Weekly Chart Activity

Month: September 2021

School of Electronics and Communication Engineering

Third Year B. Tech.

DC PROJECT REPORT

(PBL ACTIVITY)

Academic Year 2021-2022 Trimester-VII

Name of Student ---------JAY SRIVASTAVA---, TANMAY KAPIL,ATUL KUMAR---------
----

PRN Number and Batch -------1032191969-----,1032191976--1032191844,----------------------
----

Project guide------------Prof M.S.Mahajani Ma’am---

S. No. 124, Paud Road, Kothrud, Pune-411038 Maharashtra, India.
Website: www.mitwpu.edu.in

B. Tech. Trimester VIII PROJECT BASED LEARNING REPORT

On

DEMONSTRATING LINE CODING & EYE DIAGRAM USING MATLAB GUI

(Submitted by,

1032191844 ATUL KUMAR

1032191969 JAY SRIVASTAVA

1032191976 TANMAY KAPIL

Project Guide:

Prof. P.S.Mahajani

(**Year: 20 21 - 2022**

School of Electronics and Communication Engineering

MIT World Peace University, Pune

MIT World Peace University, Pune

School of Electronics and Communication Engineering

Project Report Contents

DEMONSTRATING LINE CODING & EYE DIAGRAM USING MATLAB GUI

1. Objectives

We obtain the digital data from computers which is discrete in nature. If such a discrete signal is transmitted over a band limited channel, then the signal gets dispersed. This causes the pulses to overlap and cause distortion. Such distortion is called as Inter-symbol Interference (ISI). In order to avoid this distortion, we should not transmit the discrete signal as it is on the transmission medium. The data is first converted into suitable format and the transmitted over a communication channel. These formats are called as line codes.

2. Introduction

A **line code** is the code used for data transmission of a digital signal over a transmission line. This process of coding is chosen so as to avoid overlap and distortion of signal such as inter-symbol interference.

Following are the properties of line coding −

- As the coding is done to make more bits transmit on a single signal, the bandwidth used is much reduced.
- For a given bandwidth, the power is efficiently used.
- The probability of error is much reduced.
- Error detection is done and the bipolar too has a correction capability.
- Power density is much favorable.
- The timing content is adequate.
- Long strings of **1s** and **0s** is avoided to maintain transparency.

Types of Line Coding

There are 3 types of Line Coding

- Unipolar
- Polar
- Bi-polar

2.1 Definations of Line Coding

- **Unipolar RZ**: In this case if bit is ‘1’ then logic high level (A) is transmitted only for first half bit period ($T_b/2$) & if bit is ‘0’, logic low level 0 is transmitted. Here the voltage level is either +A or zero; hence it is a unipolar format.
- **Unipolar NRZ:** In this case if bit is ‘1’ then logic high level (A) is transmitted for full bit period (Tb) and if bit is ‘0’ then logic low level (0) is transmitted. As the pulses have either +A or 0 amplitude it is called as a unipolar format. Due to longer pulse duration, the NRZ pulses carry more energy than the RZ pulses.
- **Bipolar RZ:** In this case, if bit is ‘1’ then logic high level (+A/2) is transmitted for half bit period (Tb /2) and then logic low level for remaining half bit period. If bit is ‘0’ then negative high level (-A/2) is transmitted for first half bit period (Tb/2) and then logic low level for remaining half bit period.
- **Bipolar NRZ (AMI):** In this case, if bit is '1' then high level (+A) is transmitted for full bit period (Tb). If next bit (Not necessary consecutive) is also '1' then negative high level (-A) is transmitted. i.e. for every ‘1’, sign of high level is altered. If bit is zero, then logic low level 0 is transmitted. This format is used in the PCM-TDM T1 system for digital telephony.

Split Phase Manchester : In this case, if bit is '1' then logic high level is transmitted for first half bit period (+A/2) followed by logic low level (-A/2) for next half bit period. If bit is '0' then logic low level (-A/2) is transmitted for first half bit period followed by logic high level (+A/2) for next half bit period. Local Area Network (LAN) such as Ethernet, uses the Manchester Code for signal transmission over the network

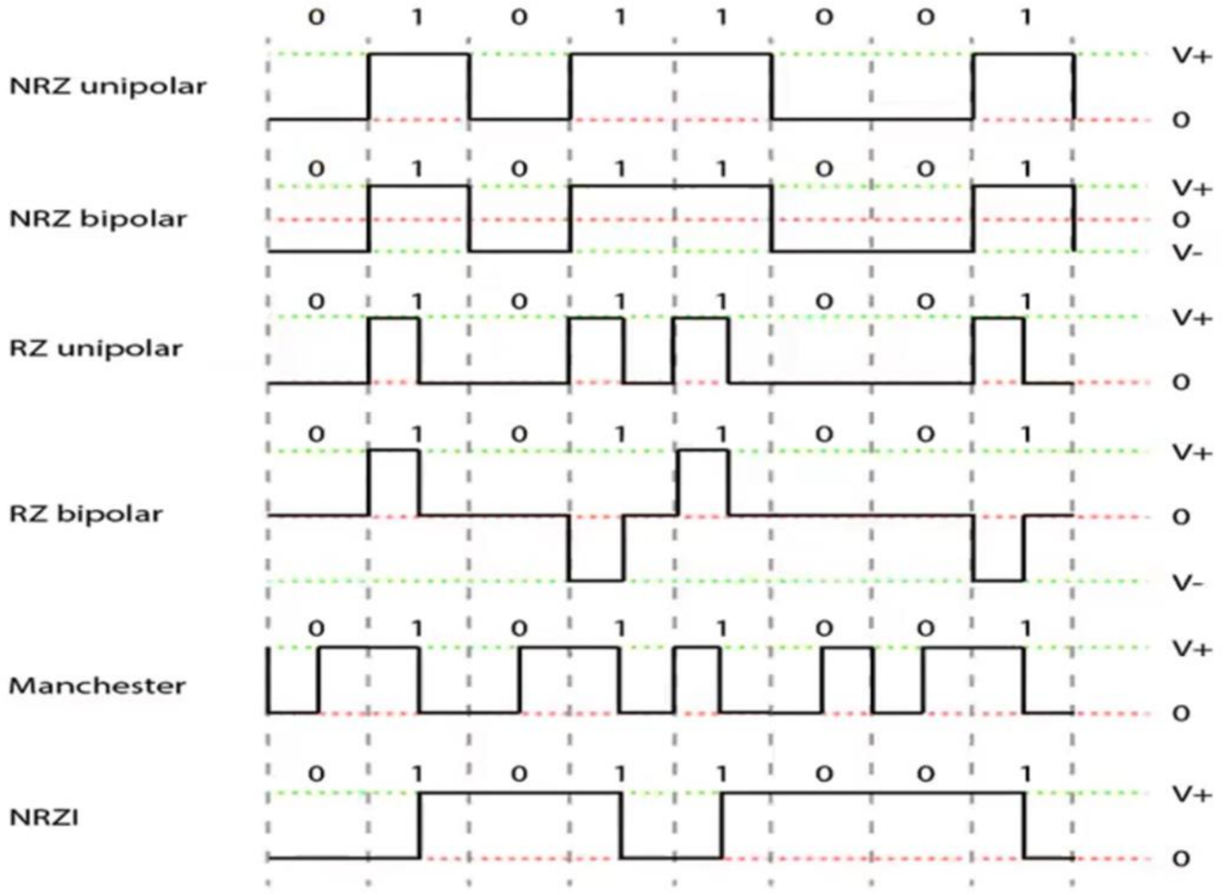


Figure 1

2.2 Power Spectral Density of Line Codes

The PSD can be evaluated by using either a deterministic or a stochastic technique. To evaluate PSD deterministic technique, the waveforms for a line codes that result from a particular data is used .The approximate PSD is then evaluated by using the equation

$$P(f) = \sum_{n=-\infty}^{n=\infty} C_n \, |\delta f(t - nf_o|^2 \text{ -------------------- Eq. 5.1}$$

