

## Contenido

Laboratorio 1: señales y espectros	6
Breve introducción teórica del tema	6
Figura 1.1: señales analógicas y digitales	6
Ondas sinusoidales	6
Figura 1.2: Onda Sinusoidal.	7
Descripción detallada de los programas desarrollados con impresiones de las pantallas	8
Figura 1.3: porción del programa para graficar los parámetros recibidos	8
Figura 1.4: ejemplo de gráfico de seno con parámetros introducidos por el usuario	9
Figura 1.5: ejemplo de gráfico de seno con parámetros introducidos por el usuario	9
Programa Matlab para graficar la aproximación a una señal cuadrada mediante suma de señales senoidales.	10
Figura 1.6: porción del programa de aproximación a una señal cuadrado	10
Figura 1.7: gráfico de señal con armónicos	10
Figura 1.8: gráfico de señal con armónicos	11
Desarrollo de guía “señales y espectros”	11
SEÑAL ESCALÓN:	11
Figura 1.9: señal escalón	11
SEÑAL PULSO:	12
Figura 1.10: señal pulso	12
SEÑAL SAMPLING:	12
Figura 1.11: señal sampling	12
SEÑAL IMPULSO:	13
Figura 1.12: señal impulso	13
SEÑAL TRIANGULAR:	13
Figura 1.13: señal triangular	13
SEÑAL DIENTE DE SIERRA DE PERIODO:	14
Figura 1.14: señal diente de sierra de periodo	14
SEÑAL EXPONENCIAL:	14

Figura 1.15: señal exponencial	15
SEÑAL CUADRADA:	15
Figura 1.16: señal cuadrada	15
ANÁLISIS DE UNA SEÑAL:	16
Figura 1.17: resultado del código	17
Figura 1.18: resultado del código	17
Conclusión	17
Laboratorio 2: transmisión de datos	18
Breve introducción teórica del tema	18
Capacidad del canal según Nyquist	18
Capacidad del canal	18
Capacidad del canal según Shannon	18
Descripción detallada de los programas desarrollados con impresiones de las pantallas.	19
Figura 2.1: cómo ejecutar código fuente	21
Figura 2.2: resultado de los parámetros ingresados	21
Desarrollo de guía “transmisión de datos”	21
Conclusión:	22
Laboratorio 3: codificación y modulación	23
Breve introducción teórica del tema Codificación y Modulación	23
Figura 3.1: gráfico de codificación y modulación	23
Técnicas de codificación digital	23
Tabla 3.1: técnicas de codificación digital	24
Figura 3.2: técnicas de modulación para la cadena de bits 01001100011	24
Descripción detallada de los programas desarrollados con impresiones de las pantallas.	24
Figura 3.3: codificación NRZ-L con la cadena de bits 01001110111	25
Figura 3.4: codificación NRZI con la cadena de bits 00110011000	26
Figura 3.5: codificación Bipolar-AMI con la cadena de bits 00110011000	26
Figura 3.6: codificación Pseudoternario con la cadena de bits 01001110011	27
Figura 3.7: codificación Manchester con la cadena de bits 10101010111	27

Figura 3.8: codificación Manchester Diferencial con la cadena de bits 01011010100	28
Desarrollo de guía “codificación y modulación”	36
Figura 3.9: simulación con la señal seno con 32 niveles	37
Figura 3.10: archivo excel generado en la simulación con 32 niveles.	37
Figura 3.11: archivo excel generado en la simulación con 8 niveles	38
Conclusión	39
Laboratorio 4: medios de transmisión	40
Breve introducción teórica del tema	40
Ganancia de una antena	40
Figura 4.1: partes de una antena	40
Distancia máxima entre dos antenas	41
Propagación en la trayectoria visual	41
Figura 4.2: propagación en la trayectoria visual (por encima de los 30MHz).	41
Refracción	41
Línea de visión óptica y de radio.	42
Figura 4.3: Horizonte óptico y de radio.	43
Descripción detallada de los programas desarrollados con impresiones de las pantallas	43
Tabla 4.1: resultados del desarrollo	43
Figura 4.4: ganancia de una Antena de 2mts de diámetro y 12GHz de frecuencia de trabajo.	44
Figura 4.5: ganancia de una Antena de 5mts de diámetro y 12GHz de frecuencia de trabajo	44
Figura 4.6: Distancia máxima entre dos antenas, ambas de 54 metros de altura.	45
Figura 4.7: Distancia máxima entre dos antenas, ambas de 75 metros de altura.	45
Conclusión	57
Laboratorio 5: Instalación de Linux CentOS (Servidor)	59
Breve introducción teórica del tema	59
Entrando a detalle	59
Algunas de sus características principales de CentOS son:	59

Instalación de una máquina virtual	60
Primera etapa: crear máquina virtual en Virtual Box	60
Figura 5.1: añadir nueva máquina	60
Figura 5.2: añadir nueva máquina	60
Figura 5.3: agregar nombre de usuario y de máquina	61
Figura 5.4: asignar memoria y procesador a la máquina virtual	61
Figura 5.5: asignar espacio en disco duro	61
Figura 5.6: terminar de crear máquina virtual	62
Figura 5.7: inicializar máquina virtual	62
Segunda etapa: instalación	63
Figura 5.8: instalar CentOS 7	63
Figura 5.9: elegir idioma de la máquina	63
Figura 5.10: realizar configuraciones iniciales	64
Figura 5.11: determinar contraseña del usuario ROOT	64
Figura 5.12: finalizar la instalación de CentOS 7	65
Figura 5.13: máquina inicializada	65
Conclusión	66

# Laboratorio 1: señales y espectros

## Breve introducción teórica del tema

Los diferentes tipos de información, como la voz, los datos, las imágenes y los videos, pueden representarse mediante señales electromagnéticas. Cualquier señal electromagnética, ya sea analógica o digital, se compone de una serie de frecuencias específicas. Un parámetro esencial para describir una señal es su ancho de banda, que indica el rango de frecuencias presentes en la señal. Por lo general, un ancho de banda mayor permite una mayor capacidad para transportar información. Las líneas de transmisión pueden tener problemas o defectos que afectan a las señales analógicas o digitales utilizadas en la comunicación. El éxito en la transmisión de datos depende principalmente de la calidad de la señal transmitida y de las características del medio de transmisión.

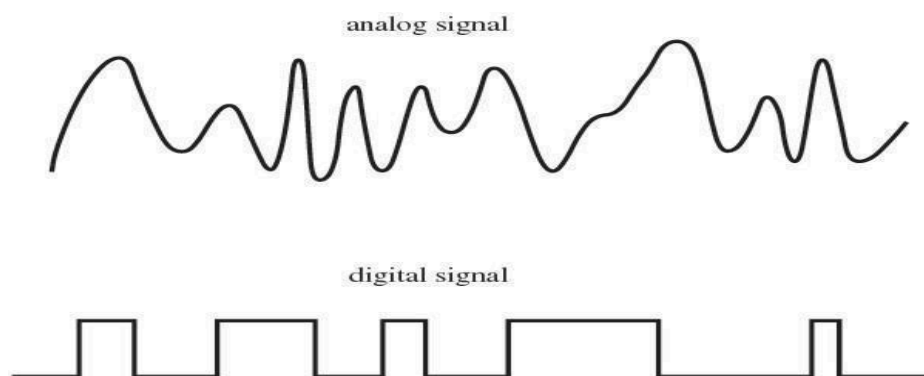


Figura 1.1: señales analógicas y digitales

## Ondas sinusoidales

Las ondas más básicas son las sinusoidales, aquellas cuyos valores cambian con el tiempo y/o el espacio de manera similar a las funciones seno o coseno. Cada una de estas ondas se define por su frecuencia  $f$ , que indica cuántas veces se repite el movimiento en un período de tiempo determinado y se mide en Hertz (Hz), o ciclos por segundo. También se puede describir por su período  $T = 1 / f$ , que es el tiempo que tarda en repetirse y se mide en segundos, su amplitud  $A$ , que es el valor máximo que alcanza y se mide en unidades de longitud (usualmente en micras o centímetros), y su fase, que indica en qué punto del ciclo se encuentra la onda en un tiempo o lugar de referencia específico. Amplitud de onda: el valor máximo que alcanza una corriente eléctrica. También se conoce como valor de pico o valor de cresta.

**Período:** es el tiempo en segundos durante el cual la corriente repite su valor. Este es el intervalo que separa dos puntos consecutivos con el mismo valor en la onda

sinusoidal. El período es el inverso de la frecuencia y se puede expresar matemáticamente con la fórmula:  $T = 1 / F$ .

Como se mencionó anteriormente, la frecuencia es simplemente el número de ciclos por segundo, medido en Hertz (Hz), que la corriente alterna alcanza. Es el inverso del período y se puede representar matemáticamente de la siguiente manera:  $F = 1 / T$ .

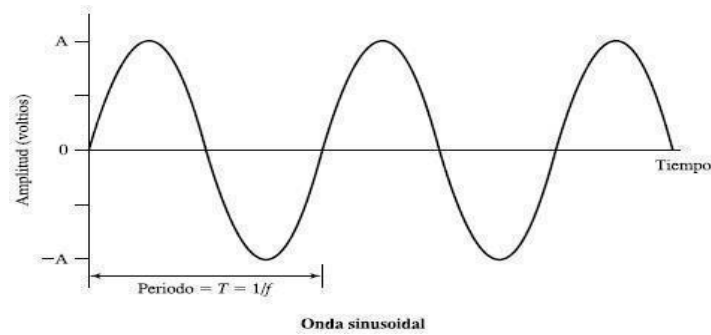
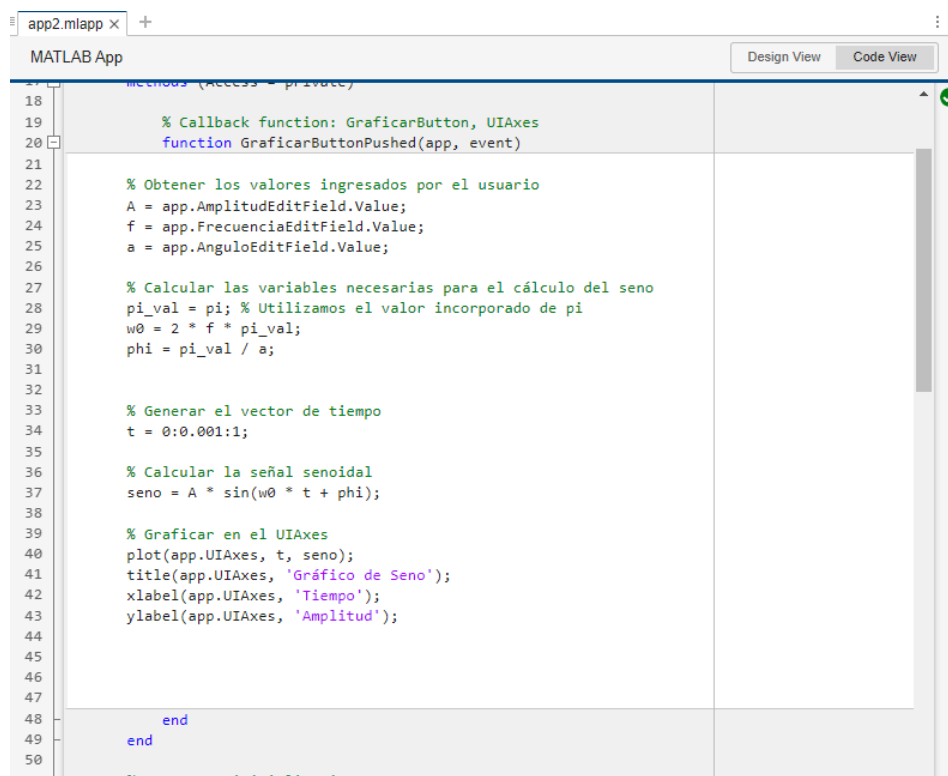


Figura 1.2: Onda Sinusoidal.

## Descripción detallada de los programas desarrollados con impresiones de las pantallas

En la primera parte se procede a programar en el software MatLab, un pequeño código con interface gráfica, donde se solicita al usuario mediante los elementos EditField que ingrese los valores de amplitud, frecuencia y fase. Esta pequeña aplicación cuenta con un botón cuyo código en el ButtonPushed es el siguiente para poder graficar: Aquí se observa el código que se introduce en el evento GraficarButtonPushed donde se recibe los parámetros a través de editfields o textinput y se los guarda en las variables para su cálculo posterior. Por último, se realizan los cálculos pertinentes, y en la última parte del código se procede a graficar. Cabe mencionar que para el gráfico se usó un elemento llamado UIAxes que se agregó al aplicativo.



```
18 % Callback function: GraficarButton, UIAxes
19 function GraficarButtonPushed(app, event)
20
21 % Obtener los valores ingresados por el usuario
22 A = app.AmplitudEditField.Value;
23 f = app.FrecuenciaEditField.Value;
24 a = app.AnguloEditField.Value;
25
26 % Calcular las variables necesarias para el cálculo del seno
27 pi_val = pi; % Utilizamos el valor incorporado de pi
28 w0 = 2 * f * pi_val;
29 phi = pi_val / a;
30
31 % Generar el vector de tiempo
32 t = 0:0.001:1;
33
34 % Calcular la señal senoidal
35 seno = A * sin(w0 * t + phi);
36
37 % Graficar en el UIAxes
38 plot(app.UIAxes, t, seno);
39 title(app.UIAxes, 'Gráfico de Seno');
40 xlabel(app.UIAxes, 'Tiempo');
41 ylabel(app.UIAxes, 'Amplitud');
42
43 end
44
45 end
46
47
48
49
50
```

Figura 1.3: porción del programa para graficar los parámetros recibidos

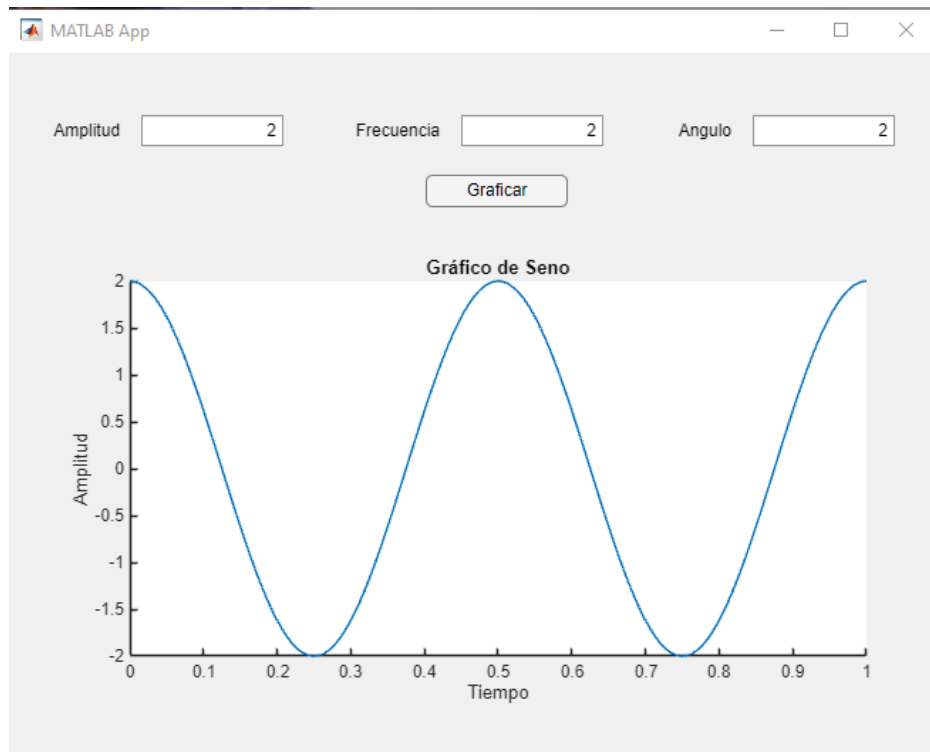


Figura 1.4: ejemplo de gráfico de seno con parámetros introducidos por el usuario

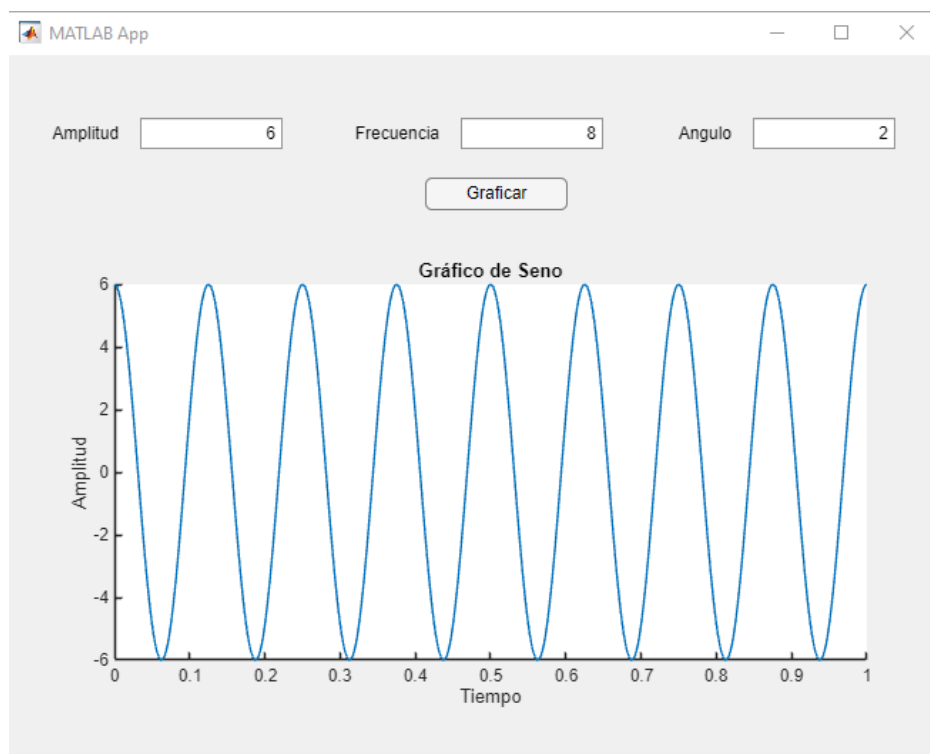


Figura 1.5: ejemplo de gráfico de seno con parámetros introducidos por el usuario



## Programa Matlab para graficar la aproximación a una señal cuadrada mediante suma de señales senoidales.

El único parámetro que solicita al usuario es la cantidad de componentes de la sumatoria.

