Project 1: Analog Communication via Amplitude Modulation

1 Introduction

In this project, you will build and analyze complete AM communication systems based on different modulation and demodulation techniques.

1.1 Modulation

In the Simulink part of the project, you will make use of two modulation techniques, namely

1. Double Sideband suppressed carrier AM (DSB-SC-AM),
2. Conventional AM (DSB-AM).

Consider the *message signal* $m(t)$ to be transmitted over an additive white Gaussian noise (AWGN) channel by Amplitude Modulation (AM). A modulated DSB-SC-AM signal is generated by

$$u(t) = m(t)c(t), \tag{1}$$

where $c(t) = A_c \cos(w_c t)$ is the *carrier signal*. A modulated conventional AM signal is generated by

$$u(t) = (1 + \alpha m(t))\, c(t), \tag{2}$$

where $0 \leq \alpha \leq 1$ stands for the *modulation index*.

1.2 Reception

The received signal is the sum of the modulated signal and AWGN:

$$r(t) = u(t) + n(t), \tag{3}$$

where $n(t)$ is a white Gaussian process with the power spectral density of P_n. Note that, P_n is varied to model channels with different noise characteristics.

1.3 Demodulation

The information carrying message signal is extracted from the modulated signal via *demodulation*. In this project, you will make use of two types of it:

1. Coherent demodulation,
2. Envelope detection.

Coherent demodulation consists of a *local oscillator* and a *low-pass filter*. The local oscillator multiplies (mixes) the received signal with a sinusoidal, which is identical to the carrier signal. Then the mixed signal is passed through a lowpass filter to extract the message signal. On the other hand, the envelope detection is not coherent. In this type of detection, the received signal is first squared, then its high frequency components are filtered out by a low-pass filter. The low-pass filter output must be scaled appropriately in order to get the exact match of the message signal.

2 Part I: Simulink

You will design two AM communication systems in Simulink with the modulation/demodulation techniques mentioned in the previous section. Consider the *message signals*

$$m_1(t) = \cos(100\pi t), \tag{4}$$

$$m_2(t) = \left\{ \begin{array}{ll} 1, & 0 \leq t \leq 0.05 \\ -2, & 0.05 < t \leq 0.1 \\ 0, & \text{else} \end{array} \right\}, \tag{5}$$

$$m_3(t) = \text{sinc}(100t). \tag{6}$$

Now, follow the instructions below.

1. Design seperate sources to generate $m_1(t), m_2(t), m_3(t)$. Plot the messages in the time-domain and frequency domain. Don't forget to set an appropriate *sampling frequency* and *simulation time*. Make sure you are getting smooth curves as discussed in class.

2. Read the bandwidths of the messages on the spectrum analyzer and give their values. Which parameters determine their bandwidths? Comment.

3. Design a Simulink subsystem to implement DSB-SC-AM according to the Eq. (1). Set $w_c = 1000\pi$ rad/s.

4. Open a new blank model and name it as "DSB_SC_AM_system". Using the subsystem you just designed, build a model which modulates $m_1(t)$ with DSB-SC-AM. Plot the modulated signal in the time-domain and frequency domain. Now comment on the plots. What changed in the time-domain and frequency-domain during this modulation?

5. Design a Simulink subsystem to implement conventional AM, according to Eq. (2). Set $\alpha = 0.85$, $w_c = 1000\pi$ rad/s.

6. Open a new blank model and name it as "conv_AM_system". Using the subsystem you just designed, build a model which modulates $m_1(t)$ with the conventional AM. Plot the modulated signal in the time-domain and frequency domain. Now comment on the plots. What changed in the time-domain and frequency-domain during this modulation?

7. Consider the signal in Eq. (1). Write down the equations governing the synchronous demodulator. Explain how the message signal is extracted from the modulated signal. What should be the frequency of the local oscillator? What should be the cut-off frequency of the low-pass filter?

8. Design a Simulink subsystem to implement the synchronous demodulator.

9. Now insert the synchronous demodulator into "DSB_SC_AM_system". Perform demodulation and plot the demodulated signal in the time-domain and frequency domain. Now comment on the plots. What changed in the time-domain and frequency-domain during the demodulation? Does the demodulated signal resemble the message signal?

10. Consider the signal in Eq. (2). Write down the equations governing the envelope detector. Explain how the message signal is extracted from the modulated signal. What should be the cut-off frequency of the low-pass filter?

11. Design a Simulink subsystem to implement the envelope detector.

12. Now insert the envelope detector into "conv_AM_system". Perform demodulation and plot the demodulated signal in the time-domain and frequency domain. Now comment on the plots. What changed in the time-domain and frequency-domain during the demodulation? Does the demodulated signal resemble the message signal?

13. Consider both the "DSB_SC_AM_system" and "conv_AM_system" models. Implement AWGN on both systems with the "Band-Limited White Noise" block. Now generate *received signals* for both systems according to Eq. (3) with $P_n = \{10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}\}$ and perform demodulation. Plot the demodulated signals both in the time-domain and frequency domain. How does AWGN change the demodulated signals in both systems? Comment.

3 Part II: Arduino

In this part, you will design an AM communication system using the Arduino board as the communication channel. Follow the instructions given below.

1. By using the given sound creator code[5.1] code and $m(t) = \cos(20\pi t)$, create an AM signal using

$$y(t) = (A + m(t))cos(2\pi f_c t). \tag{7}$$

   Choose the value of A and explain how you chose it. Additionally, determine the carrier frequency and explain your reasoning.

