# TECNOLÓGICO NACIONAL DE MÉXICO

## INSTITUTO TECNOLÓGICO DE OAXACA

**ASIGNATURA:** FUNDAMENTOS DE TELECOMUNICACIONES

**CATEDRÁTICO:** MCC. VALVERDE JARQUÍN REYNA

**ALUMNO:** GARCÍA GARCÍA JOSÉ ÁNGEL

**GRUPO:** ISB

**HORA:** 12:00 – 13:00

**CARRERA:** INGENIERÍA EN SISTEMAS COMPUTACIONALES

**REPORTE DEL TRABAJO DE UNIDAD 3**

OAXACA DE JUÁREZ, OAX, 07/DICIEMBRE/2020

# ÍNDICE GENERAL

ÍNDICE DE IMÁGENES....2
PLANTEAMIENTO DEL PROBLEMA....3
DISEÑO DE LA SOLUCIÓN....3
1.- Manchester....3
2.- Manchester Diferencial....6
3.- Graficar....9
4.- Función del Botón Graficar....12
HERRAMIENTAS UTILIZADAS....14
CÓDIGO....14
RESULTADOS OBTENIDOS....20
ENLACE DEL VÍDEO....30
CONCLUSIÓN....31

# ÍNDICE DE IMÁGENES

Ilustración 1 - Transiciones en Manchester .....3
Ilustración 2 - Función para modulación Manchester .....5
Ilustración 3 - Función para modulación Manchester Diferencial.....8
Ilustración 4 - Función para graficar .....11
Ilustración 5 - Función para el botón graficar.....13
Ilustración 6 - Interfaz grafica.....20
Ilustración 7 - Graficas del libro .....21
Ilustración 8 - Grafica de prueba del libro de Manchester.....21
Ilustración 9 - Grafica de prueba del libro de Manchester Diferencial.....22
Ilustración 10 - Grafica de ambas modulaciones de prueba del libro .....23
Ilustración 11 - Grafica del foro de Manchester .....24
Ilustración 12 - Grafica de prueba del foro de Manchester .....24
Ilustración 13 - Grafica del foro de Manchester Diferencial .....25
Ilustración 14 - Grafica de prueba del foro de Manchester Diferencial .....25
Ilustración 15 - Ambas graficas de prueba del foro.....26
Ilustración 16 - Graficas de modulaciones de 1 0 1 0 0 1 0 .....27
Ilustración 17 - Graficas de modulaciones de 1 1 1 0 0 1 1 .....28
Ilustración 18 - Graficas de modulaciones de 1 0 0 1 1 1 1 1 0 0 1 1 1 1 .....29

# PLANTEAMIENTO DEL PROBLEMA

Para realizar la práctica de esta unidad, el catedrático nos brindó la información necesaria para trabajar mediante la plataforma Moodle, que consistía en una breve descripción de las modulaciones que debíamos desarrollar. Ya luego se nos explicó a mayor detalle mediante una vídeo llamada a través de Meet.

El trabajo o problema solicitado, es crear un programa en Matlab que, al introducir un tren de bits, se le aplique la modulación Polar Manchester y Manchester Diferencial, mostrando así la gráfica de la señal ya modulada de cada una de estas. Entonces, la idea es sencilla, simular estas modulaciones a partir de un tren de bits.

# DISEÑO DE LA SOLUCIÓN

Para darle solución a lo planteado anteriormente fue relativamente sencillo, ya que en clases anteriores se nos había explicado sobre estas modulaciones, inclusive realizamos algunos ejercicios teóricos y prácticos de tarea para reforzar mucho más el conocimiento sobre estas.

La función que se empleó de Matlab para graficar fue:

**plot**: Nos sirve para graficar en 2D y se utilizó para representar la senoidal, específicamente para la gráfica analógica con un periodo diferente de 0 y está representada en el dominio de tiempo.

Es la única herramienta para realizar los gráficos, lo interesante fue el poder crear esa función para graficar.

## 1.- Manchester

Manchester consiste en usar la inversión en mitad de cada intervalo de bit para sincronizar y para representar bits, no tiene retorno a cero, además que en esta se realizan transiciones, existen sólo dos transiciones que se pueden realizar. Estas transiciones que se pueden realizar son:

- Transición de negativo a positivo que representa el 1
- Transición de positivo a negativo que representa el 0

De tal forma que podemos tener las siguientes representaciones para cada transición de forma gráfica:

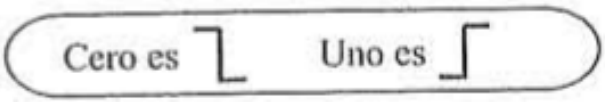


*Ilustración 1 - Transiciones en Manchester*

- Vamos a tener una función que se llama Manchester y recibirá *bits* que es el tren de bits al que vamos a modularizar. Y *app* es para poder graficar en nuestras graficas.

```
function results = Manchester(app, bits)
```

- Lo primero es que declaramos un arreglo que contendrá el par de transiciones a realizar.

```
valTrans = [];
```

- Vamos recorrer toda nuestra taza de bits, para poder obtener la transición de cada bit. Para eso vamos a hacer uso de for, entonces tendremos algo así:

```
for i = 1 : length(bits)
```

  - ✓ Ahora, verificamos si el bit que está actualmente es un 1 o es un 0, porque cómo vimos anteriormente, cada bit tiene diferente transición. La transición es un vector generado entre dos límites y solo es temporal para rellenar el vector de transacciones.

    - ❖ Verificamos si el bit actual es un cero.

```
if bits(i) == 0
```

    - ❖ Si es un cero, entonces su transición cómo vinos al principio, va de positivo a negativo, y cómo este método no tiene retorno a cero, podemos plantear una transición del valor 1 a -1.

```
transicion = [1 -1];
```

    - ❖ Pero, si esto es falso, es decir, estábamos diciendo que el bit es un 1.

```
else
```

    - ❖ Entonces lo que hacemos, es aplicar una transición de negativo a positivo, lo contrario a la transición del bit 0. Por lo que nuestra transición va del valor -1 a 1.

```
transicion = [-1 1];
```

    - ❖ Ahora, hemos terminado de evaluar las dos transiciones, entonces cerramos las sentencias de comparación.

```
end
```

  - ✓ Por último, vamos a agregar la transición calculada a nuestro arreglo de transiciones. Esto con la finalidad de almacenar todas las transiciones posibles.

```
valTrans = [ valTrans transicion];
```

- Ahora, sólo cerramos nuestro for, que se encarga de iterar todo nuestro tren de bits.

```
end
```

- Ahora vamos a usar nuestra función graficar, esta será explicada más adelante. Le pasamos un 1 para indicar que corresponde al método de Manchester.

```
graficar(app, 1, length(bits), valTrans);
```

- Finalmente, cerramos nuestra función.

```
end
```

De tal forma que nuestra función para realizar la modulación de Manchester queda así:

```
function results = Manchester(app, bits)
    valTrans = [];
    for i = 1 : length(bits)
        if bits(i) == 0
            transicion = [1 -1];
        else
            transicion = [-1 1];
        end
        valTrans = [ valTrans transicion];
    end
    graficar(app, 1, length(bits), valTrans);
end
```

*Ilustración 2 - Función para modulación Manchester*

Hasta ahora, ya hemos terminado lo que respecta a la modulación Manchester, debido a que ya tenemos los valares de las transiciones, listos para ser graficados. La función que nos ayuda a graficar la vamos a ver más adelante, ya que será usada igual para Manchester Diferencial.

## 2.- Manchester Diferencial

Esta modulación es similar a la anterior, ya que se basa en la transición. Pero para esta, la representación de cada bit es diferente, de modo que se tienen las siguientes reglas o consideraciones:

- Una transición significa un 0 binario.
- La ausencia de una transición sígnica un 1 binario.

Ahora, dicho lo anterior podemos partir de la misma forma, ya que igual tenemos dos transiciones, pero se aplican en casos diferentes si lo comparamos con el Manchester.

