Taller DSP

Ricardo Urbina - A00395489

Kevin Nieto - A00395466

1. Observando las gráficas de PSD (densidad espectral de potencia) para los diferentes esquemas de códigos de línea, ¿qué código de línea tiene el mayor ancho de banda y cuál el menor? ¿Cómo se explica esto con base en la teoría?

Al observar las gráficas de densidad espectral de potencia (PSD) de los códigos de línea, es posible identificar cómo cada esquema afecta el ancho de banda, la sincronización, y la estabilidad de la señal.

Ancho de Banda
1. Código Manchester: Este esquema tiene el mayor ancho de banda. En Manchester, cada bit tiene una transición en el centro del intervalo de tiempo del bit. Debido a esta transición constante, Manchester utiliza el doble del ancho de banda en comparación con los esquemas sin tantas transiciones, como NRZ.

2. NRZ (Non-Return to Zero): Este código utiliza el menor ancho de banda. En NRZ, la señal permanece constante durante todo el tiempo de bit y solo cambia entre bits cuando el valor del bit cambia (por ejemplo, de 0 a 1 o de 1 a 0). Al no tener transiciones adicionales dentro de cada bit, este esquema requiere menos ancho de banda.

Explicación teórica: Según la teoría de la densidad espectral de potencia, los códigos de línea con más transiciones tienden a distribuir su energía a lo largo de un espectro más amplio, incrementando su ancho de banda. En Manchester, la transición en el centro de cada bit genera una mayor cantidad de componentes de frecuencia, por lo que requiere un ancho de banda mayor. En cambio, NRZ, al tener menos cambios de estado, concentra la energía en frecuencias más bajas, ocupando menos ancho de banda.

2. Describe las características de sincronización los códigos en la presencia de a una cadena larga de unos o de ceros. Por ejemplo, compara el comportamiento del código Manchester (que tiene una transición en cada bit) con el On-Off NRZ Explique ¿por qué Manchester proporciona mejor sincronización?

Características de Sincronización
La sincronización es crucial para evitar errores en la decodificación, especialmente con cadenas largas de bits iguales (como "1111" o "0000").

1. Manchester: Tiene una transición en cada bit, lo que asegura una señal de sincronización constante. Incluso si hay una cadena larga de "1"s o "0"s, cada bit tendrá una transición, y esta consistencia facilita que el receptor mantenga el tiempo y la sincronización sin necesidad de un reloj externo.

2. NRZ (On-Off NRZ): Carece de transiciones internas dentro de cada bit. Esto implica que, en presencia de una cadena larga de "1"s o "0"s, no ocurrirán transiciones y el receptor podría perder la sincronización con el flujo de datos. Esto es problemático en sistemas asíncronos, ya que el receptor depende de transiciones periódicas para ajustarse al momento en que inicia cada bit.

Manchester garantiza una transición en cada bit, lo cual actúa como un “reloj” embebido. Este comportamiento permite al receptor identificar de forma clara los límites de cada bit, facilitando una recuperación confiable de los datos incluso sin un reloj externo. En cambio, en NRZ, la falta de transiciones en cadenas de bits idénticos (por ejemplo, muchos "1"s seguidos) puede llevar a que el receptor pierda el conteo de bits, provocando errores en la sincronización y, por ende, en la interpretación de los datos.

Punto 2

Para la elección del código de línea a usar para ambos protocolos de comunicación se tomará en cuenta los siguientes criterios para tomar la decisión:

Problemas de sincronización.
Eficiencia energética.
Velocidad de transmisión de símbolos
Implicaciones del formato 8N1.

