American International University- Bangladesh

Department of Computer Engineering

COE 3201: Data Communication Laboratory

Title: Message Passing and Receiving Using Modulator (part 1: Transmitter Side)

Abstract:

This tutorial is designed to-

1.To understand the concept of message encoding and decoding.
2.To understand the concept of serial transmission and reception of message.
3.To develop understanding of data transmission and reception process.

Introduction:

Consider the problem of transmitting and receiving a text message, such as

Data Communication is fun!

over a waveform channel such as a twisted pair cable or a wireless RF (radio frequency) link. The design of a system that can accomplish this task requires the following ingredients:

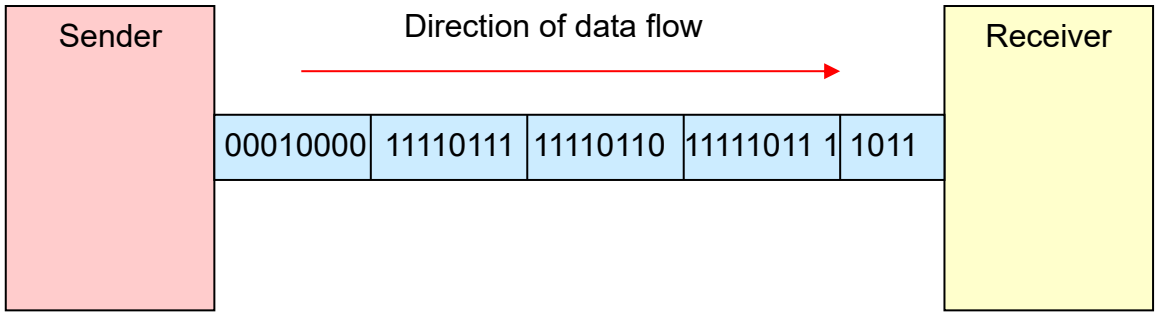
In transmitter side:

1. **Step 1:** Encoding of the letters of the alphabet, the numbers, punctuation, etc. For example, “A” could be encoded as 0, “B” as 1, “C” as 2, etc.
2. **Step 2:** Conversion of the encoded message into a serial data stream, e.g., of 0’s and 1’s in the case of a binary transmission system.
3. **Step 3:** Modulation by the serial data stream of a CT waveform that can be transmitted through the waveform channel.

In receiver side:

4. **Step 4:** Demodulation of the received waveform at the output of the waveform channel to obtain the received serial data stream.
5. **Step 5:** Conversion of the received serial data stream to a sequence of character codes.
6. **Step 6:** Decoding of the received character codes to the received message.

Step 1: Decimal to Serial Binary Conversion: To transmit a message over a communication channel with one input and one output, the bits of each ASCII encoded character need to be sent serially, one after another. One convention that needs to be established is whether the LSB (least significant bit) or MSB (most significant bit) of each code is sent first.



Synchronous transmission

Another consideration is how many bits should be sent per character. Even though the ASCII code has seven bits, it is customary to send 8 bits per character and set the MSB to zero. Thus, if the convention of sending the LSB first is used, the word **Red** (decimal 82,101,100) is converted to the binary data sequence (commas are only shown for clarity, they are not part of the data sequence)

0 1 0 0 1 0 1 0, 1 0 1 0 0 1 1 0, 0 0 1 0 0 1 1 0

In Matlab the following function (**asc2bin)** can be built to convert from a text string txt to a binary data sequence **dn**:

```
function dn = asc2bn(txt)

dec=double(txt)   %Text to ASCII (decimal)
p2=2.^(0:-1:-7)   % 2^0,2^-1,.......,2^-7
B=mod(floor(p2'*dec),2)  %Decimal to binary conversion
                          %Columns of B are bits of chars
dn=reshape(B,1,numel(B));%Bytes to serial conbversion
end
```

```
clc;
clear all;
close all;
Transmitted_Message= 'Red'
%Converting Information Message to bit%
x=asc2bn(Transmitted_Message); % Binary Information
```

```
bp=.000001;
% bit period
disp(' Binary information at Trans mitter :');
disp(x);
```