- Declaramos nuestra función, en la que *bits* es el tren de bits y *app* es para la grafica en la que hará que corresponde a las de la app creada.

```
function results = ManchesterDiferencial(app, bits)
```

- Declaramos nuestro arreglo que contendrá los valores de las transiciones. Y consideramos que nuestro tren de bits se llama bits, al igual que el de Manchester.

```
valTrans = [];
```

- Debido a que podemos tener una transición inicial, entonces debemos validar primero con que valor empieza el tren de bits. Si el valor del primer bit es un 0.

```
if bits(1) == 0
```

  - ✓ Nos indica que debemos realizar una transición de negativo a positivo, es decir, de -1 a 1.

```
transicion = [-1 1];
```

- En casa contrario

```
else
```

  - ✓ Si es falso, entonces decimos que el primer valor del tren de bits es 1. Para esto hacemos una transición de positivo a negativo, haciendo así el valor de transición de 1 a -1.

```
transicion = [1 -1];
```

- Cerramos nuestro if que verificaba cómo iniciar.

```
end
```

- Ahora guardamos esa primera transición en nuestro vector de transiciones.

```
valTrans = [ valTrans transicion];
```

- Vamos a declarar una variable que nos permitirá obtener el ultimo valor de la transición del bit anterior al que estamos.

```
k = 2;
```

- Procedemos a iterar nuestro tren de bits que se llama bits. Entonces hacemos una de la estructura for. Nótese que lo empezamos desde el segundo bit, porque ya hemos evaluado el primer bit.

```
for i = 2 : length(bits)
```

  - ✓ Iterando nuestro tren de bits, verificamos si apartir del segundo bit, es un 0.

```
if bits(i) == 0
```

    - ❖ Si el bit, es un 0, entonces esto nos indica que existen una transición, entonces lo que se hace es invertir la transición del bit anterior. Se observa que hacemos uso de nuestra variable k, que nos ayuda a poder obtener las transiciones necesarias.

```
tran = [valTrans(k - 1) valTrans(k)];
```

  - ✓ Si nuestro bit no es un 0, entonces es 1.

```
else
```

    - ❖ Lo que se hace es seguir con la misma transición del bit anterior, entonces obtenemos dichos valores de nuestro vector de transiciones y nos ayudamos de nuestra variable k.

```
tran = [valTrans(k) valTrans(k - 1)];
```

  - ✓ Terminamos de comparar nuestros bits.

```
end
```

  - ✓ Incrementamos nuestra variable k en 2, esto para obtener el siguiente valor de transición de otro bit.

```
k = k + 2;
```

- ✓ Por último, guardamos nuestra transición del bit en nuestro vector de transiciones.

```
valTrans = [valTrans tran];
```

- ➢ Cerramos nuestro for.

```
end
```

- ➢ Vamos a ejecutar nuestra función para graficar. Le pasamos un 0 para indicar que corresponde al método de Manchester Diferencial.

```
graficar(app, 0, length(bits), valTrans);
```

- ➢ Cerramos nuestra función.

```
end
```

Ya hemos terminado de obtener nuestros valores de nuestras transacciones. De tal forma que nuestro método para Manchester diferencial queda de la forma:

```
function results = ManchesterDiferencial(app, bits)
        valTrans = [];
        if bits(1) == 0
            transicion = [-1 1];
        else
            transicion = [1 -1];
        end
        valTrans = [ valTrans transicion];
        k = 2;
        for i = 2 : length(bits)
            if bits(i) == 0
                tran = [valTrans(k - 1) valTrans(k)];
            else
                tran = [valTrans(k) valTrans(k - 1)];
            end
            k = k + 2;
            valTrans = [valTrans tran];
        end
        graficar(app, 0, length(bits), valTrans);
    end
end
```

*Ilustración 3 - Función para modulación Manchester Diferencial*

## 3.- Graficar

Hasta ahora hemos detallado los métodos para obtener los vectores de transición de cada modulación, entonces nos faltaría graficarlos. Para esto vamos a crear un método que, a partir de la gráfica (*app*), el número del método a graficar (*val*), el número de bits que tiene el tren de bits (*n*) y el vector de transiciones (*valTrans*), puede realizar el grafico correspondiente.

- La cabecera de nuestra función entonces se vería algo así:

```
function results = graficar(app, val, n, valTrans)
```

- Primero, declaramos nuestra variable para poder obtener los valores del vector de transición.

```
k = 1;
```

- Luego declaramos una variable que aumentará de 0.5 en 0.5, esto para poder graficar en la mitad de cada intervalo de transición.

```
l = 0.5;
```

- Creamos nuestro vector de tiempo, para poder realizar la gráfica.

```
T = 0 : 0.001 : length(bits);
```

- La idea, es poder recorrer todo nuestro vector de transiciones e ir graficando en términos del mismo. La idea es crear una función que almacene todos los puntos de nuestro valor de transición y esa función es la que vamos a graficar.

```
for j = 1: length(T)
```

  - ✓ Vamos a guardar en nuestra función y, en el índice de j el valor de nuestra transación que esté en el indice de la variable k. Notese que k no incrementa como lo hace j, por lo que en varias iteraciones k permanece constante.

```
y(j) = valTrans(k);
```

  - ✓ Verificamos si nuestro valor del vector de tiempo en j, es mayor que nuestro valor actual de l. Esto para indicar que cambie la posición para graficar y así graficar el siguiente desplazamiento o transición.

```
if T(j) > l
```

- ❖ Incrementamos k en 1 para poder indicar que ya vamos en la siguiente transición.

```
k = k + 1;
```

- ❖ Incrementamos l para en 0.5 para indicar que ya graficaremos en otro valor del tiempo.

```
l = l + 0.5;
```

- ✓ Cerramos el if.

```
end
```

- ➢ Cerramos el for, ya hemos obtenido todos los puntos para graficar y se encuentra en la variable y.

```
end
```

- ➢ Ahora, ya sabemos que graficar, es decir, ya sabemos cómo se verá la señal que ya fue modularizada anteriormente. Pero nos falta saber en dónde graficarla, ya que cómo mencioné al principio, este método sirve para ambas modulaciones.
- ➢ Comparamos el valor que pedimos de entrada, que corresponde a la variable val, si el valor es 1 entonces se refiere a una modulación de Manchester y si es falso, entonces es una modulación de Manchester Diferencial.
- ➢ Verificamos si valor de la variable val corresponde a 1.

```
if val == 1
```

- ✓ Si el valor es 1, entonces vamos a indicar que se graficará en la gráfica de nombre *GraficaM* que corresponde a la de Manchester, le indicamos en que valores del tiempo que corresponde a nuestro vector de tiempo *T*, además le indicamos la variable *y* que es nuestra función y por último que grafique de color rojo la línea.

```
plot(app.GraficaM, T, y, 'r');
```

- ✓ También, limitamos los variables del cuadrante para nuestra grafica que corresponde a Manchester y es *GraficaM* para que muestre desde -2 y 2 en la parte de la amplitud (Eje Y) y que muestre desde 0 hasta el número de bits el tiempo (Eje X).

```
axis(app.GraficaM, [0 length(bits) -2 2]);
```

- Si el valor de la variable val es diferente de 1, entonces quiere decir que debemos graficar para una modulación de Manchester Diferencial.

```
else
```

  - ✓ Al igual que lo mencionado para la grafica de Manchester, solo que ahora lo que haremos es graficar en *GraficaMD* que corresponde a la gráfica de la modulación Manchester Diferencial.

```
plot(app.GraficaMD, T, y, 'r');
```

  - ✓ Igual que para la grafica de Manchester, solo que ahora lo aplicaremos en *GraficaMD* que corresponde a la grafica de la modulación Manchester Diferencial.

```
axis(app.GraficaMD, [0 length(bits) -2 2]);
```

  - ✓ Por último, solamente cerramos la comparación.

```
end
```

- Cerramos nuestra función finalmente.

```
end
```

De tal forma que nuestra función para graficar y con el nombre *graficar* se visualiza de la siguiente forma:

```
function results = graficar(app, val, n, valTrans)
    k = 1;
    l = 0.5;
    T = 0 : 0.001 : n;
    for j = 1: length(T)
        y(j) = valTrans(k);
        if T(j) > l
            k = k + 1;
            l = l + 0.5;
        end
    end
    if val == 1
        plot(app.GraficaM, T, y, 'r');
        axis(app.GraficaM, [0 n -2 2]);
    else
        plot(app.GraficaMD, T, y, 'r');
        axis(app.GraficaMD, [0 n -2 2]);
    end
end
```

*Ilustración 4 - Función para graficar*

Hasta este punto ya tenemos las modulaciones y el método para graficar. Entonces lo que nos falta es poder darle el tren de bits a dichas modulaciones.

