PRÁCTICA No. 6 CODIGOS DE LINEA.

Como se recordará, existen diferentes tipos de codificación en los sistemas digitales. En los sistemas *banda base* es común utilizar códigos de línea con objeto de conformar el espectro de la señal de comunicaciones (por ejemplo, para eliminar la componente continua, facilitar la recuperación de sincronismo, etc.). En esta práctica vamos a simular digitalmente (representándolas mediante sus muestras) señales conformadas mediante diferentes códigos de línea. Así mismo podremos observar en el osciloscopio las señales reales que viajan por el canal.

5.1 códigos de línea.

La transmisión de datos en forma digital implica una cierta codificación. A la forma de transmisión donde no se usa una portadora se la conoce como transmisión en banda base.

Los códigos de línea son usados para este tipo de transmisión. Existen varios tipos de códigos, entre ellos Unipolar NRZ, Polar NRZ, Unipolar RZ, Bipolar RZ (AMI), Manchester, Manchester diferencial, B8ZS, HDB3.

5.1a. Sustento Teórico.

En telecomunicaciones, un *código en línea (modulación en banda base)* es un código elegido para ser usado en un sistema de comunicación como soporte para la transmisión.

Los códigos en línea son frecuentemente usados para el transporte digital de datos. Estos códigos consisten en representar la amplitud de la señal digital transportada respecto al tiempo. La representación de la onda se suele realizar mediante un número determinado de impulsos. Estos impulsos representan los 1s y los 0s digitales. Los tipos más comunes de codificación en línea son el NRZ, AMI, pseudoternario, manchester, manchester diferencial, B8ZS y HDB3.

Algunas de las características deseables de los códigos de línea son:

Autosincronización: contenido suficiente de señal de temporización (reloj) que permita identificar el tiempo correspondiente a un bit.

Capacidad de detección de errores: la definición del código incluye el poder de detectar un error.

Inmunidad al ruido: capacidad de detectar adecuadamente el valor de la señal ante la presencia de ruido (baja probabilidad de error).

Densidad espectral de potencia: igualación entre el espectro de frecuencia de la señal y la respuesta en frecuencia del canal de transmisión.

Ancho de banda: contenido suficiente de señal de temporización que permita identificar el tiempo correspondiente a un bit.

Transparencia: independencia de las características del código en relación a la secuencia de unos y ceros que transmita

Objetivos general:

Entender la forma que los códigos de línea cumplen en cierta medida las propiedades que los caracterizan.

DESARROLLO: El siguiente listado contiene un script para MATLAB y así poder ver las características de los siguientes códigos de línea, en diversos niveles de voltaje asignados a los estados lógicos: Unipolar, bipolar y Polar

- Unipolar NRZ.
- Unipolar RZ.
- Polar RZ.
- AMI NRZ.
- AMI RZ.
- Polar Manchester.

Con éste código, crear un script llamado *line_coding* y correrlo con estos parámetros. Varíe los siguientes parámetros y saque sus conclusiones.

```
A = 1;                                  % Amplitud del pulso
T = 1;                                  % duracion del bit
bit = [ 1 0 1 1 1 0 0 0 0 0 1 1 0 0 1 0 1 0];   % Stream de bits
polar = [ -A +A ];                      % mapeo Polar
M = 32;                                 % Oversampling ratio (must be even)
T_bits = T*length(bit);                 % Duration of bit stream
t = 1/M:1/M:T_bits ;                    % Time scale
```

Listado para similar los códigos de línea.

