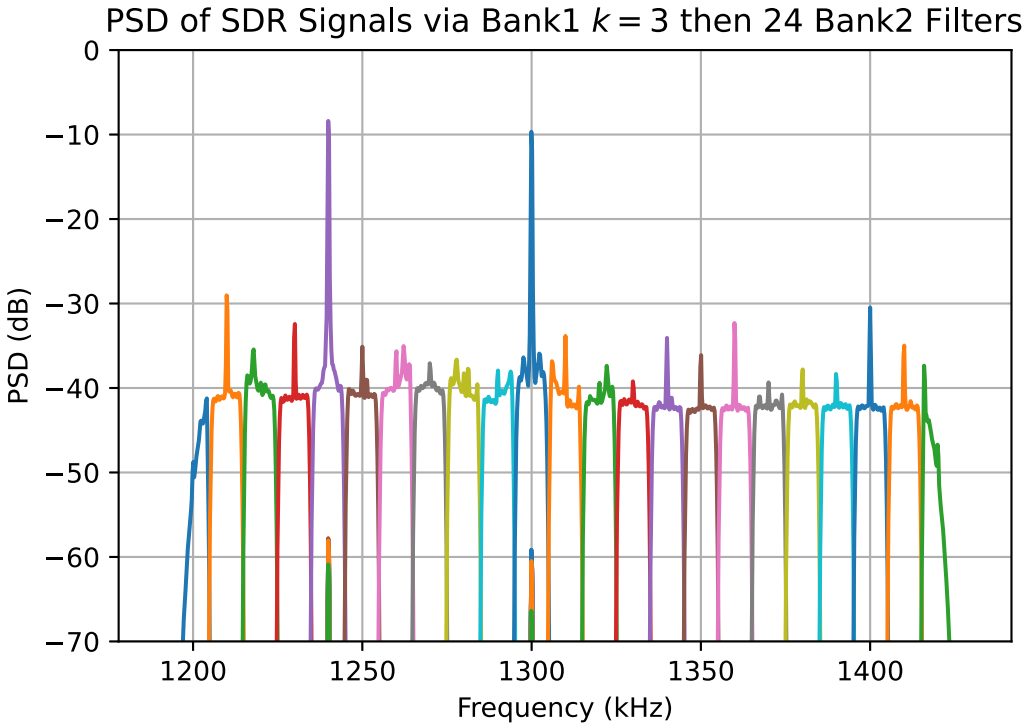

Figure 10: Stage 2 SDR spectrum for `k=3` spread over the 24 bands.

AM radio message signals we finally pick a stage2 signal of interest. The two strong signals from bank1, `k=3`, have now been isolated. The row index in `x_bank2` is needed so we can further process this channel. We do a reverse index lookup using the function `freq2idx()` defined earlier in the sample notebook. We will work with the `fax_des_kHz` frequencies, obtained by scaling to kHz from Hz units the `fax_des` output array. Note in the plot of Figure 10 the frequency axis is trans-

lated using the capture center frequency of 1070 kHz and the subband `k=3` offset frequency of 240 kHz to shift the plotting frequency axis to `f_x_in + 1070 + 240`. The calculations below detail how this is done. As an alternative you can just count the subband index start from the far left of the PSD plot know that the first subband is k=0.

```
Convert to kHz
fax2_kHz = fax2/1e3
fax_des2_kHz = fax_des2/1e3
print('fax_actual kHz: ',rint(fax2_kHz*10)/10 + 1070 + 1*240)
print('fax_desired kHz: ',fax_des2_kHz + 1070 + 1*240)
```

```
fax_actual kHz:  [1199.8 1209.8 1219.8 1229.8 1239.9 1249.9 1259.9 1269.9 1279.9 1290.
 1300.  1310.  1320.  1330.  1340.1 1350.1 1360.1 1370.1 1380.1 1390.2
 1400.2 1410.2 1420.2]
fax_desired kHz:  [1200. 1210. 1220. 1230. 1240. 1250. 1260. 1270. 1280. 1290. 1300.
1310.
 1320. 1330. 1340. 1350. 1360. 1370. 1380. 1390. 1400. 1410. 1420.]
```