## 4.- Función del Botón Graficar

La idea es tener una función que nos permita obtener el tren de bits del usuario, y también saber que modulación graficar.

- Declaramos una función que recibe *app* que corresponde a la app del programa y un *event* que son los eventos de los componentes.

```
function BtnGraficarButtonPushed(app, event)
```

- Ahora, lo primero es obtener el tren de bits introducido por el usuario, esto es introducido en nuestro elemento *TrendebitsEditField* que es un componente para introducir datos.

```
entrada_tren = app.TrendebitsEditField.Value;
```

- Hasta este momento tenemos el tren de bits, pero lo tenemos en formato de caracteres, entonces lo debemos convertir a una matriz de 1 y 0.

```
bits = str2num(entrada_tren);
```

- Ahora vamos a obtener la modulación a graficar.

```
op = app.SelMod.Value;
```

- Vamos a limpiar las gráficas y los campos.

```
BtnBorrarButtonPushed(app, event);
```

- Cómo limpiamos todos los campos, vamos a volver a colocar en la caja de entrada el tren de bits que había introducido el usuario.

```
app.TrendebitsEditField.Value = entrada_tren;
```

- Verificamos la variable *op* para verificar si se va a aplicar la modulación Manchester.

```
if strcmpi(op,'Manchester')
```

- ✓ Ejecutamos nuestra función *Manchester* que habíamos explicado anteriormente.

```
Manchester(app, bits);
```

➢ Si no es la modulación Manchester, entonces verificamos si el valor de la variable *op* corresponde a 'Ambas', es decir, si vamos a graficar las dos modulaciones.

```
elseif strcmpi(op,'Ambas')
```

- ✓ Ejecutamos nuestra función de *ManchesterDiferencial* que creamos.

```
ManchesterDiferencial(app, bits);
```

- ✓ Ejecutamos nuestra función *Manchester* que habíamos explicado anteriormente.

```
Manchester(app, bits);
```

➢ Si esto es falso, quiere decir que no es ninguna de las anteriores opciones.

```
else
```

- ✓ Entonces, esto nos indica que debemos ejecutar nuestra función de *ManchesterDiferencial* porque es nuestro único caso que ejecutaríamos si no es ninguno de los anteriores.

```
ManchesterDiferencial(app, bits);
```

➢ Finalmente cerramos nuestra función.

```
end
```

La función entonces queda:

```
function BtnGraficarButtonPushed(app, event)
    entrada_tren = app.TrendebitsEditField.Value;
    bits = str2num( entrada_tren );
    op = app.SelMod.Value;
    BtnBorrarButtonPushed( app, event );
    app.TrendebitsEditField.Value = entrada_tren;
    if strcmpi( op, 'Manchester' )
        Manchester( app, bits );
    elseif strcmpi( op,'Ambas' )
        ManchesterDiferencial( app, bits );
        Manchester( app, bits );
    else
        ManchesterDiferencial( app, bits );
    end
end
```

*Ilustración 5 - Función para el botón graficar*

## HERRAMIENTAS UTILIZADAS

- Computadora con sistema operativo Windows o donde te permite tener Matlab.
- Matlab versión R2020a
- Libro proporcionado por el catedrático.

## CÓDIGO

```
classdef GarciaGarciaJoseAngel_TrabajoUnidad3 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                matlab.ui.Figure
        GraficaM                matlab.ui.control.UIAxes
        GraficaMD               matlab.ui.control.UIAxes
        BtnGraficar             matlab.ui.control.Button
        GarcaGarcaJosngelLabel  matlab.ui.control.Label
        Label                   matlab.ui.control.Label
        BtnBorrar               matlab.ui.control.Button
        TrendebitsLabel         matlab.ui.control.Label
        TrendebitsEditField     matlab.ui.control.EditField
        ModulacinDropDownLabel  matlab.ui.control.Label
        SelMod                  matlab.ui.control.DropDown
    end


    methods (Access = private)

        function results = graficar(app, val, n, valTrans) % Función para graficar
            k = 1; % Variable para recorrer el vector de transiciones de la
modulación
            l = 0.5; % Variable para graficar la transición en intervalores de 0.5
            T = 0 : 0.001 : n; % Vector de tiempo
            for j = 1: length( T ) % Recorremos todo el vector de tiempo
                y( j ) = valTrans( k ); % Guardamos en nuestra función y la
transición correspondiente
                if T( j ) > l % Verificamos si ya tenemos que camibar de transición
                    k = k + 1; % Incrementamos k, así cambiaremos de transición
                    l = l + 0.5; % Incrementamos l que corresponde al lugar de la
otra transición
                end
            end
            if val == 1 % Verificamos si vamos a graficar una modulación Manchester
                plot( app.GraficaM, T, y, 'r' ); % Graficamos nuestra función de
variable y
                axis( app.GraficaM, [ 0 n -2 2 ] ); % Colocamos los valares de los
ejes para visualizar mejor
            else % Nos indica que graificaremos Manchester Diferencial
                plot( app.GraficaMD, T, y, 'r' ); % Graficamos nuestra función de
variable y
```

```
                axis( app.GraficaMD, [ 0 n -2 2 ] ); % Colocamos los valares de los
ejes para visualizar mejor
            end
        end % Fin de nuestra función graficar

        function results = Manchester(app, bits) % Función para obtener
transiciones de modulación Manchester
            valTrans = []; % Declaramos un vector de transiciones
            for i = 1 : length( bits ) % Recorremos nuestro tren de bits
                if bits( i ) == 0 % Verificamos si el bit actual es un 0
                    transicion = [ 1 -1 ]; % Realizamos una transición de poisitivo
a negativo
                else % Nos indica que es un bit 1
                    transicion = [ -1 1 ]; % Realizamos una transición de negativo
a positivo
                end
                valTrans = [ valTrans transicion]; % Agregamos la transicion al
vector de transiciones
            end
            graficar( app, 1, length( bits ), valTrans ); % Graficamos mediante
nuestra función para graficar
        end % Fin de nuestra función Manchester

        function results = ManchesterDiferencial(app, bits) % Función para obtener
transiciones de modulación Manchester Diferencial
            valTrans = []; % Declaramos un vector de los valores de la transición
            if bits( 1 ) == 0  % Verificamos si el primer bit es 0
                transicion = [ -1 1 ]; % Realizamos una transición de negativo a
positivo
            else % Nos indica que el bit es un 1
                transicion = [ 1 -1 ]; % Realizamos una transición de positivo a
negativo
            end
            valTrans = [ valTrans transicion ]; % Agregamos a nuestra transición al
vector de transiciones
            k = 2; % Declaramos una variable que nos permitirá obtener el ultimo
valor de la transición
            for i = 2 : length(bits) % Recorremos nuestro trend e bits
                if bits( i ) == 0 % Si el bit actual es 0
                    tran = [ valTrans( k - 1 ) valTrans( k ) ]; % Mantenemos la
transición del bit anterior
                else % Nos indica que el bit actual es 1
                    tran = [ valTrans( k ) valTrans( k - 1 ) ]; % No hay transición
y realizamos la inversión de la transición del bit anterior
                end
                k = k + 2; % Incrementos nuestra variable
                valTrans = [ valTrans tran ]; % Agregamos la transición al vector
de transiciones
            end
            graficar( app, 0, length( bits ), valTrans ); % Graficamos mediante
nuestra función para graficar
        end
    end % Fin de nuestra función Manchester Diferencial
```