Transmitted_Message =

Red

Binary information at Trans mitter :

Columns 1 through 16

0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 0

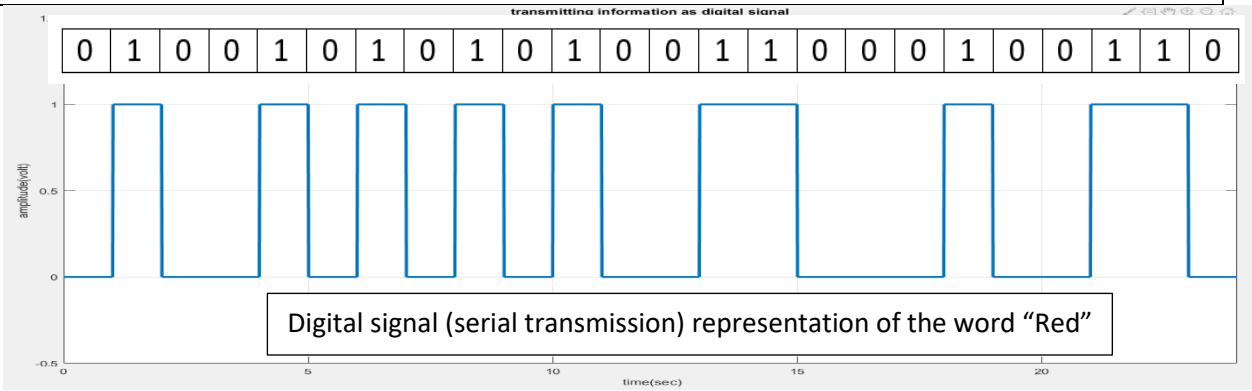
Columns 17 through 24

0 0 1 0 0 1 1 0

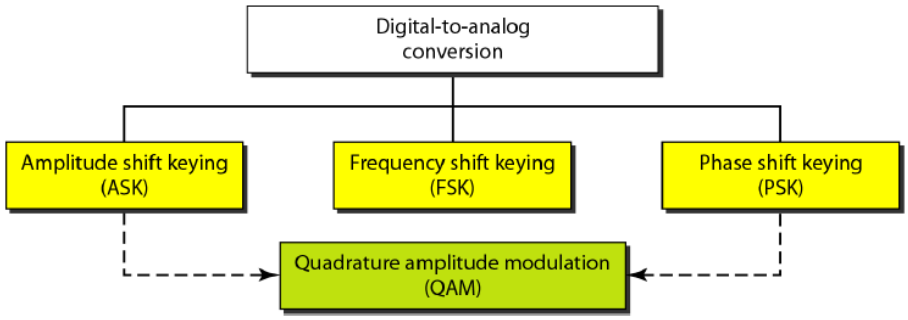
Step 2: Representation of transmitting binary information as digital signal

```
%XX representation of transmitting binary information as
digital signal XXX
bit=[];
for n=1:1:length(x)
    if x(n)==1;
       se=5*ones(1,100);
    else x(n)==0;
        se=zeros(1,100);
    end
     bit=[bit se];
end
t1=bp/100:bp/100:100*length(x)*(bp/100);
subplot(4,1,1);
plot(t1,bit,'lineWidth',2.5);grid on;
axis([ 0 bp*length(x) -.5 6]);
```

```
ylabel('amplitude(volt)');
xlabel(' time(sec)');
title('Transmitting information as digital signal');
```

