Sefa Kayraklık
2015401066

Project 1

Part I: Simulink

1

The given messages $m_1(t)$, $m_2(t)$, and $m_3(t)$ are generated by using sinusoidal wave source; three separate step sources; and a clock source and a function generator block; respectively. The block diagrams of the messages are shown in the Fig. 1.

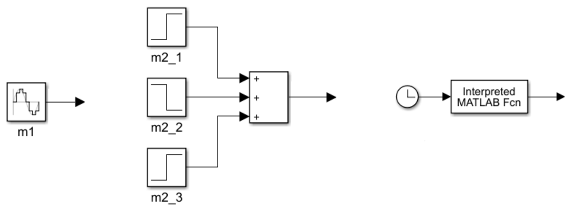

Fig. 1. The block diagram of $m_1(t)$, $m_2(t)$, and $m_3(t)$

The time domain and frequency domain representations of the messages are shown in the Fig. 2, Fig. 3, and Fig. 4, respectively.

The sample time and simulation time for $m_1(t)$ and $m_2(t)$ are selected as $5 * 10^{-4}$ and *1 second*; and for $m_3(t)$, they are chosen as $5 * 10^{-4}$, and *2 seconds* (but it starts from *-1 second*).

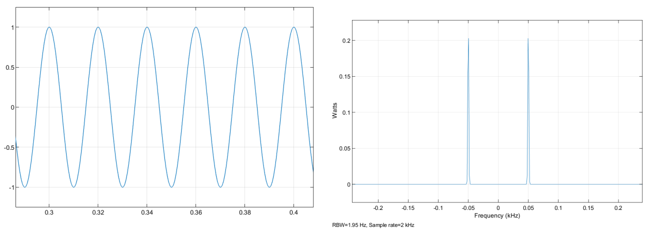

Fig. 2. The time and frequency domain plots of the $m_1(t)$ signal

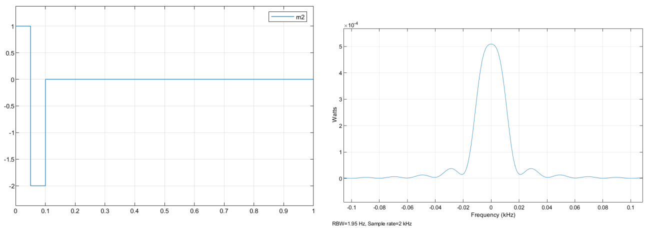

Fig. 3. The time and frequency domain plots of the $m_2(t)$ signal

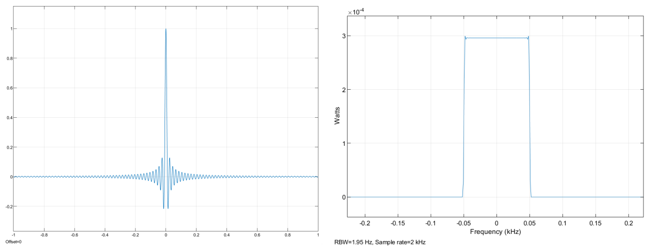

Fig. 4. The time and frequency domain plots of the $m_3(t)$ signal

2

The bandwidth of a signal can be determined by observing the width of the signal is the signal is band limited; if not, the width and the location of the main lope level of the signal's spectrum is the determinant part for bandwidth. So, The bandwidth of the messages, $m_1(t)$, $m_2(t)$, and $m_3(t)$ can be observed from the spectrum analyzer as $50Hz$, $20Hz$, and $50Hz$; respectively.

The bandwidth of a signal is mainly determined by its frequency. If the signal is not periodic then its fourier transform will display the contribution of each frequency component, and conventionally, the half power level of the frequency decides the bandwidth.

3

A subsystem named *DSB_SC_AM* is designed to modulate the input with carrier frequency, f_c, 500 Hz ($w_c = 1000\pi$) and gives the modulated output. The block diagram of the subsystem is shown in the Fig. 5

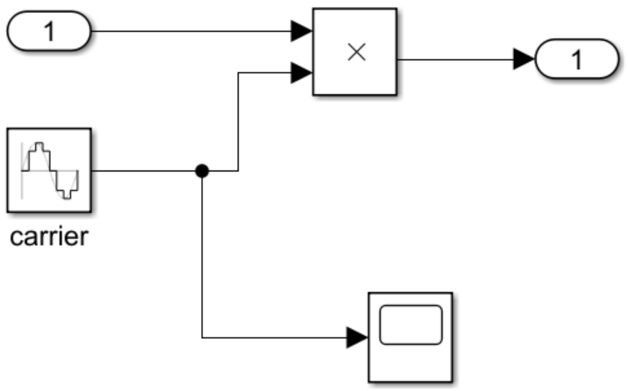

Fig. 5. The block diagram of *DSB_SC_AM*

4

The message signal $m_1(t)$, which is given in the part 1, is modulated using the above mentioned subsystem with carrier frequency, f_c, of $500Hz$ ($w_c = 1000\pi$). The time and frequency domain plots of the modulated signal are given in the Fig.6, respectively.

