College of Engineering and Applied Sciences

ESE 342 - Communication Systems

# Envelope Detector

*Authors:*
Kyle Han
Johnny Liu

November 29, 2022

# Table of Contents



# List of Figures

# 1 Assignment

Design an envelope detector for demodulating an amplitude modulated signal arriving at the reciever. The signal arriving at the input of the demodulator is given by

$$v_c(t) = 5.0[1 + \mu x(t)]cos(2\pi f_c t + \phi_0) \quad \mu V \tag{1}$$

where the modulation index, $\mu = 0.3$, $\phi_0$ is an arbitrary phase angle and $f_c = 98.6$ MHz is the carrier frequency. Generate a music signal x(t) for testing and verification of your envelope detector.

You must model the various electrical components, such as the diode, and process the signal in the time domain. You should display the signal at various stages of processing. Input should be taken from a music file and the output file should be played using any audio player.

# 2 Introduction

This report will lay out the design and construction of an Envelope Detector of an amplitude modulated signal. Deviating from the assignment, we will not be using 98.6 MHz as our carrier frequency due to the computational complexity required.

This design assumes common positive power rails are available, such as 5V. This design also assumes standard X7R capacitors and standard resistors, according to the E24 standard.

Specific to the MATLAB code, it can take in a file of any number of channels. However, it will only output an audio file with 1 audio channel. Additionally, we have amplified the audio signal before modulating and interpolated samples of the audio file to respect the analog world.

# 3 Theory

In this section, we will calculate the RC value of the parallel resistor and capacitor configuration

The envelope can be defined as

$$e_c(t) = A_c(1 + \mu cos(\omega_m t)) \tag{2}$$

where $A_c$ is the carrier amplitude, $\mu$ is the modulation index, and $\omega_m$ is the angular frequency of the modulated signal

The voltage of the RC circuit is given as

$$v_c(t) = V_0 e^{-\frac{t}{RC}} \tag{3}$$

Looking at this qualitatively, we know that the voltage of the RC circuit will follow the diode whenever it is on the rise. When the diode reaches a peak, it will then drop. However, the RC circuit doesn't allow for this drop to be so sudden. The voltage will actually follow the RC circuit's discharge until the diode is on the rise again, meeting the voltage requirement of the RC circuit to charge up. Therefore, we can assume the following

$$A_c(1 + \mu cos(\omega_m t)) >= V_0 e^{-\frac{t}{RC}} \tag{4}$$

since the ideal envelope would be a smooth curve that conveniently touches all peaks, we know that the voltage of the RC circuit will be less than the ideal envelope This is also true for the slope of the ideal and practical envelope. We can utilize algebraic techniques through derivatives to derive RC Using (4), we can obtain the following by taking the derivatives on both sides

$$[\frac{d}{dt}]A_c(1 + \mu cos(\omega_m t)) \geq [\frac{d}{dt}]V_0 e^{-\frac{t}{RC}} \tag{5}$$

$$-A_c\mu\omega_m \sin\omega_m t \geq -\frac{1}{RC}V_0 e^{-\frac{t}{RC}} \tag{6}$$

Realize that when we took the derivative of the exponential, it comes in the form of the coefficient times the original function.

$$\frac{\mu\omega_m \sin\omega_m t}{1+\mu\cos\omega_m t} \leq \frac{1}{RC} \tag{7}$$

Now, when we look at the numerator and denominator of the left hand side, we have to somehow find expressions for our trigonometric functions since they don't provide stable values for RC. To have a clue on the next procedure, we should start with trying to derive each trigonometric function as a linear expression. Seeing that oftentimes it is easier to derive when knowing an equation equals 0, we should take the derivative of both sides yet again.

$$\frac{(1+\mu\cos\omega_m t)(\mu\omega_m^2\cos\omega_m t) - (\mu\omega_m \sin\omega_m t)(-\mu\omega_m \sin\omega_m t)}{(1+\mu\cos\omega_m t)^2} \leq 0 \tag{8}$$