To evaluate PSD using stochastic technique, a digital code can be represented by followingequation:

$$P(t) = \sum_{n=0}^{n=\infty} a_n f(t - nTs) \quad \text{----------------- Eq. 5.2}$$

Where,

f (t) = symbol pulse shape

Ts = Duration of one symbol = Time to send one bit i.e. = T_b for binary signalling

The general equation for PSD of a digital signal is:

$$P(f) = \frac{|F(t)|^2}{Ts} \sum_{k=-\infty} a_n f(t - nTs) e^{j2\pi kfTs} \quad \text{----------------- Eq. 5.3}$$

Where, F (t) = Fourier Transform of pulse shape f(t)R (k) = Autocorrelation of the data

$$I = \sum_{i=1}^{I} a_n (a_n + k)_i P_i \quad \text{----------------- Eq. 5.4}$$

Polar NRZ Signalling : For Polar signalling, the possible levels for a’s are +A and -A volts. The demerit of Polar NRZ is that it has large PSD near dc. The merit of this coding is that it is easy to generate, although positive & negative power supplies are required.

- **Unipolar RZ Signalling:** For RZ signalling, the pulse duration is T_b /2 instead of T_b, as used in NRZ signalling. The demerit of Unipolar RZ is that it requires 3dB more signal power than polar signalling for the same probability error. Moreover, the spectrum is not negligible for frequencies near dc.
- **Bipolar RZ Signalling:** The PSD for a bipolar signal can also be obtained using eq. 1(a). The permitted value for ao are +A,-A and a binary 0 is represented by an =0. The demerits of bipolar signalling are that the receiver has to distinguish between there levels (+A,-A and 0), instead of just two levels in other signalling formats discussed above. The merit of this type of signalling is that it has single error detection capabilities built in. Any violations can be easily detected.
- **Manchester Signalling:** The null BW of the Manchester format is twice that of the bipolar BW. However, the Manchester code has a zero dc level on a bit by bit basis. Moreover, a string of zeros will not cause a loss of the clocking signal.

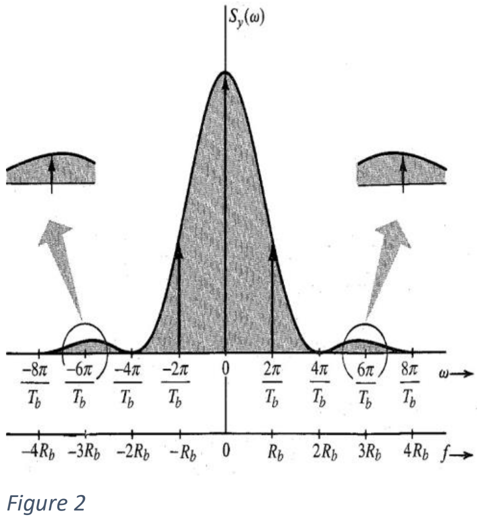


Figure 2

2.3 **EYE DIAGRAM**

It is called an eye diagram, or eye pattern, because the pattern looks like a series of eyes between a pair of rails for several types of coding schemes. It is created by taking the time domain signal and overlapping the traces for a certain number of symbols.
If we are sampling a signal at a rate of 10 samples per second and we want to look at two symbols, then we would cut the signal every 20 samples and overlap them.
In practice, it is created on an oscilloscope display by repetitively sampling the digital signal from a receiver and placing it into the vertical input of the scope while the data rate triggers the horizontal sweep.

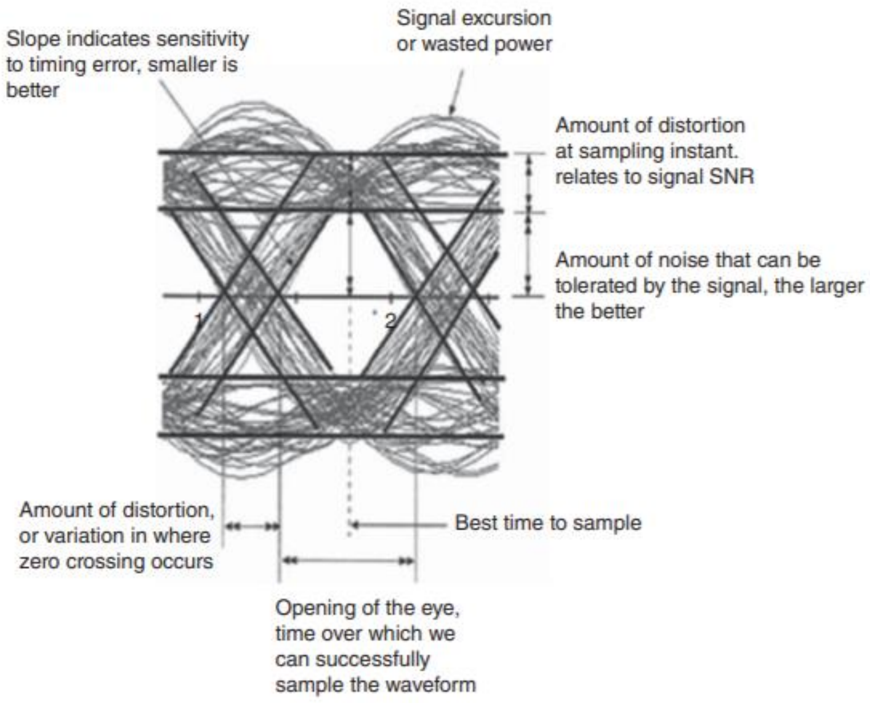
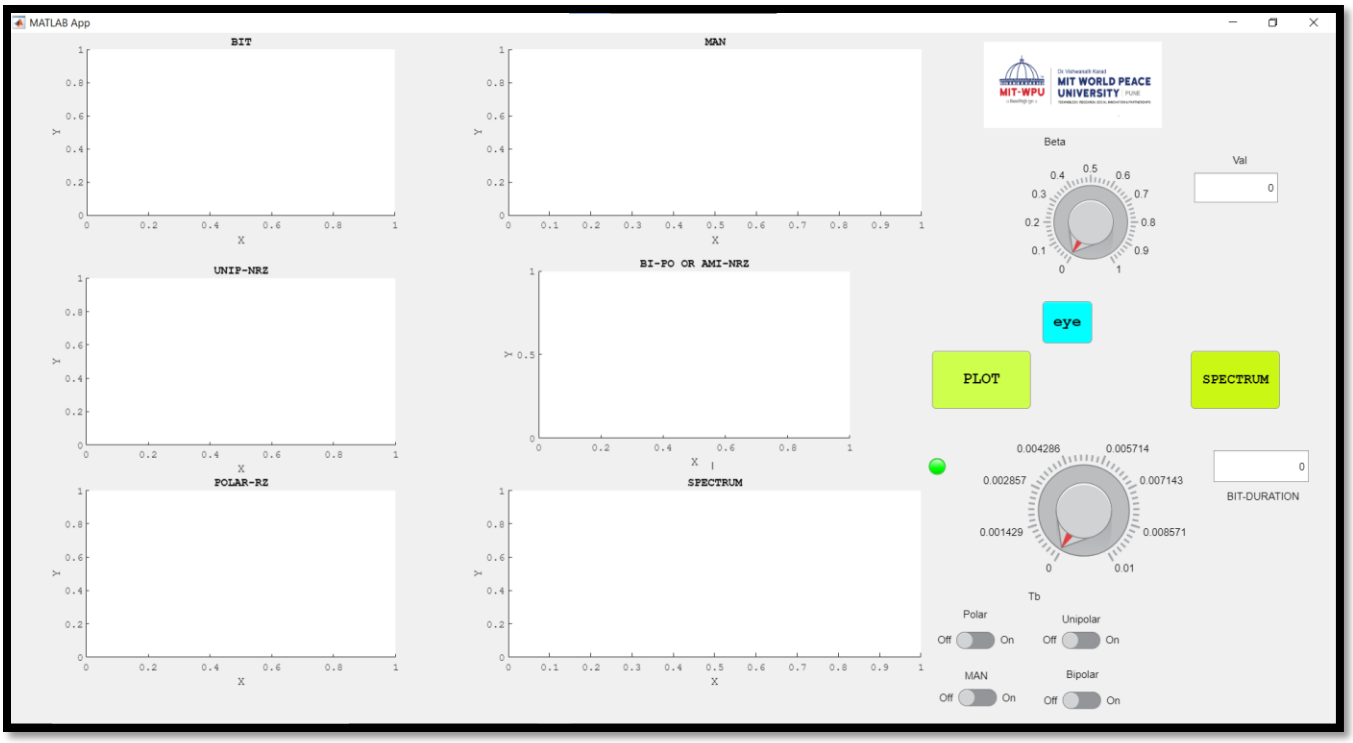