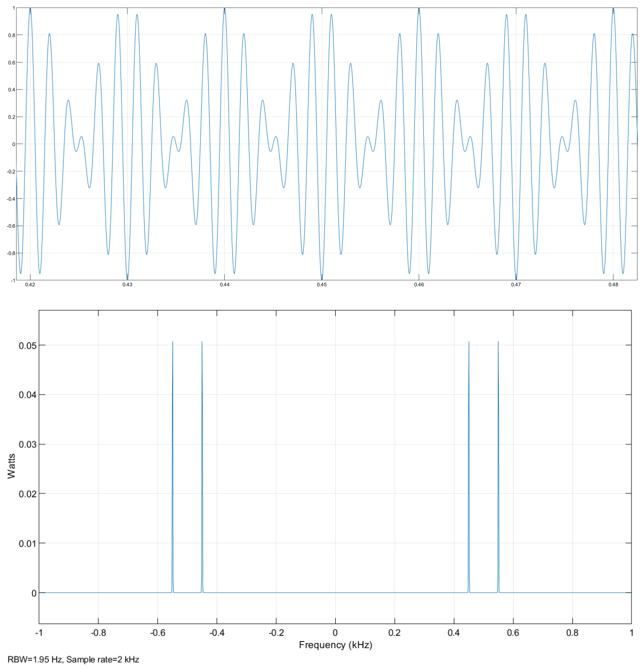

Fig. 6. The time and frequency domain plots of the modulated signal

It can be seen in the time domain that the message signal is multiplied by a cosine with much higher frequency (500 Hz); so, now, the modulated signal is a sum of two cosine signals with 450 Hz and 550 Hz frequencies.
Moreover, the above finding can be observed from the frequency domain representation: the message signal is shifted 500 Hz to positive and negative sides by convolving with two diracs, which form the carrier cosine signal.

5

A subsystem named *conv_AM* is designed to modulate the input with carrier frequency, f_c, 500 Hz ($w_c = 1000\pi$), and modulation index, α, 0.85 and gives the modulated output. The block diagram of the subsystem is shown in the Fig. 7

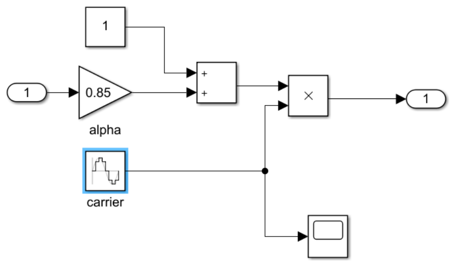

Fig. 7. The block diagram of *conv_AM*

6

The message signal $m_1(t)$, which is given in the part 1, is modulated using the above mentioned subsystem with modulation index, α, 0.85 and the carrier frequency, f_c, of $500Hz$ ($w_c = 1000\pi$). The time and frequency domain plots of the modulated signal are given in the Fig.8, respectively.

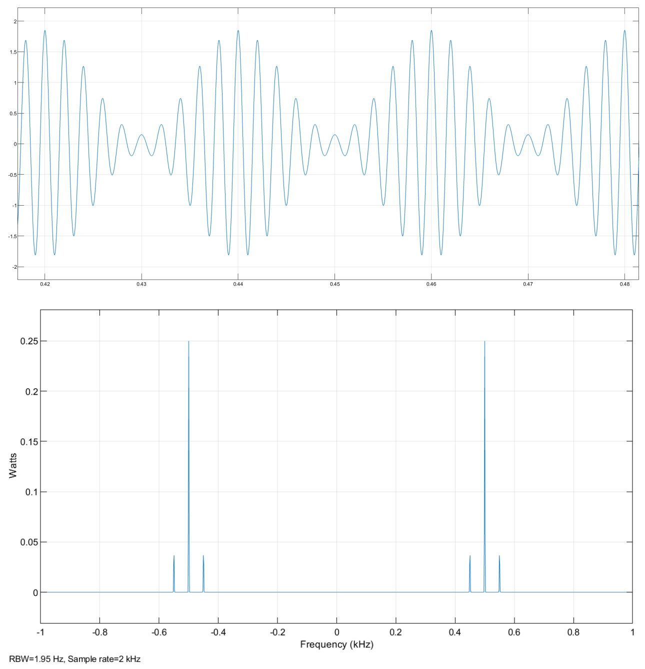

Fig. 8. The time and frequency domain plots of the modulated signal

It can be observed from the Eq. (2) which is given in the project description that the modulated signal has 3 different cosine signals, two of them is the same as the previous modulation method and the third one is the carrier signal. In the conventional AM, the carrier signal is also transmitted. Also, this can be seen in the frequency spectrum: the spectrum has three pairs of diracs which represent the cosine signals, the one in the middle is the carrier and the side ones are the message signal as in the previous method.

7

The synchronous demodulator uses a cosine signal with the same frequency as the carrier; so that it can move the message signal to the level where it was before the modulation. Namely, the demodulator multiplies the received signal with the same carrier signal and obtains two signals, the one is in the baseband, which is the sent message and the other one is in the $2f_c$ band. Also, the demodulator should have a gain of 2 since multiplication of cosines brings a coefficient of 1/2. In order to obtain the received signal, after the multiplication, the resulting signal should pass through a low pass filter. The equation governing the demolutation method is shown in the following expression:

$$\begin{aligned}&y(t) * 2c(t) = m(t) * c(t) * 2c(t)\\&= m(t) * cos(w_c t) * cos(w_c t) = m(t)(1 + cos(2w_c t))\\&= m(t) + m(t)cos(2w_c t) \xrightarrow{LPF} m(t) = \hat{m}(t)\end{aligned}$$

In order to shift the received signal exactly the same location as before, the frequency of the demodulator's local oscillator should be the same as the carrier signal. In addition, the cut-off frequency of the low pass filter is needed to be just higher than the bandwidth of the sent signal. Therefore, in our case, the local oscillator frequency is $500Hz$ and the low pass filter's cut-off frequency is $60Hz$.