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

        % Button pushed function: BtnGraficar
        function BtnGraficarButtonPushed(app, event)
            entrada_tren = app.TrendebitsEditField.Value; % Obtenemos el tren de
bits como string
            bits = str2num( entrada_tren ); % Mapeamos ese tren de bits de string a
una matriz de 1 y 0
            op = app.SelMod.Value; % Obtenemos que modulación aplicar
            BtnBorrarButtonPushed( app, event ); % Borramos todos los campos
            app.TrendebitsEditField.Value = entrada_tren; % Colocamos nuestros
campo de tren de bits
            if strcmpi( op, 'Manchester' ) % Verificamos si aplicaremos Manchester
                Manchester( app, bits ); % Ejecutamos la modulación de Manchester
            elseif strcmpi( op,'Ambas' ) % Verificamos si aplicaremos ambas
modulaciones
                ManchesterDiferencial( app, bits ); % Ejecutamos la modulación de
Manchester Diferencial
                Manchester( app, bits ); % Ejecutamos la modulación de Manchester
            else % Si no fue ninguna de las anteriores, entonces es la Manchester
Diferencial
                ManchesterDiferencial( app, bits ); % Ejecutamos la modulación de
Manchester Diferencial
            end
        end % Fin de la función para el botón Graficar

        % Button pushed function: BtnBorrar
        function BtnBorrarButtonPushed(app, event)
            app.TrendebitsEditField.Value = ' '; % Limpiamos el campo de tren de
bits
            app.GraficaM.clo; % Limpiamos la grafica de Manchester
            axis( app.GraficaM, [ 0 1 0 1 ] ); % Asignamos los ejes a la grafica de
M
            app.GraficaMD.clo; % Limpiamos la grafica de Manchester Diferencial
            axis( app.GraficaMD, [ 0 1 0 1 ]); % Asignamos los ejes a la grafica de
MD
        end % Fin de la función para el botón de Borrar
    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.Color = [0.3294 0.7608 0.9451];
            app.UIFigure.Position = [100 100 843 699];
            app.UIFigure.Name = 'MATLAB App';

            % Create GraficaM
            app.GraficaM = uiaxes(app.UIFigure);
```

```
            title(app.GraficaM, 'Manchester')
            xlabel(app.GraficaM, 'Tiempo')
            ylabel(app.GraficaM, 'Amplitud')
            app.GraficaM.FontName = 'Consolas';
            app.GraficaM.FontWeight = 'bold';
            app.GraficaM.XGrid = 'on';
            app.GraficaM.TitleFontWeight = 'normal';
            app.GraficaM.BackgroundColor = [0.9294 0.6941 0.1255];
            app.GraficaM.Position = [37 325 770 274];

            % Create GraficaMD
            app.GraficaMD = uiaxes(app.UIFigure);
            title(app.GraficaMD, {'Manchester Diferencial'; ''})
            xlabel(app.GraficaMD, 'Tiempo')
            ylabel(app.GraficaMD, 'Amplitud')
            app.GraficaMD.FontName = 'Consolas';
            app.GraficaMD.FontWeight = 'bold';
            app.GraficaMD.XGrid = 'on';
            app.GraficaMD.BackgroundColor = [0.9294 0.6941 0.1255];
            app.GraficaMD.Position = [37 31 770 272];

            % Create BtnGraficar
            app.BtnGraficar = uibutton(app.UIFigure, 'push');
            app.BtnGraficar.ButtonPushedFcn = createCallbackFcn(app,
@BtnGraficarButtonPushed, true);
            app.BtnGraficar.BackgroundColor = [0.8314 0.4745 0.8314];
            app.BtnGraficar.FontName = 'Consolas';
            app.BtnGraficar.FontWeight = 'bold';
            app.BtnGraficar.Position = [740 643 69 23];
            app.BtnGraficar.Text = 'Graficar';

            % Create GarcaGarcaJosngelLabel
            app.GarcaGarcaJosngelLabel = uilabel(app.UIFigure);
            app.GarcaGarcaJosngelLabel.FontSize = 14;
            app.GarcaGarcaJosngelLabel.FontWeight = 'bold';
            app.GarcaGarcaJosngelLabel.FontAngle = 'italic';
            app.GarcaGarcaJosngelLabel.Position = [632 1 175 22];
            app.GarcaGarcaJosngelLabel.Text = 'García García José Ángel';

            % Create Label
            app.Label = uilabel(app.UIFigure);
            app.Label.HorizontalAlignment = 'center';
            app.Label.FontName = 'Consolas';
            app.Label.FontSize = 14;
            app.Label.FontWeight = 'bold';
            app.Label.FontAngle = 'italic';
            app.Label.Position = [1 673 842 24];
            app.Label.Text = 'Fundamentos de Telecomunicaciones - Graficador de
Submodulación de Polar Manchester y Manchester Diferencial';

            % Create BtnBorrar
            app.BtnBorrar = uibutton(app.UIFigure, 'push');
            app.BtnBorrar.ButtonPushedFcn = createCallbackFcn(app,
@BtnBorrarButtonPushed, true);
            app.BtnBorrar.BackgroundColor = [0.898 0.9294 0.298];
```

```
            app.BtnBorrar.FontName = 'Consolas';
            app.BtnBorrar.FontWeight = 'bold';
            app.BtnBorrar.Position = [740 610 67 23];
            app.BtnBorrar.Text = 'Borrar';

            % Create TrendebitsLabel
            app.TrendebitsLabel = uilabel(app.UIFigure);
            app.TrendebitsLabel.HorizontalAlignment = 'right';
            app.TrendebitsLabel.FontName = 'Consolas';
            app.TrendebitsLabel.FontSize = 14;
            app.TrendebitsLabel.FontWeight = 'bold';
            app.TrendebitsLabel.Position = [37 643 105 23];
            app.TrendebitsLabel.Text = 'Tren de bits:';

            % Create TrendebitsEditField
            app.TrendebitsEditField = uieditfield(app.UIFigure, 'text');
            app.TrendebitsEditField.HorizontalAlignment = 'center';
            app.TrendebitsEditField.FontName = 'Consolas';
            app.TrendebitsEditField.FontSize = 14;
            app.TrendebitsEditField.FontWeight = 'bold';
            app.TrendebitsEditField.Position = [157 644 352 22];

            % Create ModulacinDropDownLabel
            app.ModulacinDropDownLabel = uilabel(app.UIFigure);
            app.ModulacinDropDownLabel.HorizontalAlignment = 'right';
            app.ModulacinDropDownLabel.FontName = 'Consolas';
            app.ModulacinDropDownLabel.FontSize = 14;
            app.ModulacinDropDownLabel.FontWeight = 'bold';
            app.ModulacinDropDownLabel.Position = [52 610 90 22];
            app.ModulacinDropDownLabel.Text = 'Modulación:';

            % Create SelMod
            app.SelMod = uidropdown(app.UIFigure);
            app.SelMod.Items = {'Manchester', 'Manchester Diferencial', 'Ambas'};
            app.SelMod.FontSize = 14;
            app.SelMod.Position = [157 610 181 22];
            app.SelMod.Value = 'Manchester';

            % 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 = GarciaGarciaJoseAngel_TrabajoUnidad3