Slope indicates sensitivity to timing error, smaller is better
Signal excursion or wasted power
Amount of distortion at sampling instant. relates to signal SNR
Amount of noise that can be tolerated by the signal, the larger the better
1
2
Amount of distortion, or variation in where zero crossing occurs
Best time to sample
Opening of the eye, time over which we can successfully sample the waveform

Figure 3

- Eye opening quality (height, peak-to-peak) ↔ Additive noise/intersymbol interference
- Eye overshoot/undershoot ↔ Peak distortion due to interruption in the signal path
- Eye width ↔ Timing synchronization and jitter effects.

3. Experiment and Results

3.1MATLAB GUI



3.1.1 CODE FOR BUTTONS

- Plot Button

```
N=10;
val=N;
inbits=randi([0,1],1,val);
% inbits=[1 0 1 0 1 0 1 0 1 0];
A=1;
Tb=app.TbKnob.Value;
Rb=1/Tb;
Fs=4*Rb;
% N=length(inbits);
tTb = linspace(0,Tb);
x_uninrz=[];
x_unirz=[];
x_polrz=[];
x_polnrz=[];
x_manchester=[];
for k1=1:N
```

```
x_uninrz=[x_uninrz A*inbits(k1)*ones(1,length(tTb))];
end
for k2=1:N
x_unirz = [x_unirz A*inbits(k2)*ones(1,length(tTb)/2)
0*inbits(k2)*ones(1,length(tTb)/2)];


end
for k3=1:N
c1 = ones(1,length(tTb)/2);
b1 = zeros(1,length(tTb)/2);
p1 = [c1 b1];
x_polrz= [x_polrz ((-1)^(inbits(k3)+1))*(A/2)*p1];
end
for k4=1:4
x_polnrz = [x_polnrz ((-1)^(inbits(k4) + 1))*A/2*ones(1,length(tTb))];
end
            for k5 = 1:N
            c2 = ones(1,length(tTb)/2);
            b2 = -1*ones(1,length(tTb)/2);
            p2 = [c2 b2];% Concatenation
            x_manchester = [x_manchester ((-1)^(inbits(k5)+1))*A/2*p2];
            end
            T = linspace(0,N*Tb,length(x_uninrz));

            f=1;
            for m=1:N
                if inbits(m)==1
                if f==1
                    nn(m)=1;
                    f=-1;
                else
                    nn(m)=-1;
                    f=1;
                end
                else
                    nn(m)=0;
                end
            end

             %disp(inbits);
             %disp(nn);
             i2=1;
      for i1=1:N
          inbits_plot(i2:N*length(tTb))=inbits(i1);   %% generating bit pattern
for bit duration
          i2=i2+length(tTb);
      end
      plot(app.UIAxes,T,inbits_plot);
      ylim([-2 2]);
      plot(app.UIAxes2,T,x_uninrz);
       ylim([-2 2]);
       plot(app.UIAxes3,T,x_polrz);
       ylim([-1 1]);
       plot(app.UIAxes4,T,x_manchester);
       ylim([-1 1]);
       j2=1;
```

```
        for j1=1:N
            nn_plot(j2:N*length(tTb))=nn(j1);
            j2=j2+length(tTb);
        end
        plot(app.UIAxes5,T,nn_plot);
        ylim([-2 2]);
end
```

- SPECTRUM BUTTON

```
   Tb=app.TbKnob.Value;
            Rb=1/Tb;
           f=0:0.05*Rb:2*Rb;
           x=f*Tb;
          value=app.PolarSwitch.Value;
          if(strcmp(value,'On'))
            P=Tb*(sinc(x).*sinc(x));
            plot(app.UIAxes6,f,P,'r');
            app.ILamp.Color='r';
          end
          value1=app.UnipolarSwitch.Value;

          if(strcmp(value1,'On'))
              P1=0.5*Tb*(sinc(x).*sinc(x))+ 0.5 *dirac(f);
              plot(app.UIAxes6,f,P1,'g');
              app.ILamp.Color='g';
          end
          value3=app.MANSwitch.Value;
          if(strcmp(value3,'On'))
              P2=Tb*(sinc(x/2)).^2.*(sin(pi*x/2)).^2;
              plot(app.UIAxes6,f,P2,'b');
              app.ILamp.Color='b';
          end
          value4=app.BipolarSwitch.Value;
          if(strcmp(value4,'On'))
              P3=Tb*(sinc(x/2)).^2.*(sin(pi*x)).^2;
              plot(app.UIAxes6,f,P3,'m');
              app.ILamp.Color='m';
          end


        end
```