8

The proposed synchronous demodulator is implemented in Simulink and a subsystem named *Coherent_Demodulator* is designed with the gain of 2, the local oscillator frequency is set to $500Hz$ and the low pass filter's cut-off frequency is set to $60Hz$. The block diagram of the subsystem is shown in the Fig. 9.

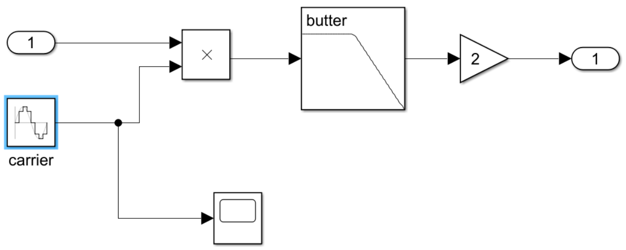

Fig. 9. The block diagram of *Coherent_Demodulator*

9

The received signal, $y(t)$, which is generated in the part 4, is demodulated with the above mentioned subsystem, and the time and frequency domain plots of the demodulated signal are given in the Fig. 10, respectively.

So, it can be seen in the time domain that the original signal is extracted from the modulated signal. The process of obtaining original signal can be observed more clearly from the frequency domain: The modulated signal is shifted to baseband and $2f_c$ band by multiplying with the carrier, via the low pass filter, the unwanted signals are removed and the sent signal is acquired.

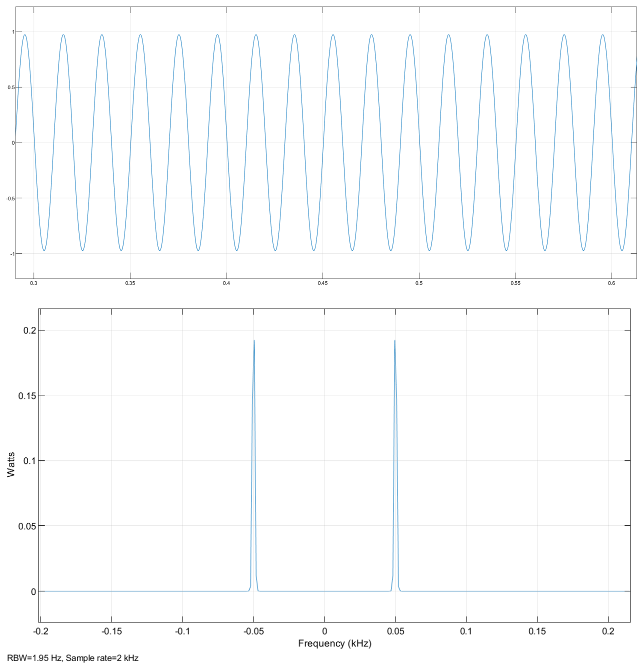

Fig. 10. The time and frequency domain plots of the demodulated signal

10

The envelope detector processes the received signal by first squaring and then filtering out the high frequency component. After the filter, a DC blocker should be used to remove the DC part of the result. Finally, with the appropriate gain value, the received signal can be recovered. The following expressions show the procedure:

$$\begin{aligned}&y(t) * y(t) = (1 + \alpha m(t))c(t) * (1 + \alpha m(t))c(t)\\&= (1 + \alpha m(t))^2 * c(t)^2\\&= (1 + 2\alpha m(t) + \alpha^2 m(t)^2)\frac{1}{2}(1 + cos(2w_c t))\\&\xrightarrow{LPF} \frac{1}{2}(1 + 2\alpha m(t)) \xrightarrow[Gain]{DCBlocker} m(t) = \hat{m}(t)\end{aligned}$$

In order to obtain the original message, the cut-off frequency of the low pass filter is needed to be just higher than the bandwidth of the sent signal since it should remove both the high component of the carrier part ($cos(2w_c t)$), and the squared sent message ($\alpha^2 m(t)^2$). Therefore, in our case, the low pass filter's cut-off frequency is set to $60Hz$. Also, the gain should be selected as $1/\alpha$, and an DC blocker block should be used to remove the DC part of the resulting signal.

11

The proposed envelope detector is implemented in Simulink and a subsystem named *Envelope_Detector* is designed with the gain of $1/0.85$, the low pass filter's cut-off frequency is set to $60Hz$, and a DC blocker. The block diagram of the subsystem is shown in the Fig. 11.

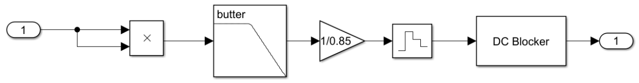

Fig. 11. The block diagram of *Envelope_Detector*

12

The received signal, $y(t)$, which is generated in the part 6, is demodulated with the above mentioned subsystem, and the time and frequency domain plots of the demodulated signal are given in the Fig. 12, respectively.