En esta segunda parte se realizó una pequeña modificación a la aplicación para que además de los parámetros anteriormente comentados, se incluye la cantidad de señales. Aquí mostramos el código con algunos ejemplos de su visualización gráfica.

```
app_1.mlapp x | +
MATLAB App
Design View Code View
21 % Button pushed function: GraficarButton
22 function GraficarButtonPushed(app, event)
23
24 % Leer valores de entrada desde los campos de texto
25 frequency = app.FrecuenciaEditField.Value;
26 amplitude = app.AmplitudEditField.Value;
27 phase = app.FaseEditField.Value;
28 niveles = app.CantidadsealesEditField.Value;
29
30 % Definir el número de puntos (simulando el ancho del canvas)
31 canvas_width = 1000; % Número de puntos en el eje x
32
33 % Inicializar el vector de datos
34 x = linspace(0, pi, canvas_width);
35 y = zeros(1, canvas_width);
36
37 % Calcular el valor de 'k'
38 k = niveles * 2;
39
40 % Calcular la señal
41 for i = 1:canvas_width
42     xi = x(i);
43     for j = 1:k
44         if mod(j, 2) ~= 0
45             y(i) = y(i) + amplitude * (1 / j) * sin(2 * pi * frequency * j * xi + phase);
46         end
47     end
48 end
49
50 % Graficar la señal
51 plot(app.UIAxes, x, y);
52 xlabel(app.UIAxes, 'x');
53 ylabel(app.UIAxes, 'y');
54 title(app.UIAxes, 'Señal con armónicos');
55
56
57
```

Figura 1.6: porción del programa de aproximación a una señal cuadrado

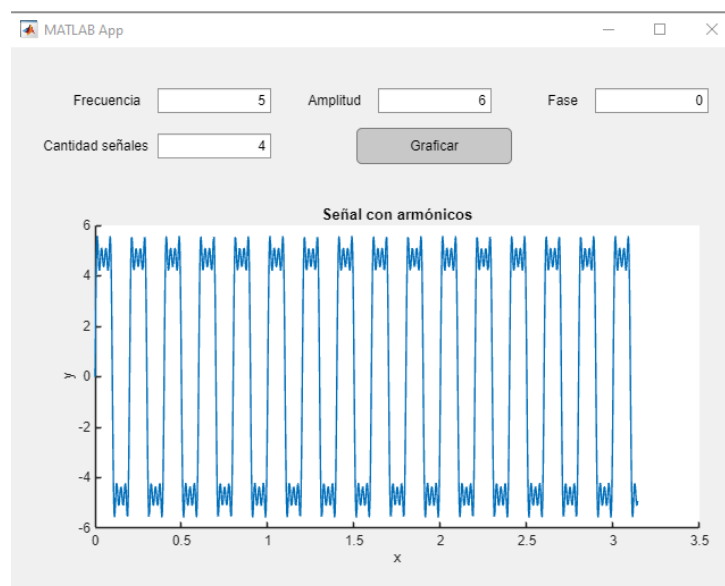


Figura 1.7: gráfico de señal con armónicos

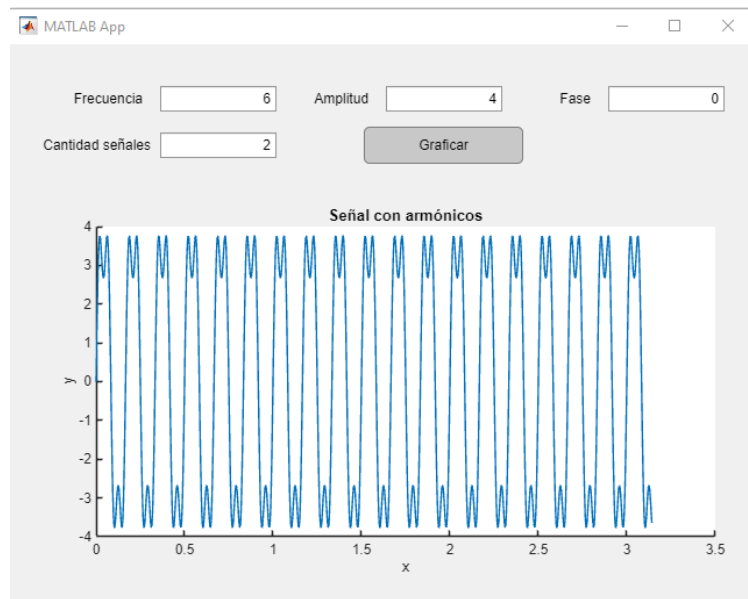


Figura 1.8: gráfico de señal con armónicos

## Desarrollo de guía “señales y espectros”

Señales analógicas típicas:

### SEÑAL ESCALÓN:

% Ejemplo de señal escalón

```
t=-10:0.01:10;
```

```
f_escalon=[zeros(1,1000),ones(1,1001)];
```

```
plot(t,f_escalon);
```

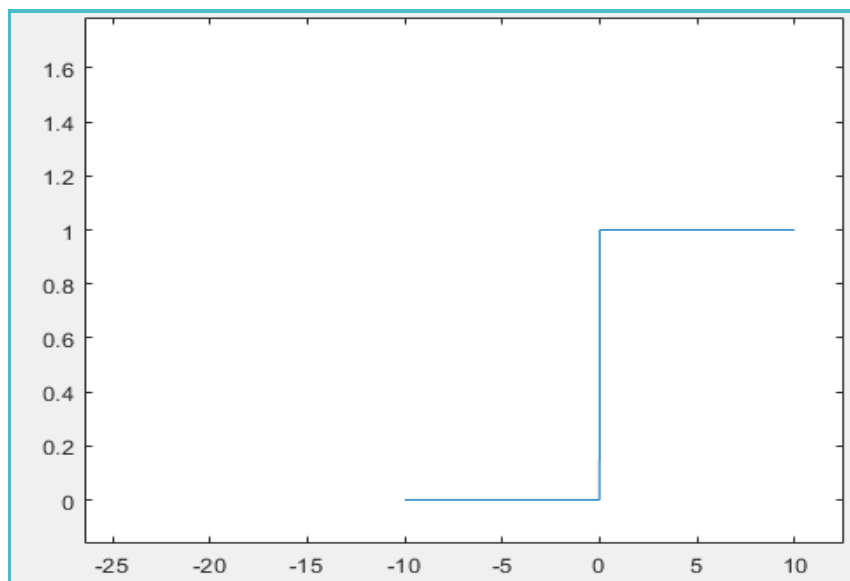


Figura 1.9: señal escalón

## SEÑAL PULSO:

% Ejemplo de señal pulso

```
t=-10:0.01:10;
```

```
f_pulso=[zeros(1,950),ones(1,101),zeros(1,950)];
```

```
plot(t,f_pulso);
```

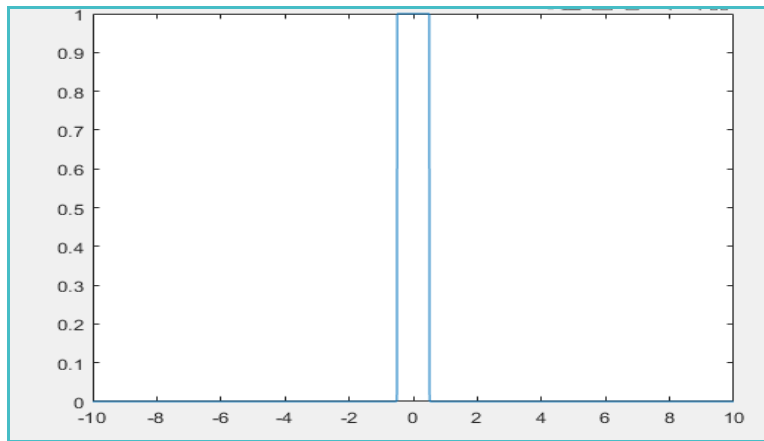


Figura 1.10: señal pulso

## SEÑAL SAMPLING:

% Ejemplo de señal sampling

```
t=-10:0.01:10;
```

% Señal sampling nula en  $t=n\pi$ ,  $n=1,2,\dots$

```
f_sampling=sin(t)./t;
```

```
plot(t,f_sampling);
```

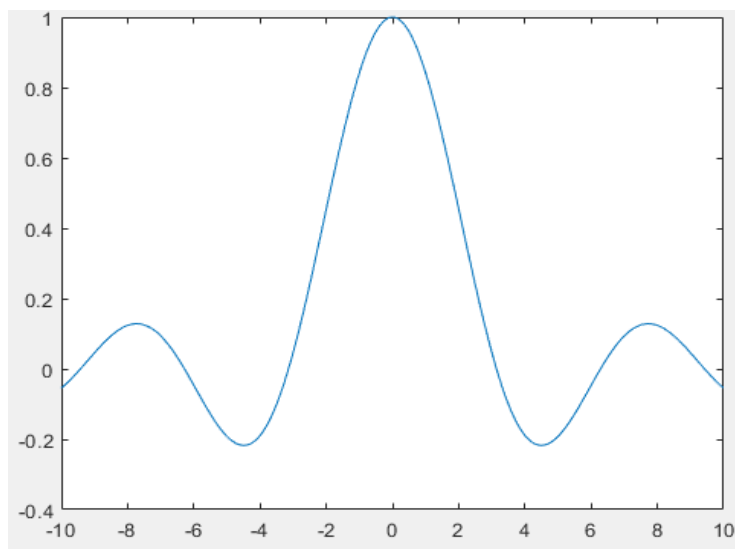


Figura 1.11: señal sampling

## SEÑAL IMPULSO:

% Ejemplo de señal impulso

```
t=-10:0.01:10;
```

```
f_impulso=[zeros(1,1000),1,zeros(1,1000)];
```

```
plot(t,f_impulso);
```

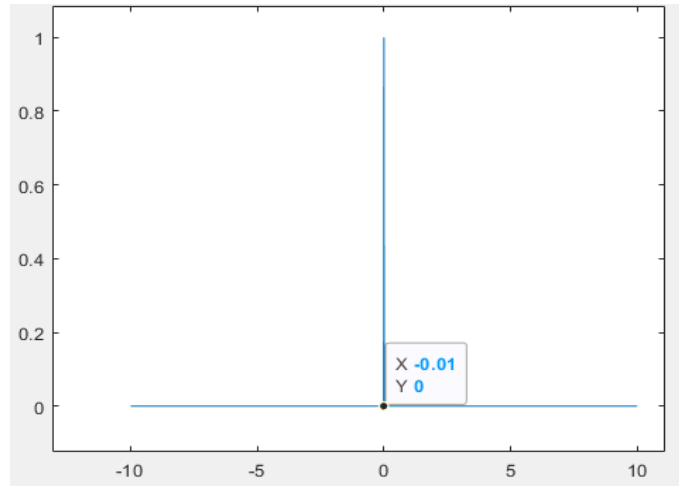


Figura 1.12: señal impulso

## SEÑAL TRIANGULAR:

% Ejemplo de señal triangular de periodo 0.1Hz

% Es un caso particular de señal diente de sierra con width=0.5

```
t=-10:0.01:10;
```

```
f_triangular=sawtooth(2*pi*0.1*t,0.5);
```

```
plot(t,f_triangular);
```

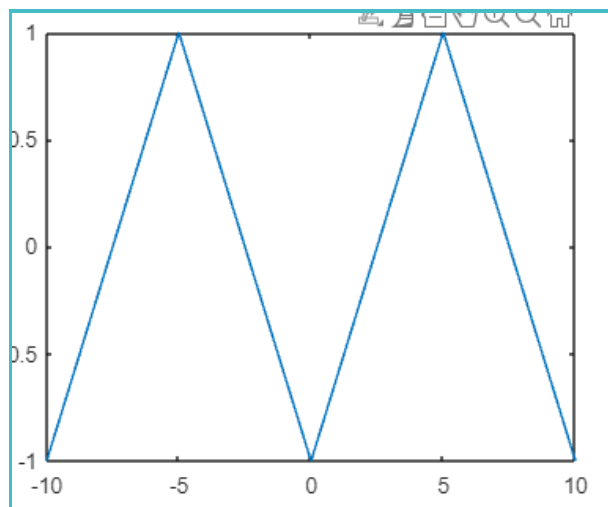


Figura 1.13: señal triangular

## SEÑAL DIENTE DE SIERRA DE PERIODO:

```
% Ejemplo de señal diente de sierra de periodo  
  
%0.1Hz sawtooth(x,width) señal en diente de sierra  
  
%con periodo  $2\pi$  para los elementos del vector x.  
  
%El parámetro "width" es un escalar entre 0 y 1, y  
  
%describe la fracción del periodo  $2\pi$  en el que  
  
%ocurre el máximo.  
  
t=-10:0.01:10;  
  
width=0.10;  
  
f_sierra=sawtooth(2*pi*0.1*t,width);  
  
plot(t,f_sierra);
```

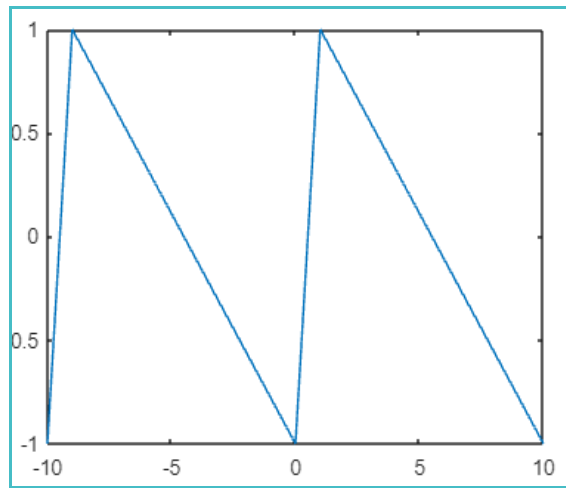


Figura 1.14: señal diente de sierra de periodo

## SEÑAL EXPONENCIAL:

```
% Ejemplo de señal exponencial decreciente  
  
t=-10:0.01:10;  
  
% tau: constante de tiempo (RC)  
  
tau=200e-2;  
  
f_expon=exp(-t/tau);  
  
plot(t,f_expon);
```

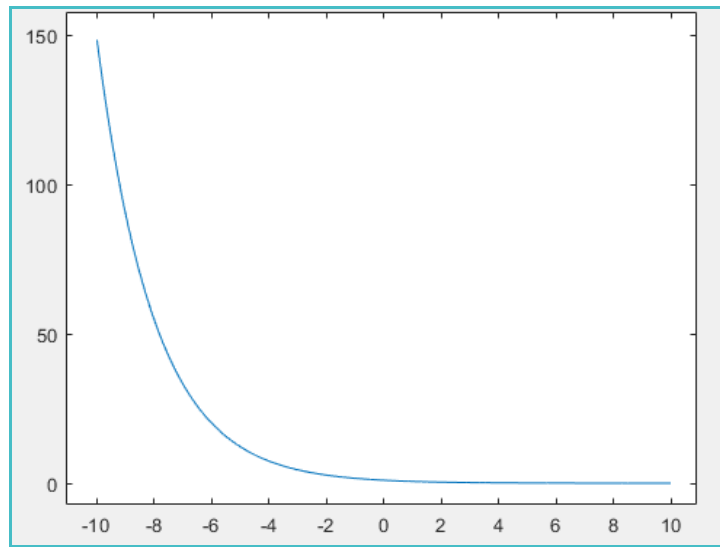


Figura 1.15: señal exponencial

### SEÑAL CUADRADA:

```
% Ejemplo de señal cuadrada de frecuencia 0.5Hz

% square(x,duty) genera una onda cuadrada de periodo 2*pi con un duty cycle dado

t=-10:0.01:10;

duty=50; % porcentaje del periodo en el que la señal es positiva

f_cuadrada=square(2*pi*0.5*t,duty);

plot(t,f_cuadrada);
```

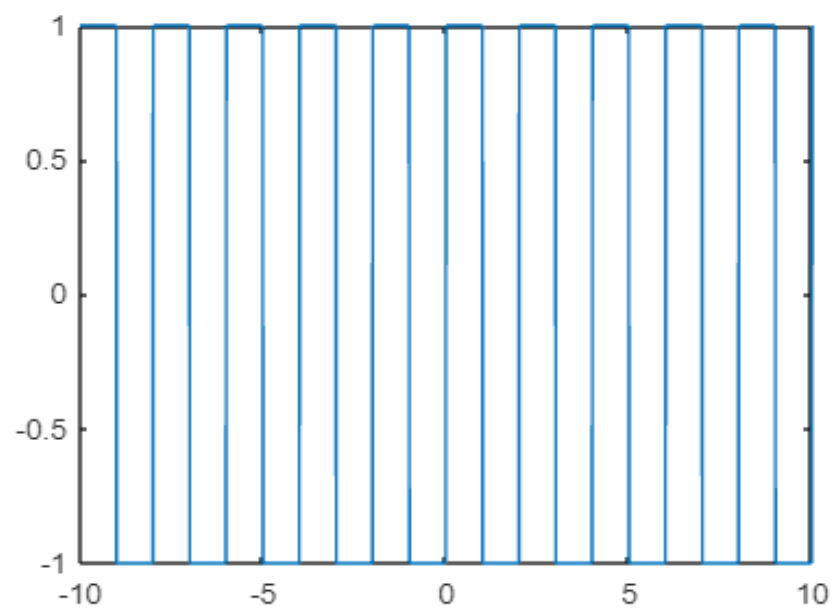


Figura 1.16: señal cuadrada

## ANÁLISIS DE UNA SEÑAL:

A una señal formada por la suma de dos senoides de distinta frecuencia y afectarla de un ruido blanco, se observa el espectro para identificar las principales frecuencias intervinientes.

