Sefa Kayraklık
2015401066

Project 2

EE 479
Fall 2020

Part I: Simulink

1

The *sin wave* block is used to generate the given message signal, $m(t) = -sin(2\pi 50t)$, with sampling time of $10^{-5}s$ and simulation time of $2s$. And for the spectrum analyzer the sampling time is increased to $5 * 10^{-4}s$ via *zero-order hold* block. The message signal is verified through *scope* in time domain and *spectrum analyzer* in frequency domain.

2

The bandwidth of the message signal can be observed from the spectrum analyzer as $50Hz$. As discussed in the previous project, the determinant part for the bandwidth is the frequency of the signal. Therefore, the message signal has the frequency of $50Hz$, so its bandwidth is $50Hz$.

3

A subsystem named *FM* is designed to modulate the input with A_c of 1, the deviation constant, k_f, of 30, and the carrier frequency, f_c, of 200 Hz and gives the information-carrying phase (the integral result) and the modulated signal. And for the integral block, the initial condition is set to $3/5$ since the simulation time starts at 0 second. The block diagram of the subsystem is shown in the Fig. 1.

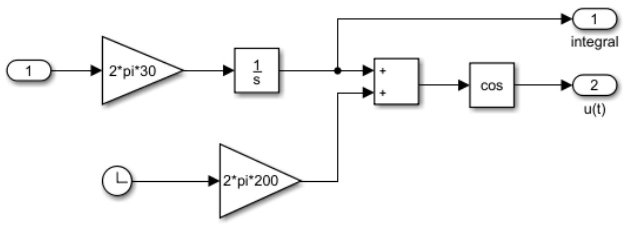

Fig. 1. The block diagram of *FM*

4

The expected pattern of the message signal is two diracs at the frequencies of $50Hz$ and $-50Hz$ for frequency domain and a sinusoidal wave having -1 amplitude and $50Hz$ frequency for time domain.
The expected pattern of the integral of the message signal is two diracs at the same frequencies but with lower power for frequency domain since its amplitude decreases to $3/5$ and a cosine wave having $3/5$ amplitude and $50Hz$ frequency for time domain.
The expected pattern of the modulated signal is two diracs at the frequencies of $200Hz$ and $-200Hz$ as carrier signal, four diracs at the frequencies of $-150Hz$, $-250Hz$ and $150Hz$, $250Hz$ as main lopes and at four diracs with much lower power at the frequencies of $-100Hz$, $-300Hz$ and $100Hz$, $300Hz$ as side lopes for frequency domain and a sinusoidal wave with increase and decrease of its frequency as the message signal values increase or decrease, respectively for time domain.

The time domain representations of the message signal, its integral, and the modulated signal are shown in the Fig. 2, the frequency representations of their powers are shown in the Fig. 3.

