LAB#06

***NAME* : ABDULLAH ZUNORAIN**

***REG NO* : 19JZELE0338**

***SUBJECT* : DATA COMMUNICATION(LAB)**

***SUBMITTED TO* : DR. UZAIR GILLANI**

***SECTION* : A**

***DEPT* : ELECTRICAL COMM**

***CAMPUS* : JALOZAI**

Simulation of Line Encoding Schemes in Matlab (Part I)

OBJECTIVES OF THE LAB

In this lab, we will cover the following topics:

- How to write matlab code that encodes digital data into digital signals?
- Study sample matlab programs for schemes like NRZ-L and Bipolar-AMI

4.1 LINE ENCODING SCHEMES

A line encoding scheme simply converts binary data i.e. a sequence of bits in to a digital signal. A variety of such techniques are available among which the suitable one is selected by comparing factors like signal spectrum, clocking mechanism, error detection, signal interference & noise immunity, and cost & complexity of one scheme with the other one.

This lab focuses on the implementation of following line encoding schemes:

- Uniplor
- NRZ-L
- Bipolar-AMI

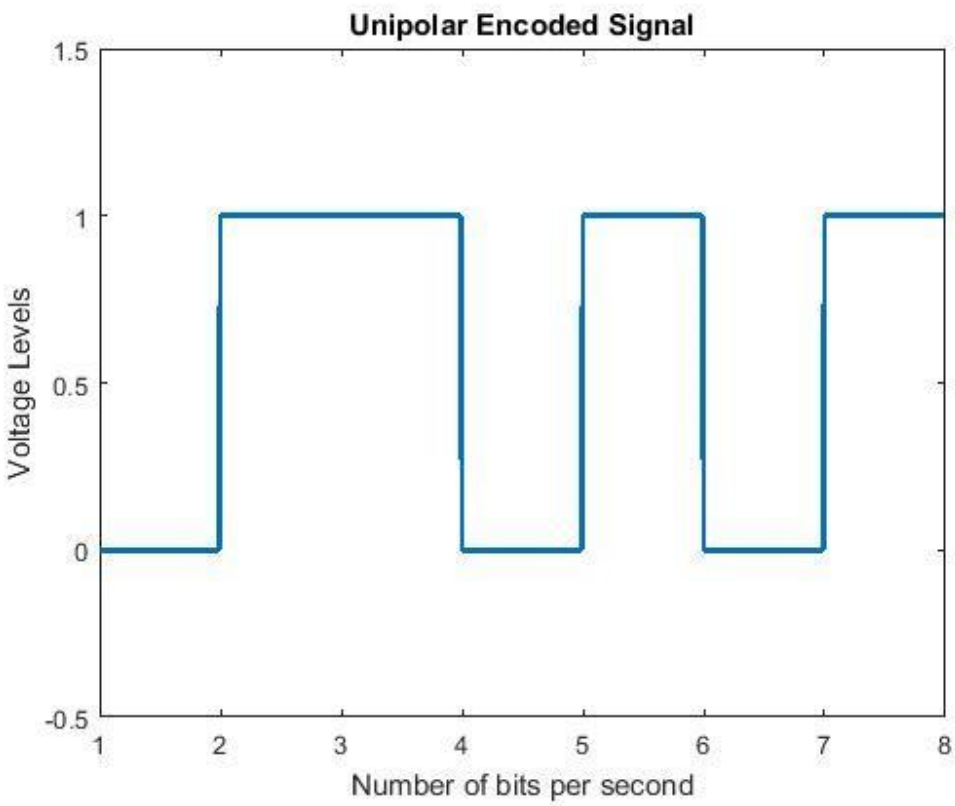
4.1.1 Simple algorithm for encoding digital signals

Here a simple algorithm is presented that produces a unipolar signal in matlab. Unipolar encoding is the one in which both binary-one and binary-zero has same algebraic sign i.e. all positive or all negative.