### Script:

```
clear all

Frec1=input('Frecuencia de la 1o senoidal? (Hz.)');

Frec2=input('Frecuencia de la 2o senoidal? (Hz.)');

tiempo=input('Tiempo de duración de la señal? (seg.)');

fm=input('frecuencia de muestreo? (Hz.)');

tm=1/fm; %tiempo de muestra

N=floor(tiempo/tm);

t = 0:tm:(N-1)*tm; %vector de tiempo muestreado a una frecuencia de fm Hz.

x = sin(2*pi*Frec1*t)+sin(2*pi*Frec2*t); %Señal pura

ruido=2*randn(size(t));

figure(1)

subplot(121)

plot(t(1:N/10),x(1:N/10))

title('Señal pura')

subplot(122)

plot(t(1:N/10),ruido(1:N/10))

title('Señal ruido')

y = x + ruido;

figure(2)

plot(t(1:N/10),y(1:N/10))

title('Señal afectada del ruido')

xlabel('tiempo (milisegundos)')

Y=abs(fft(y,500)); % Transformada de Fourier analizada para 500 puntos de la señal

f = fm*(0:250)/500; % vector de frecuencias para el gráfico del espectro

figure(3)
```

```

plot(f,Y(1:251))

title('Espectro de frecuencias')

xlabel('frecuencia (Hz)')

```

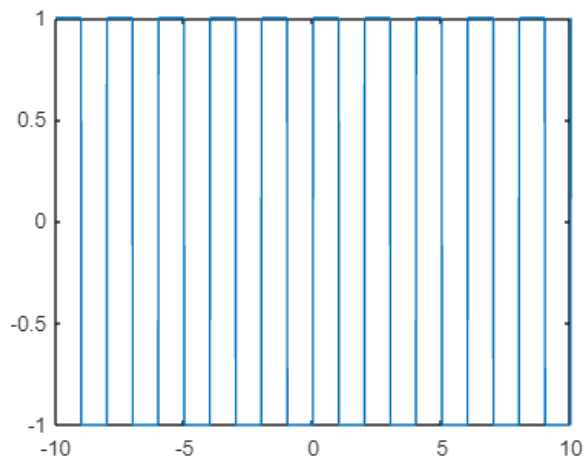


Figura 1.17: resultado del código

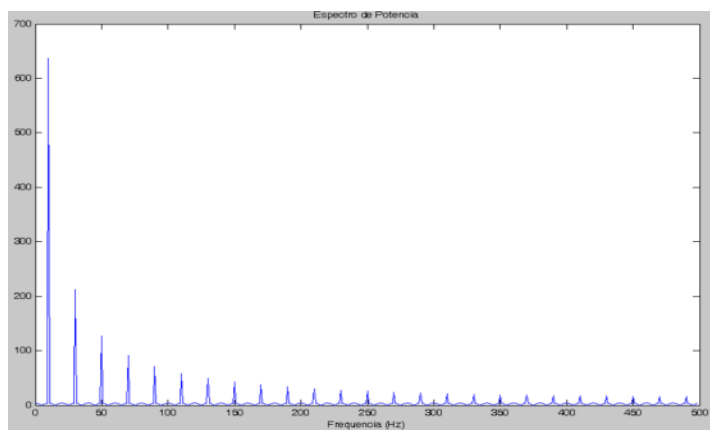


Figura 1.18: resultado del código

## Conclusión

Según los valores ingresados para graficar las ondas sinusoidales, observamos que al aumentar su amplitud se visualizan ondas cada vez más amplias. A medida que incrementamos su frecuencia, observamos una mayor cantidad de ondas en un mismo intervalo de tiempo. Además, al aumentar la fase, la onda se origina en ángulos cada vez mayores.

En el caso de las ondas cuadradas, la cantidad de componentes determina la forma de la onda. A medida que aumenta la cantidad de componentes o armónicos, la onda presenta más "picos" y "valles" en un mismo intervalo de tiempo, y los cambios



abruptos ocurren siempre en el mismo intervalo. Cuanto mayor es la cantidad de componentes, más se asemeja la onda a una forma cuadrada.

## Laboratorio 2: transmisión de datos

### Breve introducción teórica del tema

#### Capacidad del canal según Nyquist

En su teoría, Nyquist creó un canal que no tiene ruido. Por lo tanto, la velocidad de transmisión del canal depende del ancho de banda del canal.

En 1927, determinaron que la velocidad máxima de transmisión en bits por segundo para un canal con ancho de banda  $W$  (Hz) es, según la fórmula de Nyquist:

$$C = 2 * W * \log_2(M)$$

Donde:

- $C$  es la capacidad de transmisión del canal en BPS.
- $W$  es el ancho de banda del canal en Hz
- $M$  son los niveles de la señal.

#### Capacidad del canal

Llamamos capacidad del canal a la velocidad expresada en bits por segundo (BPS), a la que se pueden transmitir los datos en un canal o ruta de comunicación. La velocidad binaria se incrementa al aumentar el número de niveles de señal para codificar un símbolo, sin embargo, esta velocidad expresada en BPS tiene un límite, y este depende de:

1. El ancho de la banda disponible, cuyas limitaciones provienen directamente de las propiedades físicas de los medios de transmisión. También ocurre por limitaciones que se imponen en el transmisor para prevenir interferencia con otras fuentes que comparten el mismo medio.
2. El nivel de ruido existente
3. Los niveles de señales a transmitir. Mientras más niveles haya, el espacio entre niveles disminuye, por lo que hay más probabilidades de que ocurra un error debido al ruido.

#### Capacidad del canal según Shannon

Un canal en la vida real siempre tendrá ruido, por lo que Shannon desarrolló su fórmula para poder determinar la máxima tasa de bits de un canal medido en decibelios que define la característica del canal. En esta fórmula, a diferencia de la

anterior, no se especifica el nivel de señal M, lo cuál significa que no importa qué cantidad de niveles tenga, no se podrá conseguir una velocidad mayor que la capacidad del canal.

$$C = W * \log_2(1 + SNR)$$

- Donde SNR es la proporción relación señal/ruido entre la potencia de la señal transmitida y la potencia del ruido que lo interfiere.

## **Descripción detallada de los programas desarrollados con impresiones de las pantallas.**

Usamos lenguaje Python y lo ejecutamos en Notebooks de Google Colab para calcular la capacidad de un canal de comunicación usando las fórmulas de Nyquist y Shannon. Para ello, usamos las siguientes bibliotecas:

- numpy: que provee las funciones necesarias cálculos numéricos.
- matplotlib: que facilita la generación de los gráficos
- ipywidgets: biblioteca usada para armar de forma más fácil la interfaz gráfica.

### **Código:**

```
import matplotlib.pyplot as plt

import numpy as np

import ipywidgets as widgets

from ipywidgets import interact

# Capacidad del canal según shannon

def capacidad_shannon(ancho_banda, snr):

    return ancho_banda * np.log2(1 + snr)

# Capacidad del canal según Nyquist

def capacidad_nyquist(ancho_banda, niveles):

    return 2 * ancho_banda * np.log2(niveles)

# Interfaz gráfica

ancho_banda_slider = widgets.FloatSlider(min=0, max=1e6, step=1e3, value=1e5, description='Ancho de Banda (Hz):')

niveles_slider = widgets.IntSlider(min=2, max=256, step=1, value=16, description='Niveles:')

snr_slider = widgets.FloatLogSlider(base=10, min=0, max=5, step=0.1, value=10, description='SNR:')
```

```

# Función para actualizar la gráfica

def actualizar_grafica(ancho_banda, niveles, snr):

    bw_range = np.linspace(0, ancho_banda, 500) # Rango de valores de ancho de banda.

    # Capacidades calculadas a lo largo del rango de ancho de banda.

    nyquist_cap = capacidad_nyquist(bw_range, niveles)

    shannon_cap = capacidad_shannon(bw_range, snr)

    # Capacidades máximas según Nyquist y Shannon para los valores específicos ingresados.

    nyquist_max = capacidad_nyquist(ancho_banda, niveles)

    shannon_max = capacidad_shannon(ancho_banda, snr)

    plt.figure(figsize=(10, 6))

    plt.plot(bw_range, nyquist_cap, label='Capacidad Nyquist')

    plt.plot(bw_range, shannon_cap, label='Capacidad Shannon')

    # Líneas horizontales para las capacidades máximas

    plt.axhline(nyquist_max, color='gray', linestyle='--', linewidth=1, label=f'Nyquist Max: {nyquist_max:.2f} BPS')

    plt.axhline(shannon_max, color='gray', linestyle=':', linewidth=1, label=f'Shannon Max: {shannon_max:.2f} BPS')

    plt.xlabel('Ancho de Banda (Hz)')

    plt.ylabel('Capacidad (BPS)')

    plt.title('Capacidad del Canal según Nyquist y Shannon')

    plt.legend()

    plt.grid(True)

    plt.show()

# Interactividad

interact(

    actualizar_grafica,

    ancho_banda=ancho_banda_slider,

    niveles=niveles_slider,

    snr=snr_slider

)

```

Para ejecutar el código:

1. Hacer clic al enlace a continuación: [Notebook: capacidad del canal según Nyquist y Shannon](#)
2. Seguir la siguiente secuencia de pasos:

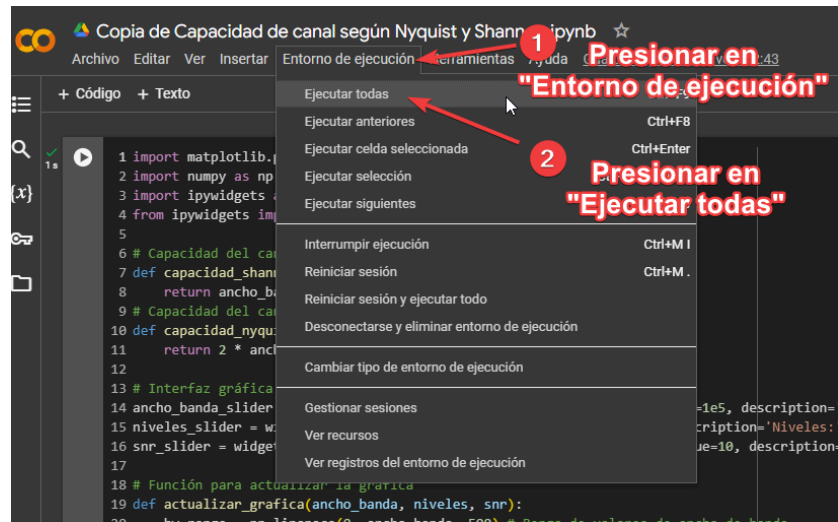


Figura 2.1: como ejecutar código fuente

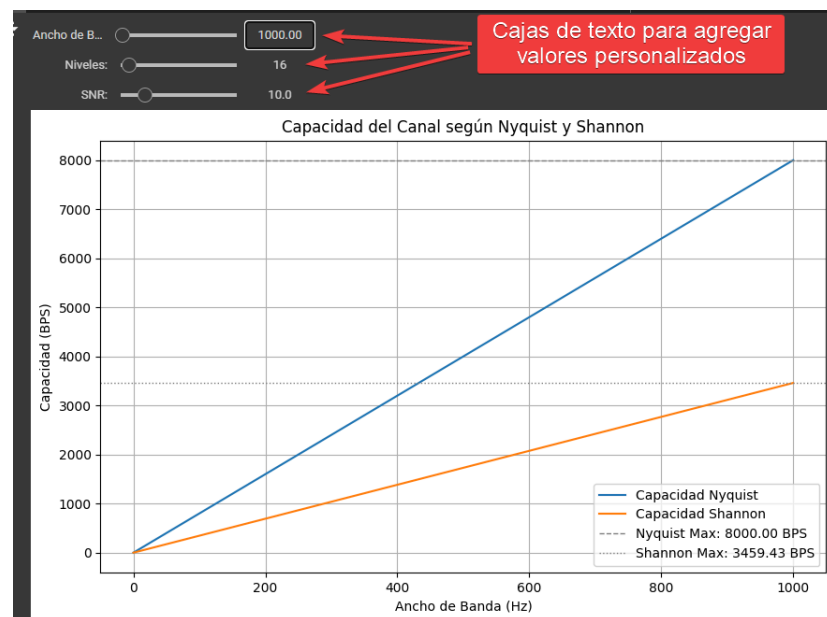


Figura 2.2: resultado de los parámetros ingresados

## Desarrollo de guía “transmisión de datos”

1. Se desea diseñar un sistema de transmisión con una velocidad de 2400 bps, utilizando una modulación PSK, en el cual solamente es admisible un error de 1 bit cada 10000 transmitidos, cual debería ser la potencia de la señal recibida si la temperatura ambiente es de 20°C. Calcular además el ancho de banda de la señal

$$\frac{E_b}{N_0} = 1 * 10^{-4}$$

Potencia de señal recibida:

$$S = kTR * \frac{E_b}{N_0} = 9,71123868 * 10^{-18}$$

Ancho de banda de la señal W:

$$W = \frac{\frac{E_b}{N_0}}{kT} = 2,47 * 10^{-16} \text{ Hz}$$

2. Si se mantiene la velocidad de transmisión y la potencia de la señal, pero la temperatura aumenta a 40°C. Cual sería la probabilidad de error esperable.

Probabilidad de error esperable:

$$\frac{E_b}{N_0} = \frac{S}{kTR} = 9,361328437 * 10^{-5}$$

3. Si se desea aumentar la velocidad de transmisión a 28800 bps manteniendo la probabilidad de error en  $10^{-4}$ , cual sería la solución?

$$S = kTR * \frac{E_b}{N_0} = 1,244853922 * 10^{-20}$$

## Conclusión:

La capacidad del canal es un concepto fundamental en la teoría de la información y es de gran importancia en el diseño y análisis de sistemas de comunicaciones, y con el código desarrollado podemos calcularlo de forma mucho más fácil y entender mejor.

## Laboratorio 3: codificación y modulación

### Breve introducción teórica del tema Codificación y Modulación

Una fuente de datos analógica o digital  $g(t)$  se codifica en una señal digital  $x(t)$  para la señalización digital. La forma de onda de  $x(t)$  dependerá de la técnica de codificación, que busca optimizar el uso del medio de transmisión. La transmisión analógica se basa en una señal continua de frecuencia constante conocida como portadora. Esta señal debe ser adecuada para el medio que se utilizará.

Los datos, se transmiten mediante la señal portadora aplicando:

- Modulación: es el proceso de codificar los datos generados por la fuente, en la señal portadora de frecuencia.
- Técnicas de modulación: es la modificación de uno o más de los tres parámetros fundamentales de la portadora (amplitud, frecuencia fase).

Por su parte, la señal de entrada se denomina señal moduladora o señal en banda base  $s(t)$ .