When taking derivations, we will analyze in the absolute maximum rating Taking equation (8),

$$(1+\mu\cos\omega_m t)(\mu\omega_m^2\cos\omega_m t) - (\mu\omega_m \sin\omega_m t)(-\mu\omega_m \sin(\omega_m t) = 0 \tag{9}$$

$$(\mu\omega_m^2\cos\omega_m t + \mu^2\omega_m^2\cos\omega_m t) + \mu^2\omega_m^2\sin^2\omega_m t = 0 \tag{10}$$

$$\cos\omega_m t + \mu = 0 \tag{11}$$

$$\cos\omega_m t = -\mu \tag{12}$$

By virtue of the trig identity

$$\sin t = \sqrt{1-\cos^2 t} \tag{13}$$

$$\sin\omega_m t = \sqrt{1-\cos^2\omega_m t} \tag{14}$$

$$\sin\omega_m t = \sqrt{1-\mu^2} \tag{15}$$

From equation (7), we will also look at the absolute maximum

$$\frac{1+\mu\cos\omega_m t}{\mu\omega_m \sin\omega_m t} = RC \tag{16}$$

$$\frac{1-\mu^2}{\mu\omega_m\sqrt{1-\mu^2}} = RC \tag{17}$$

$$\frac{\sqrt{1-\mu^2}}{\mu\omega_m} = RC \tag{18}$$

This is the absolute maximum, so we amend it

$$\frac{\sqrt{1-\mu^2}}{\mu\omega_m} \geq RC \tag{19}$$

Since modulation index is given, we just need $\omega_m$. Since we're working with audio files, the highest frequency we'll hear is 20kHz. We choose a high frequency because it will give us a more constrained bound. Choosing a more constrained bound will allow us to apply it to all other modulated angular frequencies at lower frequencies. Therefore, by substituting $\omega_m$ with 20kHz, we get a RC value of 159 $\mu$s. When we choose a common capacitance of 1 $\mu$s, resistance must be at most 150$\Omega$ since it is the closest resistor to 159$\Omega$.

Looking at the boundary for the RC circuit, we are given

$$\frac{1}{\omega_c} \leq RC \leq \frac{\sqrt{1-\mu^2}}{\mu\omega_m} \tag{20}$$

Where we are given the carrier frequency $\omega_c$=98.6kHz. Since we set capacitance to be 1nF, our resistance value must be above $10k\Omega$.

Finally, we have the equation

$$10k\Omega \leq R \leq 150k\Omega, \quad C = 1nF \tag{21}$$

# 4 Diode

We will represent the diode as the practical diode with a voltage drop and turn on voltage of 1.0V in simulation.

# 5 DC Bias

Because our input signal, measured in $\mu$V is below the minimum turn on voltage of a diode, we must bias the midpoint of the signal by at least 0.7V, which will be dropped in the diode. Thus, we can ensure enough voltage for the upper half of the waveform to pass though.

The resistances chosen for R2 and R1 are 100,000Ω and 100,000Ω, respectively to create a voltage divider resulting in 1V from a supply rail of 2V. High values were used because a voltage bias was necessary, but a large current, and therefore power waste, was not.

# 6 Peak Detector

Once the signal has passed through the diode and rectified, we now need to detect the peaks of the oncoming signal. At a fundamental level, the RC circuit charges and discharges. When charging the RC circuit, it follows the rectified waveform since the waveform is increasing voltage at an extremely fast rate. However, when the peak is met, the RC circuit can no longer follow the waveform. The RC circuit starts to discharge at a rate slower than the waveform. Imagine walking off a cliff: you will be falling at a steady rate, but the cliff side has a sharp decline. Therefore, the RC circuit will keep discharging steadily until the waveform comes back up, essentially charging the RC circuit up again.

This poses a question: how can we adjust our RC values to not discharge so much as to make the wave distorted, but also make sure we discharge enough to keep track of any decline in peaks of the waveform?

