Project 2: Analog Communication via Frequency Modulation

1 Introduction

In this project, you will build and analyze a complete FM communication system. Consider the following *message signal*

$$m(t) = -a\sin(2\pi f_m t), \tag{1}$$

to be modulated by Frequency Modulation (FM). A frequency modulated signal is obtained as

$$u(t) = A_c \cos\left(2\pi f_c t + \phi(t)\right), \tag{2}$$

where f_c is the *carrier frequency* and

$$\phi(t) = 2\pi k_f \int_{-\infty}^{t} m(\tau)d\tau \tag{3}$$

is the information-carrying phase. Here, k_f is called the *deviation constant* of FM. When $m(t)$ in (1) is modulated according to (2), we get

$$u(t) = A_c \cos\left(2\pi f_c t + \beta_f \cos\left(2\pi f_m t\right)\right). \tag{4}$$

Here,

$$\beta_f = \frac{k_f a}{f_m} \tag{5}$$

is the *modulation index* of FM. The modulated sinusoidal signal can be represented by a weighted sum of its higher order harmonics as follows.

$$u(t) = \sum_{n=0}^{\infty} A_c J_n(\beta_f)\cos(2\pi(f_c + nf_m)t), \tag{6}$$

where $J_n(\beta_f)$ is the Bessel function of the first kind and of order n. Due to (6), the bandwidth of $u(t)$ is infinite. However, it has an effective bandwidth in which most of the signal energy is carried. This bandwidth is approximately found by Carson's rule as

$$B_T = 2(\beta_f + 1)W, \tag{7}$$

where W is the bandwidth of the message.

One can demodulate $u(t)$ with a phase-locked loop (PLL), which is a negative feedback loop with a voltage controlled oscillator, and locked to the phase of the input signal. The PLL system is already implemented in Simulink with the "Phase-Locked Loop" block. You will use that block in your design. Note that the phase output of the Simulink PLL block must be further processed to accurately estimate the message signal.

2 Part I: Simulink

In this part, design a complete FM communication system in Simulink with a PLL demodulator as explained in the previous section. Consider the *message signal* in (1) with $a = 1, f_m = 50$ Hz. Now, follow the instructions below.

1. Open a new blank model and name it as "FM_system". In the model window, design a source to generate $m(t)$. Verify the message with the help of the scope and the spectrum analyzer. Don't forget to set an appropriate *sampling time* and *simulation time*. Make sure you are getting smooth curves as discussed in class.

2. Read the bandwidth of the message on the spectrum analyzer and give its value. Which parameters determine its bandwidth? Comment.

3. Design a Simulink subsystem to implement frequency modulation according to (2). Set $A_c = 1$, $k_f = 30$ and $f_c = 200$ Hz.

4. Now create the time- and frequency-domain plots of $m(t)$, the integral of the message ($2\pi k_f \int m(\tau)d\tau$) and the modulated signal $u(t)$ (2 figures, 3 curves each). Scale the axes appropriately. What are the expected patterns of the three signals in both domains? Can you observe those patterns on your plots? If not, revisit the previous step.

5. Describe the power spectrum of the modulated signal considering the message spectrum and (6).

6. Find the bandwidth of the modulated signal according to the Carson's rule. Then read its bandwidth on the spectrum analyzer. Are the two values different or the same? Explain.

7. Explain shortly, how a phase locked loop is useful for demodulation.

8. Use the "phase-locked loop" block to extract the phase of $u(t)$. Open the block parameter window to set the appropriate VCO frequency and phase values. Connect $u(t)$ to the input port of the PLL block. Get the extracted phase, $\phi(t)$ from the PD port of the PLL block. Now, create time-domain and frequency domain plots for $\phi(t)$ and $m(t)$ (2 figures, 2 curves each). Scale the axes so that the signals are readable. Now comment, is there any resemblance between $\phi(t)$ and $m(t)$?

9. Consider Eq. (3) and the time&frequency-domain plots of $\phi(t)$ and write down the equations/transfer functions to extract $m(t)$ out of it.

10. Now design a Simulink subsystem, which takes $u(t)$ as the input and outputs the demodulated signal, $\hat{m}(t)$. Insert the PLL demodulator into "FM_system". Perform demodulation and create time&frequency-domain plots for $m(t)$, $u(t)$ and $\hat{m}(t)$ (2 figures, 3 curves each). Scale the axes appropriately. Comment on the graphs. What changed in the time-domain and frequency-domain during the demodulation? Does the demodulated signal resemble the message signal?

3 Part II: Arduino

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

1. By using the given sound creator code[1] code and $m(t) = cos40\pi t$, create an FM signal using the following formula.

$$y(t) = cos\left(2\pi f_c t + 2\pi k_f \int_{-\infty}^{t} m(\tau)d\tau\right) \tag{8}$$

   Choose the values of k_f, the deviation constant, and f_c, the carrier frequency. Explain your reasoning.

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

2. Create the frequency-domain and time-domain plots for the modulated signal and show that the given FM signal's bandwidth matches the theoretical Carson's Rule.

3. 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.

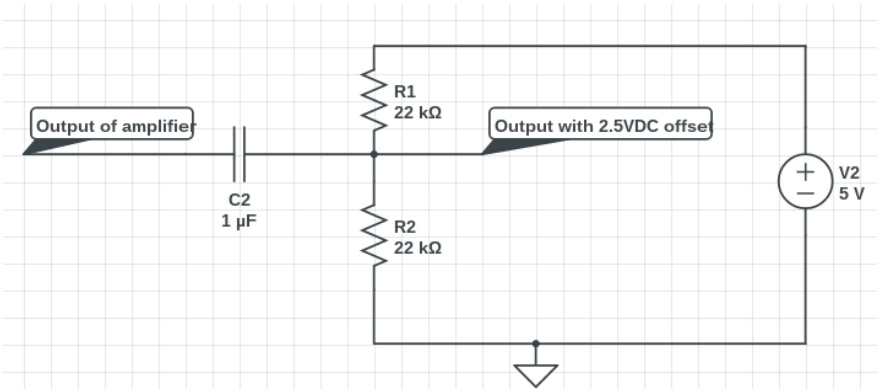

Figure 1: DC Bias Circuit.

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

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

6. 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.

7. Plot the received signal in MATLAB and observe the received FM signal.

8. Take the derivative of the received FM signal.

   Hint: Removing the DC Bias from the FM signal could be useful before demodulation. Additionally, you can use the **diff** *function from MATLAB to approximate derivative operation.*

9. After the derivative operation, observe that the received FM signal takes the shape of an AM signal. Therefore, you can apply the envelope detector to the resulting signal as you did in the AM signal demodulation.

10. 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.

    Again, the amplitude of the demodulated signal is not important as it only determines the volume of the sound. Therefore, scaling the demodulated signal according to the original signal is a good idea.

4 Report and Demonstration

You are required to prepare a report and perform an online demonstration separately 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
```