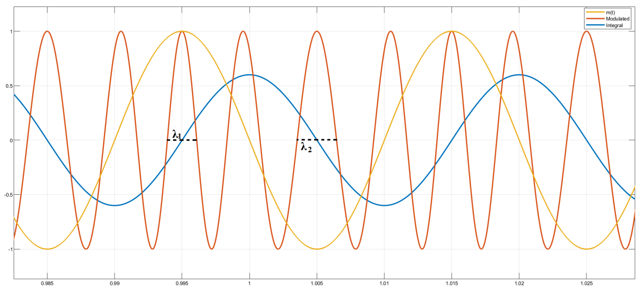

Fig. 2. The time domain representations of the signals

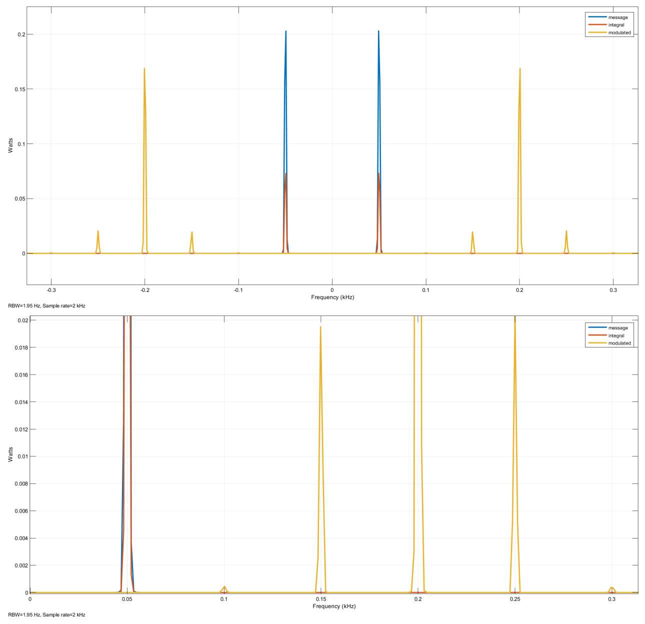

Fig. 3. The frequency domain representations of the signals and the zoomed version into the positive frequency, respectively

It can be seen from the figures that the expected can be observed from the simulation results. The frequency alteration of the modulated signal can be observed from the Fig. 2 that the half wave length of λ_1 is smaller than the λ_2 which corresponds to the lower value of the message signal;

therefore, the frequency of the modulated signal increases as the value of the message signal is raised or vice versa.

5

The modulated sinusoidal signal can be expressed by a weighted sum of its higher order harmonics as following:

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

, where $J_n(\beta_f)$ is the Bessel function of the first kind and of order n. Hence, the modulated signal is sum of cosines signals at the frequencies of $f_c + nf_m$ having different amplitudes. As the order of n increases, the amplitude of the corresponding cosine is decreased. Therefore, the power spectrum of the modulated signal can be seen as diracs at the frequencies of $f_c + nf_m$ where n starts from 0 and goes to ∞ and their magnitudes decrease as the order n increases.

6

The bandwidth of the fm modulated signal can be approximately found by Carson's rule as

$$B_T = 2(\beta_f + 1)W = 2\left(\frac{k_f a}{f_m} + 1\right) W$$

where W is the bandwidth of the message. So, the bandwidth in our case is

$$B_T = 2\left(\frac{30 * 1}{50} + 1\right) 50 = 160Hz$$

When the bandwidth is read from the power spectrum shown in the Fig. 3, if the side loops at the frequencies of $100Hz$ and $300Hz$ are included, the bandwidth is $200Hz$, yet the side loops powers are very low $(4 * 10^{-4})$ in comparison to other diracs, they can be ignored, then the bandwidth can be determined as $100Hz$. In either way, the read bandwidth is not the same as the calculated one by the Carson's rule since the formula is valid for the wideband frequency modulation where the deviation constant, k_f, is much bigger than the bandwidth of the message signal.

7

A phase locked loop (PLL) can be used to demodulate the frequency modulated signal via following the modulated input's frequency. When the PLL's input is $cos(w_c t + \phi(t))$, where $\phi(t) = k_f \int_{-\infty}^{t} m(\tau)d\tau$, then the loop filter output, output error signal, $e_0(t)$ can be expressed as following:

$$e_0(t) = \frac{1}{c}\dot{\phi}(t) = \frac{1}{c}\frac{d}{dt}\left(k_f \int_{-\infty}^{t} m(\tau)d\tau\right) = \frac{k_f}{c} m(t)$$

where $1/c$ is the gain of the loop filter. Hence, the PLL functions like an FM demodulator. (*The explanation is adopted from the Lathi's book of Modern Digital and Analog Communication Systems*)

8

The PLL block with the VCO quiescent frequency of $200Hz$ and VCO initial phase of $\pi/2$ is used to extract phase information from the *PD* port. To obtain the message signal the extracted phase signal is passed through a low pass filter to remove the high frequency components and added a gain of 3.58 to cancel the attenuation of the loop filter of the PLL. So, the subsystem named *PLL_Demodulator* is designed to demodulate the FM modulated input signal and shown in the Fig. 4.

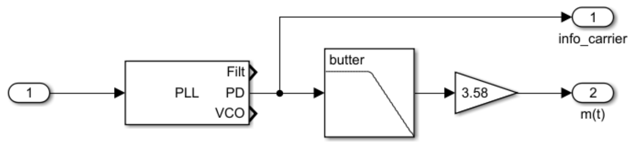

Fig. 4. The block diagram of *PLL_Demodulator*

The time domain representations of the demodulated signal and the extracted phase signal are shown in the Fig. 5, the frequency representations of their powers are shown in the Fig. 6.