            % 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
```

# RESULTADOS OBTENIDOS

La interfaz gráfica que hice es la siguiente:

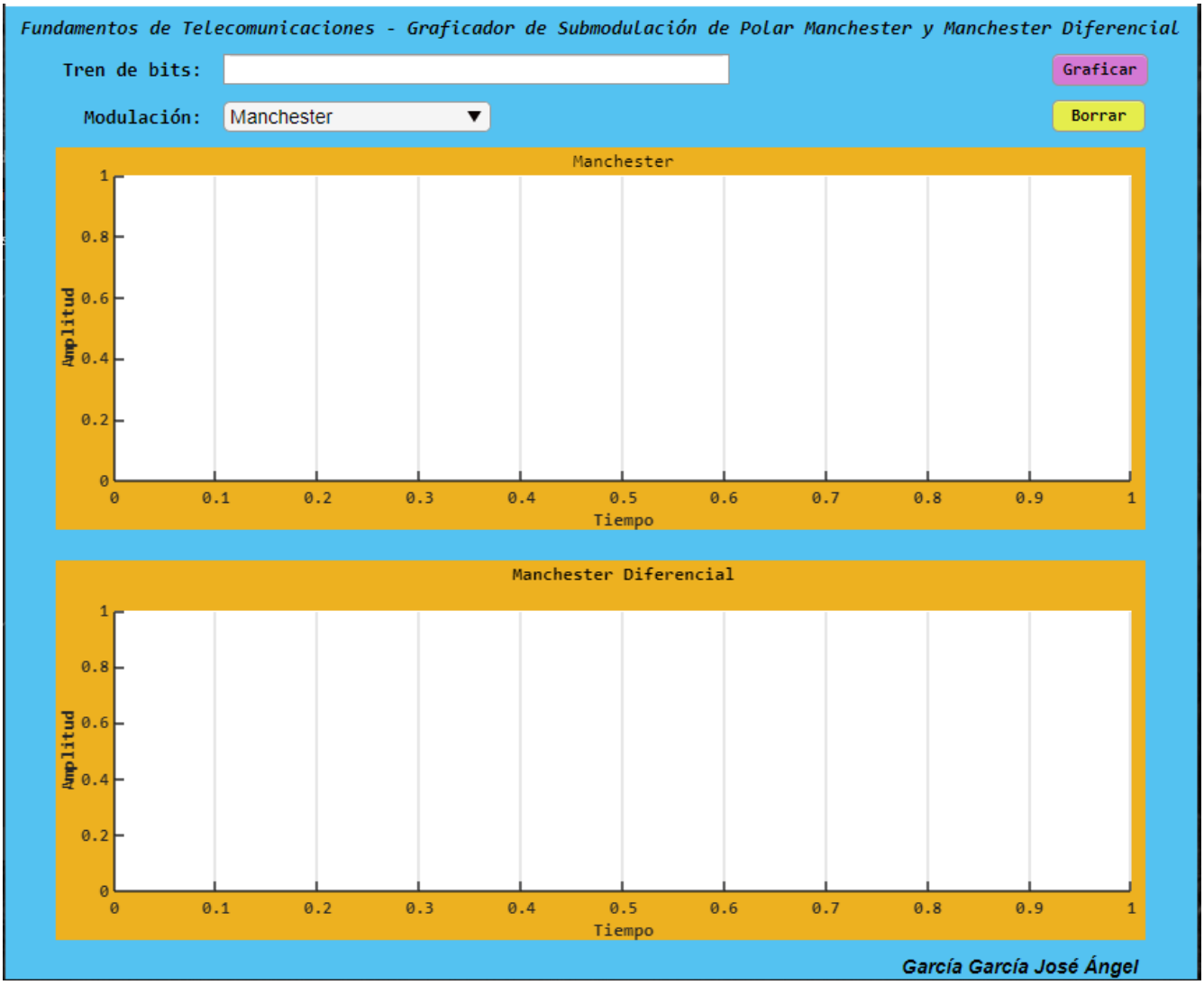


*Ilustración 6 - Interfaz grafica*

Vamos a hacer la prueba con lo siguiente:

1.- Empezamos por hacer la prueba con el tren de bits del libro.

El tren de bits que presenta el libro es **0 1 0 0 1 1 1 0**, y la gráfica de su modulación Manchester lo podemos visualizar en la siguiente imagen que es tomada del libro:

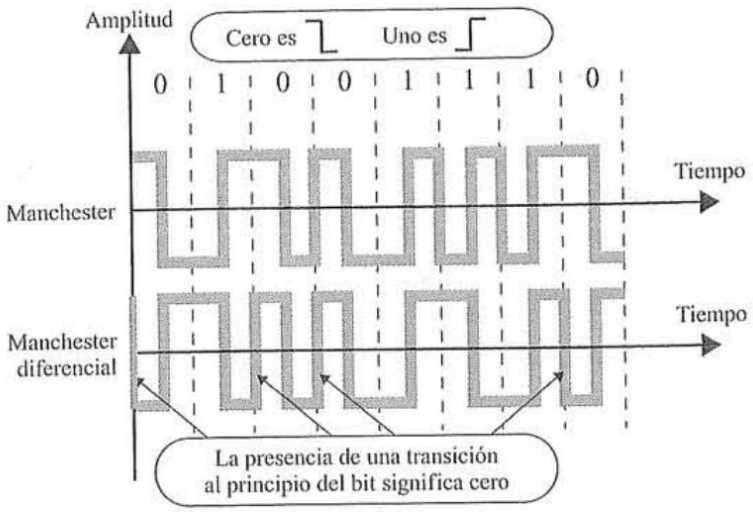


*Ilustración 7 - Graficas del libro*

Ahora, vamos a introducir dicho tren de bits a nuestro programa y seleccionar la opción de Manchester para obtener su gráfica.

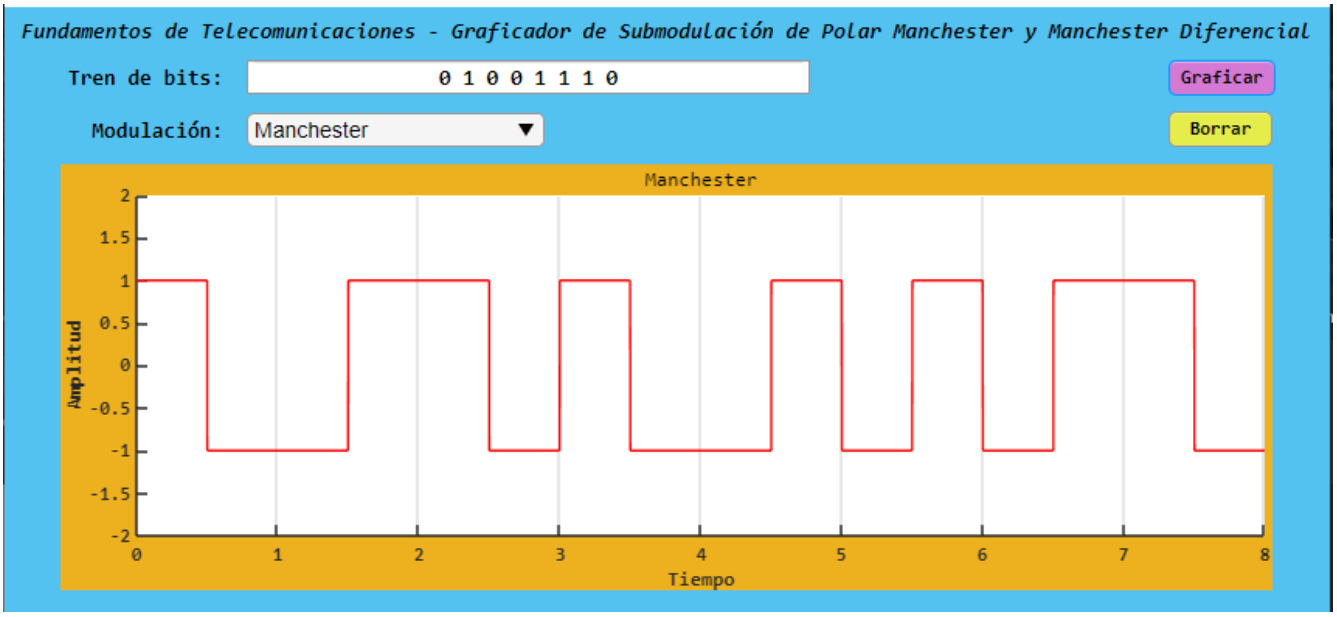


*Ilustración 8 - Grafica de prueba del libro de Manchester*