Find indices for AM radio signals at 1210, 1240, and 1300 kHz:

```
# Shift the frequencies to match the actual RF center frequencies
freq2idx([1210,1240,1300],fax_des2_kHz + 1070 + 240)
output: [1, 4, 10]
```

Final AM Demodulation

Introduce the final decimation by 30 directly on say the `k=10` signal, then gain scale it by 30 dB, envelope detect, and filter with the DC block described in the system block diagram

```
# Envelope detect using abs() and then DC block the signal
# Yes, this simple pair of operations is very trival in both analog circuitry
# and DSP code AM radio has been around for more than 100 years.
x_1300 = ss.downsample(x_bank2[10,:],30)
x_1300_env = abs(x_1300*10**(30/20)) # Gain by 30 dB
x_1300_rec = signal.lfilter([1,-1],[1,-.95],x_1300_env)
# Take a Listen
Audio(x_1300_rec,rate=fs2_Hz)
```

Problems

The tasks that are required to complete this project are detailed in the numbered items below. Everything you do will involve the use of the Jupyter notebook found in the ZIP package `Project2_f2025.zip`.

1. Using the full spectrum plot of the SDR file found in the sample notebook and with plot as shown above in Figure 6 to find frequency locations in kHz that cross above a threshold of about -29dB. Use the technique used in Project1, Problem 2. The intent is you will have about 8 or 9 large signals to demodulate and listen to from the output of `x_bank2`. This problem will give you the AM radio station channel frequencies of interest. If needed quantize the output frequencies to the nearest 10 kHz as these are the standard carrier frequency spac-

ings. You may need to increase the FFT sized used in the PSD function. Note small frequency errors are present in the receiving process due to oscillator offsets.

2. Choose one of the two lowpass filters:

```
b_lpf_stage1 = fir_h.fir_remez_lpf(105,115,0.1,60,fs_Hz/1e3)
b_lpf_stage2 = fir_h.fir_remez_lpf(4.0,5.2,0.25,50,fs1_Hz/1e3,n_bump=5)
(len(b_lpf_stage1),len(b_lpf_stage2))
```

to investigate the linear phase properties.

a) Verify that the filter is linear phase by observing coefficient symmetry.

b) Verify from a plot of the filter group delay that the filter is linear phase. Hint use `fir_h.freqz_resp_list([b_lpf],[1],mode='groupdelay_s',fs=?)` and mode 'groupdelay_s'. Is the group delay consistent with the filter tap length?

3. In this problem you are going to complete the study of `k=3` on the 2D `x_bank2` array started in the sample notebook. Listen to the two signals above -~28 dB threshold as found in Problem 1. Plot a portion of each demodulated signal `x_?freq?_rec` (you will fill in the frequency as you do the experiment, e.g., 1240 kHz and 1300 kHz). Describe what you hear.