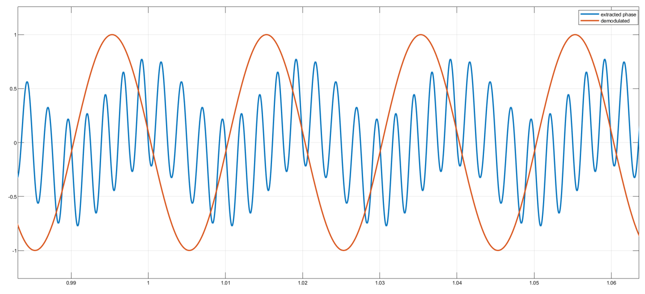

Fig. 5. The time domain representations of the signals

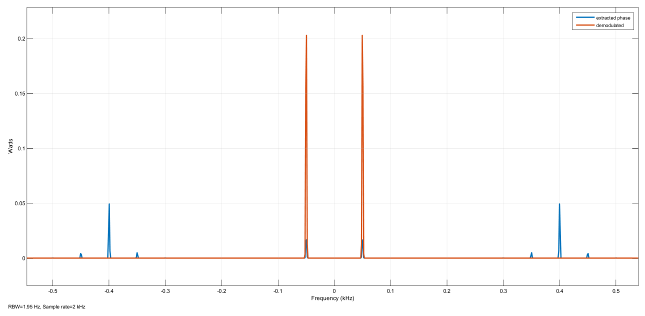

Fig. 6. The frequency domain representations of the signals

So, it can be observed from the above figures that the baseband of the extracted phase signal carries the information about the message signal. Thus, by low pass filtering the message signal is recovered from it.

9

The message signal, $m(t)$, can be recovered from the information carrying phase, $\phi(t) = 2\pi k_f \int_{-\infty}^{t} m(\tau)d\tau$, as following:

$$m(t) = \frac{1}{2\pi k_f}\dot{\phi}(t) = \frac{d}{dt}\left(\int_{-\infty}^{t} m(\tau)d\tau\right) = m(t)$$

So, in the PLL case, the derivative of the information carrying signal is obtained in the baseband of the *PD* output port.

10

The constructed subsystems of *FM* and *PLL_Demodulator* are combined to design a frequency modulation system.

The time domain representations of the message signal, the modulated signal and the demodulated signal are shown in the Fig. 7, the frequency representations of their powers are shown in the Fig. 8.

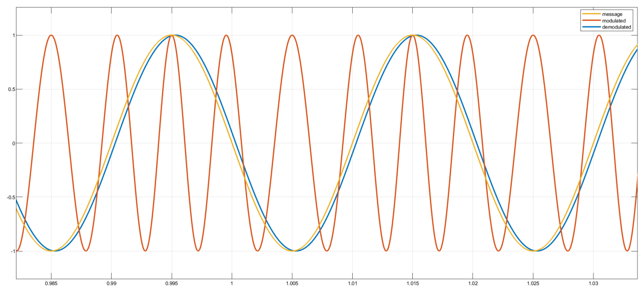

Fig. 7. The time domain representations of the signals

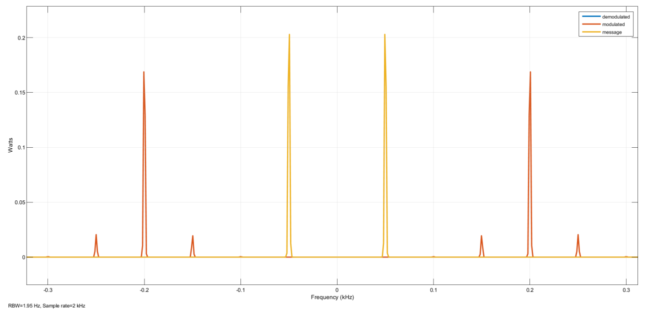

Fig. 8. The frequency domain representations of the signals

In the frequency domain, the demodulated and message signal have the same pattern, so their curves are top on each other.

So, it can be seen from the above figures that during the demodulation the message signal is extracted from the modulated signal via the PLL block. In time domain, the demodulated signal has lower values as the modulated signal's frequency gets smaller or vice verse. And in the frequency domain, the demodulated signal is recovered from the main loops at the frequencies of $150Hz$, $250Hz$ with adding a gain of 3.58.