- Eye Button

```
         function eyeButtonPushed(app, event)
   rolloff = app.BetaKnob.Value;       % Rolloff factor
   span = 6;              % Filter span in symbols
   sps = 10;
   b = rcosdesign(rolloff, span, sps);
   bnrz=ones(1,sps);
   brz=[ones(1,sps/2) zeros(1,sps/2)];
```

```
    len=250;
    d =awgn( (5*randi([0 1], len, 1) - 5),10);    % 10 dB is the AWGN SNR, 0 being
    poor, 20 being good SNR
    subplot(2,1,1)
    stem(d)
    x = upfirdn(d, b, sps); % replace bnrz with b, brz for raised cosine, bipolar
    return zero etc


    eyediagram(x,sps);
            end
```

- Full Code:

```
classdef gui_dc_final_final < matlab.apps.AppBase


% Properties that correspond to app components
properties (Access = public)
UIFigure matlab.ui.Figure
Image matlab.ui.control.Image
BITDURATIONEditFieldLabel matlab.ui.control.Label
BITDURATIONEditField matlab.ui.control.NumericEditField
TbKnobLabel matlab.ui.control.Label
TbKnob matlab.ui.control.Knob
UnipolarSwitchLabel matlab.ui.control.Label
UnipolarSwitch matlab.ui.control.Switch
PolarSwitchLabel matlab.ui.control.Label
PolarSwitch matlab.ui.control.Switch
BipolarSwitchLabel matlab.ui.control.Label
BipolarSwitch matlab.ui.control.Switch
MANSwitchLabel matlab.ui.control.Label
MANSwitch matlab.ui.control.Switch
ILampLabel matlab.ui.control.Label
ILamp matlab.ui.control.Lamp
PLOTButton matlab.ui.control.Button
SPECTRUMButton matlab.ui.control.Button
eyeButton matlab.ui.control.Button
BetaKnobLabel matlab.ui.control.Label
BetaKnob matlab.ui.control.Knob
ValEditFieldLabel matlab.ui.control.Label
ValEditField matlab.ui.control.NumericEditField
UIAxes2 matlab.ui.control.UIAxes
UIAxes matlab.ui.control.UIAxes
UIAxes3 matlab.ui.control.UIAxes
UIAxes4 matlab.ui.control.UIAxes
UIAxes5 matlab.ui.control.UIAxes
UIAxes6 matlab.ui.control.UIAxes
end


% Callbacks that handle component events
methods (Access = private)


% Callback function
```

```matlab
function ASliderValueChanged(app, event)
value = app.ASlider.Value;
app.AMPLITUDEEditField.Value=value;
end


% Callback function
function NSliderValueChanged(app, event)
value = app.NSlider.Value;
app.BITLENEditField.Value=value;
end


% Callback function
function TbKnobValueChanged(app, event)
value = app.TbKnob.Value;
app.TbEditField.Value=value;
end


% Button pushed function: PLOTButton
function PLOTButtonPushed(app, event)
N=10;
val=N;
inbits=randi([0,1],1,val);
% inbits=[1 0 1 0 1 0 1 0 1 0];
A=1;
Tb=app.TbKnob.Value;
Rb=1/Tb;
Fs=4*Rb;
% N=length(inbits);
tTb = linspace(0,Tb);
x_uninrz=[];
x_unirz=[];
x_polrz=[];
x_polnrz=[];
x_manchester=[];
for k1=1:N
x_uninrz=[x_uninrz A*inbits(k1)*ones(1,length(tTb))];
end
for k2=1:N
x_unirz = [x_unirz A*inbits(k2)*ones(1,length(tTb)/2)
0*inbits(k2)*ones(1,length(tTb)/2)];


end
for k3=1:N
c1 = ones(1,length(tTb)/2);
b1 = zeros(1,length(tTb)/2);
p1 = [c1 b1];
x_polrz= [x_polrz ((-1)^(inbits(k3)+1))*(A/2)*p1];
end
for k4=1:4
x_polnrz = [x_polnrz ((-1)^(inbits(k4) + 1))*A/2*ones(1,length(tTb))];
end
for k5 = 1:N
c2 = ones(1,length(tTb)/2);
```

```
b2 = -1*ones(1,length(tTb)/2);
p2 = [c2 b2];% Concatenation
x_manchester = [x_manchester ((-1)^(inbits(k5)+1))*A/2*p2];
end
T = linspace(0,N*Tb,length(x_uninrz));
f=1;
for m=1:N
if inbits(m)==1
if f==1
nn(m)=1;
f=-1;
else
nn(m)=-1;
f=1;
end
else
nn(m)=0;
end
end
%disp(inbits);
%disp(nn);
i2=1;
for i1=1:N
inbits_plot(i2:N*length(tTb))=inbits(i1); %% generating bit pattern for bit duration
i2=i2+length(tTb);
end
plot(app.UIAxes,T,inbits_plot);
ylim([-2 2]);
plot(app.UIAxes2,T,x_uninrz);
ylim([-2 2]);
plot(app.UIAxes3,T,x_polrz);
ylim([-1 1]);
plot(app.UIAxes4,T,x_manchester);
ylim([-1 1]);
j2=1;
for j1=1:N
nn_plot(j2:N*length(tTb))=nn(j1);
j2=j2+length(tTb);
end
plot(app.UIAxes5,T,nn_plot);
ylim([-2 2]);









end


% Callback function
function TbSliderValueChanged(app, event)
value = app.TbSlider.Value;
app.BITDURATIONEditField.Value=value;
        end
        % Value changed function: TbKnob
        function TbKnobValueChanged2(app, event)
            value = app.TbKnob.Value;
            app.BITDURATIONEditField.Value=value;
        end


        % Callback function
        function SPECTRUMButtonPushed(app, event)
       Tb=app.TbKnob.Value;

        end


        % Button pushed function: SPECTRUMButton
        function SPECTRUMButtonPushed2(app, event)
            Tb=app.TbKnob.Value;
            Rb=1/Tb;
           f=0:0.05*Rb:2*Rb;
           x=f*Tb;
          value=app.PolarSwitch.Value;
          if(strcmp(value,'On'))
            P=Tb*(sinc(x).*sinc(x));
            plot(app.UIAxes6,f,P,'r');
            app.ILamp.Color='r';
          end
          value1=app.UnipolarSwitch.Value;

          if(strcmp(value1,'On'))
              P1=0.5*Tb*(sinc(x).*sinc(x))+ 0.5 *dirac(f);
              plot(app.UIAxes6,f,P1,'g');
              app.ILamp.Color='g';
          end
          value3=app.MANSwitch.Value;
          if(strcmp(value3,'On'))
              P2=Tb*(sinc(x/2)).^2.*(sin(pi*x/2)).^2;
              plot(app.UIAxes6,f,P2,'b');
              app.ILamp.Color='b';
          end
          value4=app.BipolarSwitch.Value;
          if(strcmp(value4,'On'))
              P3=Tb*(sinc(x/2)).^2.*(sin(pi*x)).^2;
              plot(app.UIAxes6,f,P3,'m');
              app.ILamp.Color='m';
          end


        end


        % Button pushed function: eyeButton
        function eyeButtonPushed(app, event)
rolloff = app.BetaKnob.Value;       % Rolloff factor
span = 6;            % Filter span in symbols
sps = 10;
```

```
b = rcosdesign(rolloff, span, sps);
bnrz=ones(1,sps);
brz=[ones(1,sps/2) zeros(1,sps/2)];
len=250;
d =awgn( (5*randi([0 1], len, 1) - 5),10);    % 10 dB is the AWGN SNR, 0 being poor, 
20 being good SNR
subplot(2,1,1)
stem(d)
x = upfirdn(d, b, sps); % replace bnrz with b, brz for raised cosine, bipolar return 
zero etc


eyediagram(x,sps);
        end


        % Value changed function: BetaKnob
        function BetaKnobValueChanged(app, event)
            value = app.BetaKnob.Value;
            app.ValEditField.Value=value;
        end
    end


    % Component initialization
    methods (Access = private)


        % Create UIFigure and components
        function createComponents(app)


            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 910 615];
            app.UIFigure.Name = 'MATLAB App';


            % Create Image
            app.Image = uiimage(app.UIFigure);
            app.Image.Position = [640 507 221 100];
            app.Image.ImageSource = 'MITLOGO.png';


            % Create BITDURATIONEditFieldLabel
            app.BITDURATIONEditFieldLabel = uilabel(app.UIFigure);
            app.BITDURATIONEditFieldLabel.HorizontalAlignment = 'right';