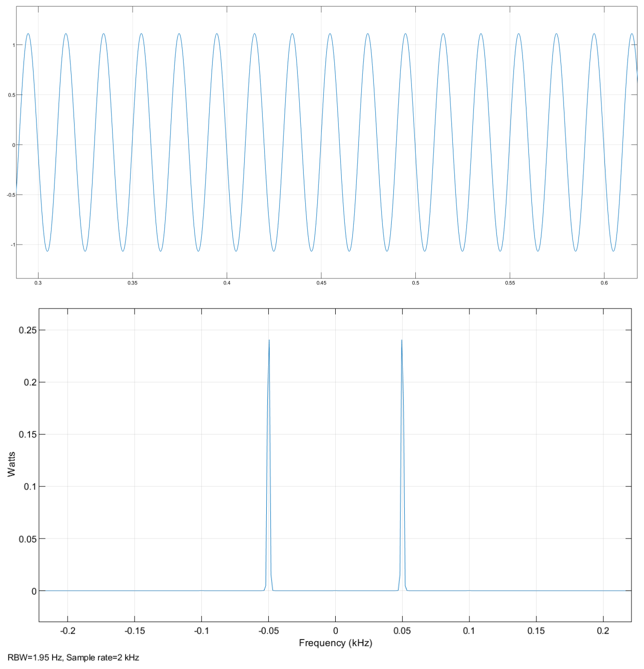

Fig. 12. The time and frequency domain plots of the demodulated signal

It can be seen in the time domain that the original signal is extracted from the modulated signal. However, the gain of the envelope detector should be lower than the implemented value since the demodulated signal has higher power than the original signal. In fact, this is unexpected since through the theoretical calculation, the gain should be $1/0.85$.

Similarly, the process of obtaining original signal can be observed more clearly from the frequency domain: The modulated signal is shifted to baseband and $2f_c$ band by multiplying with the carrier, via the low pass filter, the unwanted signals are removed and with the DC blocking operation the DC part which is formed by shifting the carrier signal is eliminated.

13

The AWGN is implemented on both systems with the *Band-Limited White Noise* block. The noise is added to the signals which are modulated with the blocks of *DSB_SC_AM*, and *conv_AM* before passing them to the demodulator blocks of *Coherent_Demodulator*, and *Envelope_Detector*, respectively.

The overall systems of communication are given in the Fig. 13 which shows the *DSB_SC_AM_system*, and *conv_AM_system*, respectively.

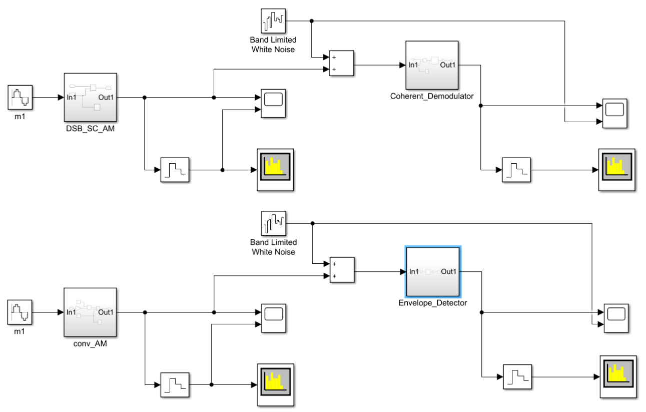

Fig. 13. The overall implemented communication systems

The powers of the AWGN are selected as $P_N = \{10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}\}$. The demodulated signals for the *DSB_ SC_AM system* are given in the Fig. 14, in the order of noise power; and for the *conv_AM_system* they are given in the Fig. 15.

It is obvious that adding a high power AWGN disrupts the received signal very severely. In the *DSB_ SC_AM system*, the noise with the 10^{-2} power is enough to destroy the original signal, on the other hand, the signal can be still obtained, not as clear as the original signal, from the signal with the 10^{-4} power. In fact, in the low power AWGN, the original signal can be demodulated as seen in the figures last two rows. In the *conv_AM_system*, the signal completely is lost with the noise of 10^{-2} and 10^{-4} power since in the demodulation process the received signal is squared, the noise also is squared; thus, the noise power becomes much higher. Therefore, it would be wise not to use the method proposed in the part 11 if the channel has high noise level.

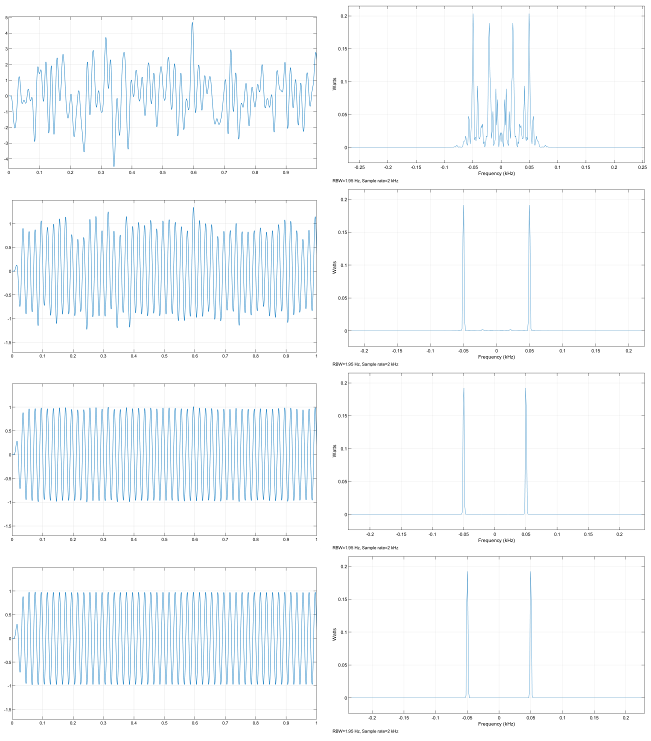
Fig. 14. The time and frequency domain plots of the demodulated signals with noise in the order of decreasing noise power