Matlab Code for Unipolar Signal:

```
clc clear
all

% original message message
= [0 1 1 0 1 0 1];

% message with redundant information at even locations data
= [0 0 1 0 1 0 0 0 1 0 0 0 1 0];

% index 'i' keeps track of message in data, while index 'j' keeps
% track of redundant information
i = 1:7; j = 1.99:7.99;

% generate timing information for plotting digital signal
tim = []; for(k = 1:7)     tim = [tim i(k) j(k)]; end

% logic for generating digital signal
signal = []; for(t=1:2:13)
   if(data(t)==0)
      signal(t:t+1) = 0;  else
      signal(t:t+1) = 1;
   end

 end
```

```
% plot signal w.r.t timing information

figure(1);plot(tim,signal,'lineWidth',2
)
 title('Unipolar Encoded Signal')
xlabel('Number of bits per second')
ylabel('Voltage Levels') axis([1 8 -0.5
1.5])
```



---TASK#01-----------------------------------

Write matlab code that generates unipolar signal with following polarities:

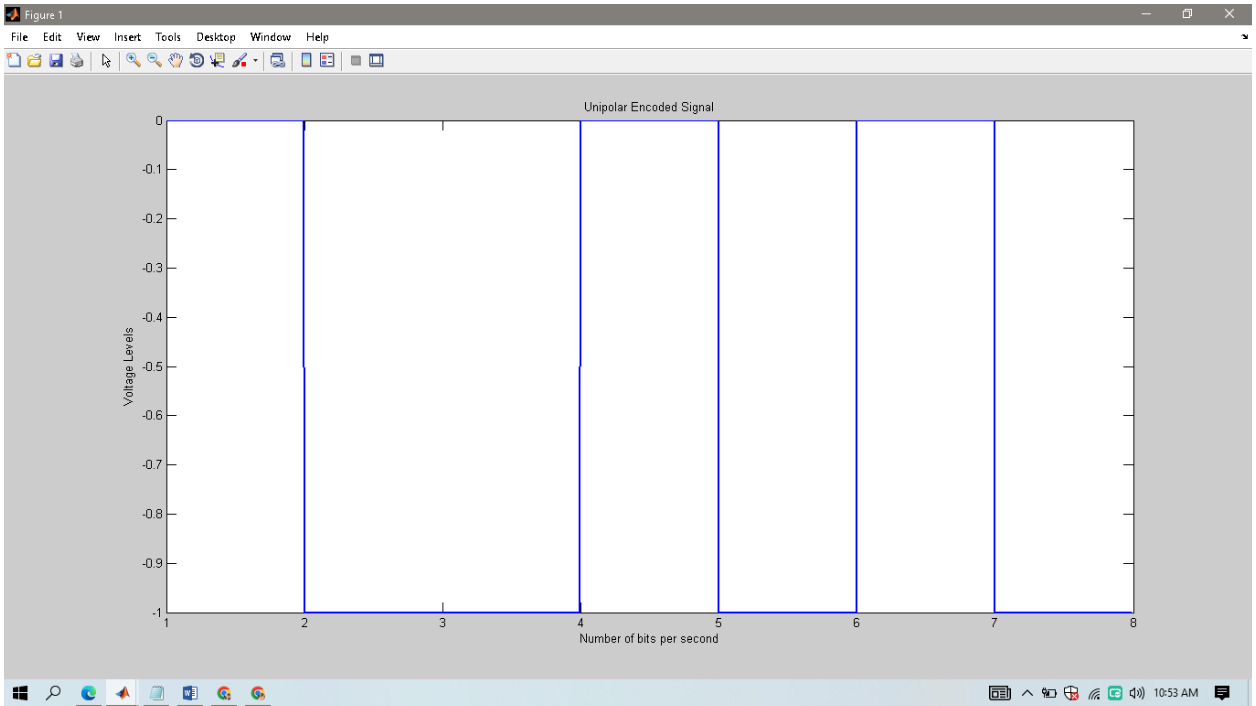
- **0V for binary-0**
- **-1V for binary-1**

ANS:

CODING:

```
% original message
message = [0 1 1 0 1 0 1];

% message with redundant information at even locations
data = [0 0 1 0 1 0 0 0 1 0 0 0 1 0];

% index 'i' keeps track of message in data, while index 'j' keeps
% track of redundant information
i = 1:7;
j = 1.99:7.99;

% generate timing information for plotting digital signal
tim = [];
for(k = 1:7)
tim = [tim i(k) j(k)];
end

% logic for generating digital signal
signal = [];
for(t=1:2:13)
if(data(t)==0)
signal(t:t+1) = 0;
else
signal(t:t+1) = -1;
end
end

% plot signal w.r.t timing information figure(1);
plot(tim,signal, 'lineWidth', 2)
title('Unipolar Encoded Signal')
xlabel('Number of bits per second')
ylabel('Voltage Levels')
```

RESULT OF THE CODING:



4.2 IMPLEMENTING NRZ-L AND PSEUDOTERNARY IN MATLAB

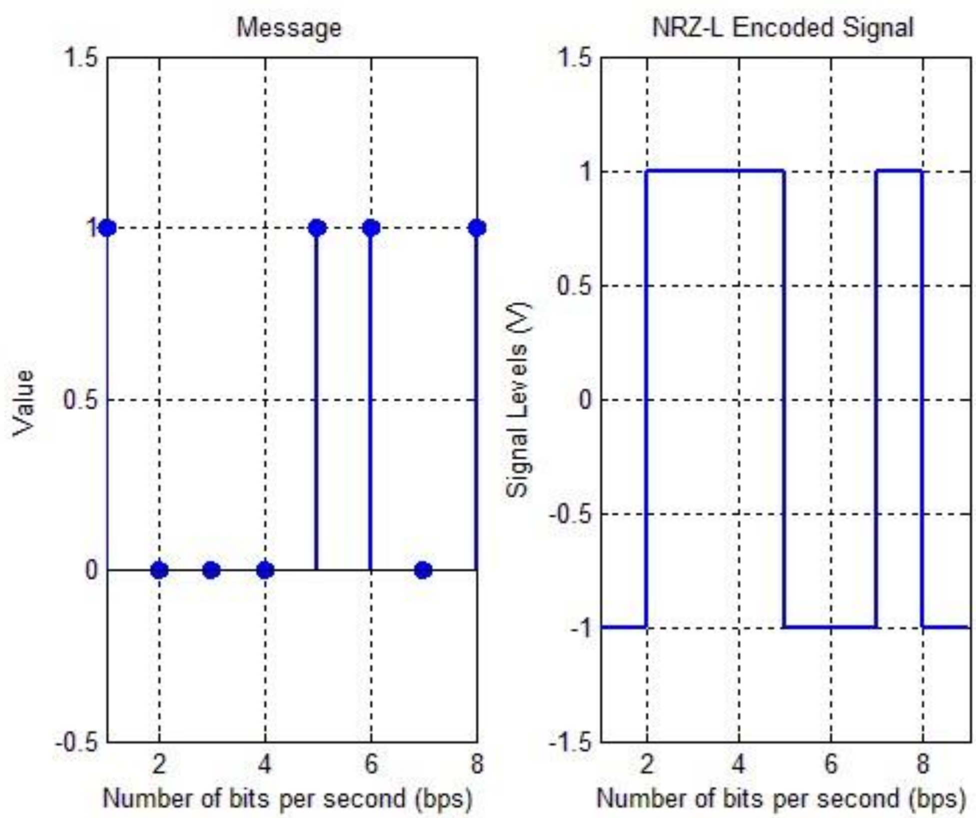
4.2.1 NRZ-L (Non-Return to Zero-Level)

NRZ-L (non-return to zero level) is the most common and easiest technique used to generate or interpret digital data by terminals and other devices. It uses a constant positive voltage to represent binary-zero and a constant negative voltage to represent binary-one.