            app.BITDURATIONEditFieldLabel.Position = [804 152 89 22];
            app.BITDURATIONEditFieldLabel.Text = 'BIT-DURATION';


            % Create BITDURATIONEditField
            app.BITDURATIONEditField = uieditfield(app.UIFigure, 'numeric');
            app.BITDURATIONEditField.Position = [818 179 62 37];


            % Create TbKnobLabel
            app.TbKnobLabel = uilabel(app.UIFigure);
            app.TbKnobLabel.HorizontalAlignment = 'center';
            app.TbKnobLabel.Position = [708 139 25 22];
app.TbKnobLabel.Text = 'Tb';


% Create TbKnob
app.TbKnob = uiknob(app.UIFigure, 'continuous');
app.TbKnob.Limits = [0 0.01];
app.TbKnob.ValueChangedFcn = createCallbackFcn(app, @TbKnobValueChanged2, true);
app.TbKnob.Position = [692 174 60 60];


% Create UnipolarSwitchLabel
app.UnipolarSwitchLabel = uilabel(app.UIFigure);
app.UnipolarSwitchLabel.HorizontalAlignment = 'center';
app.UnipolarSwitchLabel.Position = [750 113 50 22];
app.UnipolarSwitchLabel.Text = 'Unipolar';


% Create UnipolarSwitch
app.UnipolarSwitch = uiswitch(app.UIFigure, 'slider');
app.UnipolarSwitch.Position = [752 89 45 20];


% Create PolarSwitchLabel
app.PolarSwitchLabel = uilabel(app.UIFigure);
app.PolarSwitchLabel.HorizontalAlignment = 'center';
app.PolarSwitchLabel.Position = [635 118 34 22];
app.PolarSwitchLabel.Text = 'Polar';


% Create PolarSwitch
app.PolarSwitch = uiswitch(app.UIFigure, 'slider');
app.PolarSwitch.Position = [630 89 45 20];


% Create BipolarSwitchLabel
app.BipolarSwitchLabel = uilabel(app.UIFigure);
app.BipolarSwitchLabel.HorizontalAlignment = 'center';
app.BipolarSwitchLabel.Position = [755 49 43 22];
app.BipolarSwitchLabel.Text = 'Bipolar';


% Create BipolarSwitch
app.BipolarSwitch = uiswitch(app.UIFigure, 'slider');
app.BipolarSwitch.Position = [753 19 45 20];


% Create MANSwitchLabel
app.MANSwitchLabel = uilabel(app.UIFigure);
app.MANSwitchLabel.HorizontalAlignment = 'center';
app.MANSwitchLabel.Position = [637 47 32 22];
app.MANSwitchLabel.Text = 'MAN';


% Create MANSwitch
app.MANSwitch = uiswitch(app.UIFigure, 'slider');
app.MANSwitch.Position = [632 22 45 20];
```

```
% Create ILampLabel
app.ILampLabel = uilabel(app.UIFigure);
app.ILampLabel.HorizontalAlignment = 'right';
app.ILampLabel.Position = [558 228 25 22];
app.ILampLabel.Text = 'I';


% Create ILamp
app.ILamp = uilamp(app.UIFigure);
app.ILamp.Position = [598 228 20 20];


% Create PLOTButton
app.PLOTButton = uibutton(app.UIFigure, 'push');
app.PLOTButton.ButtonPushedFcn = createCallbackFcn(app, @PLOTButtonPushed, true);
app.PLOTButton.BackgroundColor = [0.8118 1 0.302];
app.PLOTButton.FontName = 'Courier New';
app.PLOTButton.FontSize = 18;
app.PLOTButton.FontWeight = 'bold';
app.PLOTButton.Position = [602 264 114 67];
app.PLOTButton.Text = 'PLOT';


% Create SPECTRUMButton
app.SPECTRUMButton = uibutton(app.UIFigure, 'push');
app.SPECTRUMButton.ButtonPushedFcn = createCallbackFcn(app, @SPECTRUMButtonPushed2,
true);
app.SPECTRUMButton.BackgroundColor = [0.7922 0.9686 0.0784];
app.SPECTRUMButton.FontName = 'Courier New';
app.SPECTRUMButton.FontSize = 16;
app.SPECTRUMButton.FontWeight = 'bold';
app.SPECTRUMButton.Position = [769 264 103 67];
app.SPECTRUMButton.Text = 'SPECTRUM';


% Create eyeButton
app.eyeButton = uibutton(app.UIFigure, 'push');
app.eyeButton.ButtonPushedFcn = createCallbackFcn(app, @eyeButtonPushed, true);
app.eyeButton.BackgroundColor = [0 1 1];
app.eyeButton.FontName = 'Courier New';
app.eyeButton.FontSize = 18;
app.eyeButton.FontWeight = 'bold';
app.eyeButton.Position = [715 340 58 49];
app.eyeButton.Text = 'eye';


% Create BetaKnobLabel
app.BetaKnobLabel = uilabel(app.UIFigure);
app.BetaKnobLabel.HorizontalAlignment = 'center';
app.BetaKnobLabel.Position = [715 480 30 22];
app.BetaKnobLabel.Text = 'Beta';


% Create BetaKnob
app.BetaKnob = uiknob(app.UIFigure, 'continuous');
app.BetaKnob.Limits = [0 1];
app.BetaKnob.ValueChangedFcn = createCallbackFcn(app, @BetaKnobValueChanged, true);
app.BetaKnob.Position = [705 418 45 45];


% Create ValEditFieldLabel
app.ValEditFieldLabel = uilabel(app.UIFigure);
app.ValEditFieldLabel.HorizontalAlignment = 'right';
app.ValEditFieldLabel.Position = [809 459 25 22];
app.ValEditFieldLabel.Text = 'Val';


% Create ValEditField
app.ValEditField = uieditfield(app.UIFigure, 'numeric');
app.ValEditField.Position = [799 420 45 35];


% Create UIAxes2
app.UIAxes2 = uiaxes(app.UIFigure);
title(app.UIAxes2, 'UNIP-NRZ')
xlabel(app.UIAxes2, 'X')
ylabel(app.UIAxes2, 'Y')
zlabel(app.UIAxes2, 'Z')
app.UIAxes2.PlotBoxAspectRatio = [1.85975609756098 1 1];
app.UIAxes2.FontName = 'Courier New';
app.UIAxes2.YTick = [0 0.2 0.4 0.6 0.8 1];
app.UIAxes2.Position = [1 228 300 185];


% Create UIAxes
app.UIAxes = uiaxes(app.UIFigure);
title(app.UIAxes, 'BIT')
xlabel(app.UIAxes, 'X')
ylabel(app.UIAxes, 'Y')
zlabel(app.UIAxes, 'Z')
app.UIAxes.PlotBoxAspectRatio = [1.85975609756098 1 1];
app.UIAxes.FontName = 'Courier New';
app.UIAxes.Position = [1 431 300 185];


% Create UIAxes3
app.UIAxes3 = uiaxes(app.UIFigure);
title(app.UIAxes3, 'POLAR-RZ')
xlabel(app.UIAxes3, 'X')
ylabel(app.UIAxes3, 'Y')
zlabel(app.UIAxes3, 'Z')
app.UIAxes3.PlotBoxAspectRatio = [1.85975609756098 1 1];
app.UIAxes3.FontName = 'Courier New';
app.UIAxes3.YTick = [0 0.2 0.4 0.6 0.8 1];
app.UIAxes3.Position = [1 44 300 185];


% Create UIAxes4
app.UIAxes4 = uiaxes(app.UIFigure);
title(app.UIAxes4, 'MAN')
xlabel(app.UIAxes4, 'X')
ylabel(app.UIAxes4, 'Y')
```

```
zlabel(app.UIAxes4, 'Z')
app.UIAxes4.FontName = 'Courier New';
app.UIAxes4.Position = [300 431 286 185];


% Create UIAxes5
app.UIAxes5 = uiaxes(app.UIFigure);
title(app.UIAxes5, 'BI-PO OR AMI-NRZ')
xlabel(app.UIAxes5, 'X')
ylabel(app.UIAxes5, 'Y')
zlabel(app.UIAxes5, 'Z')
app.UIAxes5.PlotBoxAspectRatio = [1.87116564417178 1 1];
app.UIAxes5.FontName = 'Courier New';
app.UIAxes5.YTick = [0 0.5 1];
app.UIAxes5.Position = [293 236 300 185];


% Create UIAxes6
app.UIAxes6 = uiaxes(app.UIFigure);
title(app.UIAxes6, 'SPECTRUM')
xlabel(app.UIAxes6, 'X')
ylabel(app.UIAxes6, 'Y')
zlabel(app.UIAxes6, 'Z')
app.UIAxes6.FontName = 'Courier New';
app.UIAxes6.Position = [300 44 300 185];


% Show the figure after all components are created
app.UIFigure.Visible = 'on';
end
end


% App creation and deletion
methods (Access = public)


% Construct app
function app = gui_dc_final_final


% Create UIFigure and components
createComponents(app)


% Register the app with App Designer
registerApp(app, app.UIFigure)


if nargout == 0
clear app
end
end


% Code that executes before app deletion
function delete(app)
```

```
% Delete UIFigure when app is deleted
delete(app.UIFigure)
end
end
end
```