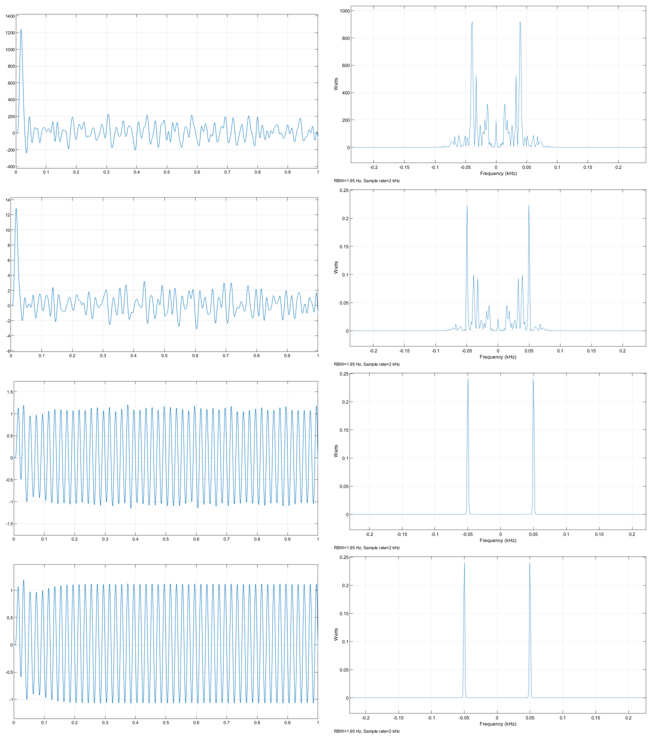
Fig. 15. The time and frequency domain plots of the demodulated signals with noise in the order of decreasing noise power

Part II: Arduino

1

The message signal, $m(t) = cos(2\pi 10t)$, is AM modulated with the following formula

$$y(t) = (A + m(t))cos(2\pi f_c t)$$

And and the modulated signal is generated with the following MATLAB 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.
A = 1.5;
fc = 500;
m = cos(2*pi*10*t);
s = (A+m).*cos(2*pi*fc*t); % Signal.
s = s /max(s) ; %scale to the signal to fit
    between 1 and 1.
%sound(s, Fs); % Creating the sound file.
audiowrite("y.wav", s, Fs);
figure(1)
plot(t,s)
title("The modulated signal")
xlabel("Time (s)")
ylabel("Value")

%% Generate a sinusoidal with a know frequency to
     determine the arduino sampling frequency

sin500hz = sin(2*pi*500*t);
audiowrite("sin.wav", sin500hz, Fs);

%% Adding AWGN

s_n1 = awgn(s,20,'measured');
audiowrite("y_n1.wav", s_n1, Fs);
figure(2)
plot(t,s_n1)
title("The modulated signal which has a 10^-^2
    power AWGN")
xlabel("Time (s)")
ylabel("Value")

s_n2 = awgn(s,60, 'measured');
audiowrite("y_n2.wav", s_n2, Fs);
figure(3)
plot(t,s_n2)
title("The modulated signal which has a 10^-^6
    power AWGN")
xlabel("Time (s)")
ylabel("Value")
```

The determination of the value of A is done by considering the demodulation method of envelop detector. For envelope detection to function properly, $A + m(t)$ should be bigger than zero and A should not be so large since it can lead to saturation in the analog output of the computer; thus, A is selected as 1.5

The carrier frequency, f_c, is selected as $500Hz$ since it is known that the sampling frequency of the arduino is smaller than $10kHz$, so any frequency much smaller than that number and higher than the transmitted signal would work.

The AM modulated signal, $y(t)$, which is obtained by the above script is shown in the Fig. 16.

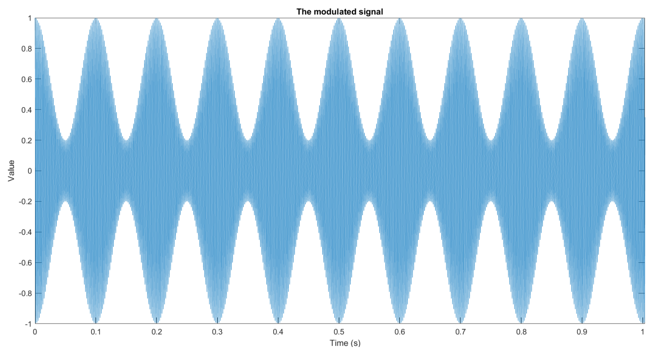

Fig. 16. The modulated signal $y(t)$

2

In order to add a DC bias to the computer analog output, the given circuit in the project description is set up to the arduino board. The Fig. 17 shows the constructed arduino setup.

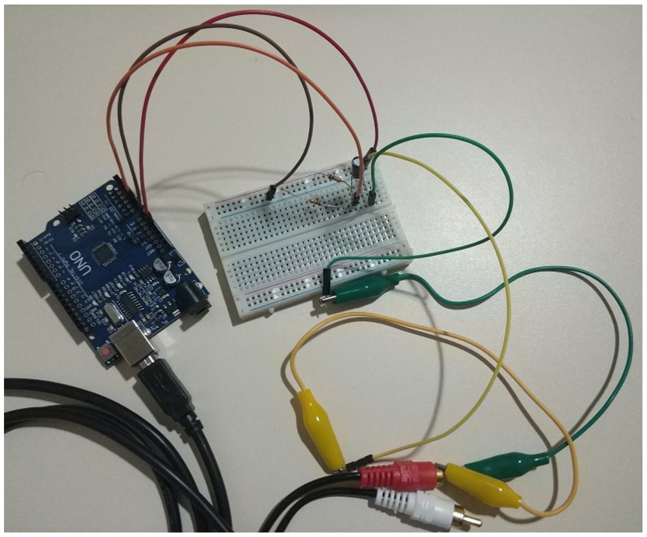

Fig. 17. The arduino setup

3

The following arduino code is used to read the AM signal from one of the analog pins of arduino board and write it to the serial interface.