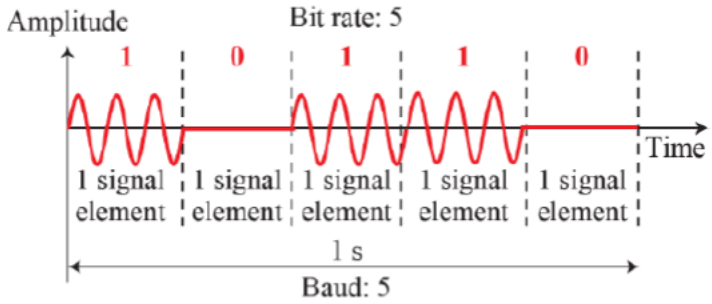


Step 3: Modulation of serial data stream to waveform: In Matlab the **asc2bin** (See A Message Passing and Receiving Using PAM (Part 1)) function can convert a text string txt to a binary data sequence. The binary data sequence is then converted into digital signals. Now, if we want to transmit the signal, we need to modulate the signal int analog.



We will perform ASK modulation. ASK: In amplitude shift keying, the amplitude of the carrier signal is varied to create signal elements Both frequency and phase remain constant while the amplitude changes.

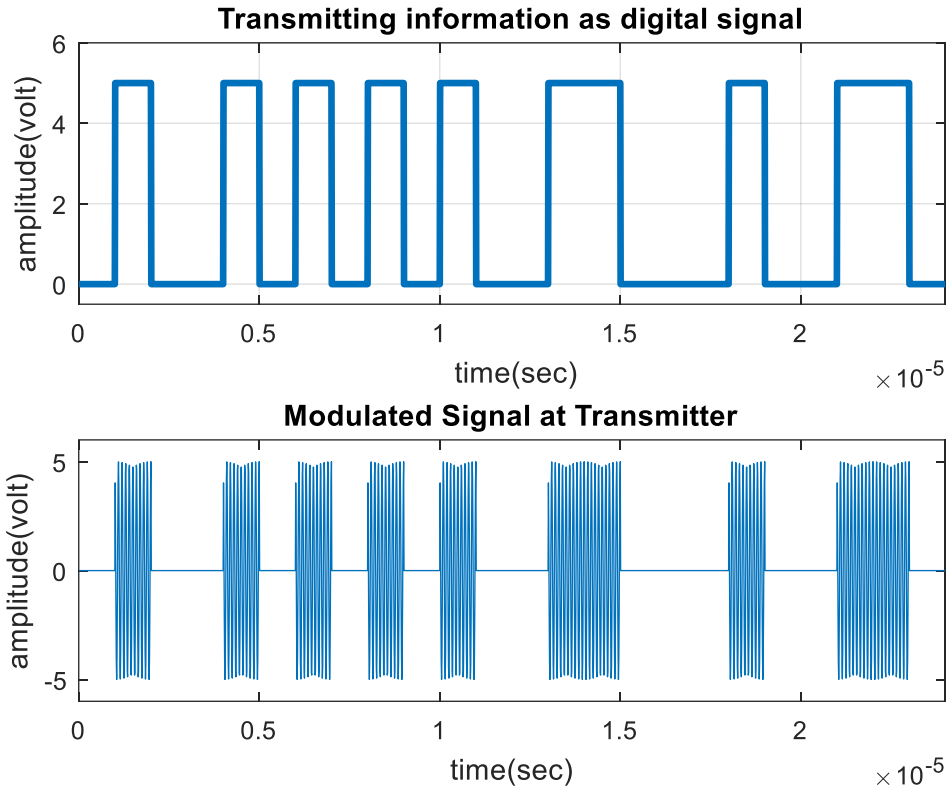
Binary amplitude shift keying



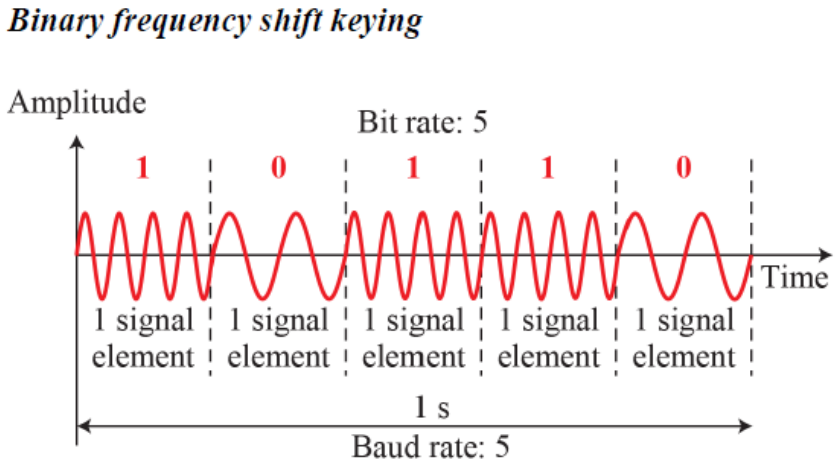
Sample code for ASK modulation in transmitting side:

```
%XXXXXXXXXXXXXXXXXXXXXXXX Binary-ASK modulation
XXXXXXXXXXXXXXXXXXXXXXXXXXXX%
A1=5;                          % Amplitude of carrier signal for
information 1
A2=0;                           % Amplitude of carrier signal for
information 0
br=1/bp;
% bit rate
f=br*10;                                                        %
carrier frequency
t2=bp/99:bp/99:bp;