4. In this problem you are going to demodulate and listen to the signals in the k=0,1, and 4 rows of x_bank1. This of course means you will have to channelize each of these signals and then downsample, demodulate, and finally listen to the above threshold signals. Comment on what you hear. The relevant code from the sample notebook that you will be working with is listed below. If you want to channelize all five and store them in memory, OK too.

```
BS_Hz_2 = 10000 # Hz
Nfft2 = 2048
N_bands2_2 = 11
N_bands_tot = 2*N_bands2_2 + 1

x_in2 = ss.downsample(x_bank1[3,:],5) # Process the k = 3 band decimated by 5
x_bank2,fax2,fax_des2  = ss.fft_filt_bank(x_in2,b_lpf_stage2,n_fft2=Nfft2,
                                          n_bands2=N_bands2_2,bs=BS_Hz_2,
                                          fs = fs1_Hz,n_band_odd=True)

figure(figsize=(6,4))
for k in range(N_bands_tot):
    # Set scale_noise = True for noise and False for SDR signal
   P_x_in,f_x_in = ss.psd(x_bank2[k,:],2**10,fs1_Hz/1e3,scale_noise=False) # change to
False for SDR signal
    plot(f_x_in + 1070 + 240,10*log10(P_x_in))
title(r'Filter Bank Characterization using White Noise with $\sigma_w^2 = 1$')
ylabel(r'PSD (dB)')
xlabel(r'Frequency (kHz)')
ylim(-70,0)
grid();
# savefig('seven_band_example.pdf')

# Convert to kHz
fax2_kHz = fax2/1e3
fax_des2_kHz = fax_des2/1e3
print('fax_actual kHz: ',rint(fax2_kHz*10)/10 + 1070 + 1*240)
print('fax_desired kHz: ',fax_des2_kHz + 1070 + 1*240)
```

```
# Shift the frequencies to match the actual RF center frequencies
freq2idx([1210,1240,1300],fax_des2_kHz + 1070 + 240)

# Envelope detect using abs() and then DC block the signal
# Yes, this simple pair of operations is very trival in both analog circuitry and DSP
code
# AM radio has been around for more than 100 years.
x_1300 = ss.downsample(x_bank2[10,:],30)
x_1300_env = abs(x_1300*10**(30/20)) # Gain by 30 dB
x_1300_rec = signal.lfilter([1,-1],[1,-.95],x_1300_env)
### Take a Listen
Audio(x_1300_rec,rate=fs2_Hz)
```

5. Verify improved band center frequency accuracy in the `x_bank1` signals when the `Nfft2` length is changed from 512 to 500. Is it now exact? Is this expected?
   a) Verify the frequency error difference.
   b) We know from the study of the FFT that FFT lengths that are a power of 2 are popular because of the speed advantages of the radix-2 algorithm. Time the below code on your machine using the Jupyter notebook cell magic `%%timeit` as shown below:

```
%%timeit
BS_Hz_1 = 240000
Nfft2 = 512 # change from 512 to 500
N_bands2_1 = 2
N_bands_tot = 2*N_bands2_1+ 1
# Select input source: noise test or SDR signal
# x_in = random.randn(100000)
x_in = am_fs1200_fc1070[:10000] # for the timeit test input just 10000 points
x_bank1,fax1,fax_des1  = ss.fft_filt_bank(x_in,b_lpf_stage1,n_fft2=Nfft2,
                                          n_bands2=N_bands2_1,bs=BS_Hz_1,
                                          fs = fs_Hz,n_band_odd=True)
```

6. Pick to two demodulated signals from `k=0,1,3,` and `4` rows of `x_bank2` to save as `wav` files to be attached to your with Project2 submission on Slack. I want to be able to listen them. Please tell me by encoding in the file name what the signal frequency is. To write a wave file simply use:

```
ss.to_wav('file_name.wav',8000,x
```

Appendix: The SDR Hardware and Software Tools

The SDR capture file that you work with in this project was obtained using two hardware components: (1) an RF up-converter and (2) an SDR receiver with USB interface. A third component is a software application for controlling the radio hardware and creating a capture file.

SDR Hardware

Shown below in Figure 11 is a photograph of the hardware components.