PART II: ARDUINO

1

The message signal, $m(t) = cos(2\pi 20t)$, is FM modulated with the following formula

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

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.
kf = 20;
fc = 500;
m = cos(2*pi*20*t);
phi = 2*pi*kf*cumsum(m)/Fs; % information
    carrying phase
s = cos(2*pi*fc*t+phi); % Signal.
s = s /max(abs(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)
hold on;
plot(t,m)
title("The modulated signal in time domain")
xlabel("Time (s)")
ylabel("Value")
legend("modulated","message");
figure(2)
f = [-duration*Fs/2:duration*Fs/2-1]/duration;
plot(f,abs(fftshift(fft(s))))
title("The modulated signal in frequency domain")
xlabel("Frequency (Hz)")
ylabel("Value")
%% Generate a sinusoidal with a know frequency to
     determine the arduino sampling frequency

sin50hz = sin(2*pi*50*t);
audiowrite("sin.wav", sin50hz, Fs);
plot(t,sin50hz)
```

The determination of the value of deviation constant, k_f, is done by considering how wide the modulated signal is wanted to occupy the frequency spectrum. In order not to occupy too much bandwidth in the frequency domain, it is chosen a small number as 20.

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.

2

The time domain representation of the FM modulated signal is shown in the Fig. 9, the frequency representation of the signal is shown in the Fig. 10.

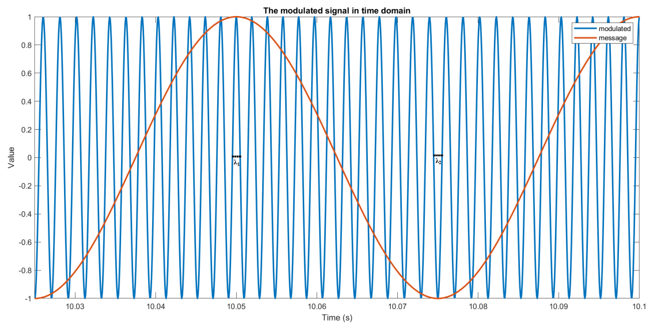

Fig. 9. The time domain representations of the signals

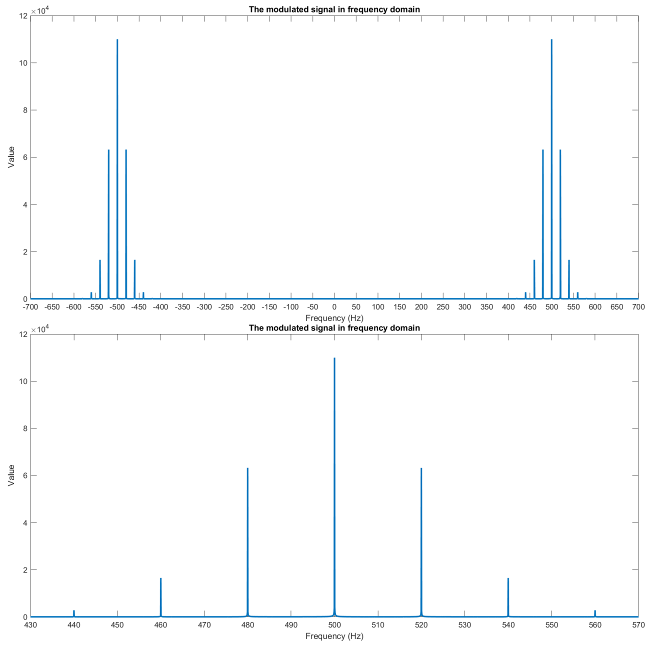

Fig. 10. The frequency domain representations of the signal and the zoomed version into the positive frequency, respectively

The frequency alteration of the modulated signal can be observed from the Fig. 9 that the half wave length of λ_1 is smaller than the λ_2 which corresponds to the lower value of the message signal; therefore, the frequency of the modulated signal increases as the value of the message signal is raised or vice versa.

The bandwidth of the fm modulated signal can be approximately found by Carson's rule as

$$B_T = 2(\beta_f + 1)W = 2\left(\frac{k_f a}{f_m} + 1\right)W$$

where W is the bandwidth of the message. So, the bandwidth in our case is

$$B_T = 2\left(\frac{20*1}{20}+1\right)20 = 80Hz$$

When the bandwidth is read from the frequency spectrum shown in the Fig. 10, it is $540Hz - 460Hz = 80Hz$ (Ignoring the side loops at the frequency of $440Hz$ and $560Hz$ since their powers are very low). Therefore, the result of the Carson's rule matches with the one read from the frequency spectrum.

3

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. 11 shows the constructed arduino setup.

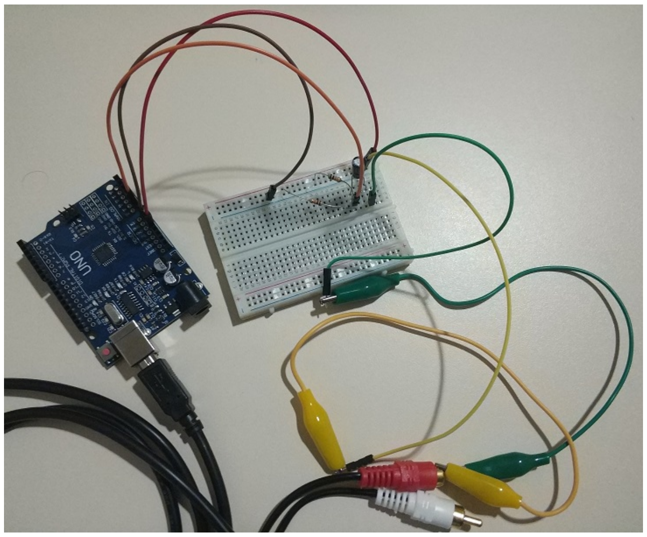

Fig. 11. The arduino setup

4

The following arduino code is used to read the FM 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
```