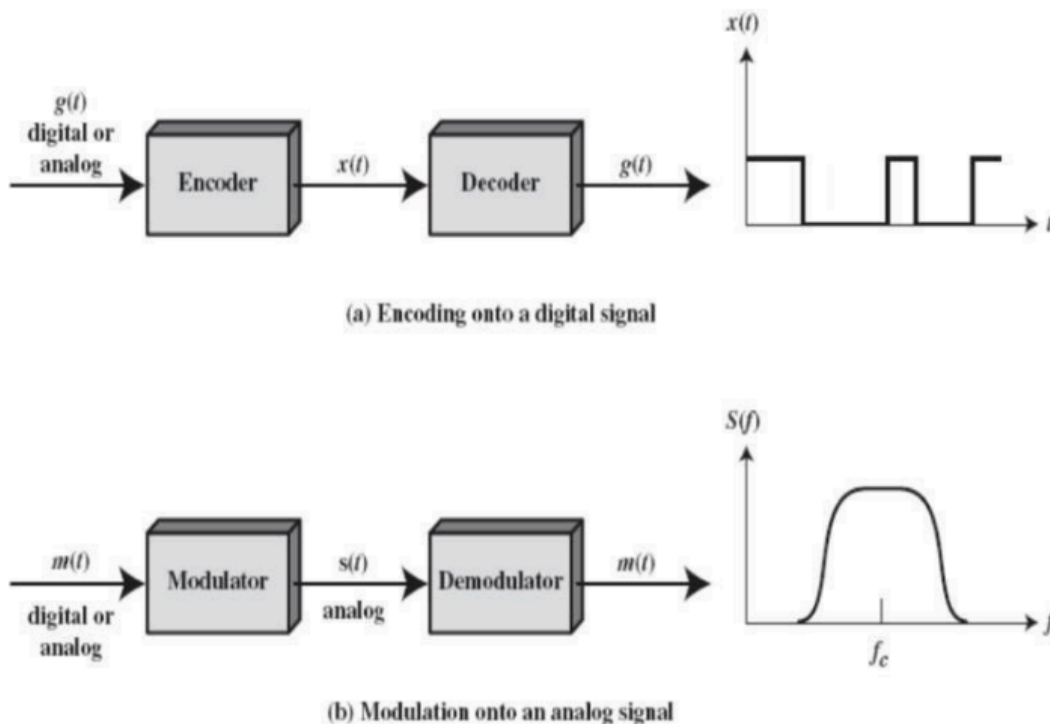


Figura 3.1: gráfico de codificación y modulación

### Técnicas de codificación digital

Nombre	Bit 0	Bit 1
No retorno a cero (NRZ-L)	Nivel de tensión alto	Nivel de tensión bajo

No retorno a cero invertido (NRZI)	No hay transición al comienzo del intervalo del bit	Hay transición al comienzo del intervalo del bit
Bipolar-AMI	No hay señal	Nivel positivo o negativo alternante
Pseudoternaria	Nivel positivo o negativo alternante	No hay señal
Manchester	Transición de nivel de tensión alto a nivel de tensión bajo en mitad del intervalo	Transición del nivel de tensión bajo al nivel de tensión alto en mitad del intervalo
Manchester Diferencial (siempre hay una transición en la mitad del intervalo)	Hay transición al principio del intervalo	No hay transición al principio del intervalo

Tabla 3.1: técnicas de codificación digital

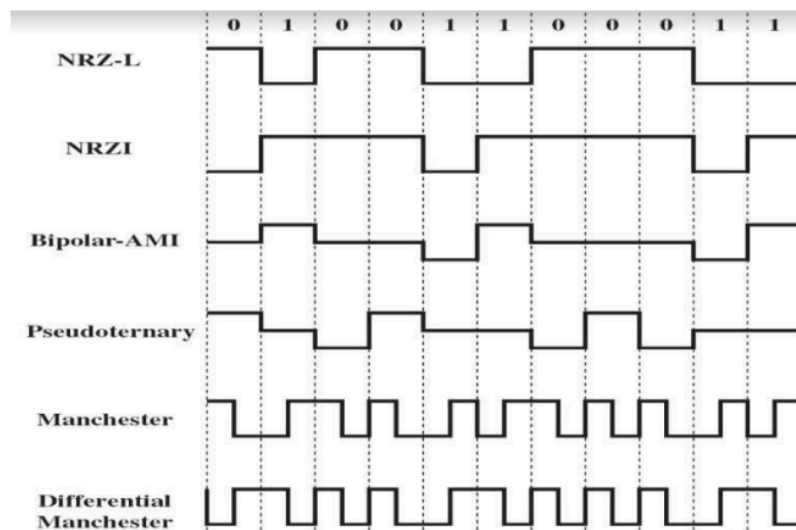


Figura 3.2: técnicas de modulación para la cadena de bits 01001100011

### Descripción detallada de los programas desarrollados con impresiones de las pantallas.

El programa, creado en C# y soportado por .NET, tiene como objetivo simular y visualizar una variedad de métodos de codificación de datos digitales utilizando señales digitales. Las técnicas de codificación utilizadas incluyen NRZ-L, NRZI, Bipolar-AMI, Pseudoternario, Manchester y Diferencial de Manchester. Sus funciones principales son:

- **Entrada de datos:** permite al usuario ingresar una cadena de bits de 11 caracteres. Estos bits representan la secuencia digital que se codificará y graficará según el método seleccionado.



- **Selección de Codificación:** mediante un menú desplegable, el usuario puede elegir el tipo de codificación que desea visualizar (NRZ-L, NRZI, Bipolar-AMI, Pseudoternario, Manchester o Manchester Diferencial).
- **Gráficos plotados de señales:** usa la librería gráfica de .NET para representar gráficamente las señales codificadas en una interfaz gráfica. Cada tipo de codificación se representa en un gráfico separado, donde se muestra visualmente la forma de onda de la señal generada.

El 1 representa el nivel de tensión positivo

El 0 representa el nivel de tensión negativo

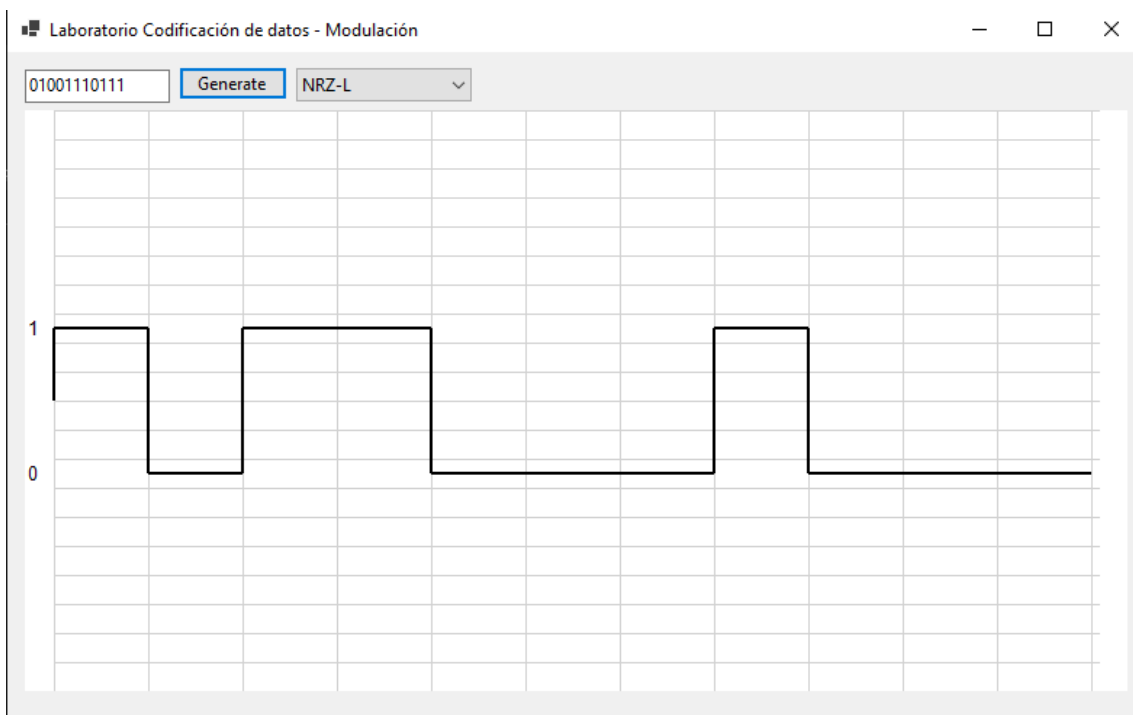


Figura 3.3: codificación NRZ-L con la cadena de bits 01001110111

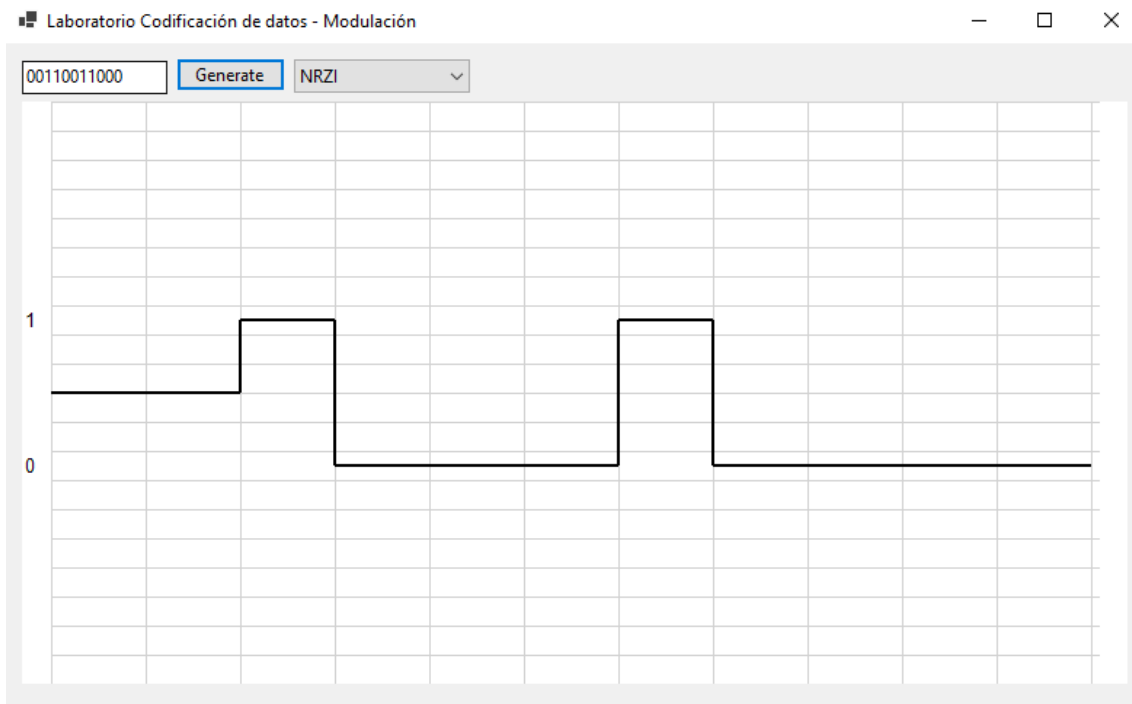


Figura 3.4: codificación NRZI con la cadena de bits 00110011000

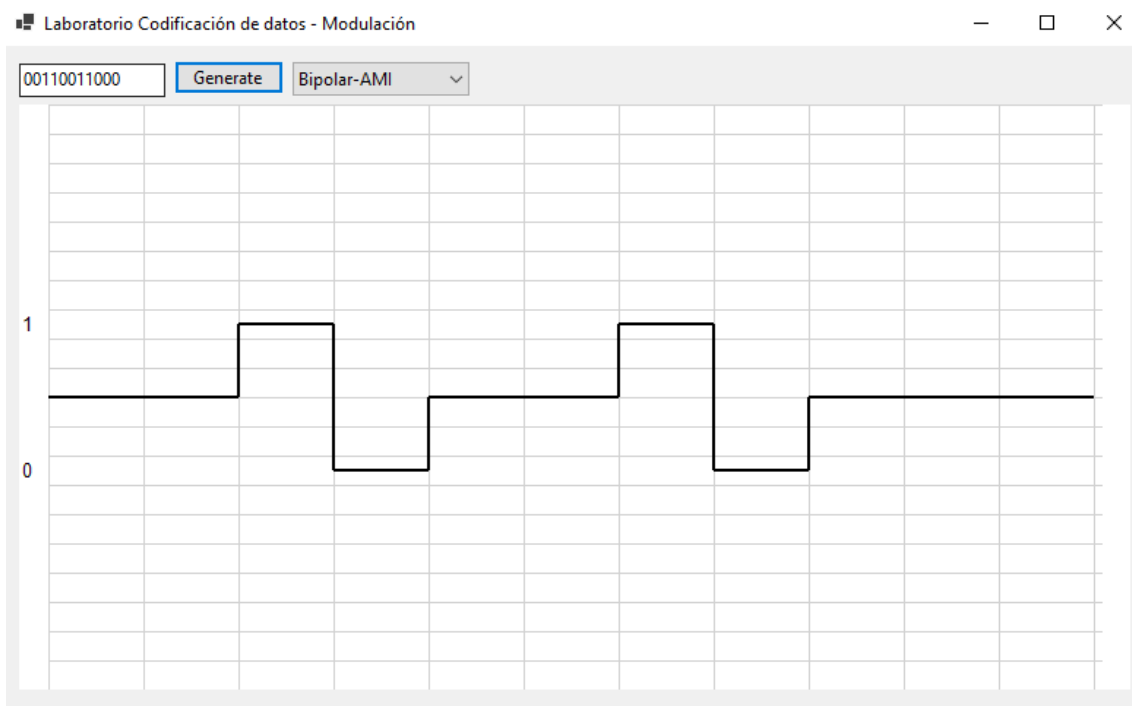


Figura 3.5: codificación Bipolar-AMI con la cadena de bits 00110011000

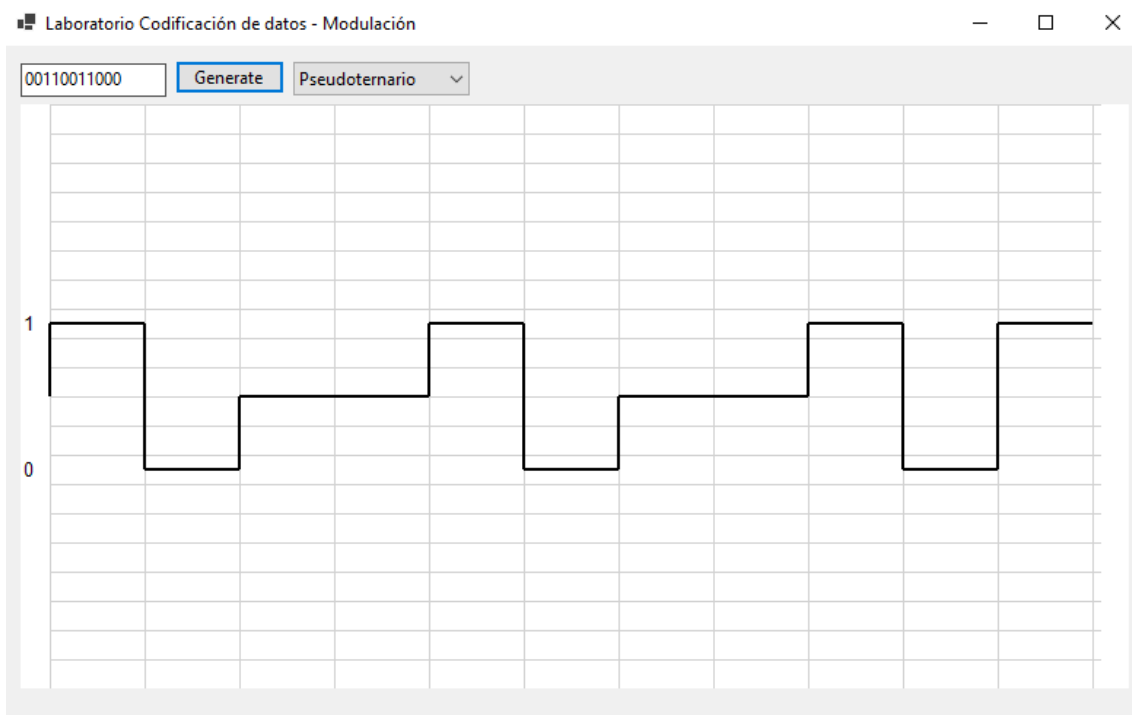


Figura 3.6: codificación Pseudoternario con la cadena de bits 01001110011

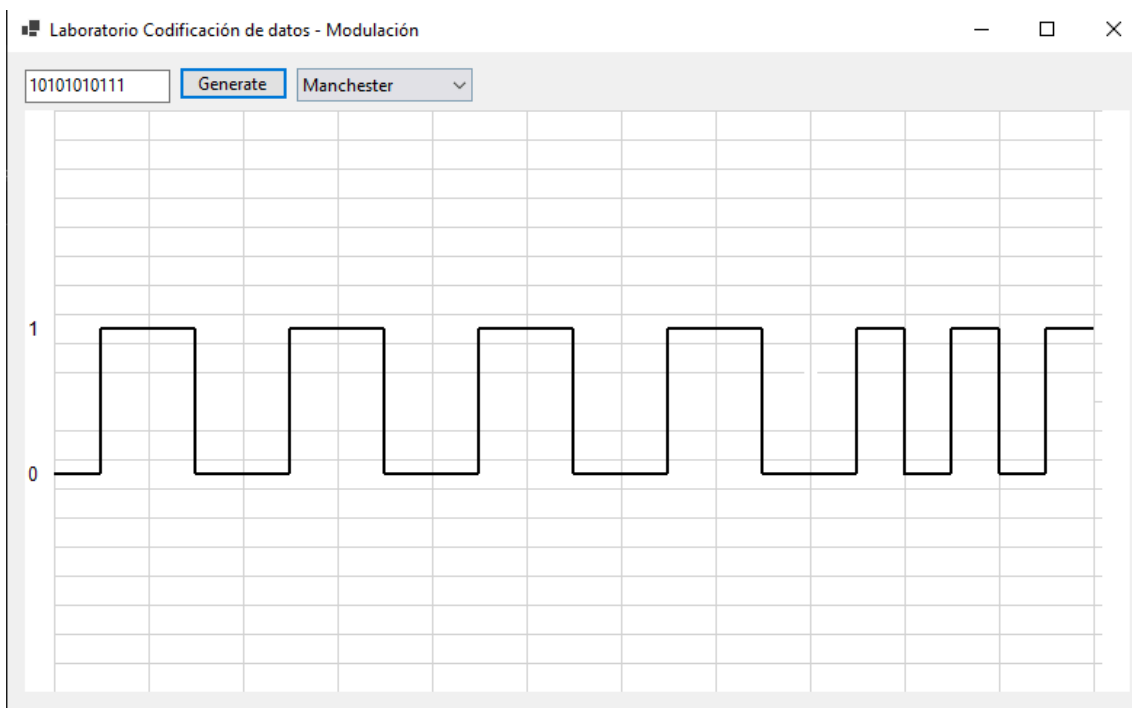


Figura 3.7: codificación Manchester con la cadena de bits 10101010111

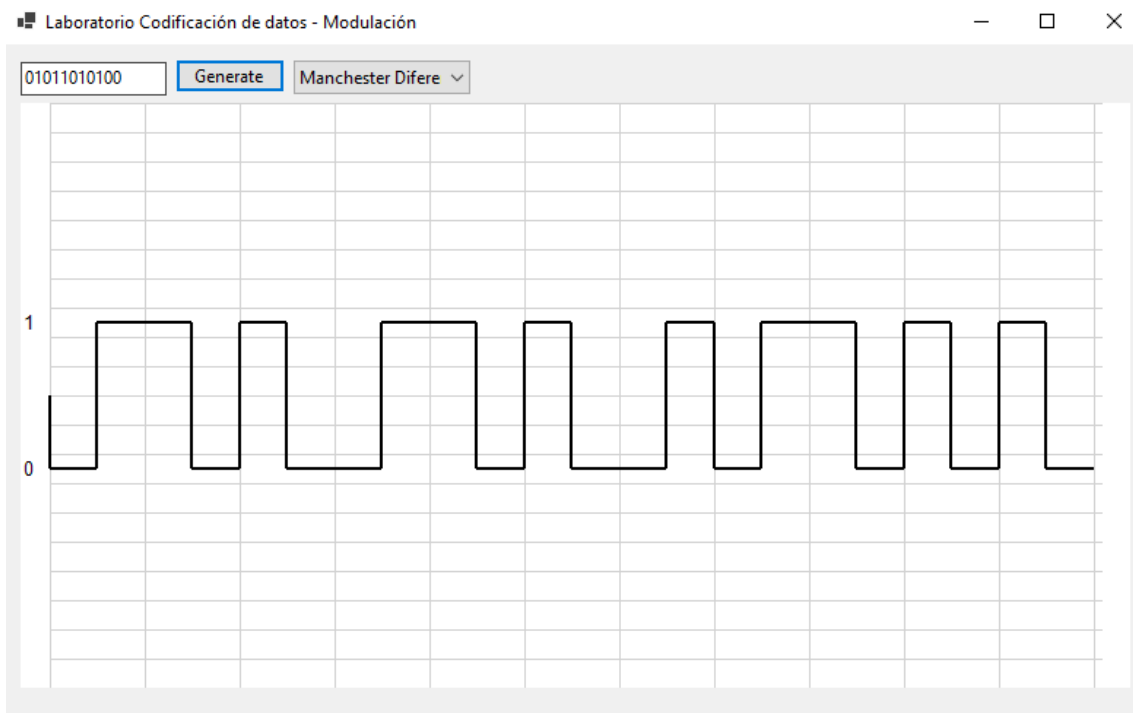


Figura 3.8: codificación Manchester Diferencial con la cadena de bits 01011010100