Se comprueba que es la misma gráfica, entonces nuestra modulación de Manchester es la correcta. Además, no se muestra la otra gráfica porque está vacía así que la omitimos.

Ahora, del mismo tren de bits, vamos a hacer la gráfica de su modulación de Manchester Diferencial. Nótese que la gráfica de Manchester no aparece porque hemos seleccionado sólo la modulación de Manchester Diferencial.

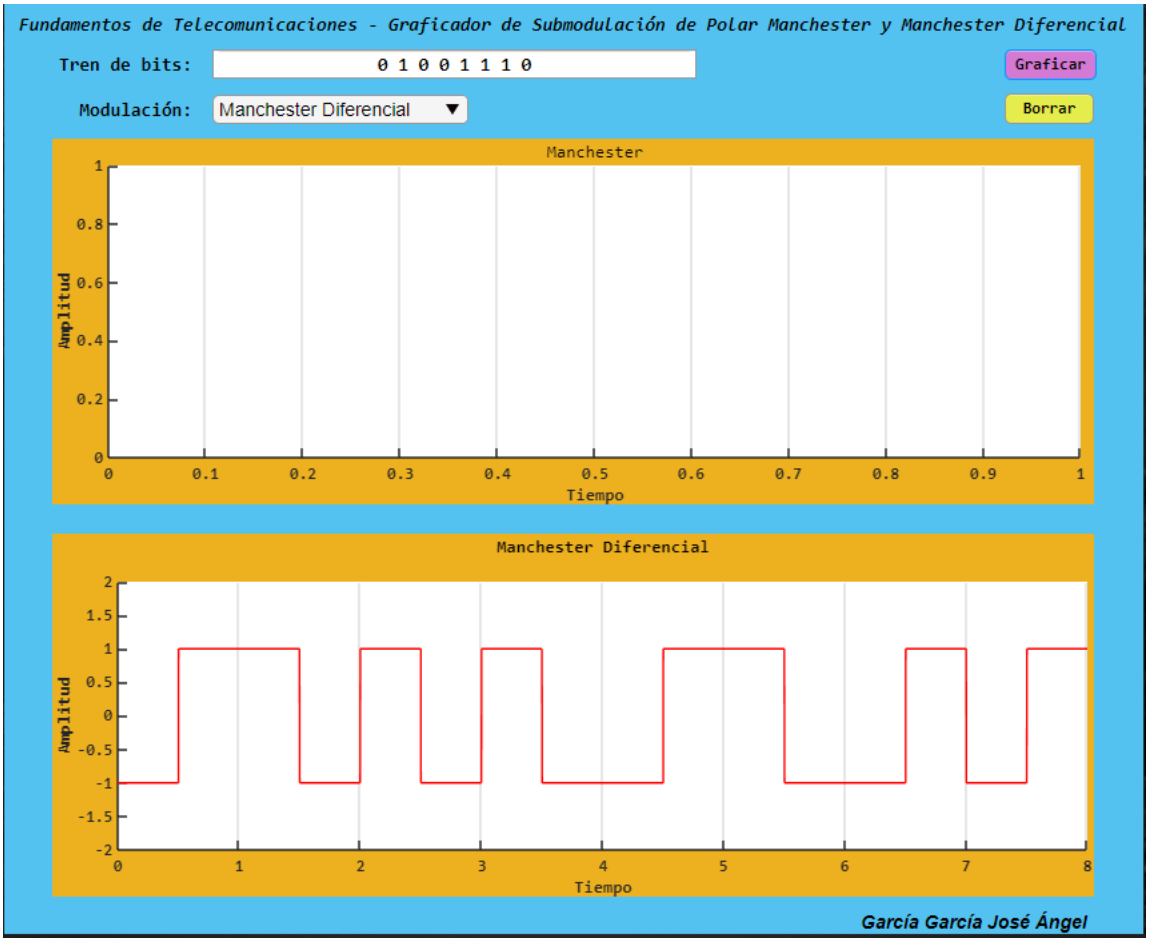


*Ilustración 9 - Grafica de prueba del libro de Manchester Diferencial*

Si comparamos la grafica presentada por el libro, y la de nuestro programa, podemos comprobar que son las mismas de acuerdo la modulación Manchester Diferencial.

Ahora si seleccionamos la opción de ambas, nos mostrará la gráfica de ambas modulaciones y se vería algo así:

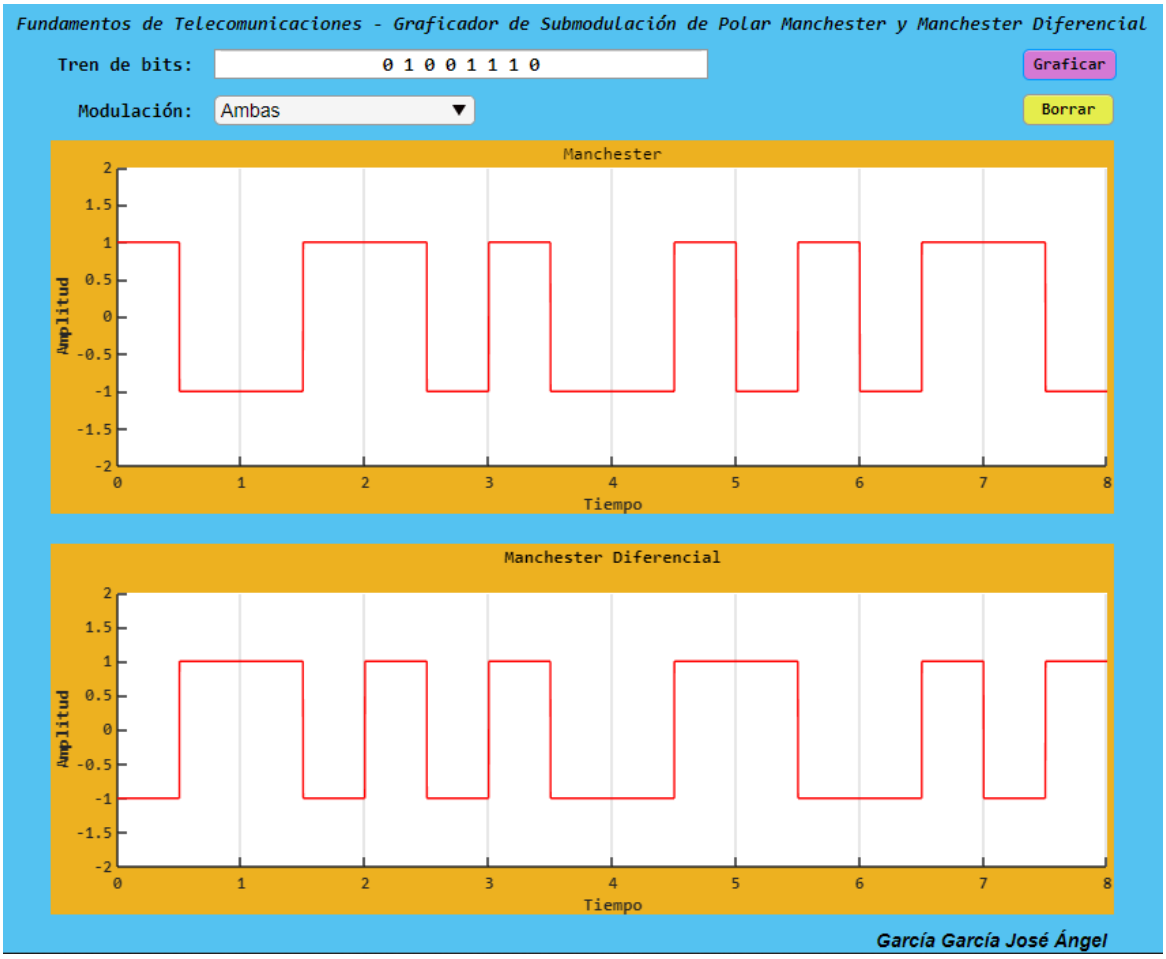


*Ilustración 10 - Grafica de ambas modulaciones de prueba del libro*

Entonces se concluye de esta prueba que está bien nuestro programa, ya que en ambas graficas de modulación el resultado concuerda con las gráficas del libro.

2.- Ahora, realizaremos la prueba con el tren de bits solicitado en la actividad del foro.

El tren de bits de dicha actividad es **1 0 1 1 1 1 0 1 0 0 0 1 1 0 0.** Y su grafica aplicando la modulación de Manchester es:

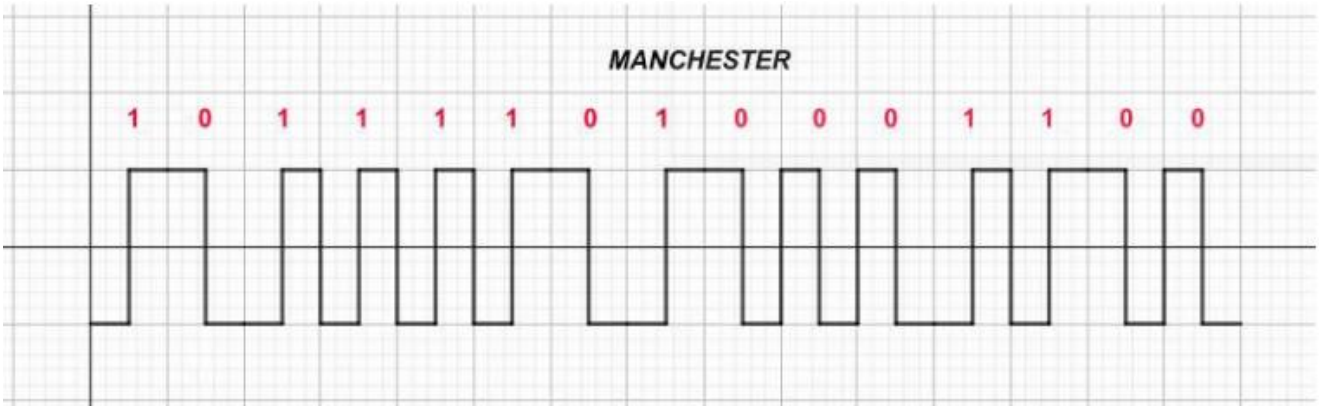


*Ilustración 11 - Grafica del foro de Manchester*

Y la salida de mi programa es la siguiente:

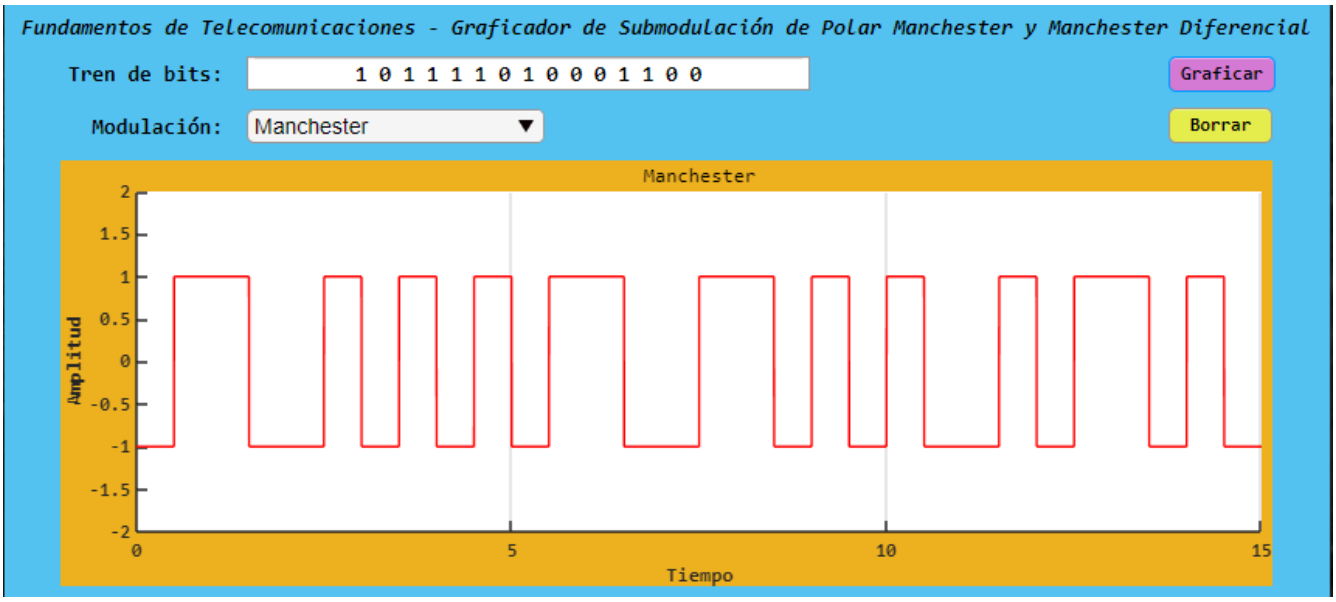


*Ilustración 12 - Grafica de prueba del foro de Manchester*

Se comprueba que la salida es la correcta, que concuerda con la del foro. Que de hecho por los comentarios de mis compañeros varios concordamos en que esa era la solución.

Y ahora vamos a graficar la de Manchester Diferencial, que de acuerdo al foro es:

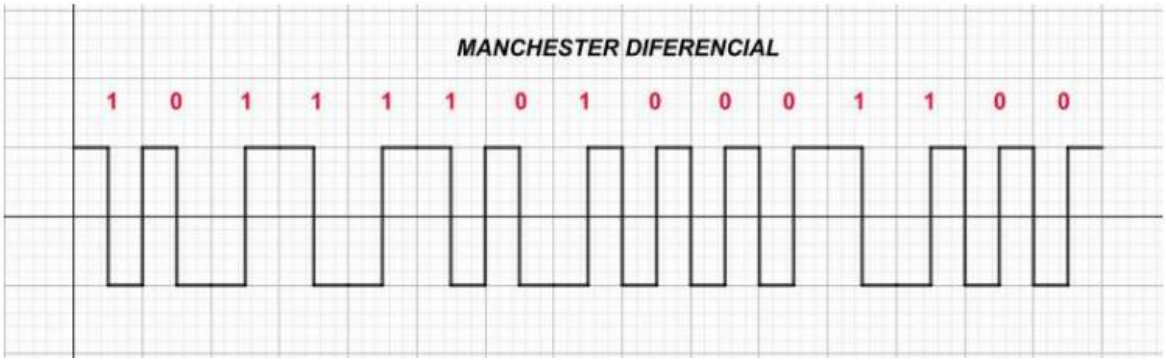