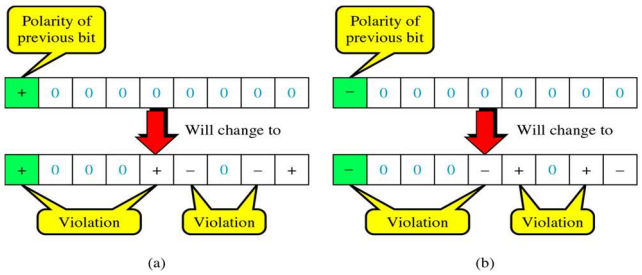
```
% File: line_coding.m
% Descripción: Ilustración de los esquemas de los códigos de línea
% Introducción a las telecomunicaciones 2020
% Tecnológico Nacional de México Campus Oaxaca;
% Departamento de Electrónica;
% DOCENTE: ING. MIGUEL ANGEL PEREZ SOLANO.

clear

A = 1;                                      % Amplitud del pulso
T = 1;                                      % duracion del bit
bit = [ 1 0 1 1 1 0 0 0 0 0 1 1 0 0 1 0 1 0];  % Stream de bits
polar = [ -A +A ];                          % mapeo Polar
M = 32;                                     % Oversampling ratio (must be even)
T_bits = T*length(bit);                     % Duration of bit stream
t = 1/M:1/M:T_bits ;                        % Time scale

bit_M= [];
for i=1:length(bit)
   bit_M = [bit_M bit(i)*ones(1,M)];        % Oversampled bit stream
end

p_NRZ(1:M)=1;                               % NRZ pulse
p_RZ(1:M/2)=1; p_RZ(M/2:M)=0;               % RZ pulse
p_M(1:M/2)=-1; p_M(M/2:M)=1;                % Manchester pulse

% (a) Unipolar NRZ
for i=1:length(bit)
   for j=(i-1)*M+1:i*M
      s_UNRZ(j) = bit(i)*A*p_NRZ(j-(i-1)*M);
   end
end

% (b) Unipolar RZ
for i=1:length(bit)
   for j=(i-1)*M+1:i*M
      s_URZ(j) = bit(i)*A*p_RZ(j-(i-1)*M);
   end
end

% (c) Polar NRZ
for i=1:length(bit)
   for j=(i-1)*M+1:i*M
      s_PNRZ(j) = polar(bit(i)+1)*p_NRZ(j-(i-1)*M);
```

```
   end
end

% (d) Polar RZ
for i=1:length(bit)
   for j=(i-1)*M+1:i*M
      s_PRZ(j) = polar(bit(i)+1)*p_RZ(j-(i-1)*M);
   end
end

% (e) AMI NRZ
st = [-A*ones(1,M)];                % Inital amplitude -A
for i=1:length(bit)
   if bit(i), st = -st; end
   for j=(i-1)*M+1:i*M
      AMI_NRZ(j) = bit(i)*st(j-(i-1)*M)*p_NRZ(j-(i-1)*M);
   end
end

% (f) AMI RZ
st = [-A*ones(1,M)];                                    % Inital amplitude -A
for i=1:length(bit)
   if bit(i), st = -st; end
   for j=(i-1)*M+1:i*M
      AMI_RZ(j) = bit(i)*st(j-(i-1)*M)*p_RZ(j-(i-1)*M);
   end
end

% (g) Manchester Polar
for i=1:length(bit)
   for j=(i-1)*M+1:i*M
      s_M(j) = polar(bit(i)+1)*p_M(j-(i-1)*M);
   end
end

% Plot all signals
subplot(7,1,1)
plot(t,bit_M); ylabel ('Bits'); axis ([ 0 T_bits  -0.1 1.1 ])
set(gca,'YTick',[0 1])
title('Ejemplos de esquemas de códigos de linea')
subplot(7,1,2)
plot(t,s_UNRZ); ylabel ('U-NRZ'); axis ([ 0 T_bits  -0.1*A 1.1*A ])
subplot(7,1,3)
plot(t,s_URZ); ylabel ('U-RZ'); axis ([ 0 T_bits  -0.1*A 1.1*A ]);
subplot(7,1,4)
plot(t,AMI_RZ); ylabel ('AMI-RZ'); axis ([ 0 T_bits  -1.1*A 1.1*A ]);
subplot(7,1,5)
plot(t,s_PNRZ); ylabel ('P-NRZ'); axis ([ 0 T_bits  -1.1*A 1.1*A ])
subplot(7,1,6)
plot(t,s_PRZ); ylabel ('P-RZ'); axis ([ 0 T_bits  -1.1*A 1.1*A ])
subplot(7,1,7)
```

```
plot(t,s_M); ylabel ('Manchester'); axis ([ 0 T_bits  -1.1*A 1.1*A ])
```

:

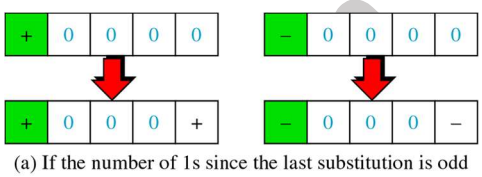
El siguiente código, es para generar un script y ver el desempeño del código de línea B8ZS. La siguiente figura muestra la regla de substitución una vez que el código encuentra 8 ceros consecutivos. Al correr este script, le pedirá ingresar en binario a codificar (16 bits). Investigar el formato para ingresar los bits:



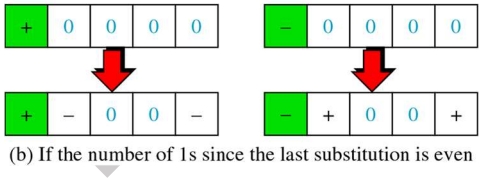

```
% File: B8ZS.m
% Descripción: Ilustración de la codificación en línea B8ZS
% Introducción a las telecomunicaciones 2020
% Tecnológico Nacional de México Campus Oaxaca;
% Departamento de Electrónica;
% DOCENTE: ING. MIGUEL ANGEL PEREZ SOLANO.

clear
bits = input('Ingrese patron binario:');
bitrate = 1;
n = 1000;
T = length(bits)/bitrate;
N = n*length(bits);
dt = T/N;
t = 0:dt:T;
x = zeros(1,length(t));
counter = 0;
lastbit = 1;
for i=1:length(bits)
  if bits(i)==0
    counter = counter + 1;
    if counter==8
      x((i-1-7)*n+1:(i-7)*n) = 0;
      x((i-1-6)*n+1:(i-6)*n) = 0;
      x((i-1-5)*n+1:(i-5)*n) = 0;
      x((i-1-4)*n+1:(i-4)*n) = lastbit;
      x((i-1-3)*n+1:(i-3)*n) = -lastbit;
      lastbit = -lastbit;
      x((i-1-2)*n+1:(i-2)*n) = 0;
      x((i-1-1)*n+1:(i-1)*n) = lastbit;
```

```
      x((i-1)*n+1:i*n) = -lastbit;
      lastbit = -lastbit;
      counter = 0;
    end
  else
    counter = 0;
    x((i-1)*n+1:i*n) = -lastbit;
    lastbit = -lastbit;
  end
end
plot(t, x, 'Linewidth', 3);
```

El siguiente código, es para generar un script y ver el desempeño del código de línea HDB3. La siguiente figura muestra la regla de substitución una vez que el código encuentra 4 ceros consecutivos. Al correr este script, le pedirá ingresar en binario a codificar (16 bits). Investigar el formato para ingresar los bits:



(a) If the number of 1s since the last substitution is odd



(b) If the number of 1s since the last substitution is even

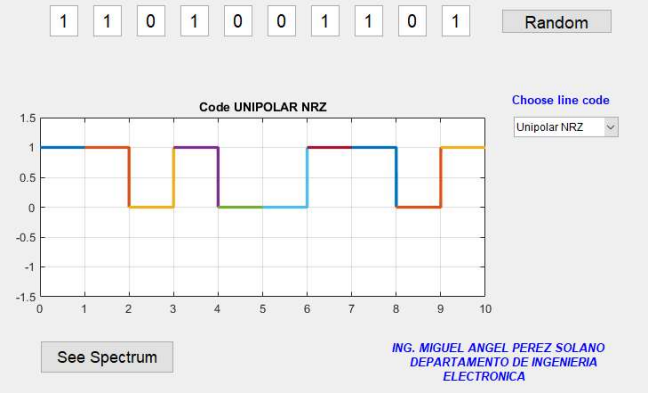
```
% File: HDB3.m
% Descripción: Ilustración de la codificación en línea HDB3.
% Introducción a las telecomunicaciones 2020
% Tecnológico Nacional de México Campus Oaxaca;
% Departamento de Electrónica;
% DOCENTE: ING. MIGUEL ANGEL PEREZ SOLANO.

clear
bits = input('Ingresar bits a codificar:');
bitrate = 1;
n = 1000;
T = length(bits)/bitrate;
N = n*length(bits);
dt = T/N;
t = 0:dt:T;
x = zeros(1,length(t));
counter = 0;
```

```
lastbit = 1;
pulse = 0;
for i=1:length(bits)
  if bits(i)==0
    counter = counter + 1;
    if counter==4
      if(mod(pulse, 2)==0)
        x((i-1-3)*n+1:(i-3)*n) = -lastbit;
        lastbit = -lastbit;
        x((i-1-2)*n+1:(i-2)*n) = 0;
        x((i-1-1)*n+1:(i-1)*n) = 0;
        x((i-1)*n+1:i*n) = lastbit;
        counter = 0;
        pulse = 0;
      else
        x((i-1-3)*n+1:(i-3)*n) = 0;
        x((i-1-2)*n+1:(i-2)*n) = 0;
        x((i-1-1)*n+1:(i-1)*n) = 0;
        x((i-1)*n+1:i*n) = lastbit;
        counter = 0;
        pulse = 0;
      end
    end
  else
    counter = 0;
    x((i-1)*n+1:i*n) = -lastbit;
    lastbit = -lastbit;
    pulse = pulse + 1;
  end
end
plot(t, x, 'Linewidth', 3);
```


El siguiente ejercicio se va a realizer de la siguiente manera:

1.- Anexo a esta práctica va una carpeta llamada line_code_GUI y contiene los siguientes archivos:

Descargarla y desde matlab abrir (open) y ejecutas el script *line_code_m*



- Con el botón *random,* se elige un *patrón binario aleatorio* y con el botón *"choose line code"* se elige el código de línea para el patrón binario. El botón "*See spectrum*" sirve para mostrar el espectro del código seleccionado. Variando el patrón bits, será la diferencia de la anchura del espectro. Saque sus conclusiones.

Es importante que se relacione la teoría con esta práctica, para obtener las competencias que se aplicarán en sus materias de especialidad.

CONCLUSIONES:

_____

_____

_____

_____

_____

PRÁCTICA No. 2 SERIE Y TRANSFORMADA RAPIDA DE FOURIER .