5

The following MATLAB code is used to read the FM signal from the serial interface, put the sampled received signal data into the data vector and to process the data in order to obtain the message signal.

```
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 = 15000; % 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=8872; % In the part 6, the sampling frequency
    of arduino is found 8872Hz

figure(2)
plot([1:15000]/Fs,data);
title("The received signal")
xlabel("Time (s)")
ylabel("Value")
%% Removing the added DC bias
data = data - 126; % The added DC value by the DC
    biasing circuit is 126

%% Taking the derivative to obtain the message
    signal as in the magnitude
der=diff(data);
figure(3)
plot([1:14999]/Fs,der)
title("The derivative of the received signal")
xlabel("Time (s)")
ylabel("Value")
%% Demodulation with the envelope detector method
    given in the Simulink part

Fpass = 30;

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

```

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

%% Synchronization of the demodulated signal
data_demodulated = data_lowpassed(1420:1420+Fs-1)
     - mean(data_lowpassed(1420:1420+Fs-1)); %
     removing Dc component
%plot(abs(fftshift(fft(data_demodulated))))

data_demodulated = data_demodulated/30.9; %
     scaling into [-1 1] range

%% PLoting the result
figure(4)
t = linspace(1/Fs,1,Fs);
m = cos(2*pi*20*t);
subplot(211)
plot(t,data_demodulated); % 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').^2)/Fs;
```

6

In order to determine the obtained sampling frequency of the arduino, a sinosoidal wave with a known frequency, in our case it is $50Hz$, 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. 12.

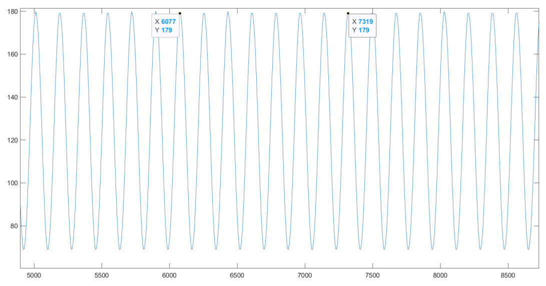

Fig. 12. The received sin wave with 50Hz frequency