ss=length(t2);
m=[];
for (i=1:1:length(x))
    if (x(i)==1)
        y=A1*cos(2*pi*f*t2);
    else
        y=A2*cos(2*pi*f*t2);
    end
    m=[m y];
end
t3=bp/99:bp/99:bp*length(x);
subplot(4,1,2);
plot(t3,m);
axis([ 0 bp*length(x) -6 6]);
xlabel('time(sec)');
ylabel('amplitude(volt)');
title('Modulated Signal at Transmitter');
```

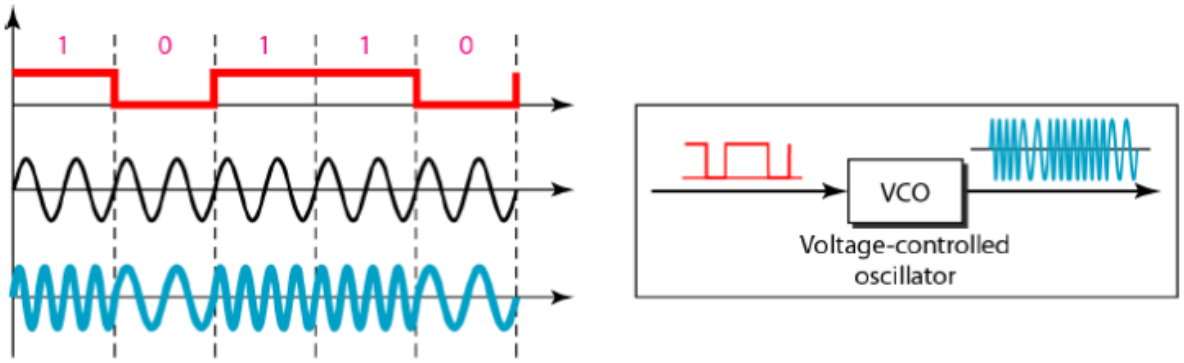
Output:



FSK: In frequency shift keying, the frequency of the carrier signal is varied to represent data The frequency of the modulated signal is constant for the duration of one signal element, but changes for the next signal element if the data element changes Both peak amplitude and phase remain constant for all signal elements



Implementation of BFSK:



We can perform FSK modulation for the input text “Digital” as well.

In receiver side:

1. **Step 4:** Demodulation of the received waveform at the output of the waveform channel to obtain the received serial data stream.
2. **Step 5:** Conversion of the received serial data stream to a sequence of character codes.
3. **Step 6:** Decoding of the received character codes to the received message.

Received signal is always different from transmitted signal:

Disturbance is a natural phenomenon. Whenever we are transmitting any kind of signal, naturally it will face some disturbance along the path it travels towards the recipient. We term these disturbances as signal impairments. One of the common form of signal impairments is noise. Receiver never receives a signal in the exact same form as it would be sent from the transmitter due to added noise to the signal.

In this experiment we also assume that the transmitted signal will be corrupted by noise before receiver receives the signal. There are different noise models present in literature for analyzing a transceiver model. Additive white Gaussian noise (AWGN) model is the most common of them all. So, we will consider our signal is transmitting through an AWGN channel before it reaches the receiver.

Additive White Gaussian Noise: A basic and generally accepted model for thermal noise in communication channels, is the set of assumptions that

- the noise is additive, i.e., the received signal equals the transmit signal plus some noise, where the noise is statistically independent of the signal.

- the noise is white, i.e., the power spectral density is flat, so the autocorrelation of the noise in time domain is zero for any non-zero time offset.
- the noise samples have a Gaussian distribution.

Mostly it is also assumed that the channel is Linear and Time Invariant. The most basic results further assume that it is also frequency non-selective.

```
disp('**********')
disp(' Message transmitted through a Transmission medium');
disp('**********')

%Channel Noise%
t4=bp/99:bp/99:bp*length(x);
Rec=awgn(m,10);
subplot(4,1,3);
plot(t4,Rec);
axis([ 0 bp*length(x) -6 6]);
xlabel('time(sec)');
ylabel('amplitude(volt)');
title('Received signal at Receiver');
```

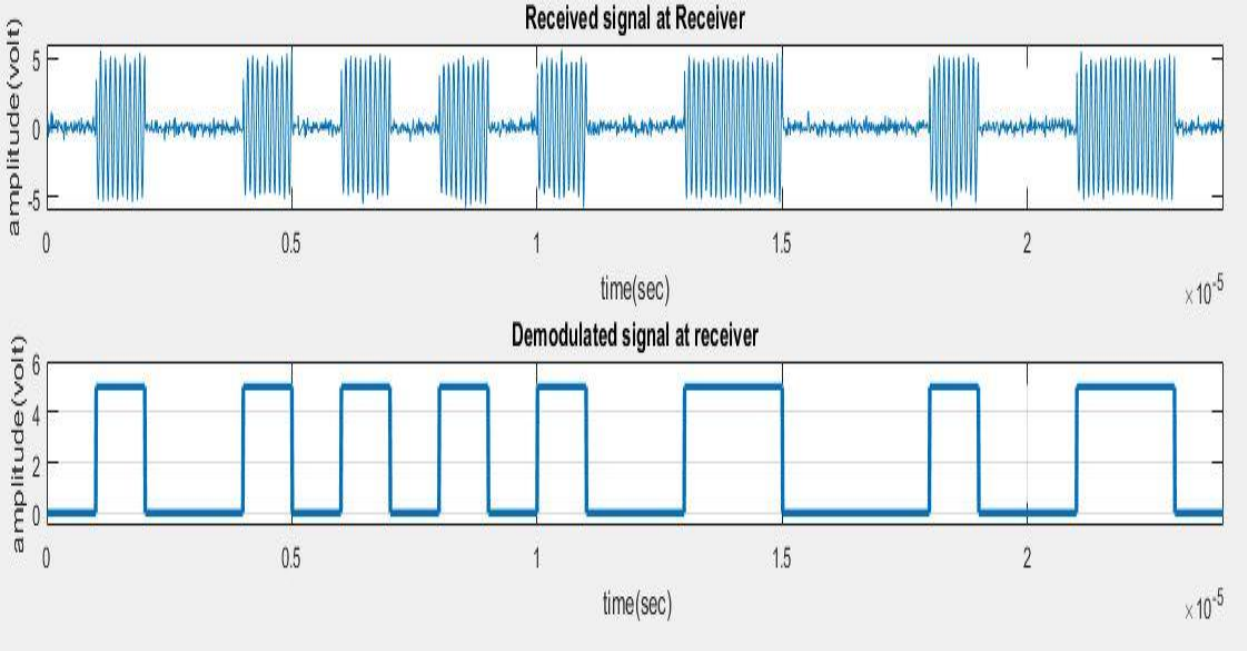
Step 4: Demodulation of the received waveform at the output of the waveform channel to obtain the received serial data stream: Before diving into how to demodulate the received waveform, we need to consider noise. So, first part of this example code shows how we can introduce noise in Matlab and then add it to our transmitted signal so that our model resembles a real-life scenario more accurately.