   Hint: Do not forget the sampling frequency of Arduino. You may want to choose carrier frequency accordingly.

2. Set up your Arduino board and DC bias circuit as it is shown in lecture to transmit the analog signal from an Aux Cable to Arduino board.

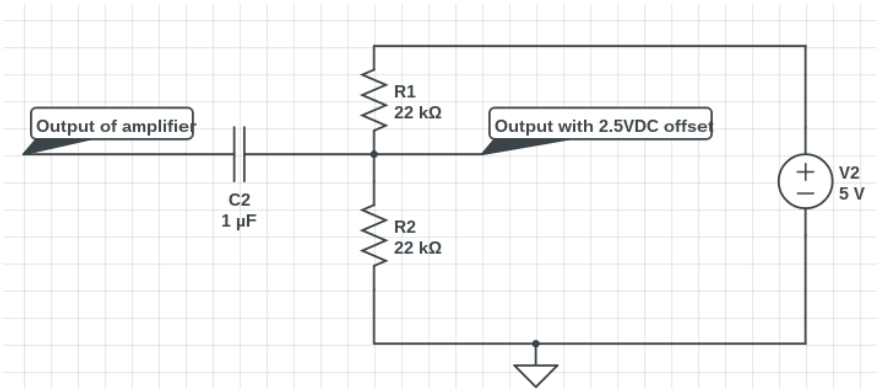

Figure 1: DC Bias Circuit.

3. Use the given Arduino Code[5.2] to read the AM signal from one of the analog pins of Arduino board and write it to the serial interface.

4. Use the given MATLAB code[5.3] to read the AM signal from the serial interface and put the sampled received signal data into the data vector.

5. Arduino boards sample the input signals with frequency of 10 kHz. However, writing the data to the serial interface and reading the data from the interface via MATLAB slows down the sampling process so the sampling frequency is generally smaller than 10 kHz. Before implementing any kind of filter, knowing the sampling rate is critical. Therefore; before proceeding further, determine the sampling frequency of the received signal. Explain how you determined the sampling frequency.

   Hint: There are various "online tone generators" on the internet which could generate sinusoidal signals. Knowing the received signal's frequency may help you to determine sampling frequency.

6. Plot the received signal in MATLAB and observe the received AM signal.

7. Apply envelope detector to the received signal to demodulate the signal. Explain how you implemented the envelope detector and how you chose the parameters of the detector.

   Hint: Removing the DC Bias from the AM signal could be useful before implementing envelope detector.

8. Plot the original signal and demodulated signal side by side. Comment on the similarities and differences between two signals. Then, calculate the Mean Square Error between the original signal and the received signal.

   Amplitude of the sound signal is not an important property since it only affects the volume of the sound. Therefore, scaling the demoudulated signal according to the original signal is a good idea.

9. Lastly add AWGN with PSD of $P_n = \{10^{-2}, 10^{-6}\}$ and repeat the above steps. Comment on the results of 3 cases.

4 Report and Demonstration

You are required to prepare a report and perform an online demonstration seperately for each part of this project.

In your reports, simply enumerate your replies to the instructions given in the previous sections. Each of your replies should provide the relevant figures, derivations and/or explanations, necessary to complete the instruction. Make sure you include the screenshots for your Simulink subsystem/system designs and the digital photo of your Arduino setup in your report under the corresponding instruction.

Your reports should be prepared electronically (on Office, LATEX, etc.). Handwritten reports will not be accepted.

The procedure for demonstrations will be announced on MOODLE.

Make a .zip file consisting of your two reports (for Part I and Part II), all Simulink and MATLAB files and upload it to MOODLE before the announced deadline.

5 Appendix

5.1 Sound Creator Code MATLAB

```
Fs = 14400;      % Sampling Frequency, generally chosen as 14400
duration = 20;       % Duaration of the sound file.
t  = linspace(0, duration, duration*Fs);         % Time Vector.
s = ;        % Signal.
s = s/max(s); %scale to the signal to fit between 1 and  1.
sound(s, Fs);        % Creating the sound file.
```

5.2 Arduino Code

```
int sensorValue = 0;
void setup() {
Serial.begin(115200);} // serial communication hizini belirleme
void loop() {
sensorValue = analogRead(A0); // A0 portundan sample alma
sensorValue = sensorValue / 4; // 10 bitlik sample'i precision kaybi yaparak 8bit'e
// cevirmis oluyoruz
Serial.write(sensorValue);} // serial port'tan PC'ye yollama
```

5.3 MATLAB Code

```
priorports=instrfind; % halihazirdaki acik portlari bulma
delete(priorports); % bu acik port'lari kapama (yoksa hata verir)
s = serial('COM1'); % bilgisayarinizda hangi port olarak define edildiyse,
% o port'u girin.. COM1, COM2, vs..

s.InputBufferSize = 10000; % serial protokolunden oturu, datayi bloklar halinde
% almaniz gerekiyor. kacar bytelik bloklar halinde  almak istiyorsaniz, onu girin

set(s,'BaudRate', 115200); % arduino'da set ettigimiz hiz ile ayni olmali
fopen(s); % COM portunu acma
while 1 % surekli okuma (ya da belli sayida blok okumak icin for kullanin)
data = fread(s);
% datayi bloklar halinde alma. bu islemi yaptiginzda 0 ile 255 arasi
% degerler iceren, yukarida InputBufferSize'i kac olarak
% belirlediyseniz o boyutta bir vektorunuz olacaktir.
end
fclose(s); % serial port'u kapatmak
```