Matlab Code for NRZ-L Signal:

```
clc clear all

% original message

message = [1 0 0 0 1 1 0 1];

% message with redundant information

 data = ones(1, 2*length(message)-1);
data(1:2:end) = message;

% index representing original message in 'data' vector i
= 1:length(message);

% index representing extra information in 'data' vector
j = 1.99:length(message)+0.99;

% generating 'time' vector by concatenating indices i & j to
% represent 'data' vector tim
= [];
for(k = 1:length(message))
   tim = [tim i(k) j(k)]; end

% generating digital signal
signal = []; N =
length(data);
for(t = 1:2:N)
   if(data(t)==0)
      signal(t:t+1) = 1;
   else  signal(t:t+1) =
      -1;
   end end

% displaying digital signal figure(1);
subplot(121);
```

```
stem(message, 'filled', 'linewidth', 2);
title('Message'); ylabel('Value');
xlabel('Number of bits per second (bps)');
axis([1 length(message) -0.5 1.5]); grid
on;

subplot(122);
plot(tim,signal,'linewidth',2); title('NRZ-L
Encoded Signal'); ylabel('Signal Levels
(V)');
xlabel('Number of bits per second (bps)');
axis([1 length(message)+1 -1.5 1.5]);
grid on;
```



---------------------------------------TASK#02------------------------------------

Write matlab code that converts the following message bits into NRZ-L (non-return to zero level) signal:

Message = [0 1 0 0 1 1 0 0 0 1 1]

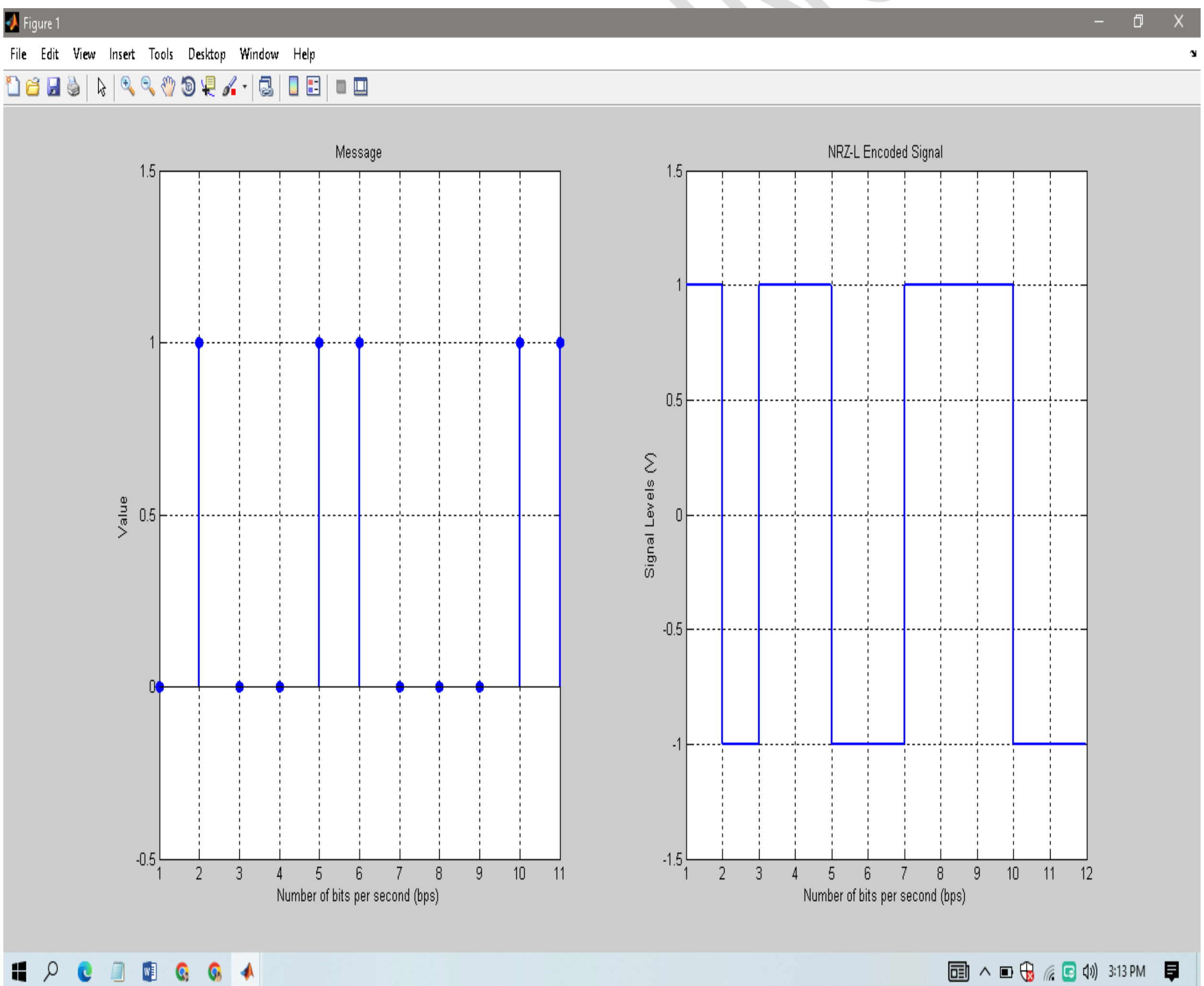
ANS:

CODING OF THE ABOVE TASK:

```
% original message
message = [0 1 0 0 1 1 0 0 0 1 1];
% message with redundant information
data = ones(1, 2*length(message)-1);
data(1:2:end) = message;
% index representing original message in 'data' vector
i = 1:length(message);
% index representing extra information in 'data' vector
j = 1.99:length(message)+0.99;
% generating 'time' vector by concatenating indices i & j to
% represent 'data' vector
tim = [];
for(k = 1:length(message))
  tim = [tim i(k) j(k)];
end
% generating digital signal
signal = []; N = length(data);
for(t = 1:2:N)
 if(data(t)==0)
signal(t:t+1) = 1;
 else  signal(t:t+1) = -1;
   end
end
% displaying digital signal figure(1);
subplot(121);
stem(message, 'filled', 'linewidth', 2);
title('Message');
ylabel('Value');
xlabel('Number of bits per second (bps)');
```

```
axis([1 length(message) -0.5 1.5]);
grid on;
subplot(122);
plot(tim,signal,'linewidth',2);
title('NRZ-L Encoded Signal');
ylabel('Signal Levels (V)');
xlabel('Number of bits per second (bps)');
axis([1 length(message)+1 -1.5 1.5]);
grid on;
```


THE RESULT OF THE CODING:



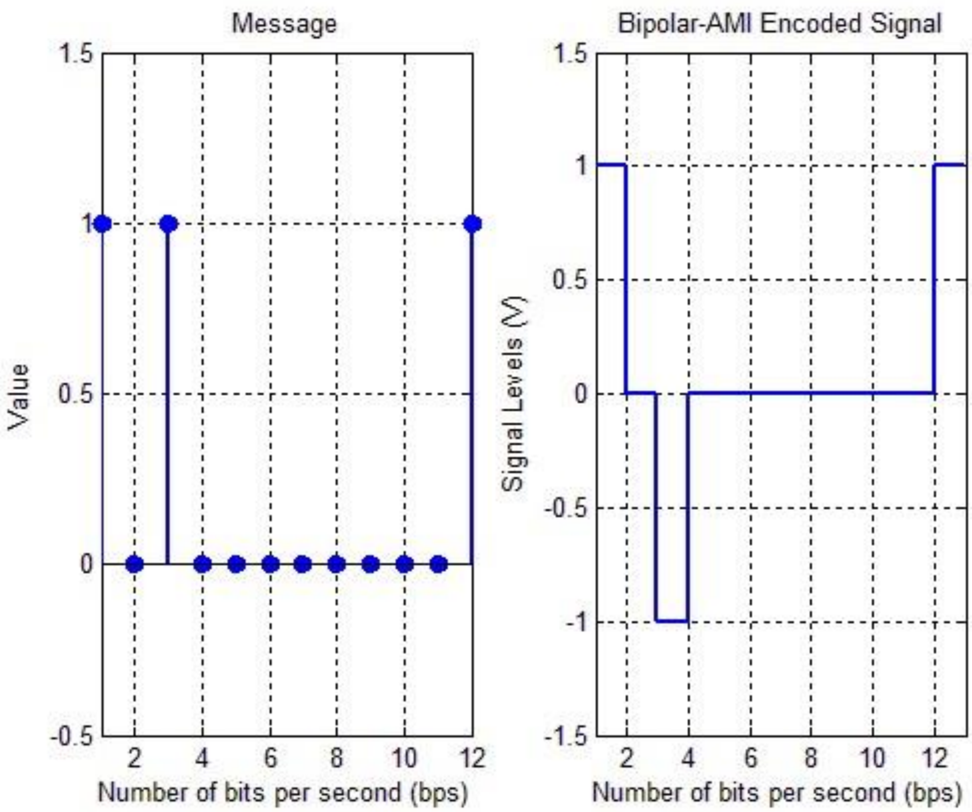
4.2.2 Bipolar-AMI (Alternate Mark Inversion)