UART: Codificación NRZ On-Off

Se propone que el formato NRZ (Non-Return to Zero) es el más adecuado para la UART, dado que es un código de línea que simplifica la transmisión de estados lógicos (1 o 0) sin necesidad de cambiar la polaridad de bit a bit, como ocurre en la codificación Manchester. Esta característica permite reducir los requisitos de frecuencia, logrando así una mayor eficiencia energética para la transmisión de datos. Sin embargo, la simplicidad del NRZ puede generar problemas de sincronización en secuencias largas de bits con el mismo valor lógico. No obstante, al utilizar el formato 8N1, el cual limita las unidades de información a un byte (8 bits netos), se minimizan estos problemas de sincronización, además de que el formato incluye bits de inicio y parada que facilitan la detección de cambios.

Dado que el protocolo UART se utiliza normalmente en distancias cortas, con poco ruido y baja interferencia electromagnética (EMI), no resulta necesario optar por un formato de codificación bipolar. Aumentar la distancia entre los cambios lógicos para mejorar la

resistencia al ruido sería innecesario en este contexto, considerando el propósito del protocolo UART.

Un factor adicional a considerar es la velocidad de transmisión de símbolos o baudios. En este caso, tanto la UART como el RS-232 se configuran para una tasa de 9600 baudios, lo que equivale a una tasa de transmisión de 9600 bps, dado que cada símbolo representa un bit. Esta velocidad relativamente baja reduce aún más las posibles implicaciones de sincronización, reforzando la conveniencia de usar un código de línea simple como NRZ On-Off.

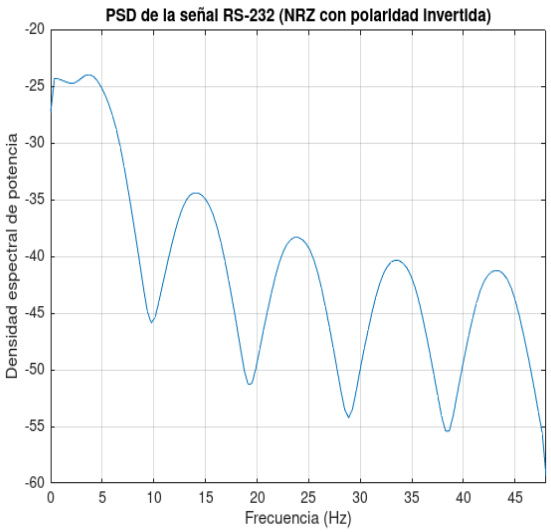
RS-232: Codificación NRZ Polar

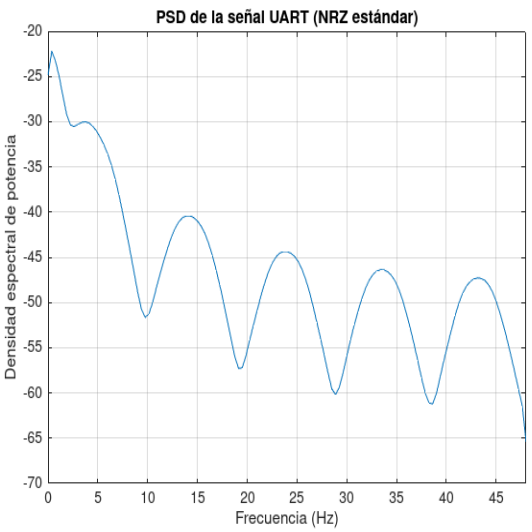
Como se mencionó anteriormente, el formato NRZ presenta una mayor eficiencia energética en comparación con el Manchester y otros códigos de línea que requieren más transiciones por cambio de valor lógico. Al configurarse a 9600 bps y con el formato 8N1, no presenta inconvenientes de sincronización.

¿Por qué, entonces, se utiliza NRZ en su modalidad polar para RS-232? Este protocolo se emplea en aplicaciones que requieren transmisiones a mayores distancias y en entornos con mayor nivel de ruido. Usar una codificación polar en el RS-232 implica una mayor separación entre los niveles lógicos, lo que dificulta que el ruido altere el valor transmitido. Además, el formato bipolar se adapta de manera natural al funcionamiento del RS-232, que es inherentemente bipolar, ofreciendo una mayor robustez frente a interferencias en aplicaciones de mayor alcance.

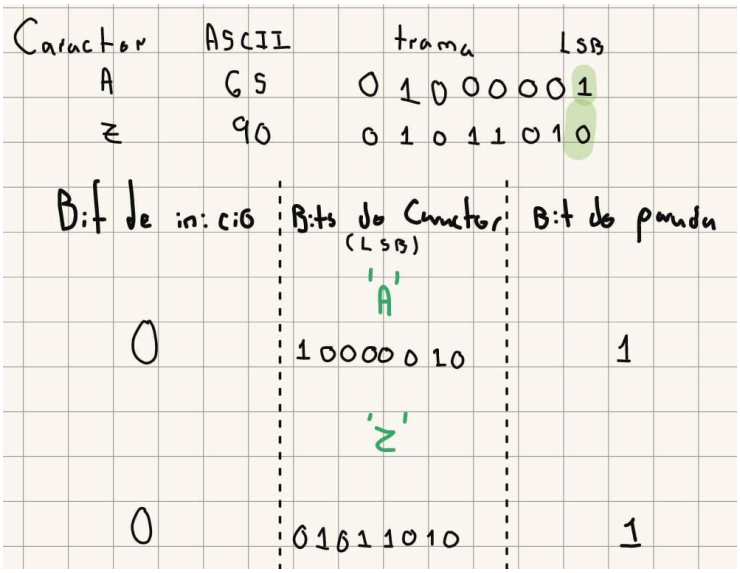
Esta elección de codificación permite que el protocolo RS-232 mantenga su integridad en condiciones adversas, aprovechando al máximo sus características para una comunicación fiable.

Gráficas PSD para UART y RS-232 ambas con 9600 baudios 8N1

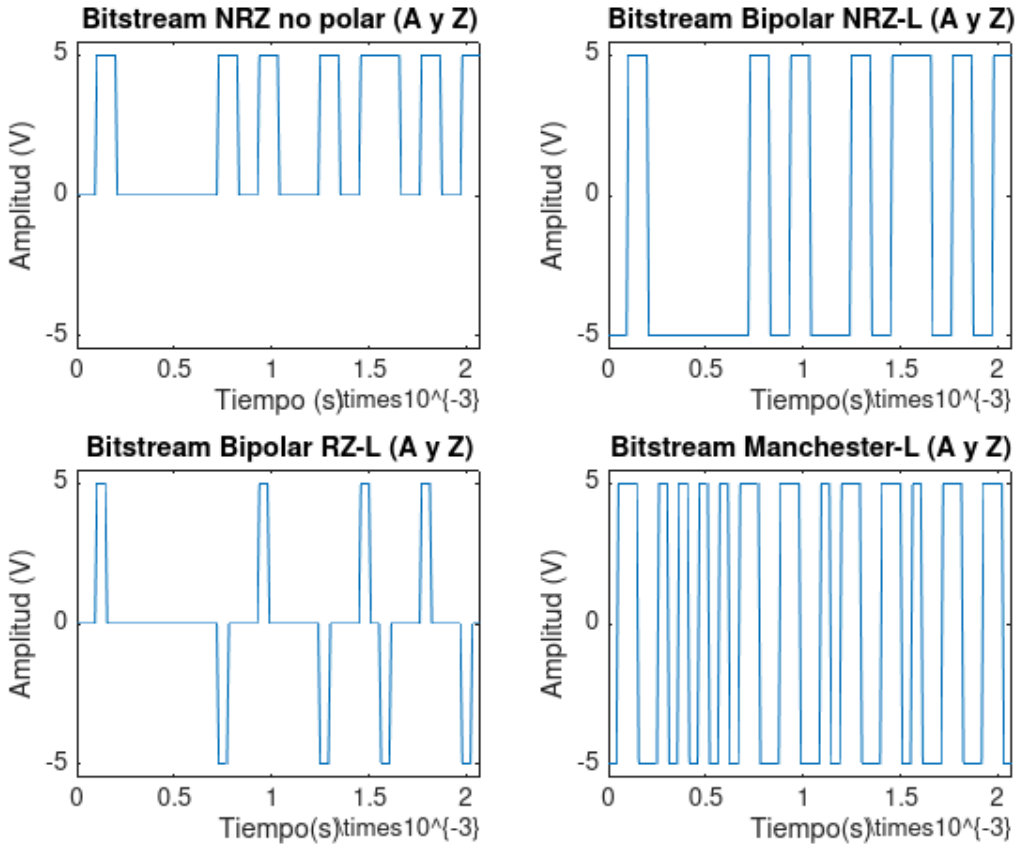




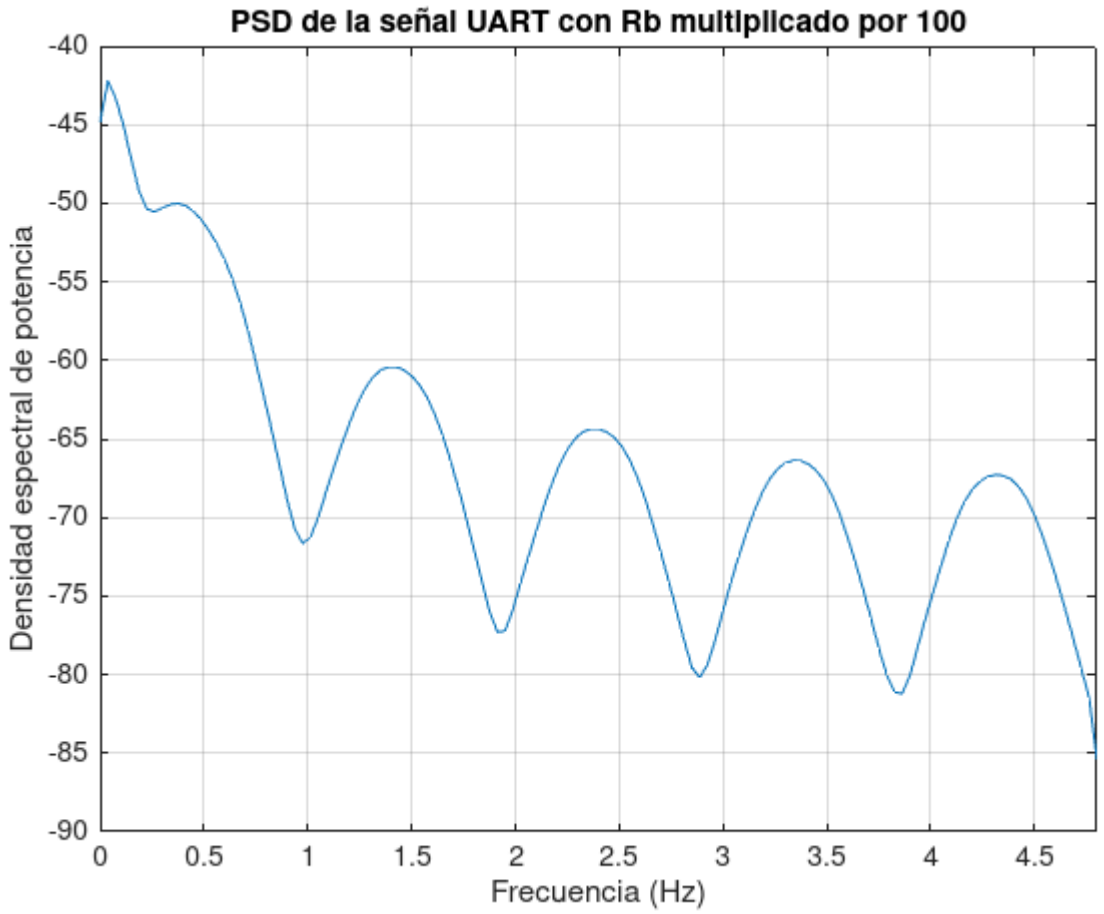
Se procede a analizar cómo estaría compuesta la trama para el carácter ‘A‘ y ‘Z’, teniendo en cuenta que tanto UART como RS-232 trabajan con el bit menos significativo.



Gráficas de bitstream para cada código de línea



Aumento 1:100 de la tasa de bits Rb -> Gráfica de PSD



Al incrementar la tasa de bits, los bits se transmiten más rápido, lo que significa que la duración de cada bit (**Tb = 1/Rb**) disminuye. Una señal con cambios más rápidos (menor Tb) tiene componentes de frecuencia más altas. Esto se debe a que las transiciones más rápidas en el tiempo requieren frecuencias más altas para ser representadas, todo esto conlleva a que el ancho de banda requerido para transmitir la señal aumente proporcionalmente con Rb.

Código:

```
clear all;
close all;
%Integrantes:
% Kevin Steven Nieto Curaca - A00395466
% Ricardo Urbina Ospina - A00395489
% Número de caracteres a transmitir
Nb = 100;
% Generación de 100 caracteres ASCII aleatorios (del 32 al 126)
data = randi([32, 126], 1, Nb);
% Parámetros de transmisión
Rb = 9600; % Tasa de bits (baudios)
fs = 10 * Rb; % Frecuencia de muestreo
Vp = 5; % Voltaje pico
% Inicialización del bitstream
bitstream = [];
% Generación del bitstream con bits de inicio y parada
for i = 1:Nb
  % Conversión del carácter a bits (8 bits)
  char_bits = de2bi(data(i), 8, 'right-msb');
  % Agregar bits de inicio (0) y parada (1)
  frame = [0 char_bits 1];
  bitstream = [bitstream frame];
end
% Codificación NRZ con polaridad invertida (RS-232)
NRZ_RS232 = [];
for bit = bitstream
  if bit == 1
     NRZ_RS232 = [NRZ_RS232 ones(1, fs/Rb)*(-Vp)]; % Lógico '1' es voltaje negativo
  else
     NRZ_RS232 = [NRZ_RS232 ones(1, fs/Rb)*(Vp)];  % Lógico '0' es voltaje positivo
  end
end
% Cálculo de la PSD usando Welch
h = spectrum.welch;
Hpsd = psd(h, NRZ_RS232, 'Fs', fs);
figure;
plot(Hpsd);
title('PSD de la señal RS-232 (NRZ con polaridad invertida)');
xlabel('Frecuencia (Hz)');
ylabel('Densidad espectral de potencia');
% Codificación NRZ estándar para UART
NRZ_UART = [];
for bit = bitstream
  if bit == 1
     NRZ_UART = [NRZ_UART ones(1, fs/Rb)*(Vp)]; % Lógico '1' es voltaje positivo
```

```
  else
    NRZ_UART = [NRZ_UART ones(1, fs/Rb)*0];    % Lógico '0' es voltaje cero
  end
end
% Cálculo de la PSD usando Welch
h = spectrum.welch;
Hpsd = psd(h, NRZ_UART, 'Fs', fs);
figure;
plot(Hpsd);
title('PSD de la señal UART (NRZ estándar)');
xlabel('Frecuencia (Hz)');
ylabel('Densidad espectral de potencia');
%%
% Caracteres a transmitir
chars = ['A', 'Z'];
data_chars = double(chars);
% Inicialización del bitstream
bitstream_chars = [];
A = de2bi(data_chars(1), 8, 'right-msb');
Z = de2bi(data_chars(2), 8, 'right-msb');
disp(A);
disp(Z);
% Generación del bitstream con bits de inicio y parada
for i = 1:length(data_chars)
  char_bits = de2bi(data_chars(i), 8, 'right-msb');
  frame = [0 char_bits 1];
  bitstream_chars = [bitstream_chars frame];
end
NRZno_polar = []; % Bipolar NRZ
for bit = bitstream_chars
  if bit == 1
    NRZno_polar = [NRZno_polar ones(1, fs/Rb)*(Vp)];
  else
    NRZno_polar = [NRZno_polar ones(1, fs/Rb)*0];
  end
end
NRZ_out = [];
for bit = bitstream_chars
  if bit == 1
    NRZ_out = [NRZ_out ones(1, fs/Rb)*(Vp)]; % '1' es +Vp
  else
    NRZ_out = [NRZ_out ones(1, fs/Rb)*(-Vp)]; % '0' es -Vp
  end
end
RZ_out = []; % Bipolar RZ
current_level = Vp;
for bit = bitstream_chars
  if bit == 1
```

```
        pulse = [ones(1, fs/(2*Rb))*(current_level) zeros(1, fs/(2*Rb))];
        RZ_out = [RZ_out pulse];
        current_level = -current_level;
    else
        RZ_out = [RZ_out zeros(1, fs/Rb)];
    end
end
Manchester_out = []; %Machebster
for bit = bitstream_chars
    if bit == 1
        symbol = [ones(1, fs/(2*Rb))*(Vp) ones(1, fs/(2*Rb))*(-Vp)];
    else
        symbol = [ones(1, fs/(2*Rb))*(-Vp) ones(1, fs/(2*Rb))*(Vp)];
    end
    Manchester_out = [Manchester_out symbol];
end
t = (0:length(NRZno_polar)-1)/fs;
% Crear una nueva figura
figure;
% NRZ no polar
subplot(2,2,1); % Posición 1 en una rejilla de 2x2
plot(t, NRZno_polar);
title('Bitstream NRZ no polar (A y Z)');
xlabel('Tiempo (s)');
ylabel('Amplitud (V)');
ylim([-Vp*1.1 Vp*1.1]);
% Bipolar NRZ-L
subplot(2,2,2); % Posición 2 en una rejilla de 2x2
plot(t, NRZ_out);
title('Bitstream Bipolar NRZ-L (A y Z)');
xlabel('Tiempo(s)');
ylabel('Amplitud (V)');
ylim([-Vp*1.1 Vp*1.1]);
% Bipolar RZ-L
subplot(2,2,3); % Posición 3 en una rejilla de 2x2
plot(t, RZ_out);
title('Bitstream Bipolar RZ-L (A y Z)');
xlabel('Tiempo(s)');
ylabel('Amplitud (V)');
ylim([-Vp*1.1 Vp*1.1]);
% Manchester-L
subplot(2,2,4); % Posición 4 en una rejilla de 2x2
plot(t, Manchester_out);
title('Bitstream Manchester-L (A y Z)');
xlabel('Tiempo(s)');
ylabel('Amplitud (V)');
ylim([-Vp*1.1 Vp*1.1]);
%%
```

```
% Nuevo Rb multiplicado por 100
Rb_high = Rb * 100;
fs_high = 10 * Rb_high;
% Recalculamos NRZ_UART con el nuevo Rb
NRZ_UART_high = [];
for bit = bitstream
  if bit == 1
     NRZ_UART_high = [NRZ_UART_high ones(1, fs_high/Rb_high)*(Vp)];
  else
     NRZ_UART_high = [NRZ_UART_high ones(1, fs_high/Rb_high)*0];
  end
end
% Cálculo de la PSD con el nuevo Rb
h = spectrum.welch;
Hpsd_high = psd(h, NRZ_UART_high, 'Fs', fs_high);
figure;
plot(Hpsd_high);
title('PSD de la señal UART con Rb multiplicado por 100');
xlabel('Frecuencia (Hz)');
ylabel('Densidad espectral de potencia');
%%
```