Sample code for demodulation in receiving side (*** this is an extension of 'sample code for modulation in transmitting side' from previous experiment. Please run that code first to get correct outcome)**:**

```
%XXXXXXXXXXXXXXXXXXXX Binary ASK demodulation
XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
mn=[];
for n=ss:ss:length(Rec)
  t=bp/99:bp/99:bp;
  y=cos(2*pi*f*t);                                          %
carrier siignal
  mm=y.*Rec((n-(ss-1)):n);
  t5=bp/99:bp/99:bp;
```

```
  z=trapz(t5,mm) ;
% intregation
  zz=round((2*z/bp));
  if(zz>2.5)                                    % logic level =
(A1+A2)/2=7.5
    a=1;
  else
    a=0;
  end
  mn=[mn a];
end
disp(' Binary information at Reciver :');
disp(mn);
%XXXXX Representation of binary information as digital signal
which achived
%after ASK demodulation
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
bit=[];
for n=1:length(mn);
    if mn(n)==1;
       se=5*ones(1,100);
    else mn(n)==0;
        se=zeros(1,100);
    end
     bit=[bit se];
end
t5=bp/100:bp/100:100*length(mn)*(bp/100);
subplot(4,1,4)
plot(t5,bit,'LineWidth',2.5);grid on;
axis([ 0 bp*length(mn) -.5 6]);
ylabel('amplitude(volt)');
xlabel(' time(sec)');
title('Demodulated signal at receiver');
```

Output:




```
Binary information at Reciver :
 Columns 1 through 9

    0     1     0     0     1     0     1     0     1

 Columns 10 through 18

    0     1     0     0     1     1     0     0     0

 Columns 19 through 24

    1     0     0     1     1     0
```

Step 5, 6: Conversion from Serial Binary to Text: Assuming that the sequence **A** can be successfully transmitted and received at the channel output, the next step is to convert it back to a textstring. This requires that the binary data string is decomposed into 8-bit segments (chopping off any extraneous bits at the end) which are then converted back to decimal ASCII codes that can then be displayed as text using the char function, which works as follows:

```
>> char([82 101 100])
ans =
Red
```

A complete function, called **bin2asc** that converts a binary data string **A** back to a text string txt is shown below.

```
function txt = bin2asc(dn)
%bin2asc    Serial binary to ASCII to text conversion
%           8 bits per char , LSB first
%           >> txt= bin2asc(dn) <<
%           where dn is binary input sequence
%                 txt is output text string
L=length(dn); %Length of input string
L8=8*floor(L/8); %Multiple of 8 Length
B=reshape(dn(1:L8),8,L8/8); %Cols of B are bits of chars
p2=2.^(0:7); %power of 2
dec=p2*B;  %Binary to decimal conversion
txt=char(dec); %ASCII (decimal) to txt
end
```

Thus, using bin2asc with asc2bin(txt) as input should return the text in txt as demonstrated below:

```
%Converting Information bit to Message%
Received_Message=bin2asc(mn)
%>>>>>>>> end of program >>>>>>>>>>>>>>>>%
```

Received_Message =

Red

Performance Task for Lab Report

(a) Transmit a text message 'Data Comm' as analog signal using QASK. Show the binary data representing the text, show the digital signal, and show the analog signal.

(b) Show the received analog signal assuming the communication channel has an SNR of 30 dB.

(c) Recover the text 'Data Comm' from the received signal.