Earlier in Theory, Equation 20, we derived the maximum and minimum RC value for peak detection. Given a chosen capacitance, we end up with a range of resistor values to pick from. In practice, we have chosen 75kΩ to be the resistance, with it being almost halfway between the two bounds, as well as being a known resistor value.

# 7 High-Pass Filter

In practice, a high-pass filter may be needed to filter out DC components of the output waveform. However for our simulation, it had little to no distortion due to the DC Bias. Therefore, the circuit has no high-pass component.

However, it would be rude to flirt with your curiosity, so we will calculate a theoretical high-pass filter. Unless you are an elephant reading this, the lowest frequency that we can hear is 20Hz

$$f = \frac{1}{2\pi RC}, \quad f = 20Hz, C = 0.047\mu F$$

$$\frac{1}{R} = 2\pi fC$$

$$R = \frac{1}{2\pi fC}$$

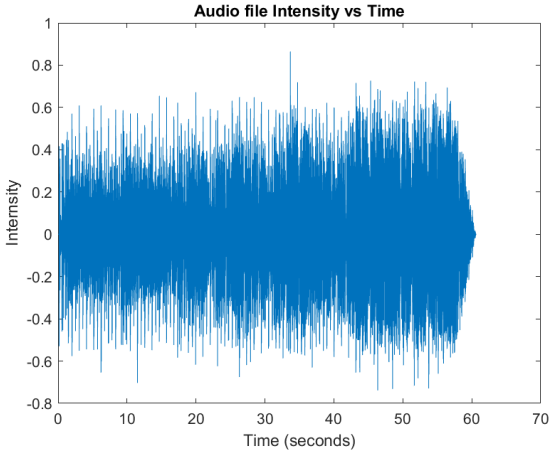
$$R = 169k\Omega$$

Looking up the known resistor values, we can use 180kΩ resistor to obtain a slightly lower frequency threshold since we don't want to filter too much of the output waveform.

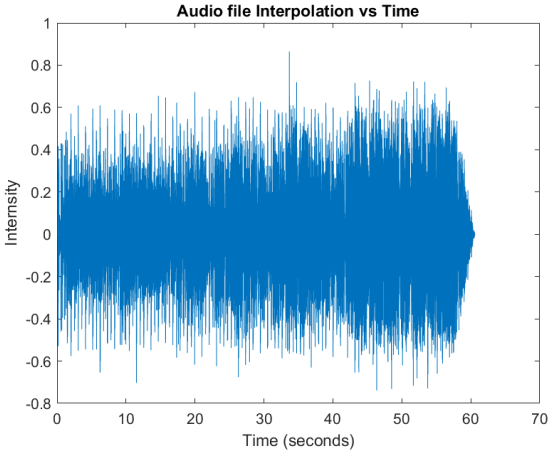
# 8 Simulation

## 8.1 Signal Generation

The audio signal's sampling rate was not enough to simulate the carrier frequency as well. If the original sampling rate was used, we would lose the carrier signal data and end up with a distorted output. Thus, in the simulation, we interpolated the audio file using interp1, and deinterpolated using a time average function at the very end. This wouldn't be necessary in a strictly analog world, but because we're simulating, we need to make several adjustments.




(a) This audio file has 2,910,576 samples.

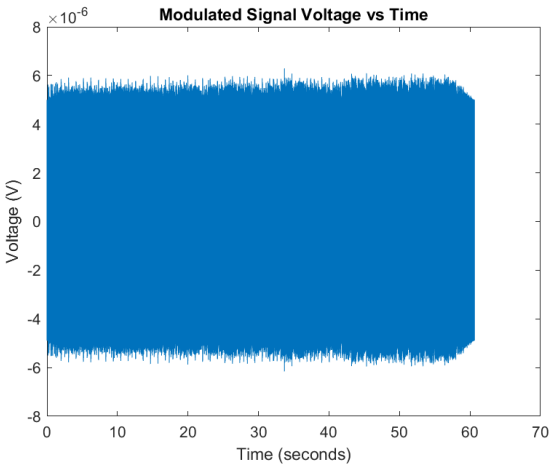



(b) This audio file has 509,350,626 samples.

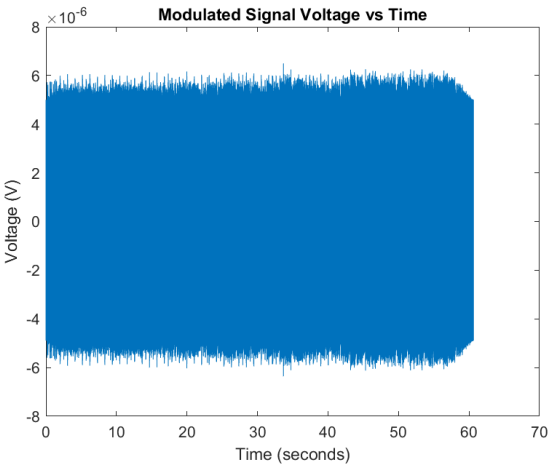
Figure 1: Figure 1a should look exactly the same as figure 1b, since interpolation will not lose any data.

## 8.2 Transmitted Wave

To simulate the transmission of the wave, we used the standard form of an amplitude modulated wave, as stated in the assignment. However, due to the low signal-to-noise ratio (SNR) in certain parts of the audio, we normalized the audio. This was done by dividing the message by the maximum intensity of the audio.




(a) Unnormalized Audio




(b) Normalized Audio

Figure 2: Figure 2a should look exactly the same as Figure 2b, since we are looking on a larger time scale.

## 8.3 Amplification

In this example, we will assume an absolutely absurd amplification of 1,000,000 once the modulated signal gets to the receiver. We believe that this is also exemplified in the real world, since crystal radios required a special, piezoelectric crystal earpiece which was sensitive to low voltages. However, in the modern world, with high impedance speakers and high voltage sound systems, amplification is necessary.

## 8.4 DC Bias

The DC Bias was added

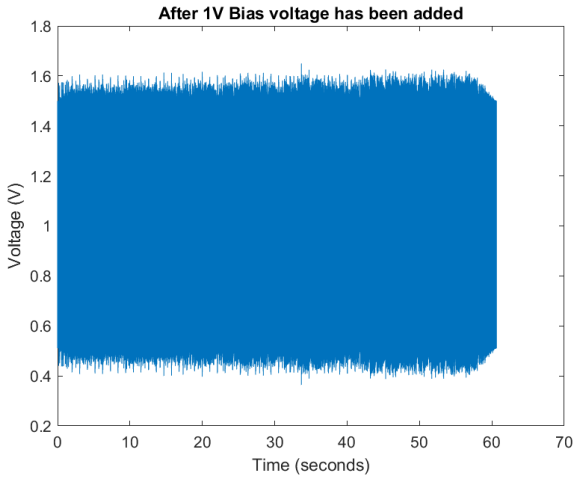



Figure 3: A 1V Bias voltage has been added

## 8.5 Diode

The diode was modeled off the practical ideal model of a diode, whose turn on voltage is 0.7. In a perfect world, the diode wouldn't deviate far from its rated forward voltage, so we will assume the perfect world in this report. The MATLAB simulation iterates through each sample. If the input value from the biasing step is lower than 0.7V, it will get recorded as 0. Otherwise, it will be copied over into the new matrix.

Additionally, we decided to model the later high pass filter in this step as well by removing all known DC components. We felt as though this simplified simulation, since the forward voltage drop would be dependant on current through the diode. By not simulating current (Which may change depending on the impedance of the headphones used to listen), we can simulate the ideal scenario.

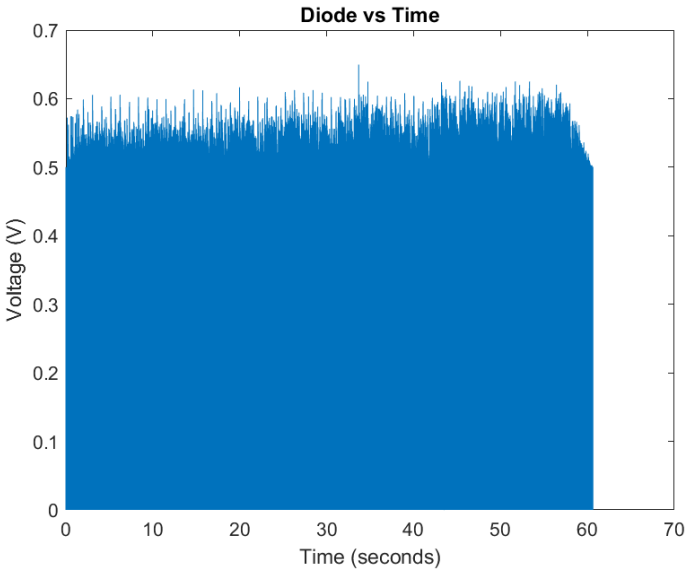



Figure 4: After the diode. Here, we see all values below 0 are cut off and the signal is now centered at 0.

## 8.6 Peak Detector

Similar to the diode, the peak detector was modeled by iterating through each sample and comparing its voltage to the previously applied voltage. If the new voltage was higher, the value would simply be copied over to a new matrix. However, if the value was lower, we'd record the last known highest value, as well as the time difference between them. The discharge capacitor equation was then applied to find the new voltage, which was copied to the new matrix.

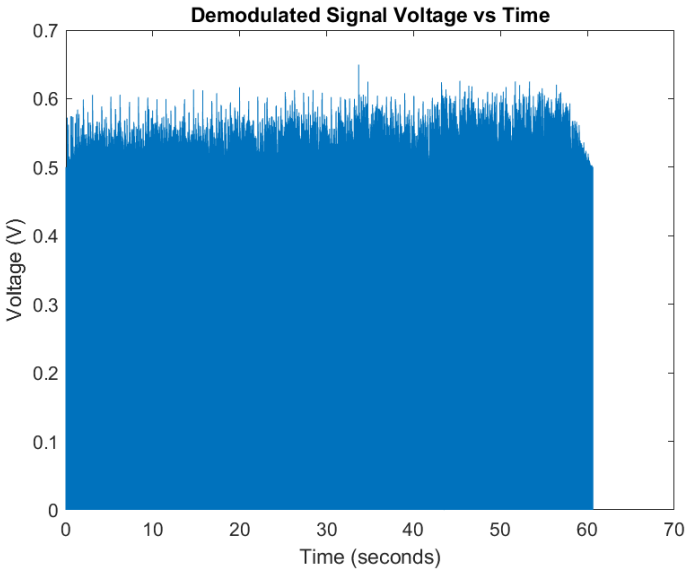



Figure 5: This may look the same as figure 4, but it lacks the 98.6MHz carrier wave. However, this cannot be seen due to the time scale of the audio file used.

### 8.7 High-Pass Filter

# 9 Conclusion

## 9.1 Errors

#### 9.1.1 18kHz

Upon playing the audio file, one could immediately tell that the output is distorted. There is a very audible hum, which we've identified as an 18kHz sine wave, probably a remnant from the peak detector.

#### 9.1.2 Distortion

When comparing the output wave to the original, we hear a nicely recovered audio sample. However, unsurprisingly, some information was lost after diode and peak detection. The diode cuts off half of the waveform. Seeing that audio files aren't totally symmetrical, theoretically we should not see a perfect copy of the wave after the rectification. On top of this, peak detection introduces a "jaggedness" of the demodulated waveform due to the RC circuit. This error is very subtle and requires a spectrum analyzer to be able to visually see the differences. On the other hand, there was also an addition of very low frequencies ranging from 20Hz to 58Hz in the output waveform. This is common in every audio file given to the simulation. However, we argue that this is due to the inherent nature of AM radio, which due to lower frequency usage, is susceptible to interference from weather, physical structures, or other electrical devices. Additionally only 5kHz of bandwidth is allocated to AM radio, meaning a faithful representation of sound is not possible. Thus, we have simulated real life.

*It's a feature, not a bug!!!* - Unknown

# Appendix

## A Matlab Code

```
%Global Variables
inputFile = "Gonna.mp3";
outputFile = "output.wav";
modulationIndex = 0.3;
phaseAngle = 0;
carrierFrequency = 560*10^3; %560kHz
R = 75000;
C = 1*10^-9;
interpolatedSample = 15;


useSample = false;
sampleFrequency = 440;
sampleR = 700;
sampleC = 1*10^-6;


%Read the audio file. Then, transpose and convert to single channel.
%y: Matrix containing the audio data. See:
↪  https://www.mathworks.com/help/matlab/ref/audioread.html#btiabil-1-y
%For MP3s: Audio data ranges between -1 and 1.
%Fs: Sample Rate
[y, Fs] = audioread(inputFile);
y = y.';
y = mean(y);
t = 0:1/Fs:(length(y)/Fs)-(1/Fs);

if useSample
    y = sin(2*pi*sampleFrequency*t);
    z=y;
    R = sampleR;
    C = sampleC;
    sound(y, Fs);
end


plot(t, y);
title("Audio file Intensity vs Time")
xlabel("Time (seconds)");
ylabel("Internsity");
if useSample
    axis([0 1/sampleFrequency -1 1]);
end
ax = gcf;
exportgraphics(ax,"Audio V Time.png");
%Creates the vector that represents the time at each sample, from 0 to the
%end

%Interpolation of Data
tInterpolated = 0:1/(interpolatedSample*carrierFrequency):(length(y)/Fs)-(1/Fs);
yInterpolated = interp1(t, y, tInterpolated);
```

```
plot(tInterpolated, yInterpolated);
title("Audio file Interpolation vs Time")
xlabel("Time (seconds)");
ylabel("Internsity");
if useSample
    axis([0 1/sampleFrequency -1 1])
end
ax = gcf;
exportgraphics(ax,"Audio V Time Interpolated.png");

%Begin Attempted FFT section

%{
%Sampling Frequency is 1/(interpolatedSample*carrierFrequency)
NFFT = length(yInterpolated);
%Length of signal is the size of the arrays
audioFFT = fft(yInterpolated, NFFT);
F = ((0:1/NFFT:1-1/NFFT)*(1/(interpolatedSample*carrierFrequency))).';

magnitudeY = abs(audioFFT);        % Magnitude of the FFT
phaseY = unwrap(angle(audioFFT));  % Phase of the FFT

%helperFrequencyAnalysisPlot1(F,magnitudeY,phaseY,length(yInterpolated));
dB_mag=mag2db(magnitudeY);
subplot(2,1,1);plot(F(1:NFFT/2),dB_mag(1:NFFT/2));title('Magnitude response of
↪  signal');
ylabel('Magnitude(dB)');
subplot(2,1,2);plot(F(1:NFFT/2),phaseY(1:NFFT/2));title('Phase response of
↪  signal');
xlabel('Frequency in kHz')
ylabel('radians');

cla()

%}
%End Attempted FFT section

vc =
↪  5.0*(1+(modulationIndex.*yInterpolated)).*cos(2*pi*carrierFrequency.*tInterpolated
↪  + phaseAngle)*(10^-6);

plot(tInterpolated, vc);
title("Modulated Signal Voltage vs Time")
xlabel("Time (seconds)");
ylabel("Voltage (V)");
ax = gcf;
if useSample
    axis([0 1/sampleFrequency -inf inf])
end
exportgraphics(ax,"Modulated Signal Before Normalization.png");

%I don't think we can do this. If we multiply the x(t) by anything, it'll
%screw with mu, the modulation index. However, what we do need to do is
%normalize x(t) to go from 1 to -1
%yInterpolated = yInterpolated*100000;

yInterpolated = yInterpolated * (1/max(yInterpolated));
```

```
%Creating the modulated signal, units are V

vc =
↪  5.0*(1+(modulationIndex.*yInterpolated)).*cos(2*pi*carrierFrequency.*tInterpolated
↪  + phaseAngle)*(10^-6);

plot(tInterpolated, vc);
title("Modulated Signal Voltage vs Time")
xlabel("Time (seconds)");
ylabel("Voltage (V)");
ax = gcf;
if useSample
    axis([0 1/sampleFrequency -inf inf])
end
exportgraphics(ax,"Modulated Signal After Normalization.png");

%Now that the signal is transmitted over the waves, lets amplify at the
%reciever module by a factor of 10^6

vc = vc*100000;

%Add the DC Bias, 0.7V
bias = vc + 0.7;
plot (tInterpolated, bias);
title("After 1V Bias voltage has been added")
xlabel("Time (seconds)");
ylabel("Voltage (V)");
ax = gcf;
if useSample
    axis([0 1/sampleFrequency -1 1])
end
exportgraphics(ax,"Bias.png");


diode = zeros(1,length(bias));
%Rectify the signal using an ideal practical diode with a forward drop of 1V.
diode = bias - 0.7;
for c = 1:length(bias)
    if diode(1,c) <= 0
        diode(1,c) = 0;
    end
end

%t = linspace(0,size(diode,2),size(diode,2));
%subplot(2,2,3)
plot(tInterpolated, diode);
title("Diode vs Time")
xlabel("Time (seconds)");
ylabel("Voltage (V)");
ax = gcf;
if useSample
    axis([0 1/sampleFrequency -1 1])
    %axis([0 0.0001 0 0.15])
end
exportgraphics(ax, "Diode Signal.png");
%axis([0 inf 0  0.000006])
```

```
%Then, we pass it through a simple peak detector consisting of a resistor
%and capacitor in parallel. The discharge of the capacitor is given to be
%V(t) = V0*e^(-t/(RC)), and the upcharge will follow the peaks. Thus, a
%piecewise function is formed.
%Charging the cap (Or when V(n-1) < V(n))
%Discharging the cap (Or when V(n-1) >= V(n))
peakDetector = zeros(1,length(diode));
peakDetector(1, 1) = diode (1, 1);
V0 = 0;
recentlyCharged = 0;
timeSinceCharge = 0;
activeCapacitor = 0;
for c = 2:length(diode)
    if diode(1, c-1) < diode(1, c) && activeCapacitor == 0
        peakDetector(1, c) = diode(1, c);
        V0 = diode(1, c);
        recentlyCharged = c;
    else
        %To find t, we need to figure out the number of samples since the
        %last charge of the capacitor, then divide by the samples per
        %second to get the time.
        %When the active capacitor discharges, we have to wait until it hits the
↪   function again
        timeSinceCharge = c - recentlyCharged; %Measured in Samples
        timeSinceCharge = timeSinceCharge/Fs; %Divided by samples per second to
↪   get seconds
        peakDetector(1, c) = V0 * exp(-(timeSinceCharge)/(R*C));
        activeCapacitor = 1;
        if peakDetector(1, c) <= diode(1, c)
            activeCapacitor = 0;
        end
    end
end
%Change the time vector

%t = linspace(0,size(peakDetector,2),size(peakDetector,2));
%subplot(2,2,4)
plot(tInterpolated, peakDetector);
title("Demodulated Signal Voltage vs Time")
xlabel("Time (seconds)");
ylabel("Voltage (V)");
ax = gcf;
if useSample
    axis([0 1/sampleFrequency -1 1])
    %axis([0.0016 0.002 0 0.15])
end
exportgraphics(ax, "Demodulated Signal.png");



x = round(carrierFrequency * interpolatedSample/Fs);
convolutionTable = zeros(1, x);
for c = 1:length(convolutionTable)
    convolutionTable(1,c) = 1/x;
end

counter = 0;
j = 1;
```

```
avg = 0;
output = zeros(1, 1);
for i = 1:length(peakDetector)
    avg = avg + peakDetector(1, i);
    counter = counter + 1;
    if (counter == x)
        output(1, j) = avg/x;
        j = j+1;
        counter = 0;
        avg = 0;
    end
end
%I tried to do it the smart way
%Add 0s until we can divide by x to get a whole number
%closeToEnd = mod(peakDetectSize, x);
%for i = size(peakDetector,2)-closeToEnd:size(peakDetector,2)+1
%    peakDetector(1, i) = 0;

%end

%audioAverage = reshape(peakDetector, round(size(peakDetector,2)/x), x);
%pspectrum(peakDetector,10000,"spectrogram")


sound(output, Fs);
audiowrite(outputFile, output, Fs);
```

## B Schematic: Power Supply

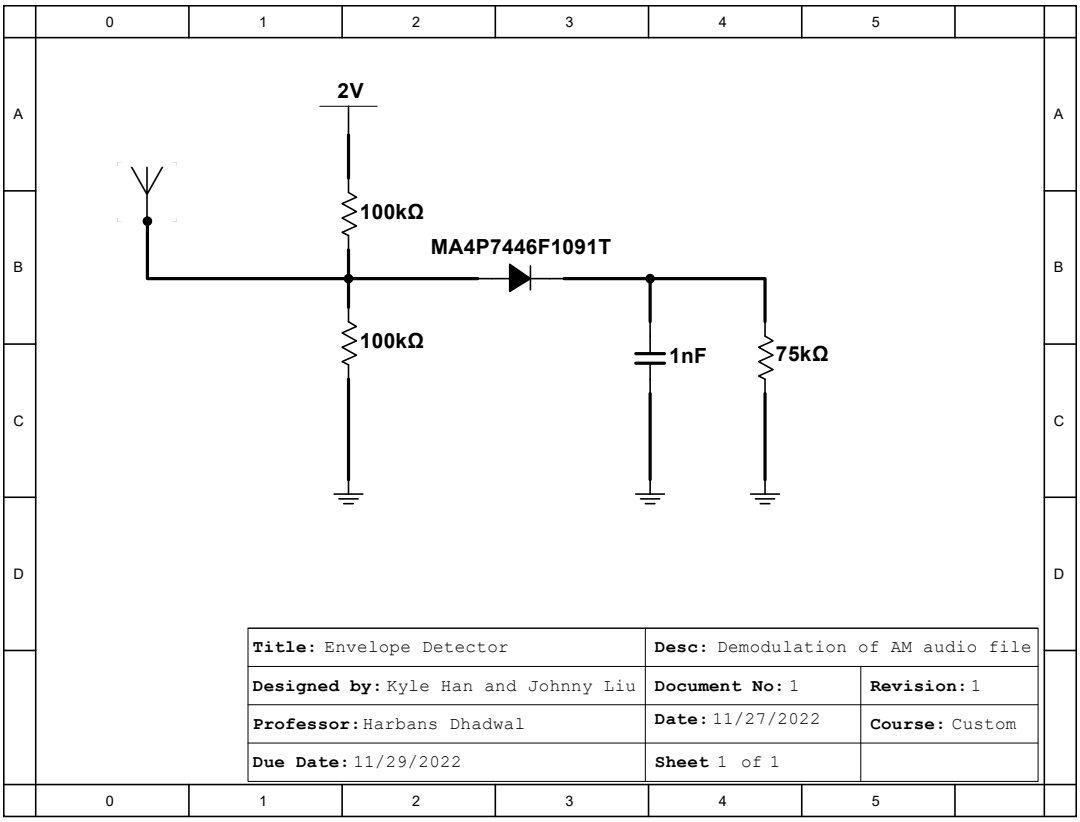



Figure 6: Envelope Detector, comprised of DC bias, diode, and RC circuit