## Código fuente

```
using System;

using System.Drawing;

using System.Windows.Forms;

namespace Laboratorio3
{
    partial class Form1
    {
        #region Windows Form Designer generated code

        private TextBox bitInput;

        private Button generateButton;

        private PictureBox graphBox;

        private ComboBox encodingType;

        /// <summary>

        /// Required designer variable.

        /// </summary>

        private System.ComponentModel.IContainer components = null;
```

```

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)
{
    if (disposing && (components != null))
    {
        components.Dispose();
    }

    base.Dispose(disposing);
}

```

```

/// <summary>

/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()
{
    this.components = new System.ComponentModel.Container();

    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

    this.ClientSize = new System.Drawing.Size(800, 450);

    this.Text = "Form1";

    this.bitInput = new TextBox();

    this.generateButton = new Button();

    this.graphBox = new PictureBox();

    this.encodingType = new ComboBox();

    //

    // bitInput

    //

    this.bitInput.Location = new Point(12, 12);

```

```

this.bitInput.MaxLength = 11;

this.bitInput.Name = "bitInput";

this.bitInput.Size = new Size(100, 20);

this.bitInput.TabIndex = 0;


//

// generateButton

//

this.generateButton.Location = new Point(118, 10);

this.generateButton.Name = "generateButton";

this.generateButton.Size = new Size(75, 23);

this.generateButton.TabIndex = 1;

this.generateButton.Text = "Generate";

this.generateButton.UseVisualStyleBackColor = true;

this.generateButton.Click += new EventHandler(this.GenerateButton_Click);


//

// graphBox

//

this.graphBox.Location = new Point(12, 40);

this.graphBox.Name = "graphBox";

this.graphBox.Size = new Size(760, 400);

this.graphBox.TabIndex = 2;

this.graphBox.TabStop = false;


//

// encodingType

//

this.encodingType.DropDownStyle = ComboBoxStyle.DropDownList;

this.encodingType.Items.AddRange(new object[] {

    "NRZ-L",

    "NRZI",

    "Bipolar-AMI",

    "Pseudoternario",

    "Manchester",

```

```

        "Manchester Diferencial"}));

this.encodingType.Location = new Point(199, 11);

this.encodingType.Name = "encodingType";

this.encodingType.Size = new Size(121, 21);

this.encodingType.TabIndex = 3;


//

// Form1

//

this.ClientSize = new Size(784, 461);

this.Controls.Add(this.encodingType);

this.Controls.Add(this.graphBox);

this.Controls.Add(this.generateButton);

this.Controls.Add(this.bitInput);

this.Name = "Form1";

this.Text = "Laboratorio Codificación de datos - Modulación";

((System.ComponentModel.ISupportInitialize)(this.graphBox)).EndInit();

}

private void GenerateButton_Click(object sender, EventArgs e)
{
    string bits = bitInput.Text;

    string encoding = encodingType.SelectedItem?.ToString();

    if (bits.Length == 11 && ValidateBits(bits) && !string.IsNullOrEmpty(encoding))
    {
        DrawGraph(bits, encoding);
    }
    else
    {
        MessageBox.Show("Introduzca una cadena binaria de 11 bits válida y seleccione un tipo de codificación.");
    }
}
}

```

```
private bool ValidateBits(string bits)
```

```
{  
    foreach (char c in bits)  
    {  
        if (c != '0' && c != '1')  
        {  
            return false;  
        }  
    }  
    return true;  
}
```

```
private void DrawGraph(string bits, string encoding)
```

```
{  
    Bitmap bmp = new Bitmap(graphBox.Width, graphBox.Height);  
    using (Graphics g = Graphics.FromImage(bmp))  
    {  
        g.Clear(Color.White);  
        Pen pen = new Pen(Color.Black, 2);  
        Pen gridPen = new Pen(Color.LightGray, 1);  
        Font font = new Font("Arial", 10);  
        Brush brush = Brushes.Black;  
  
        int yOffset = graphBox.Height / 2;  
        int xOffset = 20;  
        int xStep = (graphBox.Width - 40) / bits.Length;  
        int yStep = 20;  
        int yCurrent = yOffset;  
        int yPrev = yOffset;  
        bool lastPulsePositive = false;  
  
        // Dibujar cuadrícula  
        for (int x = xOffset; x <= graphBox.Width - xOffset; x += xStep)  
        {  
            g.DrawLine(gridPen, x, 0, x, graphBox.Height);  
        }  
    }  
}
```



```

}

for (int y = 0; y <= graphBox.Height; y += yStep)
{
    g.DrawLine(gridPen, xOffset, y, graphBox.Width - xOffset, y);
}

// Dibujar etiquetas de 1 y 0
g.DrawString("A+", font, brush, new PointF(0, yOffset - 50 - font.Height / 2));
g.DrawString("A-", font, brush, new PointF(0, yOffset + 50 - font.Height / 2));

// Dibujar las señales
for (int i = 0; i < bits.Length; i++)
{
    int bit = bits[i] == '1' ? 1 : 0;

    int xCurrent = xOffset + i * xStep;

    switch (encoding)
    {
        case "NRZ-L":
            yCurrent = bit == 1 ? yOffset + 50 : yOffset - 50; // 1 es negativo, 0 es positivo

            break;

        case "NRZI":

            if (bit == 1)
            {
                yCurrent = yPrev == yOffset + 50 ? yOffset - 50 : yOffset + 50;
            }

            else
            {
                yCurrent = yPrev;
            }

            break;

        case "Bipolar-AMI":

            if (bit == 1)
            {

```

```

        yCurrent = lastPulsePositive ? yOffset + 50 : yOffset - 50;

        lastPulsePositive = !lastPulsePositive;
    }

    else

    {

        yCurrent = yOffset;

    }

    break;

case "Pseudoternario":

    if (bit == 0)

    {

        yCurrent = lastPulsePositive ? yOffset + 50 : yOffset - 50;

        lastPulsePositive = !lastPulsePositive;

    }

    else

    {

        yCurrent = yOffset;

    }

    break;

case "Manchester":

    // 0: transición de nivel alto a bajo en la mitad del intervalo

    // 1: transición de nivel bajo a alto en la mitad del intervalo

    if (bit == 0)

    {

        // Primera mitad del intervalo: alto

        g.DrawLine(pen, xCurrent, yOffset - 50, xCurrent + xStep / 2, yOffset - 50);

        // Transición a bajo en la mitad del intervalo

        g.DrawLine(pen, xCurrent + xStep / 2, yOffset - 50, xCurrent + xStep / 2,

yOffset + 50);

        // Segunda mitad del intervalo: bajo

        g.DrawLine(pen, xCurrent + xStep / 2, yOffset + 50, xCurrent + xStep, yOffset

+ 50);

    }

    else

    {

```

```

        // Primera mitad del intervalo: bajo
        g.DrawLine(pen, xCurrent, yOffset + 50, xCurrent + xStep / 2, yOffset + 50);

        // Transición a alto en la mitad del intervalo
        g.DrawLine(pen, xCurrent + xStep / 2, yOffset + 50, xCurrent + xStep / 2,
yOffset - 50);

        // Segunda mitad del intervalo: alto
        g.DrawLine(pen, xCurrent + xStep / 2, yOffset - 50, xCurrent + xStep, yOffset
- 50);

    }

    yPrev = bit == 0 ? yOffset + 50 : yOffset - 50; // Actualizar el nivel anterior

    break;

case "Manchester Diferencial":

    if (i == 0)

    {

        yCurrent = bit == 1 ? yOffset - 50 : yOffset + 50;

    }

    else

    {

        yCurrent = bit == 0 ? (yPrev == yOffset - 50 ? yOffset + 50 : yOffset - 50) :

yPrev;

    }

    g.DrawLine(pen, xCurrent, yPrev, xCurrent, yCurrent); // Transición al comienzo
del intervalo

    yPrev = yCurrent;

    yCurrent = yCurrent == yOffset + 50 ? yOffset - 50 : yOffset + 50;

    g.DrawLine(pen, xCurrent, yPrev, xCurrent + xStep / 2, yPrev); // Mitad del
intervalo

    g.DrawLine(pen, xCurrent + xStep / 2, yPrev, xCurrent + xStep / 2, yCurrent); //
Transición en la mitad del intervalo

    g.DrawLine(pen, xCurrent + xStep / 2, yCurrent, xCurrent + xStep, yCurrent); //
Final del intervalo

    yPrev = yCurrent;

    continue;

}

g.DrawLine(pen, xCurrent, yPrev, xCurrent, yCurrent);

g.DrawLine(pen, xCurrent, yCurrent, xCurrent + xStep, yCurrent);

```

```

        yPrev = yCurrent;
    }

    graphBox.Image = bmp;
}

}

}

}

#endregion

```

## Desarrollo de guía “codificación y modulación”

### Consignas:

1. Correr una simulación con una señal seno y cambiar la cantidad de niveles a 32, observar la salida del cuantizador y generar un archivo EXCEL®, con los códigos correspondientes.
2. Correr una simulación con una señal seno y cambiar la cantidad de niveles a 8, observar la salida del cuantizador y generar un archivo EXCEL®, con los códigos correspondientes.
3. Cuales son las conclusiones que puede aportar sobre lo observado?
4. Buscar información y explicar que es el "dither", y por que se lo utiliza en PCM.
5. Buscar información acerca del código de línea 4B/5B, describirlo y averiguar que sistemas lo utilizan.
6. Buscar información acerca del código de línea 8B/10B, describirlo y averiguar que sistemas lo utilizan.

1)

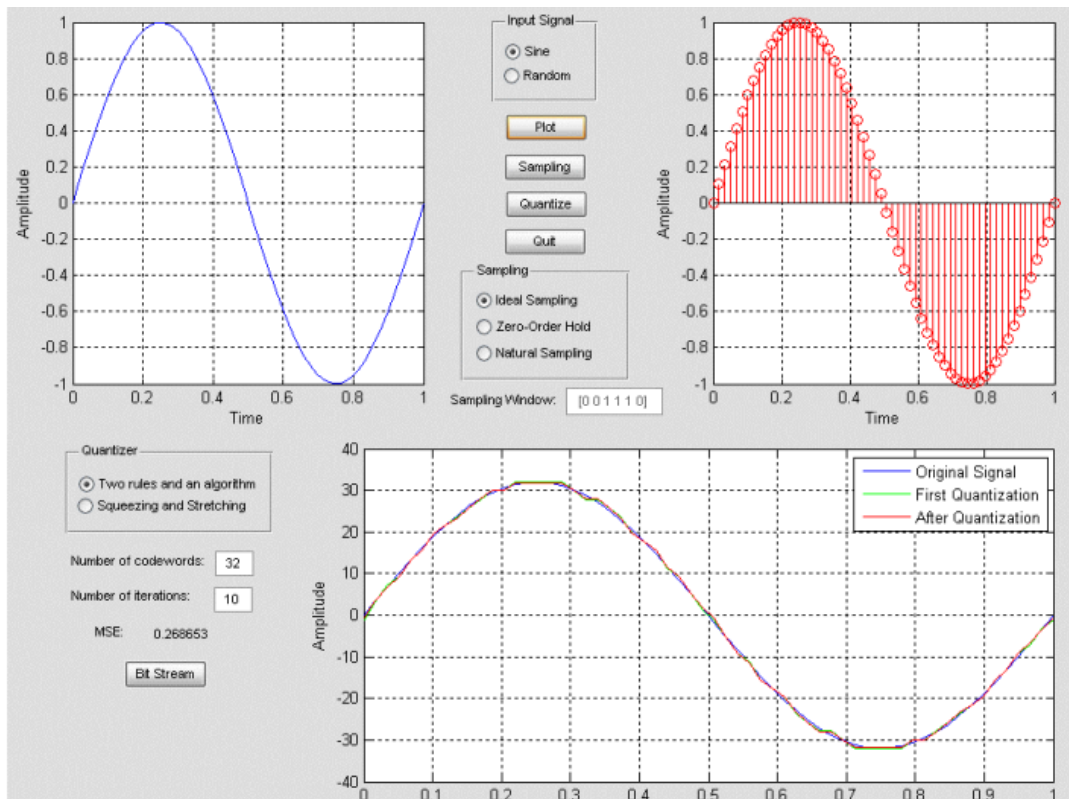


Figura 3.9: simulación con la señal seno con 32 niveles

Quantized Signal	Symbol	Digital Signal					
-31,62703659	0	0	0	0	0	0	0
-30,07396568	1	0	0	0	0	0	1
-27,96596952	2	0	0	0	0	1	0
-25,67766931	3	0	0	0	0	1	1
-23,50444016	4	0	0	0	1	0	0
-21,70597876	5	0	0	0	1	0	1
-19,75378467	6	0	0	0	1	1	0
-17,68960209	7	0	0	0	1	1	1
-15,50013283	8	0	0	1	0	0	0
-13,22278994	9	0	0	1	0	0	1
-11,65313841	10	0	0	1	0	1	0
-9,23486842	11	0	0	1	0	1	1
-6,764243956	12	0	0	1	1	0	0
-5,090031873	13	0	0	1	1	0	1
-3,401391541	14	0	0	1	1	1	0
-0,567703198	15	0	0	1	1	1	1
1,703109595	16	0	1	0	0	0	0
3,401391541	17	0	1	0	0	0	1
5,090031873	18	0	1	0	0	1	0
6,764243956	19	0	1	0	0	1	1
9,23486842	20	0	1	0	1	0	0
11,65313841	21	0	1	0	1	0	1
13,22278994	22	0	1	0	1	1	0
15,50013283	23	0	1	0	1	1	1
17,68960209	24	0	1	1	0	0	0
19,75378467	25	0	1	1	0	0	1
21,70597876	26	0	1	1	0	1	0
23,50444016	27	0	1	1	0	1	1
25,67766931	28	0	1	1	1	0	0
27,96596952	29	0	1	1	1	0	1
30,07396568	30	0	1	1	1	1	0
31,62703659	31	0	1	1	1	1	1

Figura 3.10: archivo excel generado en la simulación con 32 niveles.

Quantized Signal	Symbol	Digital Signal			
-7,55830822	0	0	0	0	0
-5,714454832	1	0	0	0	1
-3,484010454	2	0	1	1	0
-0,906359251	3	0	1	1	1
1,268902951	4	1	0	0	0
3,484010454	5	1	0	0	1
5,714454832	6	1	1	1	0
7,55830822	7	1	1	1	1

Figura 3.11: archivo excel generado en la simulación con 8 niveles

**3)** Hemos visto que a medida que aumentamos los niveles, la cuantización disminuye y la señal recuperada se acerca a la original. En los ejercicios 1 y 2, la señal recuperada se acerca a la original con errores notables al usar 8 niveles, mientras que al usar 32 niveles se acerca mucho más a la original.

**4)** El dithering, también conocido como dithering, es la aplicación intencional de una forma de ruido para aleatorizar el error de cuantificación y así evitar patrones de gran escala, como las bandas de color que se pueden ver en las imágenes. Se emplea con frecuencia en el procesamiento de datos de audio y video digital, y normalmente es una de las etapas finales de la masterización de audio en un CD. En telecomunicaciones, el código de línea 4B5B asigna grupos de datos de 4 bits a grupos de 5 bits para su transmisión. Para garantizar que habrá suficientes transiciones de estado de línea para producir una señal de autobloqueo, se seleccionan estas palabras de 5 bits en un diccionario. Un efecto colateral del código es que la misma información requiere un 25 % más de bits. Algunos sistemas que lo usan son:

- MAD1 en 1989
- Fast Ethernet en 1995
- FDDI a mitad de los años 1980

**5)** En telecomunicaciones, el 8B10B es un tipo de código de línea que asigna palabras de 8 bits a símbolos de 10 bits para mantener un buen equilibrio y proporcionar cambios de estado suficientes para permitir una recuperación de sincronización razonable. Es decir, en una cadena de al menos 20 bits, la diferencia entre los unos y los ceros no es mayor a dos, y no hay más de cinco unos o ceros seguidos. Esto contribuye a la disminución de la demanda del límite inferior del ancho de banda del canal que transmite la señal. Algunos sistemas que lo usan son:

- PCI Express

- Thunderbolt 3
- USB 3.0
- ServerNet

## **Conclusión**

Se puede usar varios métodos de codificación digital para transmitir una misma cadena de bits donde cada uno usa valores distintos para los bits 0 y 1, Además de los métodos para resolver problemas de sincronización, como las técnicas de Manchester y Manchester diferencial, que utilizan un cambio de polaridad a mitad del intervalo.

## Laboratorio 4: medios de transmisión

### Breve introducción teórica del tema

#### Ganancia de una antena

Es una medida de direccionalidad: dada una dirección, se define la ganancia de una antena como la potencia de salida en esa dirección, comparada con la potencia transmitida en cualquier dirección por una antena omnidireccional ideal (antena isotrópica).

Ej.: si una antena proporciona una ganancia de 3dB en una dirección: mejora a la antena isotrópica en esa dirección en 3dB (en un factor 2).

La ganancia de una antena no se refiere al incremento de potencia transmitida respecto a la potencia de entrada sino a una medida de direccionalidad.

Un concepto relacionado con la ganancia es el área efectiva: el área efectiva de una antena está relacionada con su tamaño físico y con su geometría. La relación entre la ganancia de una antena y su área efectiva viene dada por:

$$G = (4 \pi A_e) / \lambda^2 = (4 \pi f^2 A_e) / c^2$$

**G**: ganancia de la antena

**A<sub>e</sub>**: área efectiva

**f**: frecuencia de la portadora

**c**: velocidad de la luz ( $\approx 3 \times 10^8$  m/s).

**λ**: longitud de la portadora

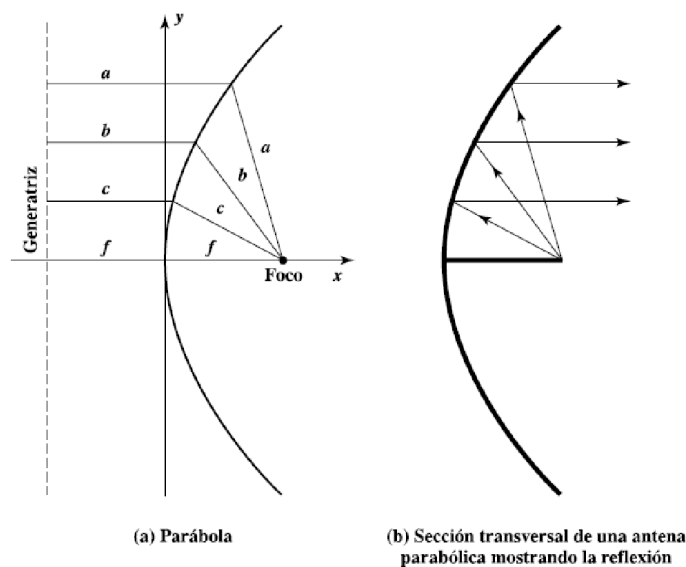


Figura 4.1: partes de una antena



## Distancia máxima entre dos antenas

### *Propagación en la trayectoria visual*

Por encima de 30MHz, los modos de propagación superficial o aérea no funcionan, por lo que las comunicaciones han de realizarse siguiendo la línea de visión (LOS, Line-of-Sight). En las comunicaciones vía satélite, las señales por encima de 30 MHz no se reflejan en la ionosfera, por lo que para esas frecuencias no es posible transmitir entre estaciones terrestres y satélites que estén por debajo de la línea del horizonte. En comunicaciones superficiales, para este modo de transmisión, la antena emisora y la receptora deben estar alineadas según la trayectoria visual efectiva. Se usa el término efectiva ya que las microondas son pandeadas o refractadas por la atmósfera. La cantidad de pandeo, e incluso la dirección seguida, dependerá de las condiciones, aunque, por lo general, las microondas siguen la curvatura de la tierra, por lo que llegaran más lejos que siguieran la línea de visión óptica.

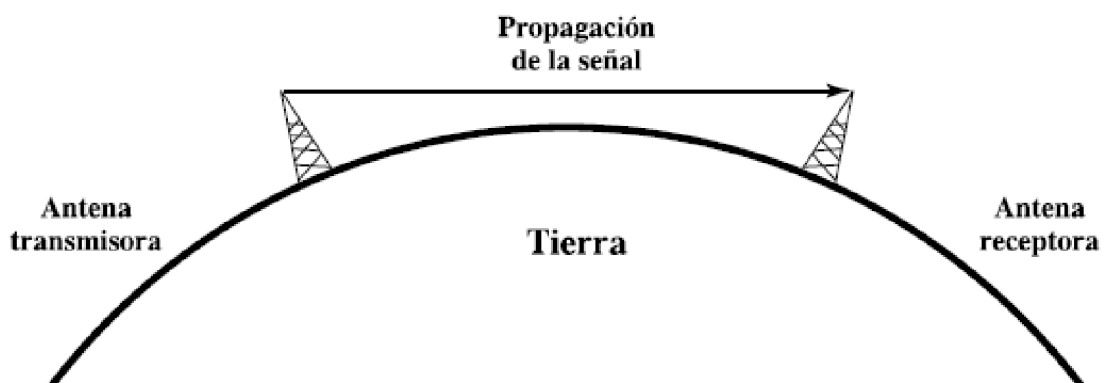


Figura 4.2: propagación en la trayectoria visual (por encima de los 30MHz).

## Refracción

La refracción se produce debido a que la velocidad de las ondas electromagnéticas es una función de la densidad del medio atravesado. En el vacío, una onda electromagnética (por ejemplo, la luz o una onda de radio), se propaga aproximadamente a  $3 \times 10^8$  m/s. Esta es la constante  $c$ , denominada velocidad de la luz, aunque en realidad se está refiriendo a la velocidad de la luz en el vacío. En el aire, agua, cristal o cualquier otro medio de transporte, o parcialmente transparente, las ondas electromagnéticas viajan a velocidades menores que  $c$ .

Cuando una onda electromagnética pasa de un medio con una densidad a otro con densidad distinta, su velocidad cambia. El efecto de esto es desviar la dirección de la onda en la separación entre los dos medios. Al pasar de un medio menos denso a otro

con densidad mayor, la onda se desviará hacia el medio más denso. Este fenómeno se puede observar sumergiendo parcialmente un palo en agua.

El índice de refracción de un medio respecto a otro es igual al seno del ángulo de incidencia dividido entre el seno del ángulo de refracción. El índice de refracción es también igual al cociente entre las velocidades respectivas en los dos medios. El índice absoluto de refracción de un medio se calcula en comparación con el del vacío. El índice de refracción varía con la longitud de onda, de forma tal que la refracción sufrida por señales con distintas longitudes de onda será diferente.

Aunque en una separación discreta entre dos medios la desviación de la onda será abrupta y de una vez, si se trata de una separación continua, en la que el índice de refracción varíe gradualmente, la onda se desviará gradualmente. Bajo condiciones normales de propagación, el índice de refracción de la atmósfera disminuye con la altura, por lo que las ondas de radio viajan más lentamente cerca de la tierra que a alturas mayores. Como consecuencia, se tiene que las ondas de radio se desvían suavemente hacia la tierra.

### **Línea de visión óptica y de radio.**

Si no hay obstáculos, la línea de visión óptica se puede expresar como:

$$d = 3,57 \sqrt{h}$$

Donde d es la distancia entre la antena y el horizonte en kilómetros y h es la altura de la antena en metros. La línea de visión efectiva, o de radio, se expresa como:

$$d = 3,57 \sqrt{Kh}$$

Donde k es un factor de ajuste que tiene en cuenta la refracción. Una buena aproximación es  $K=4/3$ . Así, la distancia máxima entre dos antenas siguiendo propagación LOS es:

$$d = 3,57 (\sqrt{Kh_1} + \sqrt{Kh_2})$$

Donde  $h_1$  y  $h_2$  son respectivamente las alturas de las antenas.

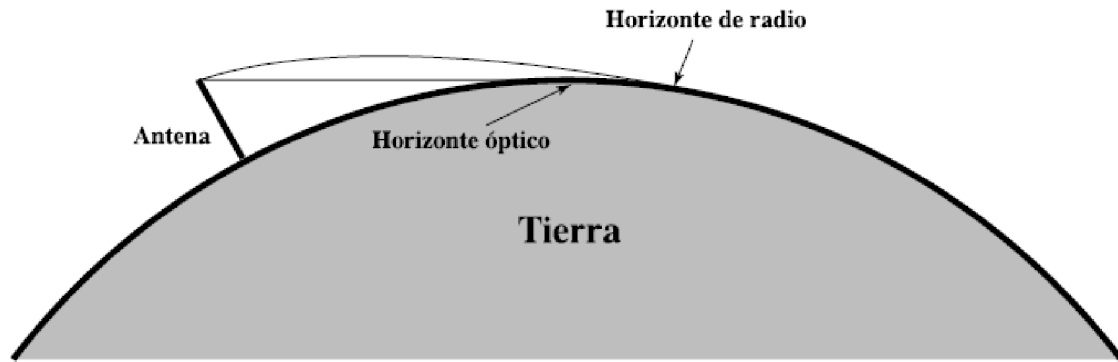


Figura 4.3: Horizonte óptico y de radio.

## Descripción detallada de los programas desarrollados con impresiones de las pantallas

Una aplicación que calcula y genera una gráfica de la Ganancia de una Antena en decibels mili-watts DBm.

Los parámetros a introducir/modificar son:

- Diámetro de la Antena: rango de 0 a 99999999 mts.
- Frecuencia de Trabajo: rango de 0 a 99999999 GHz.

Otra aplicación calcula y genera una gráfica de la Distancia Máxima entre dos Antenas en kilómetros.

Los parámetros a introducir/modificar son:

- Altura de la primera antena: rango de 0 a 999 mts.
- Altura de la segunda antena: rango de 0 a 999 mts.

### Resultados:

Ganancia de una Antena		Distancia Máxima entre Antenas	
Ejecución 1	Ejecución 2	Ejecución 1	Ejecución 2
Diámetro de la Antena: 2 mts.	Diámetro de la Antena: 5 mts.	Altura 1° Antena: 54 mts	Altura 1° Antena: 75 mts
Frecuencia de Trabajo: 12 GHz	Frecuencia de Trabajo: 12 GHz	Altura 2° Antena: 54 mts.	Altura 2° Antena: 75 mts.
Resultado	Resultado	Resultado	Resultado
45.46 dBm	53.42 dBm	60.58 Km	71.40 Km

Tabla 4.1: resultados del desarrollo



Figura 4.4: ganancia de una Antena de 2mts de diámetro y 12GHz de frecuencia de trabajo.

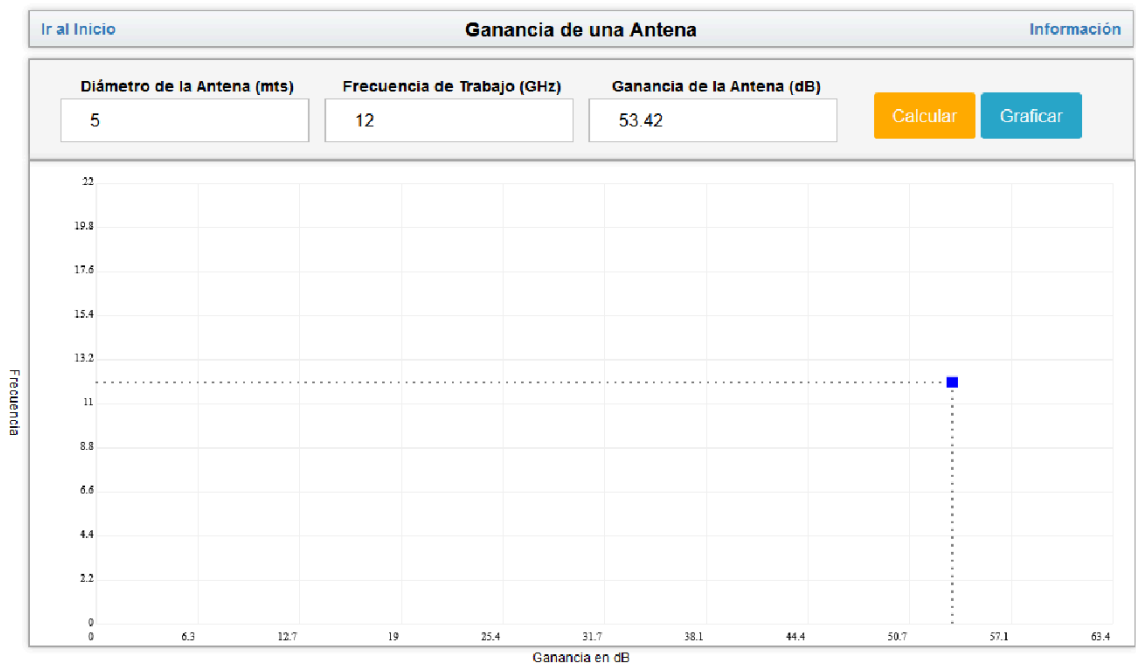


Figura 4.5: ganancia de una Antena de 5mts de diámetro y 12GHz de frecuencia de trabajo

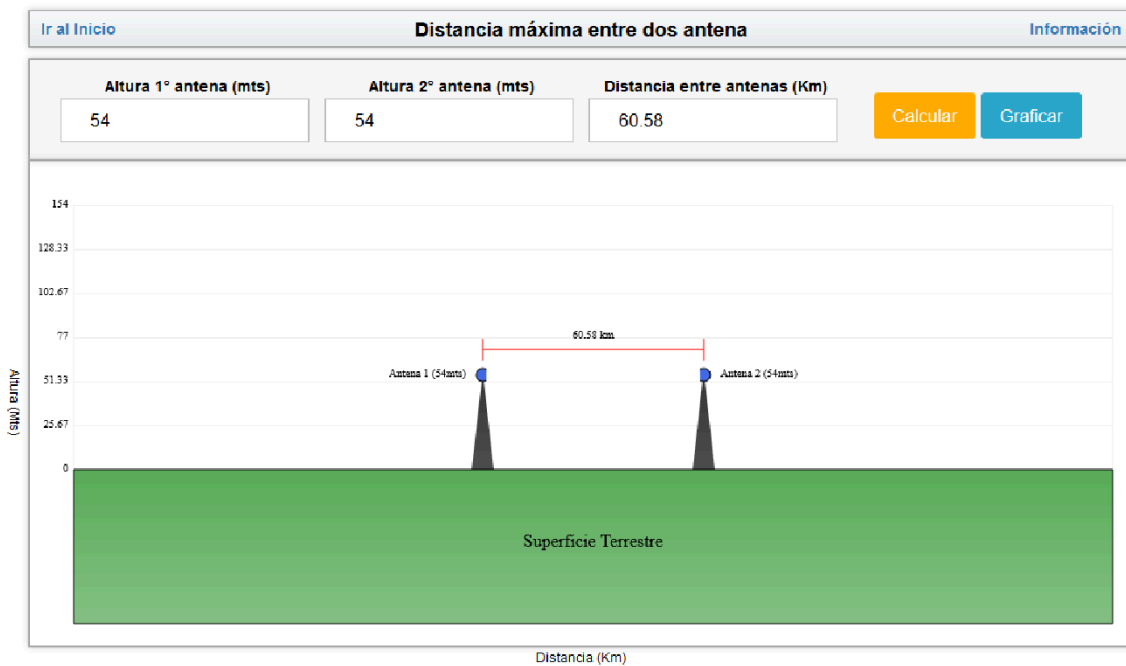


Figura 4.6: Distancia máxima entre dos antenas, ambas de 54 metros de altura.

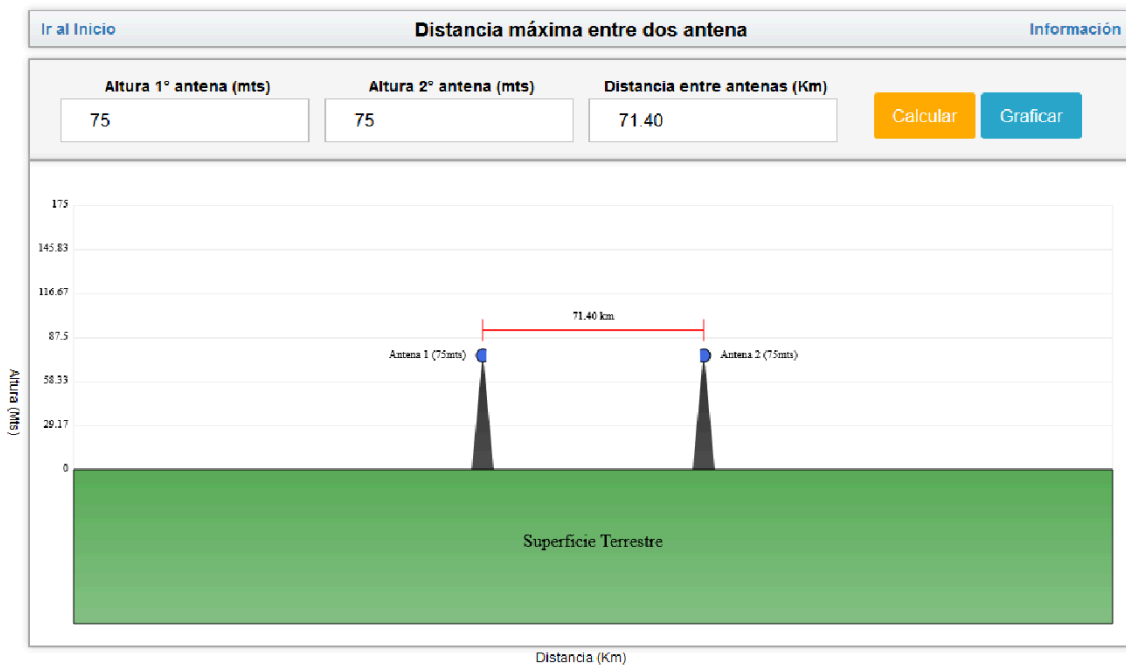


Figura 4.7: Distancia máxima entre dos antenas, ambas de 75 metros de altura.