Bipolar-AMI is a multilevel binary encoding scheme - *such schemes uses more than two signal levels to represent binary data*. Thus, in AMI a binary-0 is represented by no line signal, and binary-one is represented by a positive or negative pulse. The binary-one pulses alternates in polarity.

Matlab Code for Bipolar-AMI Signal:

```
clc clear all

% original message message = [1 0
1 0 0 0 0 0 0 0 0 1];

% message with redundant information
data = zeros(1, 2*length(message)-1);
data(1:2:end) = message;

% index representing original message in 'data' vector i
= 1:length(message);

% index representing extra information in 'data' vector
j = 1.99:length(message)+0.99;

% generating 'time' vector by concatenating indices i & j to
% represent 'data' vector tim
= [];
for(k = 1:length(message))
   tim = [tim i(k) j(k)]; end

% generating digital signal
signal = []; N =
length(data);
pre_bit = 1; for(t
= 1:2:N)
   if(data(t)==0)
      signal(t:t+1) = 0;
   elseif(data(t)==1 & pre_bit ==
      1)  signal(t:t+1) = 1;  pre_bit
      = -1;
   else  signal(t:t+1) =
      -1;  pre_bit = 1;
   end end
```

```
% displaying digital signal
figure(1); subplot(121);
stem(message, 'filled', 'linewidth', 2);
title('Message'); ylabel('Value');
xlabel('Number of bits per second (bps)');
axis([1 length(message) -0.5 1.5]); grid
on;

subplot(122);
plot(tim,signal,'linewidth',2);
title('Bipolar-AMI Encoded Signal');
ylabel('Signal Levels (V)');
xlabel('Number of bits per second (bps)');
axis([1 length(message)+1 -1.5 1.5]);
grid on;
```



--TASK 03------------------------------------

Write matlab code that converts the following message bits into Bipolar-AMI signal:

Message = [0 1 0 0 1 1 0 0 0 1 1]

ANS:

CODING OF THE GIVEN TASK:

```
 % original message
message = [0 1 0 0 1 1 0 0 0 1 1];
% message with redundant information
data = zeros(1, 2*length(message)-1);
data(1:2:end) = message;
% index representing original message in 'data' vector
i = 1:length(message);
% index representing extra information in 'data' vector
j = 1.99:length(message)+0.99;
% generating 'time' vector by concatenating indices i & j to
% represent 'data' vector
tim = [];
for(k = 1:length(message))
   tim = [tim i(k) j(k)];
end
% generating digital signal
signal = []; N = length(data);
pre_bit = 1;
for(t = 1:2:N)
 if(data(t)==0)
signal(t:t+1) = 0;
 elseif(data(t)==1 & pre_bit == 1)
signal(t:t+1) = 1;
pre_bit = -1;
 else
signal(t:t+1) = -1;
pre_bit = 1;
  end
end
% displaying digital signal
figure(1);
```

```
subplot(121);
stem(message, 'filled', 'linewidth', 2);
title('Message');
ylabel('Value');
xlabel('Number of bits per second (bps)');
axis([1 length(message) -0.5 1.5]);
grid on;
subplot(122);
plot(tim,signal,'linewidth',2);
title('Bipolar-AMI Encoded Signal');
ylabel('Signal Levels (V)');
xlabel('Number of bits per second (bps)');
axis([1 length(message)+1 -1.5 1.5]);
grid on;
```

SCREENSHOT OF THE RESULT: