Name:_____

EE354: Lab 5-AM Demodulation

This lab will have you demodulate an AM waveform using Software-Defined Radio techniques and investigate the impact of SNR on the demodulated signal's audio quality.

Background Information

For this lab, we will be transmitting the following waveform:

$$s(t) = A_c\left[1+\mu m(t)\right]\cos(2\pi f_c t)$$

Where:

$$A_c = 0.75V$$
$$\mu \approx 0.2 - 0.4$$
$$f_c = 4.7\,MHz$$

In this lab, we will be using audio output from the PC as the input to an AM Modulator on one of the Agilent 33220A function generators. To calculate the parameters of our AM signal, we have to recognize that our AM modulator scales the input, where 5V represents 100% modulation of our carrier signal. Recall that the modulation index is related to the amplitude of the carrier:

$$\mu = \frac{V_{\text{mod}}}{A_c}$$

Where: V_{mod} is the amplitude of the modulating signal [V]

A_c is the amplitude of the carrier [V]

The modulation index of the AM waveform **as produced by the signal generator** is given by:

$$\mu = \frac{V_{RMS}}{5V}$$

Where: V_{RMS} is the peak amplitude of the audio signal [V]

Then, we can solve for V_{mod} and determine the signal power:

$$p_{signal} = \frac{V_{\text{mod}}^2}{2R_L} = \frac{(\mu A_c)^2}{2R_L}$$

Part 1: AM Demodulator Design

This part of the lab will step you through creating a envelope detector in Matlab. Many of these building blocks will be used in later labs where you will capture and demodulate a real AM or FM signal. A handout entitled **"Software Demodulation of a Pure Tone"** is attached to the back of this lab and will aid you in your design.

First, enter the following code to generate a DSB-TC AM signal that has a center frequency of 4.7 MHz, modulation index of 0.6, and modulating frequency of 500 Hz.

```
clear all
close all

fs = 20e6;              % Establish our Sampling Frequency
fs_soundcard = 44.1e3; % Soundcard Sampling Frequency
fc = 4.7e6;             % Carrier Frequency is 4.7 MHz
fm = 500;               % message frequency 500 Hz
mu = 0.6;               % Modulation Index
A = 0.75;               % AM carrier amplitude

delt = 1/fs;
tstop = .125;          % Create signal for 1/8 of a second
t = 0:delt:tstop;      % Establish the time vector

m = cos(2*pi*fm*t);    % Create the Message Signal
sam = (A + mu.*m).*cos(2*pi*fc*t);    %DSBTC Creation
```

1. In software, implement an envelope detector (recall that envelope detection in Matlab is basically, an absolute value and low pass filter, see the posted course notes for an example and the handout attached to the back of this lab) to recover the original information signal. Output the resulting demodulated signal via the PC's sound card. Use the `filter_audio` function (available on the course website) to implement the LPF.

   Recall: Your sampling frequency is likely set based on the frequency and bandwidth the RF signal. As a result, the sampling frequency will be many times larger than the PC's sound card can handle. To obtain an acceptable audio sampling frequency, we have to <u>decimate</u> (remove redundant samples from) the demodulated signal. Decimation works by essentially throws away unnecessary samples. To implement, use the following commands:

```
downsample_rate = floor(fs./fs_soundcard);  % Downsampling rate to audio
fs_audio = fs./downsample_rate;
s_demod = downsample(s_envelope_detected, downsample_rate);
t_demod = downsample(t, downsample_rate); % Must downsample time as well
```

   The `downsample` command performs the decimation operation for us, and the variable `downsample_rate` tells Matlab how many samples to throw away in between samples that it retains. Notice that our net sampling frequency is reduced (which will be important when you are playing the demodulated signal on the PC's sound card)

2. Output the recovered audio signal to the PC's sound card and verify that you hear the 500 Hz tone.

Instructor Verification of Audio Recovery:_____

Part 2: Hardware Connections

1. Using an adapter provided by the instructor, connect the PC Audio Output (from the front panel speakers) to the oscilloscope input as shown in the diagram below.

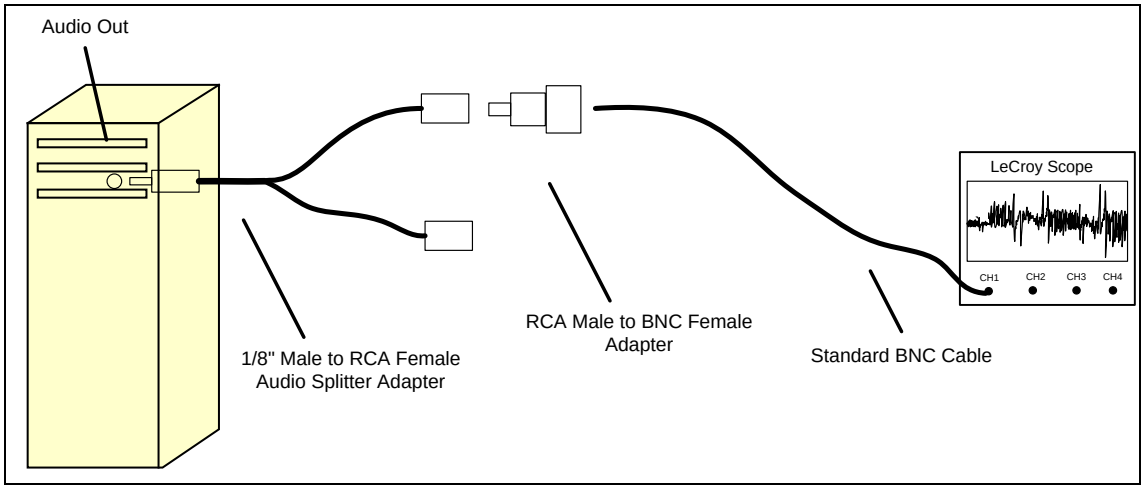

Figure 2: Illustration of the connection between the PC Audio Output and the LeCroy Oscilloscope.

2. On the PC, open the PC Volume Control panel, and set both the Main and Wave volumes to Max, as shown in Figure 2. On the PC's front panel, set the volume control thumbwheel to about 70-80% of maximum. Using the PC, navigate to your favorite YouTube music video or MP3 location and select a song to play. (**Note:** You want a song that you can play repeatedly, so a site like Pandora won't work). Play the audio file and observe the result on the oscilloscope screen.

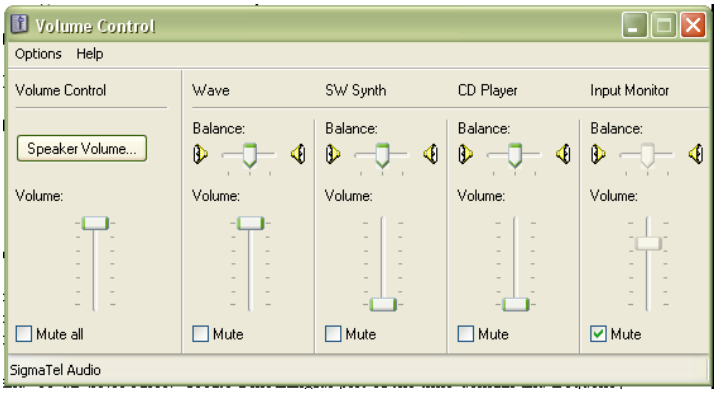

Figure 2: Volume Control Configuration

3. Adjust the timebase of the oscilloscope to display several tens or hundreds of milliseconds of the audio signal. Adjust the amplitude (volume) of the audio signal until the peak falls between 2.5V and 2.75V. Use the scope to measure the RMS voltage of the audio waveform (**Note:** It will bounce around somewhat, so make a good guess as to the average value).

   Record the Peak voltage of the audio signal: _____

   Record the RMS voltage of the audio signal: _____

   Calculate the Modulation Index of your AM Signal: _____

Part 3: Signal Capture

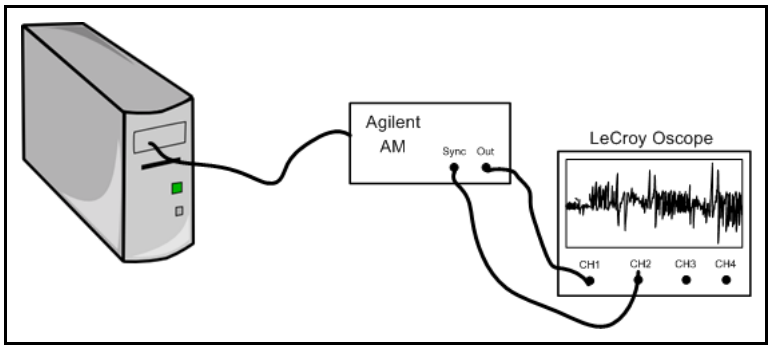

Figure 1: Measurement Setup Connections

1. To begin, use one of the Agilent 33220A function generators to create a 4.7 MHz sine wave with a peak-to-peak voltage of 1.5V. This function generator will be used to produce the AM modulated signal.
2. Now turn on the AM modulation of the 4.7 MHz function generator by using the "MOD" button, and selecting the External source.
3. Connect the output of the 4.7 MHz function generator to the Oscilloscope. Connect the "Sync" output to Channel 2 of the Oscilloscope. Be sure the Function Generator is set for a 50Ω output impedance (Utility→Output Setup), and the scope is set for a 50Ω input impedance. Configure the Scope to trigger on Channel 2, with a 500 mV trigger level, and using Normal triggering.
4. Connect the PC Audio Output (<u>from the front panel</u>) to the MOD IN input of the AM signal generator (See Figure 1).
5. Before proceeding, verify that you have the 4.7 MHz signal (with no modulation) appears on the oscilloscope screen. Verify that the signal generator produces an <u>actual amplitude</u> of 0.75V (modulation is turned on, but with no modulating signal).

<u>Note 1:</u> Verify (using the oscilloscope) that the signal generator produces an <u>actual amplitude</u> of 0.75V once the modulation is turned on, but with no modulating signal.

<u>Note 2:</u> The Anritsu spectrum analyzer will demodulate AM signals, and is a great way to verify that you have generated the AM signal properly.

6. Adjust the oscilloscope to record 5.0 Million Samples, and configure the sampling frequency to be 1.0 MHz. Your timebase on the scope screen should be 500 msec/division. Configure the scope to generate the FFT of the waveform to observe the spectrum in real time.

Question 1: At what frequency would you expect the AM waveform to appear in the FFT? What phenomenon are you taking advantage of?

_____

_____

_____

_____

7. Play the audio file on the PC. Observe the AM signal on the scope (you should see a modest amount of amplitude change). Press the "Single" button on the scope, and when the scope is finished capturing the signal, save it to a Matlab .dat file.

9. Using your Matlab envelope-detector code from the first part of this lab, load the Oscilloscope captured data into Matlab (**Note:** it's a big file, it may take some time!), demodulate the AM signal, and play it using the PC's sound card. Your demodulated signal in this case should be several seconds of whatever audio signal you were outputting from the PC.

10. Create a plot of the Time-Domain AM Signal and demodulated signal and turn them in with your report.

Based on the captured data, calculate the actual modulation index of your AM Signal: _____

Instructor Verification of Successful Audio Demod: _____

Software Demodulation of a Pure Tone

To recover the original information signal from the received AM waveform (whether in simulation or from data captured off the scope), you will need to implement an envelope detector in Matlab. The envelope detector consists of the following four steps:

1. Take the absolute value of the AM waveform to form $\left|s_{am}(t)\right|$ (where $s_{am}(t)$ is the received AM signal).
2. Low-Pass filter the rectified signal to form $\hat{m}(t)$.
3. Downsample the low-pass filter to an audio-level sampling frequency ($f_s < 44.1kHz$).
4. Output the resulting audio signal via the PC's speaker using the `sound` command.

In order to build your AM demodulator, you will find the following Matlab functions useful:

`filter_bb(x, fco, fs) –` (Available on the EE354 Course Website) Implements a simple Low Pass Filter to prevent aliasing of the audio signal when sampled. Note that `fco` must be less than half of `fs` and that `fco` is the -3dB cutoff point.

`downsample(x, N) –` (a native Matlab function) downsamples input signal X by keeping every N^{th} sample starting with the first.

Note 1: The cutoff frequency you use in the filter_bb.m function should be no lower than approximately $0.005 f_s - 0.01 f_s$, otherwise the filter will simply output a flat line.

Note 2: Because the sampling frequency is set for the RF signal, the sampling frequency is many times larger than the PC's sound card can handle. To obtain an acceptable audio sampling frequency, we need to <u>decimate</u> the demodulated signal. Decimation works by throwing away unnecessary samples. To implement, use the following commands:

```
downsample_rate = floor(fs_original./fs_soundcard);
fs_audio = fs_scope./downsample_rate;
s_demod = downsample(m_hat, dec_rate);
```

The `downsample` command performs the decimation operation for us, and the variable `dec_rate` tells Matlab how many samples to throw away in between samples that it retains. Notice that our net sampling frequency is reduced (which will be important when you are playing the demodulated signal on the PC's sound card)

For our purposes, the soundcard operates at a sampling frequency of 44.1 kHz (i.e., `fs_soundcard = 44.1e3`). For the Matlab Simulation, `fs_original` will be the sampling frequency you used to generate and simulate the AM signal. For the oscilloscope data, `fs_original` will be the sampling frequency of the oscilloscope.