## Ganancia de una Antena

//variable que contiene al elemento HTML

```
var antennaDraw = document.getElementById("antennaDraw");
```

```

var ctx = antennaDraw.getContext("2d");

var $valDiameter;

var $valFrequency;

var $valGain;

var $radio;

var $area;

var $lengthWave;

var gain;

var gainDB;

//al hacer clic en el botón calcular

$('#calcular').on('click',function(){

//introducimos diámetro de la antena

$valDiameter=parseFloat($('name=diameter').val());

if(isNaN($valDiameter)){($('name=diameter').val(0);$valDiameter=0;}else{if($valDiameter <
0){$valDiameter=0;$('name=diameter').val($valDiameter)}else{($('name=diameter').val($valDiameter)}}

//introducimos frecuencia de la señal

$valFrequency=parseFloat($('name=frequency').val());

if(isNaN($valFrequency)){($('name=frequency').val(0);$valFrequency=0;}else{if($valFrequency <
0){$valFrequency=0;$('name=frequency').val($valFrequency)}else{($('name=frequency').val($valFrequency)}}

//establecemos la fórmula de ganancia en dB

$radio=$valDiameter / 2;

$area=Math.PI * Math.pow($radio, 2);

$lengthWave=(3 * Math.pow(10,8)) / ($valFrequency * Math.pow(10,9));

$gain=(7 * $area) / Math.pow($lengthWave, 2);

$gainDB=10 * Math.log10($gain);

//mostramos ganancia de la antena en dB

($('name=gain').val($gainDB.toFixed(2)));

});

//al hacer clic en el boton Graficar

$('#graficar').on('click',function(){

```

```

//introducimos diametro de la antena

$valDiameter=parseFloat($(' [name=diameter] ').val());

if(isNaN($valDiameter)){ $(' [name=diameter] ').val(0);$valDiameter=0;}else{if($valDiameter <
0){$valDiameter=0;$(' [name=diameter] ').val($valDiameter)}else{ $(' [name=diameter] ').val($valDiameter)}}

//introducimos frecuencia de la señal

$valFrequency=parseFloat($(' [name=frequency] ').val());

if(isNaN($valFrequency)){ $(' [name=frequency] ').val(0);$valFrequency=0;}else{if($valFrequency <
0)

{$valFrequency=0;$(' [name=frequency] ').val($valFrequency)}else{ $(' [name=frequency] ').val($valFrequency)}}

//establecemos la formula de ganancia en dB

$radio=$valDiameter / 2;

$area=Math.PI * Math.pow($radio, 2);

$lengthWave=(3 * Math.pow(10,8)) / ($valFrequency * Math.pow(10,9));

$gain=(7 * $area) / Math.pow($lengthWave, 2);

$gainDB=10 * Math.log10($gain);

//mostramos ganancia de la antena en dB

($(' [name=gain] ').val($gainDB.toFixed(2)));

//graficamos

gainAntenna();

});

//función para dibujar grilla y ganancia antena

function gainAntenna(){

clearCanvas();

// Ancho de la grilla

bw = 960;

// alto de la grilla

bh = 400;

// margen interno

```

```

p = 20;

ctx.beginPath();

//escalas

escalaFrequency=parseFloat(($valFrequency+10));

scaleFreDown=parseFloat((escalaFrequency)/10);

scaleFrequency=escalaFrequency;

escalaGain=parseFloat(($gainDB+10)/10);

scaleGain=0;


//dibujamos la grilla respecto a líneas verticales

for (x = 40; x <= bw; x += 92) {

ctx.beginPath();


//creamos las líneas

ctx.moveTo(0.5 + x + p, p);

ctx.lineTo(0.5 + x + p, bh + p);

//establecemos fuente, color e información de los ejes Y

ctx.font = '10px serif';

ctx.fillStyle='#000';

ctx.fillText(Math.round(scaleGain * 10) / 10,(x+18),bh+35);

scaleGain=parseFloat(scaleGain+escalaGain);

//establecemo linea verticales

ctx.setLineDash([1, 0]);

//color de las líneas

ctx.strokeStyle = "#F2F2F2";

//mostramos

ctx.stroke();

}


//dibujamos la grilla respecto a líneas horizontales

for (x = 0; x <= bh; x += 40) {

ctx.beginPath();

```



```

//creamos las líneas

ctx.moveTo(40+p, 0.5 + x + p);

ctx.lineTo(bw + p, 0.5 + x + p);


//establecemos fuente, color e información de los ejes X

ctx.font = '10px serif';

ctx.fillStyle='#000';

ctx.textAlign="right";

ctx.fillText(Math.round(scaleFrequency * 100) /100 , 58, (x+22));

scaleFrequency=parseFloat(scaleFrequency)-parseFloat(scaleFreDown);

//establecemos líneas horizontales

ctx.setLineDash([1, 0]);

//color de las líneas

ctx.strokeStyle = "#F2F2F2";

//mostramos

ctx.stroke();

}


//alto y ancho de la gráfica

widthGraph=920;

heightGraph=400;


//cálculos para graficar líneas

percentFrequency=($valFrequency * 100) / ($valFrequency+10);

lineFrequency=419-((heightGraph/100)*(percentFrequency));

percentGain=($gainDB * 100) / ($gainDB+10);

lineGain=60+((widthGraph/100)*(percentGain));

//graficamos línea punteada horizontal

ctx.beginPath();

ctx.moveTo(60,lineFrequency);

ctx.lineTo(lineGain,lineFrequency);

ctx.setLineDash([2, 5]);

```

```

ctx.strokeStyle = "#000";

//mostramos

ctx.stroke();


//graficamos línea punteada vertical

ctx.beginPath();

ctx.moveTo(lineGain,lineFrequency);

ctx.lineTo(lineGain,420);

ctx.setLineDash([2, 5]);

ctx.strokeStyle = "#000";

//mostramos

ctx.stroke();

//graficamos punto (rectángulo) del cálculo final de ganancia

ctx.beginPath();

ctx.fillStyle="blue";

ctx.fillRect(lineGain-5, lineFrequency-5,10,10);

ctx.stroke();

}


//función para resetear canvas

function clearCanvas(){

ctx.beginPath(); // clear existing drawing paths

ctx.save(); // store the current transformation matrix


// Use the identity matrix while clearing the canvas

ctx.setTransform(1, 0, 0, 1, 0, 0);

ctx.clearRect(0, 0, antennaDraw.width, antennaDraw.height); ctx.restore(); // restore the
transform

}

})();

```

**Distancia Máxima entre dos antenas.**

```

//variable que contiene al elemento HTML

var antennaDraw = document.getElementById("antennaDraw");

var ctx = antennaDraw.getContext("2d");

var $valHeightOne;

var $valHeightTwo;

var $approx=4/3;

var $distance;

//al hacer clic en el botón calcular

$('#calcular').on('click',function(){

//introducimos altura de la primer antena

$valHeightOne=parseFloat($('name=height_one').val());

if(isNaN($valHeightOne)){ $('name=height_one').val(0);$valHeightOne=0;}else{if($valHeightOne <
0){$valHeightOne=0; $('name=height_one').val($valHeightOne)}else{ $('name=height_one').val($valHeightOne)}}

//introducimos altura de la segunda antena

$valHeightTwo=parseFloat($('name=height_two').val());

if(isNaN($valHeightTwo)){ $('name=height_two').val(0);$valHeightTwo=0;}else{if($valHeightTwo <
0){$valHeightTwo=0; $('name=height_two').val($valHeightTwo)}else{ $('name=height_two').val($valHeightTwo)}}

//establecemos formula de distancia entre antenas

$distance=3.57
*(Math.pow(($approx*$valHeightOne),(1/2))+Math.pow(($approx*$valHeightTwo),(1/2)));

//mostramos valor de la distancia en KM

($('name=distance').val($distance.toFixed(2));

});

//al hacer clic en el botón Graficar

$('#graficar').on('click',function(){

//introducimos altura de la primer antena

$valHeightOne=parseFloat($('name=height_one').val());

if(isNaN($valHeightOne)){ $('name=height_one').val(0);$valHeightOne=0;}else{if($valHeightOne <

```

```
0){$valHeightOne=0;$('[name=height_one]').val($valHeightOne)}else{$('[name=height_one]').val($valHeightOne)}}}
```

```
//introducimos altura de la segunda antena
```

```
$valHeightTwo=parseFloat($('name=height_two').val());
```

```
if(isNaN($valHeightTwo)){($('name=height_two').val(0);$valHeightTwo=0;}else{if($valHeightTwo < 0){$valHeightTwo=0;$('[name=height_two]').val($valHeightTwo)}else{$('[name=height_two]').val($valHeightTwo)}}}
```

```
//establecemos formula de distancia entre antenas
```

```
$distance=3.57
```

```
*(Math.pow(($approx*$valHeightOne),(1/2))+Math.pow(($approx*$valHeightTwo),(1/2)));
```

```
//mostramos valor de la distancia en KM
```

```
($('name=distance').val($distance.toFixed(2)));
```

```
//graficamos antenas
```

```
distanceAntenna();
```

```
});
```

```
//función grilla, antenas y distancia
```

```
function distanceAntenna(){
```

```
//reiniciamos graficas
```

```
clearCanvas();
```

```
// Ancho de la grilla
```

```
bw = 960;
```

```
// alto de la grilla
```

```
bh = 270;
```

```
// margen interno
```

```
p = 20;
```

```
ctx.beginPath();
```

```
//escalas para grilla
```

```
$maximumHeight=($valHeightOne >= $valHeightTwo)? $valHeightOne:$valHeightTwo;
```

```

escalaHeight=parseFloat($maximumHeight+(($maximumHeight >= 50)? 100:15));

scaleHeiDown=parseFloat((escalaHeight)/6);

scaleHeight=escalaHeight;

ctx.beginPath();

//creamos las líneas

ctx.moveTo(40,40);

ctx.lineTo(40, bh+10 );

ctx.moveTo(bw+p,40);

ctx.lineTo(bw+p, bh+10 );

ctx.setLineDash([1, 0]);

//color de las líneas

ctx.strokeStyle = "#F2F2F2";

//mostramos

ctx.stroke();

//dibujamos la grilla respecto a líneas horizontales

for (x = 20; x <= bh; x += 40) {

ctx.beginPath();

//creamos las líneas

ctx.moveTo(20+p, 0.5 + x + p);

ctx.lineTo(bw + p, 0.5 + x + p);

//establecemos fuente, tamaño y color de letra

ctx.font = '10px serif';

ctx.fillStyle='#000';

ctx.textAlign="right";

//establecemos valores para eje Y

ctx.fillText(Math.round(scaleHeight * 100) /100 , 35, (x+22));

```

```

scaleHeight=parseFloat(scaleHeight)-parseFloat(scaleHeiDown);

ctx.setLineDash([1, 0]);

//color de las líneas
ctx.strokeStyle = "#F2F2F2";

//mostramos
ctx.stroke();
}

//distancia entre antenas
$AntPos=0
$startPos=0;

//a mayor altura separamos las antenas un poco más
for($i=0; $i <= 260; $i+=20){
$startPos=$startPos+25;
if($i <= $distance){$AntPos=$startPos;}
}

//altura de antenas
drawHeight=parseFloat($maximumHeight+(($maximumHeight >= 50)? 100:15));
porcentHeight1=($valHeightOne * 100) / drawHeight;
porcentHeight2=($valHeightTwo * 100) / drawHeight;
lineHeight1=bh-(((bh-45)/100)*(porcentHeight1));
lineHeight2=bh-(((bh-45)/100)*(porcentHeight2));

//altura máxima
maxHeight=(lineHeight1 <= lineHeight2)? lineHeight1: lineHeight2;

//antena uno
ctx.beginPath();

```

```

//creamos un gradiente de color

var my_gradient=ctx.createLinearGradient(0,0,0,900);

posAntenna1=510-$AntPos;

my_gradient.addColorStop(0,"black");
my_gradient.addColorStop(1,"white");

ctx.fillStyle=my_gradient;

//generamos un triangulo para simular una antena

ctx.moveTo(0+posAntenna1,lineHeight1);

ctx.lineTo(-10+posAntenna1,280);

ctx.lineTo(10+posAntenna1,280);

ctx.fill();


//generamos la parábola de la antena

ctx.beginPath();

ctx.strokeStyle = "red";

ctx.fillStyle = "#4169E1";

ctx.arc(posAntenna1,lineHeight1+3,6,1,5.3);

ctx.fill();

ctx.stroke();

//antenas dos

ctx.beginPath();


//creamos un gradiente de color

var my_gradient2=ctx.createLinearGradient(0,900,0,0);

posAntenna2=510+$AntPos;

my_gradient2.addColorStop(0,"white");
my_gradient2.addColorStop(1,"black");

ctx.fillStyle=my_gradient2;


//generamos un triángulo para simular una antena

ctx.moveTo(0+posAntenna2,lineHeight2);

ctx.lineTo(-10+posAntenna2,280);

```

```

ctx.lineTo(10+posAntenna2,280);

ctx.fill();

//generamos la parábola de la antena

ctx.beginPath();

ctx.strokeStyle = "#000";

ctx.fillStyle = "#4169E1";

ctx.arc(posAntenna2, lineHeight2+3, 6, Math.PI+1,2.1, false);

ctx.fill();

ctx.stroke();

//superficie terrestre

ctx.beginPath();

var my_gradient3=ctx.createLinearGradient(0,0,0,800);

my_gradient3.addColorStop(0,"green");

my_gradient3.addColorStop(1,"white");

ctx.strokeStyle = "#000";

ctx.fillStyle = my_gradient3;

ctx.rect(40, 280, 940,140);

ctx.fill();

ctx.stroke();

//textos

ctx.beginPath();

ctx.font = '16px serif';

ctx.fillStyle='#000';

ctx.fillText("Superficie Terrestre",573, 350);

//texto antena1

ctx.font = '10px serif';

ctx.fillText("Antena 1 ("+$valHeightOne+"mts)",posAntenna1-15,lineHeight1+5);

//texto antena2

ctx.font = '10px serif';

```



```

ctx.textAlign="left";

ctx.fillText("Antena 2 ("+$valHeightTwo+"mts)",posAntenna2+15, lineHeight2+5);

//línea de distancia entre las antenas

ctx.strokeStyle = "red";

ctx.moveTo(posAntenna1,maxHeight-20);

ctx.lineTo(posAntenna2,maxHeight-20);

ctx.moveTo(posAntenna1,maxHeight-30);

ctx.lineTo(posAntenna1,maxHeight-10);

ctx.moveTo(posAntenna2,maxHeight-30);

ctx.lineTo(posAntenna2,maxHeight-10);

ctx.font = '10px serif';

ctx.fillStyle='#000';

ctx.textAlign="center";

ctx.fillText($distance.toFixed(2)+" km",510, maxHeight-30);

ctx.fill();

ctx.stroke();

}

//función para resetear canvas

function clearCanvas(){

ctx.beginPath(); // clear existing drawing paths

ctx.save(); // store the current transformation matrix

// Use the identity matrix while clearing the canvas

ctx.setTransform(1, 0, 0, 1, 0, 0);

ctx.clearRect(0, 0, antennaDraw.width, antennaDraw.height);

ctx.restore(); // restore the transform

}

```

## Conclusión

En la aplicación de Ganancia de una Antena, utilizando los valores de 2 metros (ver Figura 1. 4.3) y 5 metros (ver Figura 1. 4.4) de diámetro, con una misma frecuencia de trabajo (12 GHz), observamos que, a mayor diámetro de la antena obtendremos mayor ganancia.

En la aplicación de distancia máxima entre dos antenas (ver Figura 1. 4.5 y 4.6), podemos observar que, la distancia máxima entre dos antenas depende de la altura de cada antena, a mayor altura, mayor distancia, esto nos posibilita, entre otras cosas, saber la limitante de distancia y que equipos electrónicos/informáticos serían los óptimos.

## **Laboratorio 5: Instalación de Linux CentOS (Servidor)**

### **Breve introducción teórica del tema**

Cómo introducción se expondrá los pasos para la instalación del sistema operativo Linux CentOS solicitado por la cátedra, en una máquina virtual (en este caso de Oracle Virtual Machine -VirtualBox-).

### **Entrando a detalle**

CentOS es *Community Enterprise Operating System* que en español se traduce como "Sistema operativo para empresas basados en los aportes de la comunidad". Y es un proyecto opensource abierta a la comunidad que lo creó basado en el código fuente liberado por Red Hat.

### **Algunas de sus características principales de CentOS son:**

- Distribución de Linux estable.
- Alto rendimiento y disponibilidad.
- Elevado nivel de seguridad.
- Fiabilidad y Estabilidad.
- Orientado a Servidores.
- Libre y Gratuito.

Para poder realizar su instalación hay que tener disponible e instalado el programa VirtualBox.

Luego poseer un ISO de instalación para proseguir.

# Instalación de una máquina virtual

## Primera etapa: crear máquina virtual en Virtual Box

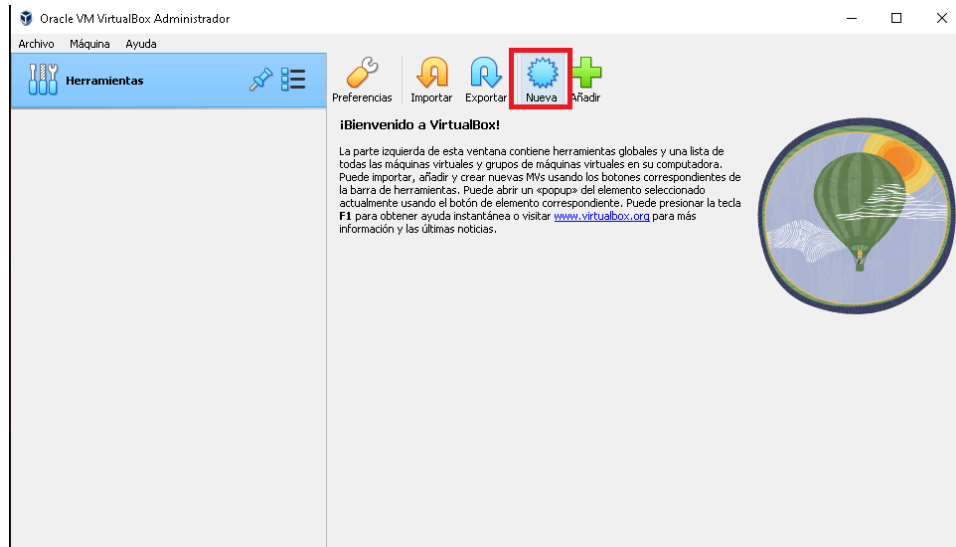


Figura 5.1: añadir nueva máquina

Primero debemos dar clic en nueva.

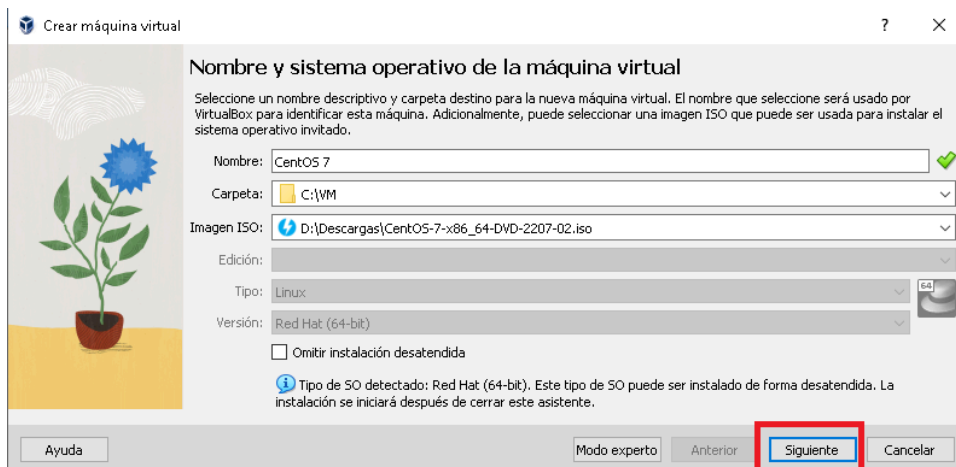


Figura 5.2: añadir nueva máquina

A continuación, asignamos un nombre, la ubicación de la máquina y el ISO correspondiente y proseguimos con “Siguiente”.

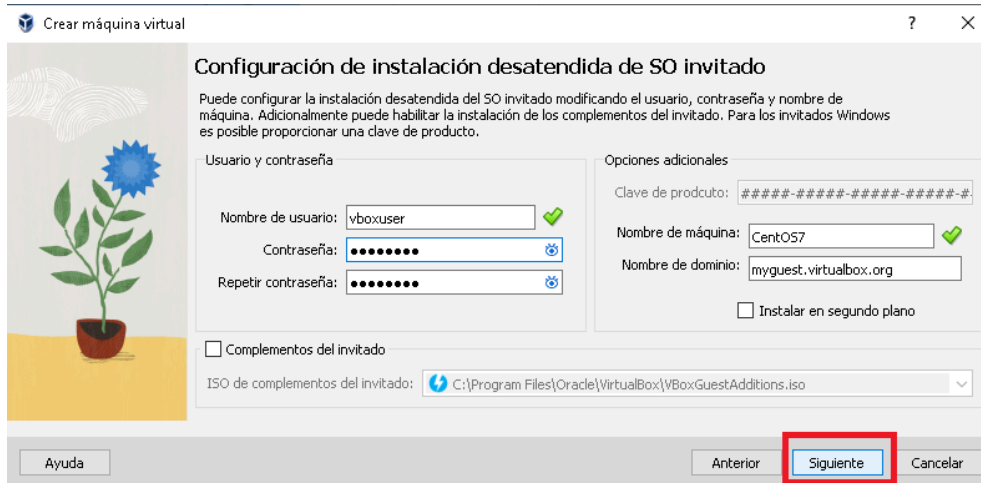


Figura 5.3: agregar nombre de usuario y de máquina

Damos en siguiente ya que este apartado correspondería a temas de seguridad propios de virtualbox.

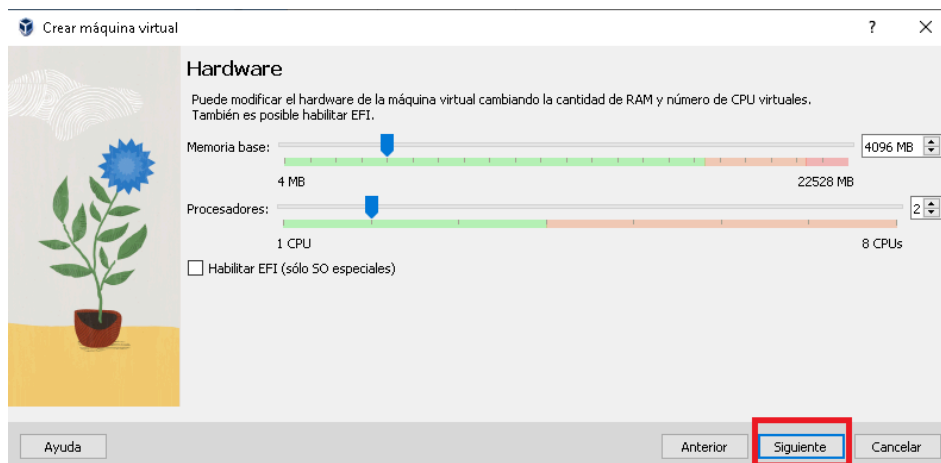


Figura 5.4: asignar memoria y procesador a la máquina virtual

Asignamos memoria RAM total y cantidad de núcleos.

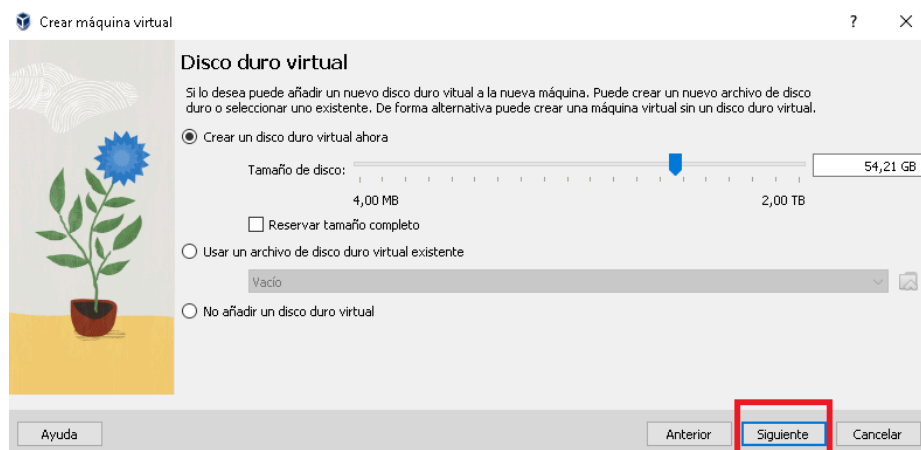


Figura 5.5: asignar espacio en disco duro

Determinamos el tamaño total del disco virtual.

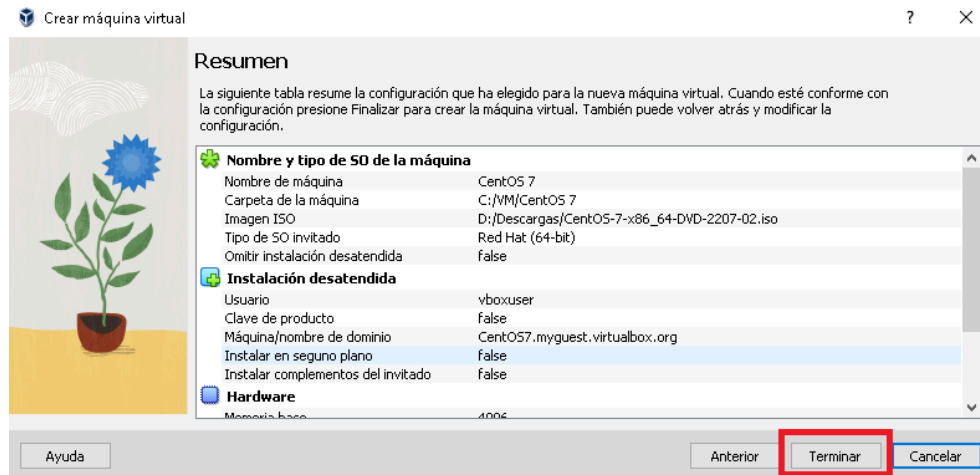


Figura 5.6: terminar de crear máquina virtual

Finalizamos con el botón “Terminar”

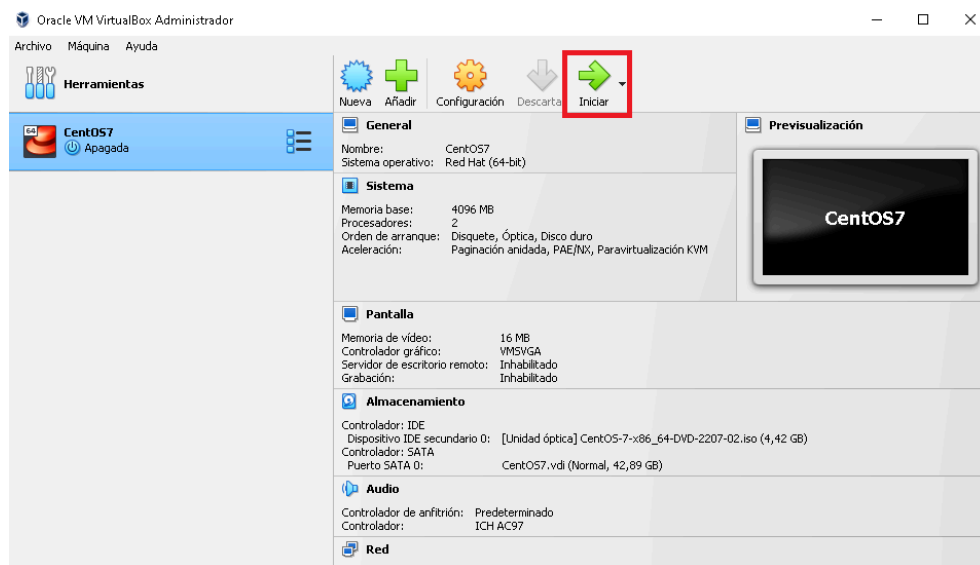


Figura 5.7: inicializar máquina virtual

Finalmente damos en iniciar finalizando el procedimiento a realizar por virtualbox.

## Segunda etapa: instalación

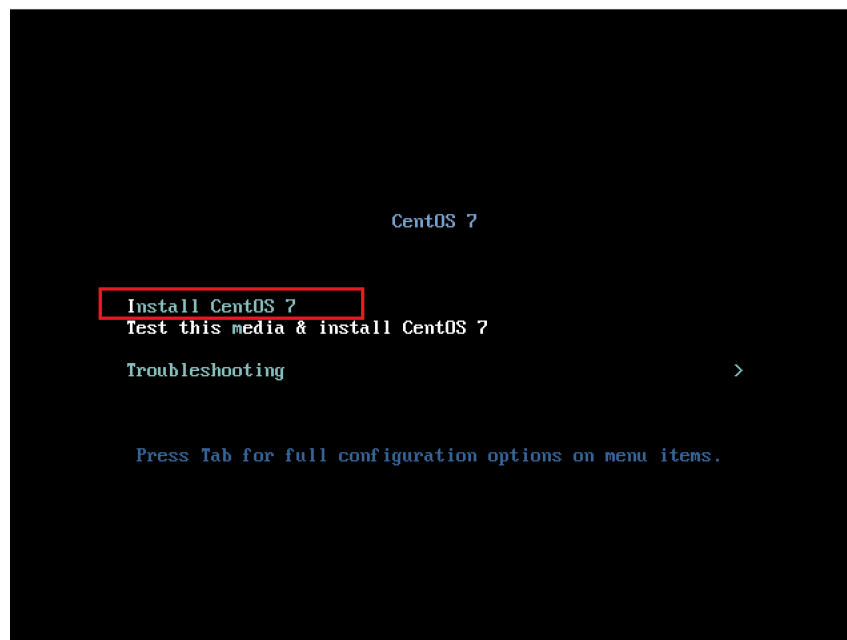


Figura 5.8: instalar CentOS 7

Damos enter en la opción resaltada.

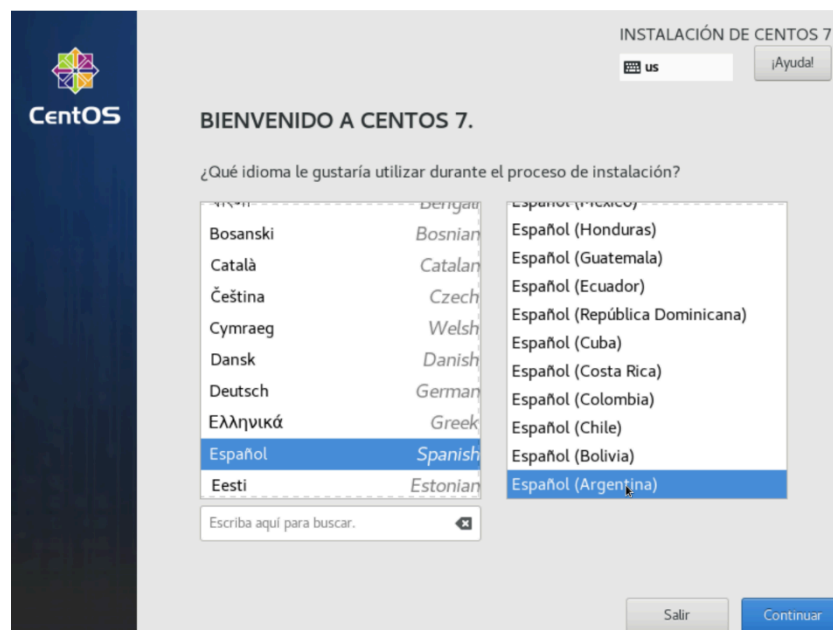


Figura 5.9: elegir idioma de la máquina

Determinamos idioma y continuamos



Figura 5.10: realizar configuraciones iniciales

Realizamos las configuraciones necesarias: en este caso se resaltó antes del estado que se muestra en la imagen, determinar horario, teclado y destino de la instalación.



Figura 5.11: determinar contraseña del usuario ROOT

Mientras la instalación prosigue, se solicitará que se determine contraseña de acceso ROOT y un usuario





Figura 5.12: finalizar la instalación de CentOS 7

Finalizada la instalación reiniciamos con el botón.

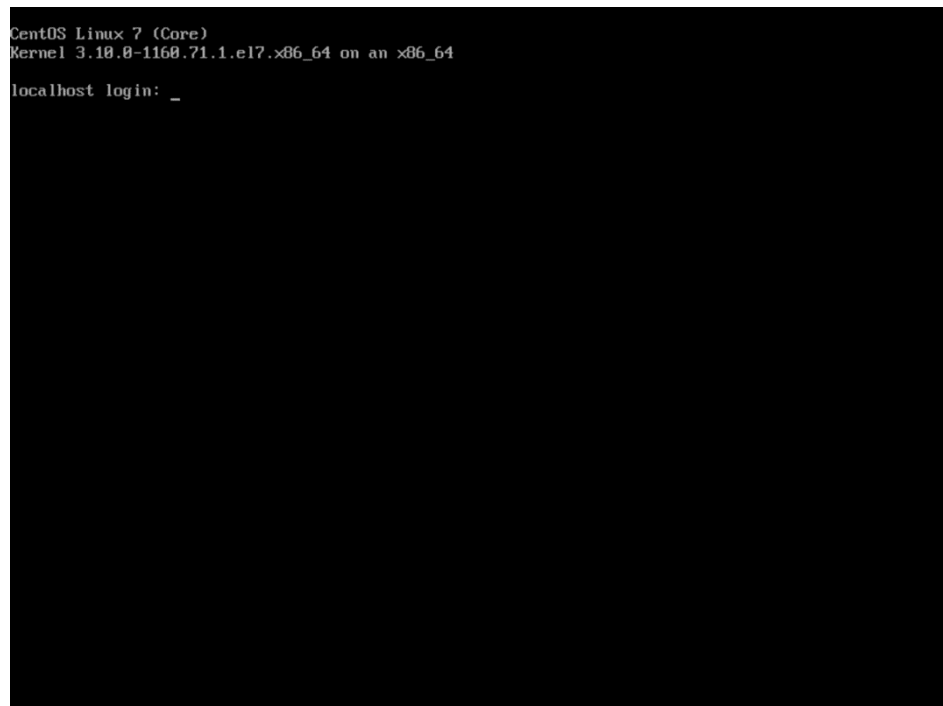


Figura 5.13: máquina inicializada

Y listo. Accedemos con el usuario establecido y luego podremos presionar “Tab” para ver las instrucciones disponibles.

## **Conclusión**

La conclusión a la que llegamos es que CentOS es de instalación fácil siempre que se lo configure correctamente. La lista de comandos con los que se dispone nos ayuda a aplicar el conocimiento adquirido en esta cátedra. El sistema busca ser de mayor utilidad al momento de manipular servidores de manera rápida y segura en el mercado permitiendo al usuario establecer los comandos e instrucciones necesarias para dicho fin.