```
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 iprecision kaybi yaparak 8 bit e
    cevirimis oluyoruz
Serial.write(sensorValue); } // serial port dan
    PC ye yollama
```

4

The following MATLAB code is used to read the AM signal from the serial interface and put the sampled received signal data into the data vector.

```
priorports=instrfind; % halihazirdaki acik
    portlari bulma
delete(priorports); % bu acik port lari kapama (
    yoksa hata verir)
s = serial('COM3'); % bilgisayarinizda hangi port
     olarak define edildiyse,
% o portu 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

time0 = tic;
figure(1);
while toc(time0) < 5 % 5 sn ye boyunca data al
   data = fread(s);
   drawnow;
   plot(data)
% datayi bloklar halinde alma. bu islemi
    yaptiginzda 0 ile 255 arasi
% degerlericeren , yukarida InputBufferSize i kac
     olarak
% belirlediyseniz o boyutta bir vektorunuz
     olacaktir .
end
fclose(s); % Serial port u kapatmak

%% Plot received signal (the last 10000 samples
    of the input signal)
Fs=8910; % In the part 5, the sampling frequency
    of arduino is found 8910Hz

figure(2)
plot([1:10000]/Fs,data);
title("The received signal")
xlabel("Time (s)")
ylabel("Value")
%% mapping the data into [-1 1] range

data = data - 122; % The added DC value by the DC
     biasing circuit is 122
data = data/60; % mapping the data into [-1 1]
plot(data)
%% Demodulation with the envelope detector method
     given in the Simulink part

Fpass = 15;

data_squared = data.*data;
%plot(abs(fftshift(fft(data_squared))))

data_lowpassed = lowpass(data_squared, Fpass, Fs,
     'Steepness',0.95);
%plot(abs(fftshift(fft(data_lowpassed))))

data_demodulated = data_lowpassed - 0.26; %
    removing Dc component
%plot(abs(fftshift(fft(data_demodulated))))

data_demodulated = 4.25*data_demodulated; %
    scaling into [-1 1] range

%% PLoting the result
figure
t = linspace(1/Fs,1,Fs);
m = cos(2*pi*10*t);
subplot(211)
plot(t,data_demodulated(Fs/10+1:1.1*Fs)); % a 1
    sec interval is plotted
title("The demodulated signal")
xlabel("Time (s)")
ylabel("Value")
subplot(212)
plot(t,m)
title("The sent signal")
xlabel("Time (s)")
ylabel("Value")

mean_sq_err = sum((m-data_demodulated(Fs
    /10+1:1.1*Fs)').^2)/Fs;
```

5

In order to determine the obtained sampling frequency of the arduino, a sinosoidal wave with a known frequency, in our case it is $500Hz$, is generated and is given to the arduino analog input. The received sine wave is read using the MATLAB code given below, and shown in the Fig. 18.

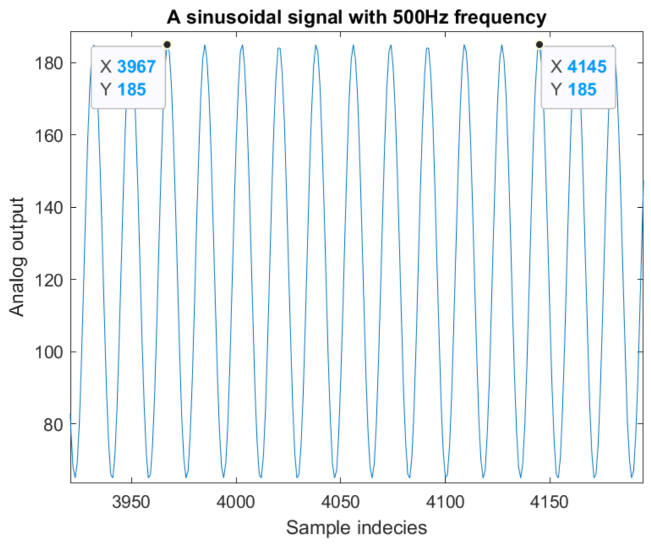

Fig. 18. The received sin wave with 500Hz frequency

So, the period of the generated sin wave is $T = \frac{1}{500}s$, and there exist 178 data points between ten waves. Therefore, the sampling frequency of ardoino can be calculated as approximately $178/(10T) = 8900Hz$

6

The received signal is shown in the Fig. 19 and the zoomed version is shown in the Fig. 20.

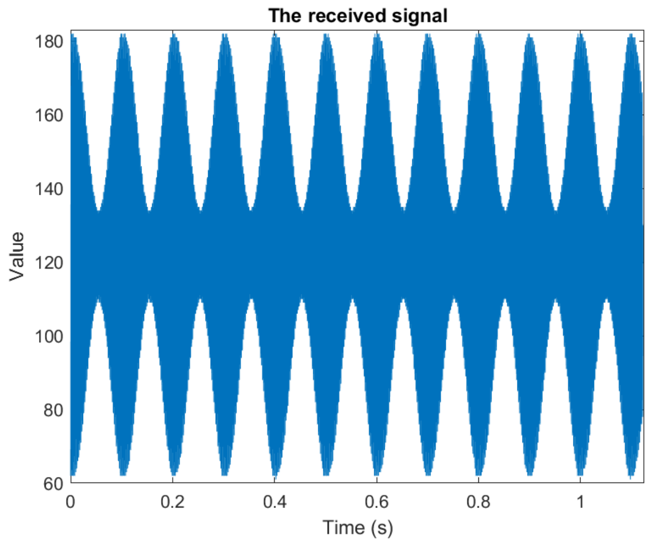

Fig. 19. The received signal

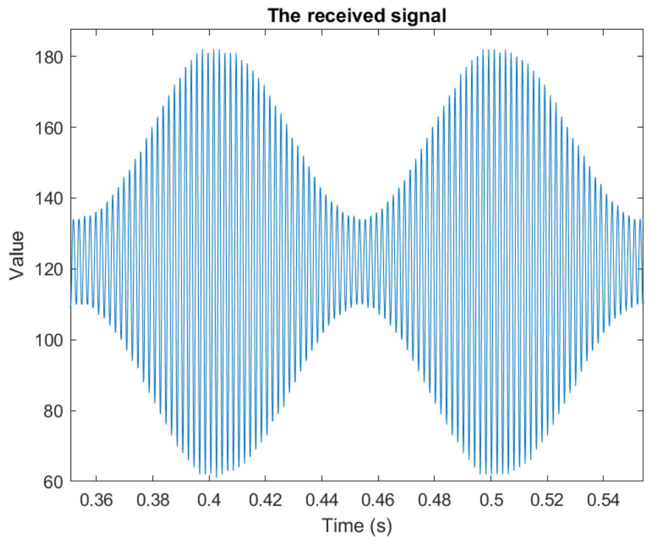

Fig. 20. The zoomed version of the received signal

It can be seen from the figures that the AM modulated signal in the part 1 is received and read correctly.

7

The proposed method of envelope detector in the Simulink part of the project is also implemented in this part. Therefore, first, the received signal is squared, passed through a low pass filter with $15Hz$ cut-off frequency and then DC part of the resulting signal is removed. Thus, in the end the received signal is demodulated. As in the Simulink part, the cut-off frequency is selected as $15Hz$, since the sent signal has $10Hz$ bandwidth so in order to pass only the original signal the cut-off frequency should be higher than $10Hz$ and also not so large.

The function *lowpass*, with the parameters of $15Hz$ cut-off frequency, 8910 sampling frequency and 0.95 steepness which specifies the transition band steepness, is used for the filtering.

The DC component of the low passed signal is subtracted for the DC blocking.

The MATLAB script is given in the part 4.

8

The demodulated signal within one second time interval and the sent signal are plotted in the Fig. 21, respectively.

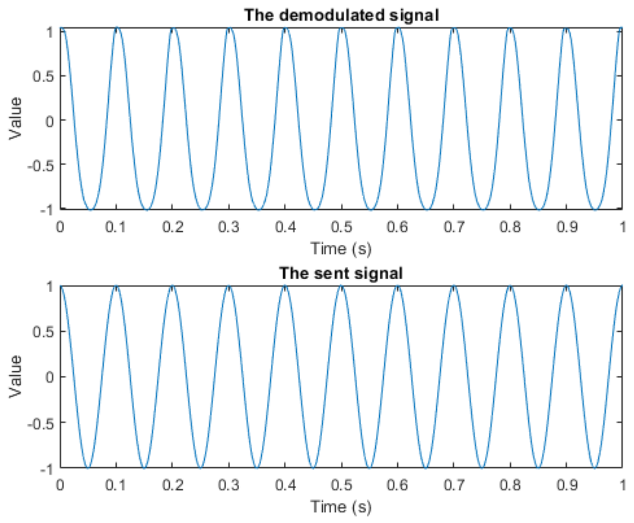

Fig. 21. The demodulated signal and sent signal, respectively