4. Results

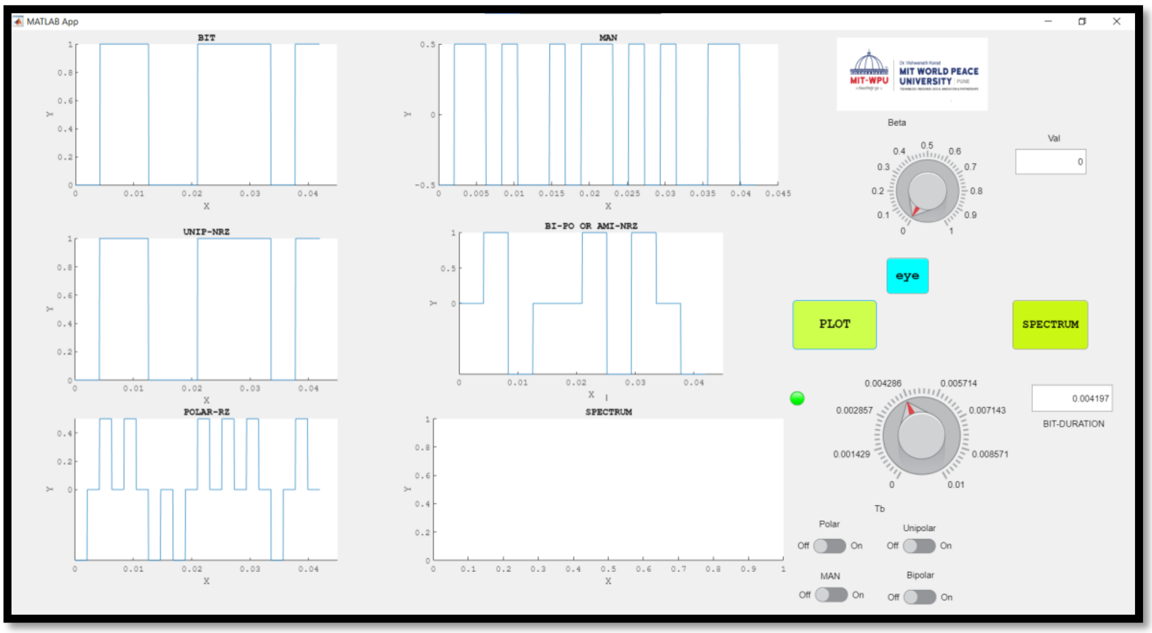


Figure 4

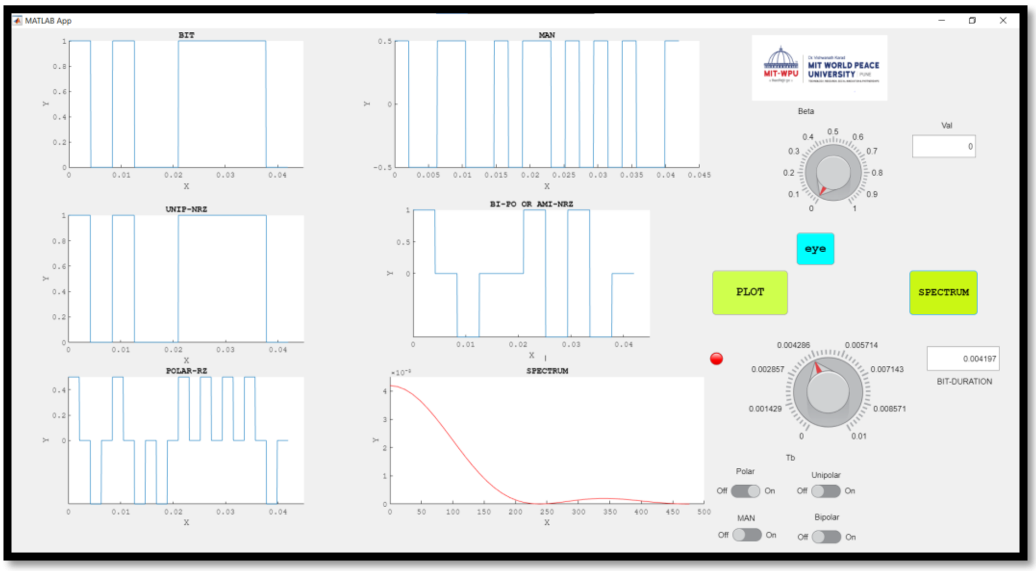


Figure 5

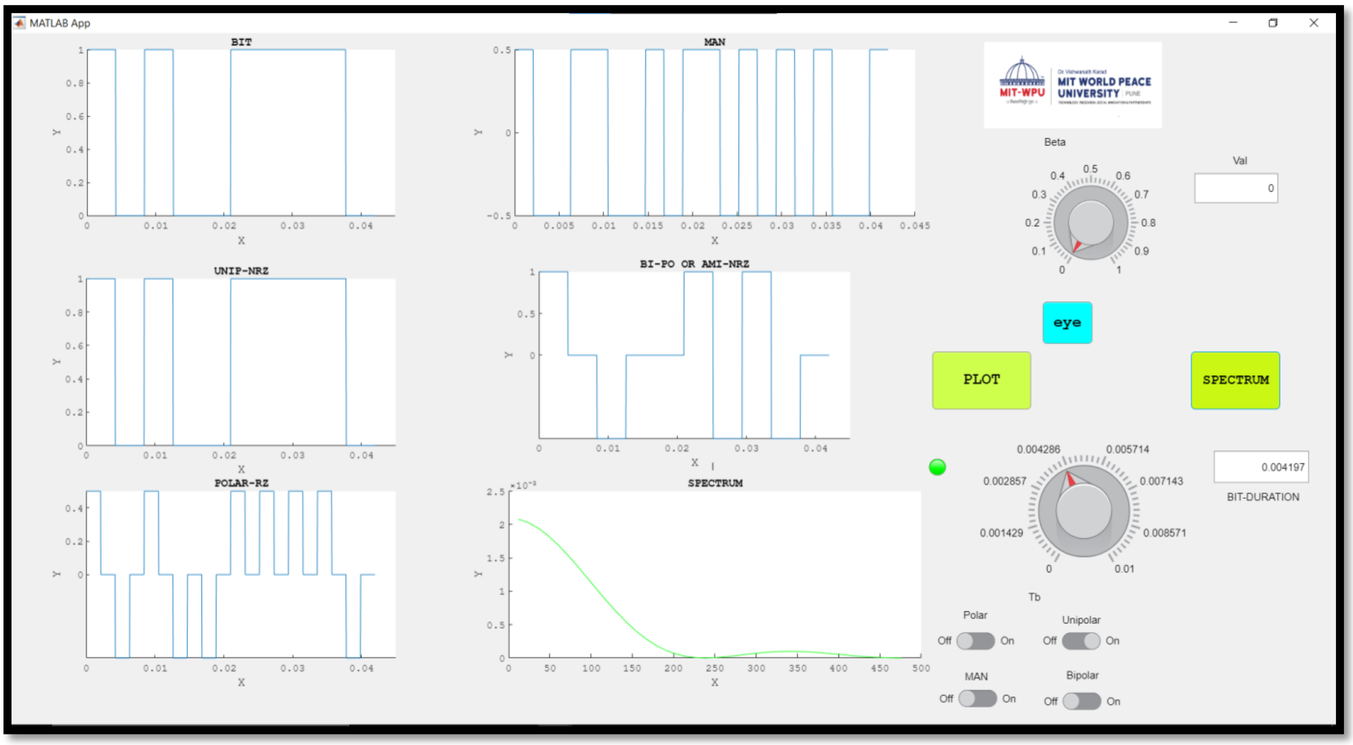


Figure 6

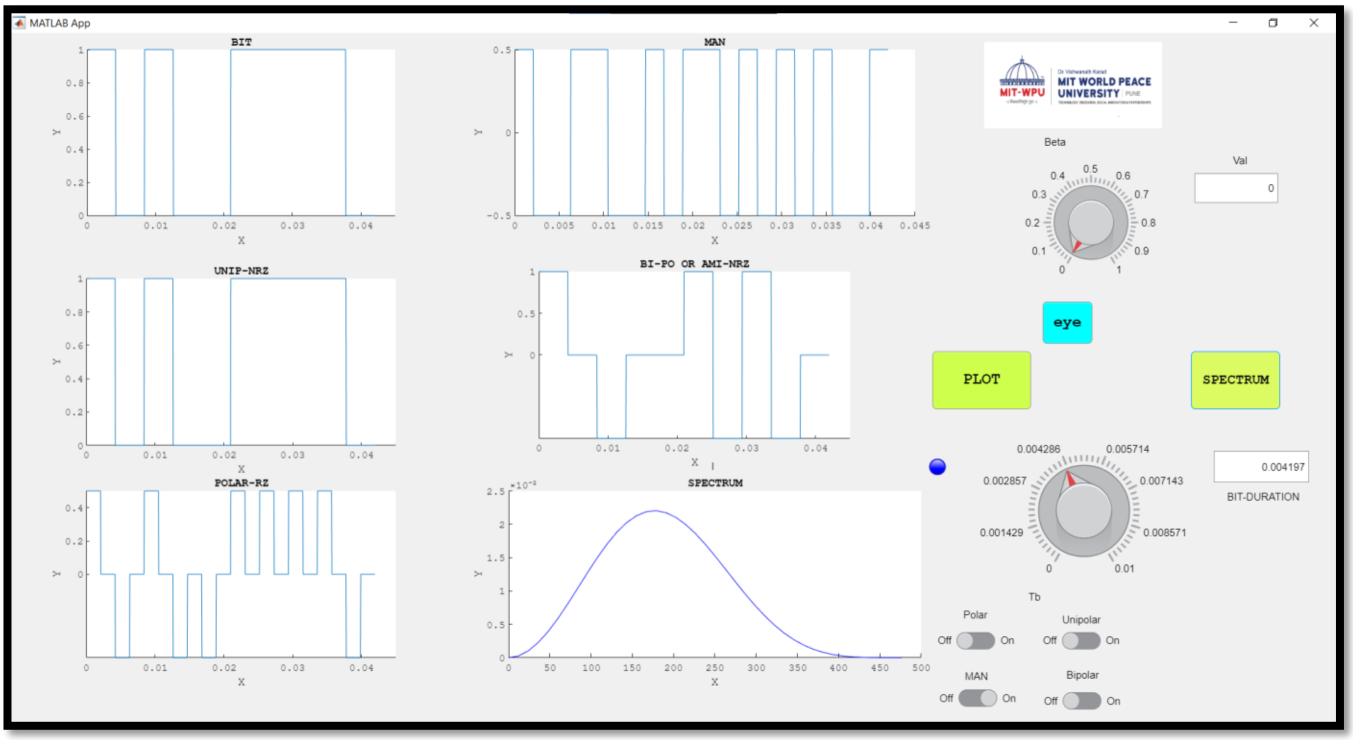
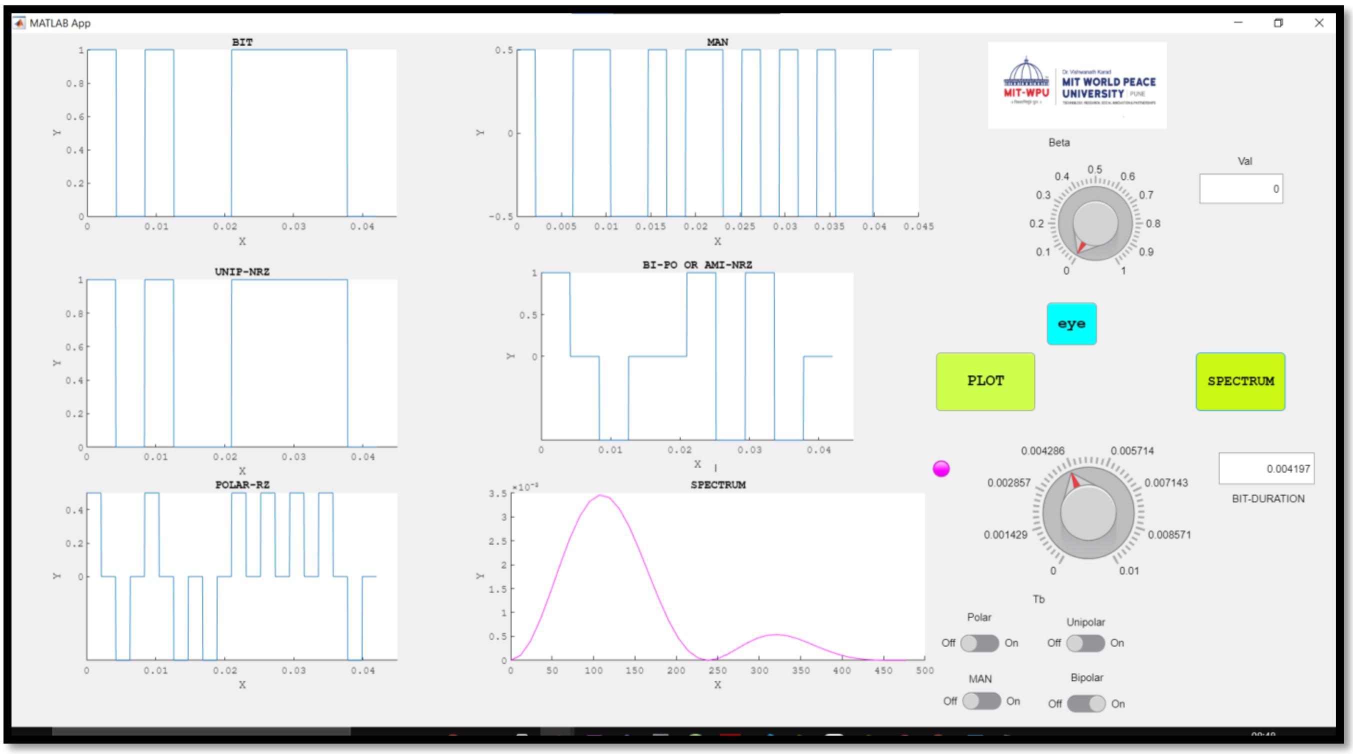


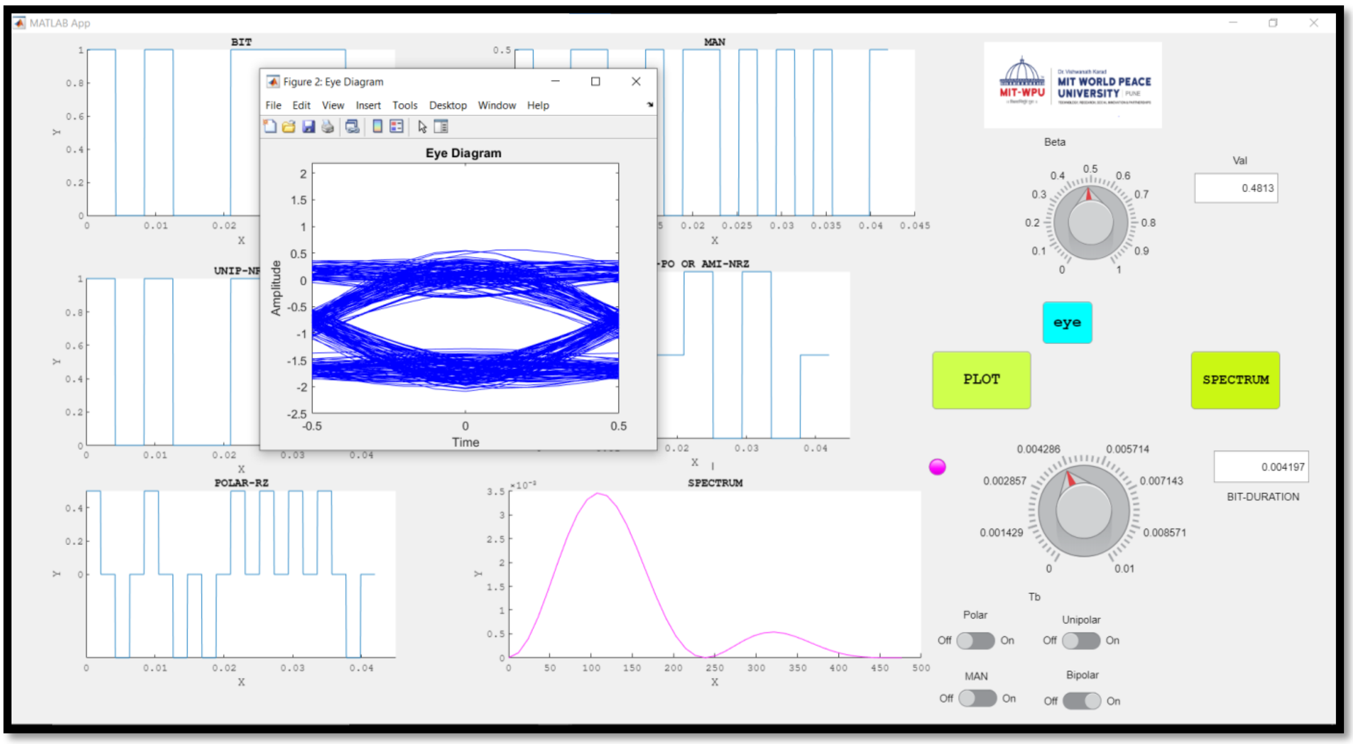
Figure 7



Val 0

BIT-DURATION 0.004197

Figure 8



Val 0.4813

BIT-DURATION 0.004197

Figure 9

5. **Conclusions**

Thus from the above Project Based Learning , We have concluded the analysis of different Line coding using MATLAB App designer GUI and verification of eye diagram using MTALAB App designer GUI and also Plotted different line coded waves and eye diagram and the PSD of Line coding using app designer.

6. **References**

- https://www.tutorialspoint.com/digital_communication/digital_communication_line_codes.html
- https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119279068.app2
- https://in.mathworks.com/discovery/matlab-gui.html?requestedDomain=
- https://www-elec.inaoep.mx/~rogerio/Tres/MatLab%207,%20Creating%20Graphical%20User%20Interfaces.pdf
-