So, the period of the generated sin wave is $T = \frac{1}{50}s$, and there exist 1242 data points between seven waves. Therefore, the sampling frequency of ardoino can be calculated as approximately $1244/(11T) = 8872Hz$.

7

The received signal and its the zoomed version are shown in the Fig. 13.

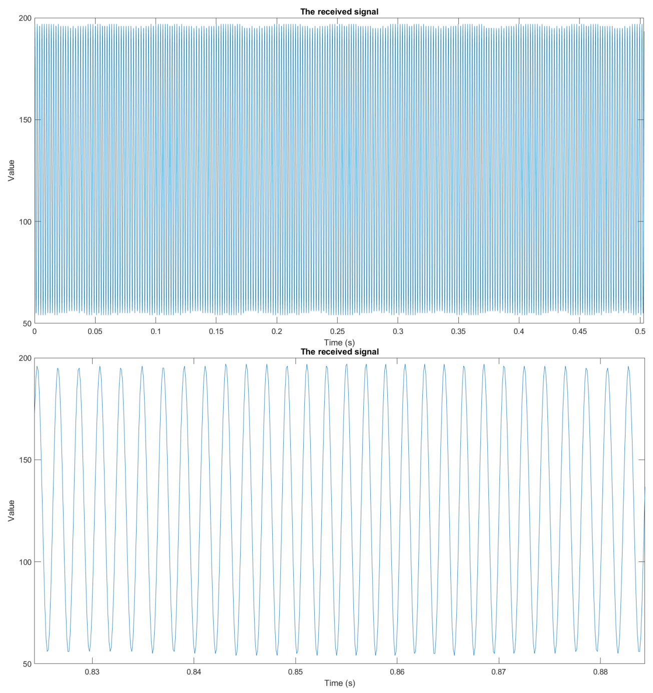

Fig. 13. The received signal, and its zoomed version, respectively

It can be seen from the figures that the FM modulated signal in the part 1 is received and if the zoomed version is inspected closely, the frequency alteration can be observed.

8

When the derivative of the received signal is taken, the received FM signal takes the shape of an AM signal. The obtained derivative signal is shown in the Fig. 14.

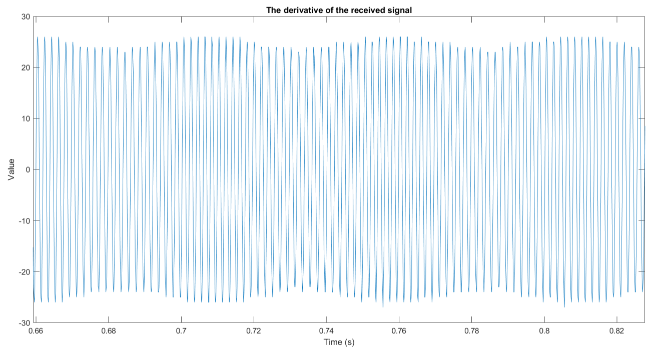

Fig. 14. The derivative of the received signal

9

Since by taking the derivative of FM signal and obtaining an AM signal, the implemented demodulation method of envelope detector in the previous project can be used to recover the sent message signal. Therefore, the same procedure is followed as in the previous project with low pass filter with $30Hz$ cut-off frequency.

The MATLAB script is given in the part 5.

10

The demodulated signal within one second time interval and the sent signal are plotted in the Fig. 15, respectively.

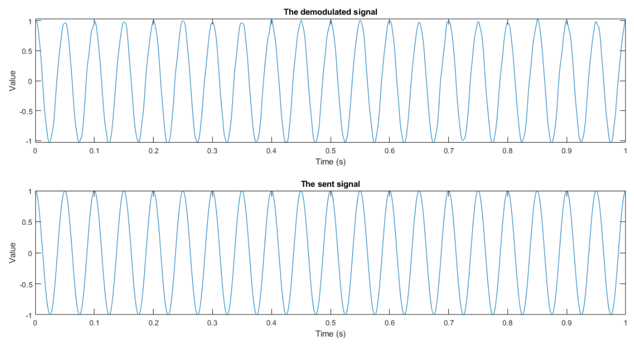

Fig. 15. The demodulated signal and sent signal, respectively

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 and stable as the original signal.

The Mean Square Error between the original signal and the demodulated signal is calculated as 0.0022.

Appendix

The MATLAB and Simulink codes that are used in this project is in the *EE479Project2(Sefa Kayraklık).zip* file. The content of file is following m, slx files and the reports:

- FM_System.slx for the parts of the simulink
- Part2_SoundGen.m, Part2_ReadData.m, and sketch_nov11a.ino for the arduino part
- EE479Project2_Part1.pdf and EE479Project2_Part2.pdf for the reports