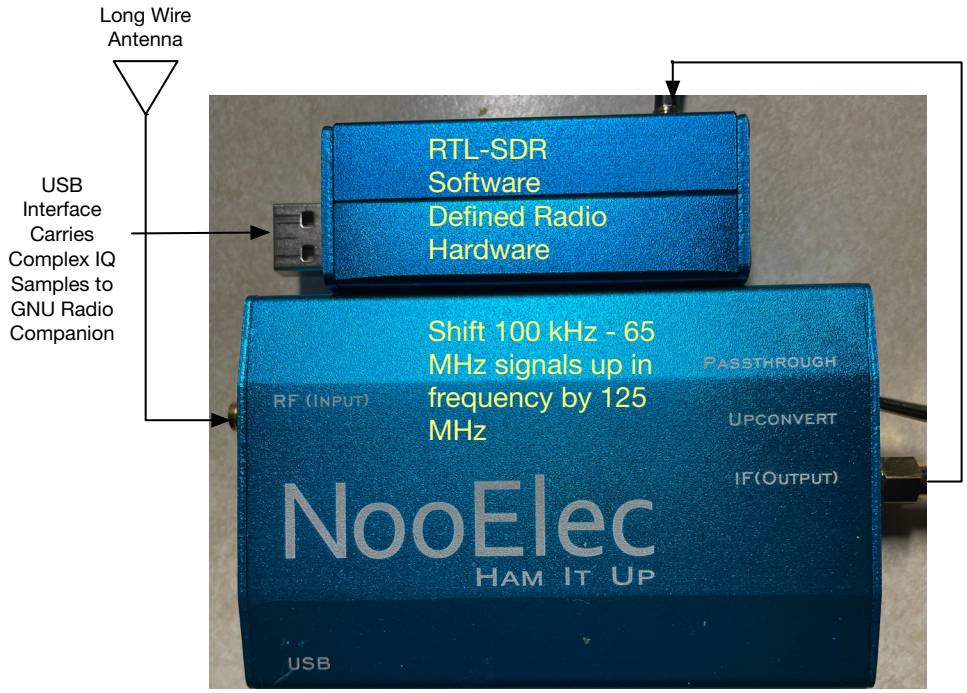

Figure 11: The SDR hardware, which includes an RF up converter and the RTL-SDR radio that interfaces to a computer using USB.

SDR Software Application

The *GNU Radio Companion* flow diagram created to collect the radio complex baseband samples

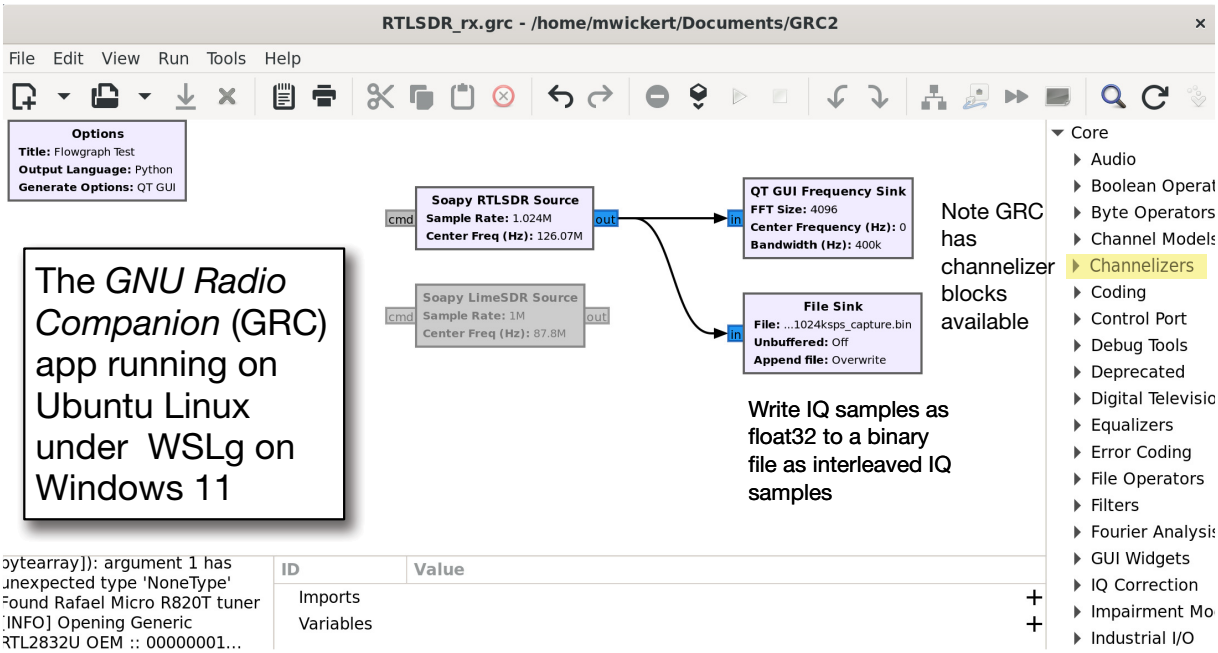

Figure 12: GNU Radio Companion flow diagram used to make the file capture; this flow diagram also display a real-time spectrum when the app is running.

is shown in Figure 12.

Channelizer for AM Radio Fall 2025 Hints

1. Recall from Project 1 you can use find spectral peaks as a particular index in a PSD array. Using the corresponding frequency axis array you can look-up the frequencies of interest. Since AM station carrier frequencies line at even multiples of 10 kHz, you may need to round the frequency values you find to the nearest 10 kHz value. You can round numpy arrays using e.g.,

```
# Demo rounding an array using numpy rint on an array
f_array = random.rand(10)*1000 + 500
f_array_round = np.rint(f_array/100)*100 # Round to nearest 100
(f_array,f_array_round)
```

```
(array([1182.6601353 ,  567.54783532, 1415.60413099, 1016.89490285,
         883.14345403,  961.72595202, 1239.51792409,  654.71833078,
         602.22135925,  613.36653992]),
 array([1200.,  600., 1400., 1000.,  900., 1000., 1200.,  700.,  600.,
         600.]))
```

2. This problem revolves around basic linear phase FIR filter theory from Chapter 5. Here you consider a specific filter design that has length 305 or 389 taps. Just verify the properties and show you work.

3. In this problem you finish the work that was started in the sample Jupyter notebook. You may find one of the signals is the result of an inband intermodulation product. See the comments for Problem 4 below.

4. As a heads up when you listen to the signals coming from bank1 channels $k = 0$ and $k = 4$ you will find yourself choosing a 10 kHz wide channel where the signal is/ not centered in the middle of a 10 kHz band, i.e. the signals appear to be shifted high or low by about 5 kHz. AM radio stations according to the FCC rules are spaced by even multiples of 10 kHz. When you listen to these signals you should note that they approximately match another signal on a different channel. What is going on?

   When multiple strong signals are present in a radio receiver analog front-end a nonlinear behavior known as *intermodulation distortion* can occur. Suppose we have a signal at f_1 and one at f_2. A third-order intermodulation product signal will lie in the AM band at say $|2f_1 - f_2|$ or $|2f_2 - f_1|$. So as an example consider $2 \times 1460 - 1600 = 1320$. The new signal lies at an even multiple of 10 kHz. In our case the new signal lies at an odd multiple of 5 kHz, e.g., 1325 kHz. I do not fully understand how this can unless one of the intermod frequencies is not another AM radio, but some strong signal that is offset by 5 kHz or so. No worries in working this problem, just interesting to know that things like this can happen with radio receivers.

5. Follow the instructions provided. You should find the filter center frequencies are now exact. There will be a difference in execution time between the 512 pt and 500 pt FFTs. The use of the cell magic `%%timeit` will provide results. Do take the advice given in the problem statement code example and only run 10000 samples through bank1. If do not do this you will be waiting a very long time for `%%timeit` to complete its code timing results.

6. Just save your two favorites as described making sure the sampling rate set to 8000 Hz.

Project2 2025 Point Assignments

Problem	Points per part
1. Peak search of SDR PSD carrier frequencies above ~-28dB	List of frequencies10 pts
2. Lowpass Filter 1 or 2 evidence of linear phase	Parts (a) Show coefficient symmetry 5 pts (b) Show group delay and mean delay 5 pts
3. Listening to strong stations in `x_bank2` in the `k=3` channel coming from `x_bank1`	Extending from the code in the sample notebook and the PSD plot of Figure 10 fully demodulate and describe what you hear from the ~3 strong stations. 10 pts
4. Demodulate and listen to the remaining strong signals in `x_bank1` channels 0, 1, and 4	Similar to Problem 3 fully channelize and demodulate the indicated `x_bank1` signals so that you can listen and comment on what you hear. This should be approximate 8 signals (~3 in the `k=0` band, ~2 in the `k=1` band, ~3 in the `k=4` band) 20 pts
5. Explore increased center frequency accuracy in `x_bank1` bands	Parts (a) Verify the error 5 pts (b) Time the filter bank code to compare Nfft2 512 vs 500 10 pts
6. Collect your two best message wav files	10 pts
Total	75 total points