It can be observed from the demodulated signal that the proposed method in the previous part works well. The overall shape of the demodulated signal is very similar to the sent signal. However, if the peaks of the demodulated signal are zoomed, one can see that the peaks are not as smooth as the original signal. Also, the peaks are not located at exactly the multiples of 0.1 seconds since the sampling frequency of the arduino is determined approximately.

The Mean Square Error between the original signal and the demodulated signal is calculated as 0.048.

9

The above steps are repeated for the noisy modulated signals which has a 10^{-2} and 10^{-6} power AWGN noise.

The modulated and noisy signals are shown in the Fig. 22, respective to their noise power.

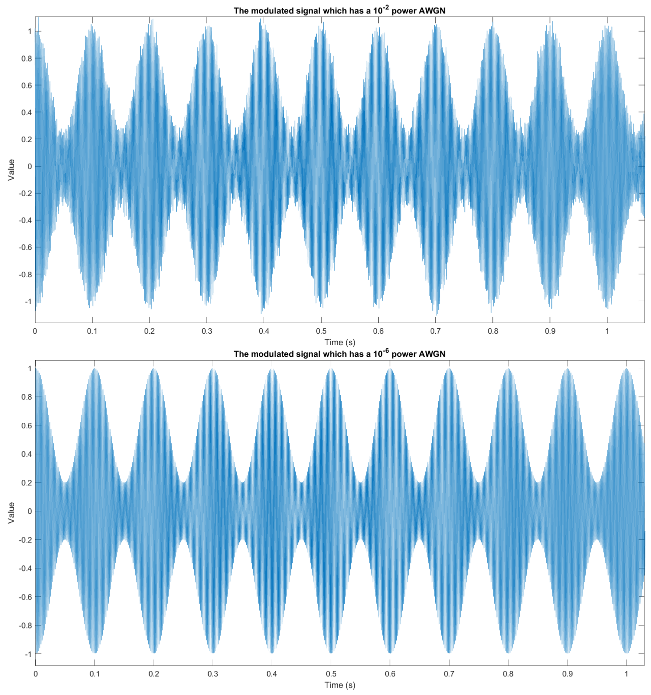

Fig. 22. The modulated and noisy signals with noise power of 10^{-2} and 10^{-6}, respectively

The received signals are shown in the Fig. 23, respective to their noise power.

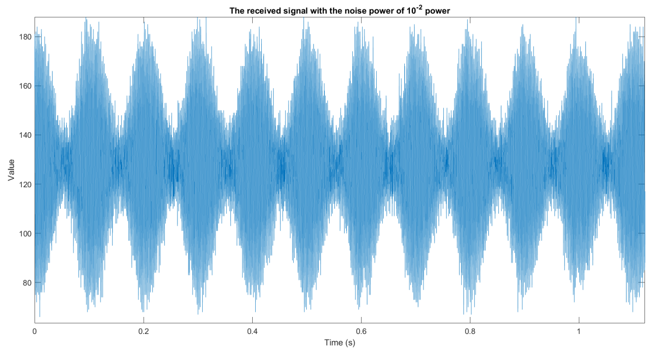
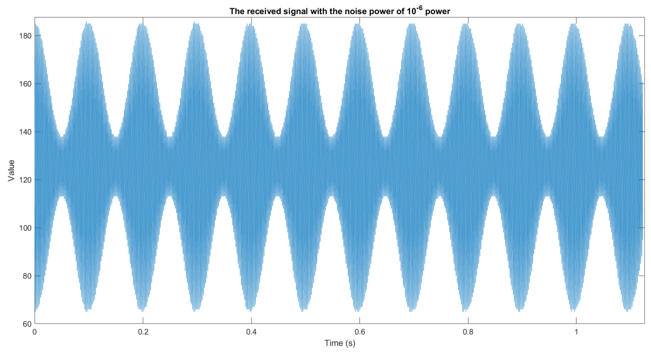

Fig. 23. The received signals with noise power of 10^{-2} and 10^{-6}, respectively

The demodulated signal within one second time interval are plotted in the Fig. 24, respective to their noise power.

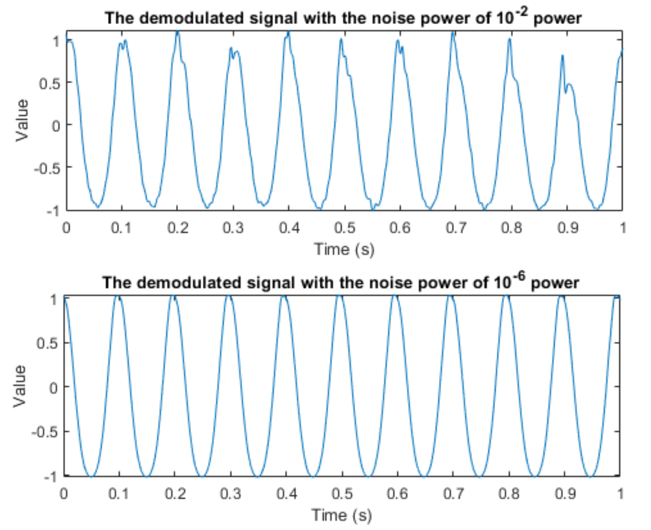

Fig. 24. The demodulated signals with noise power of 10^{-2} and 10^{-6}, respectively

The Mean Square Error between the original signal and the demodulated signals are calculated as 0.0459, 0.0583, respectively.

It can be seen from the demodulated signal plots that adding AWGN disrupts the shape of the signal and increases the mean square error between the sent signal. However, the noise with 10^{-6} power does not alter the received signal so much since its power is not so large to cause a big difference between the original signal yet the caused error can be observed in the mean square error metric.

Appendix

The MATLAB and Simulink codes that are used in this project is in the *EE479Project1(Sefa Kayraklık).zip* file. The content of file is following m, slx files and the reports:

- part1_1_m1.slx, part1_1_m2.slx, and part1_1_m3.slx for the part 1 of the simulink
- DSB_SC_AM_System.slx, and conv_AM_system.slx for the remaing parts of the simulink
- Part2_SoundGen.m, Part2_ReadData.m, and sketch_nov11a.ino for the arduino part
- EE479Project1_Part1.pdf and EE479Project1_Part2.pdf for the reports