*Ilustración 13 - Grafica del foro de Manchester Diferencial*

Y la salida de mi programa es la siguiente de acuerdo a la modulación de Manchester Diferencial:

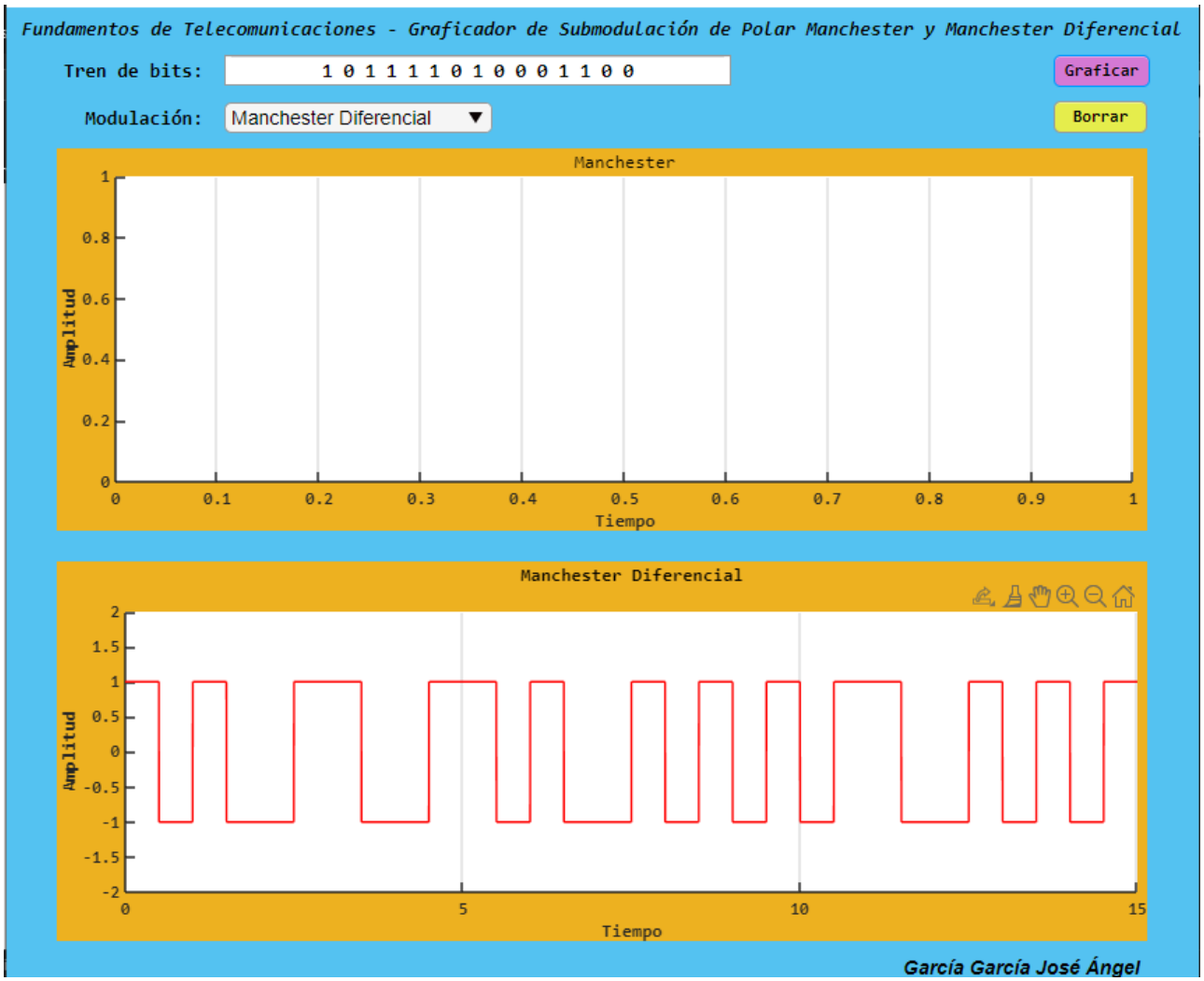


*Ilustración 14 - Grafica de prueba del foro de Manchester Diferencial*

Por último, podemos visualizar ambas graficas de la modulación:

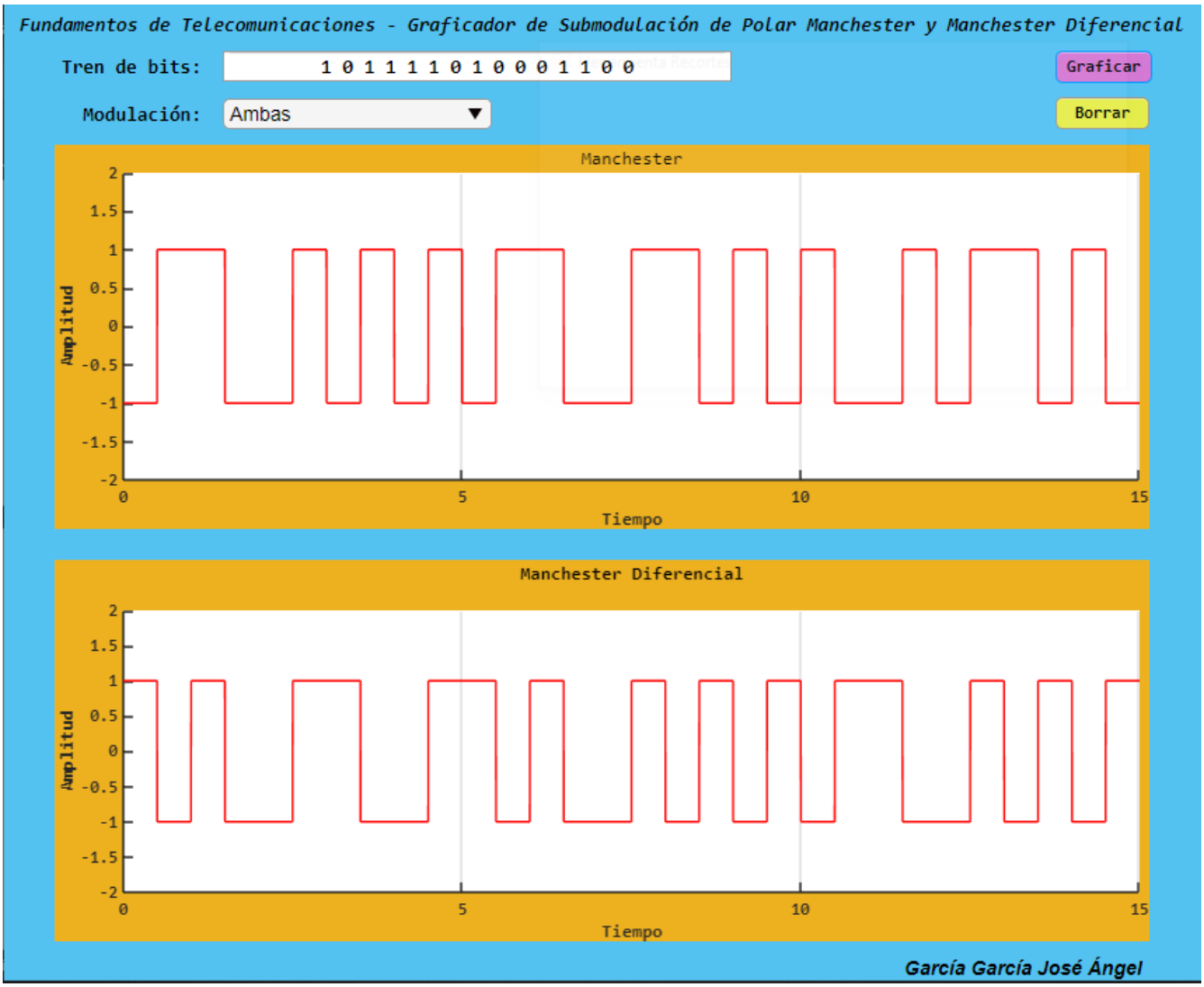


*Ilustración 15 - Ambas graficas de prueba del foro*

Como podemos observar, se comprueba que en ambas modulaciones la gráfica corresponde a las gráficas del ejercicio del foro, por lo que nuestro programa está bien, funciona correctamente de acuerdo a lo solicitado en el planteamiento del problema.

Ahora, vamos a hacer otras pruebas, en la que vamos visualizar las dos modulaciones de cada uno de los siguientes trenes de bits.

Para un tren de bits de **1 0 1 0 0 1 0**

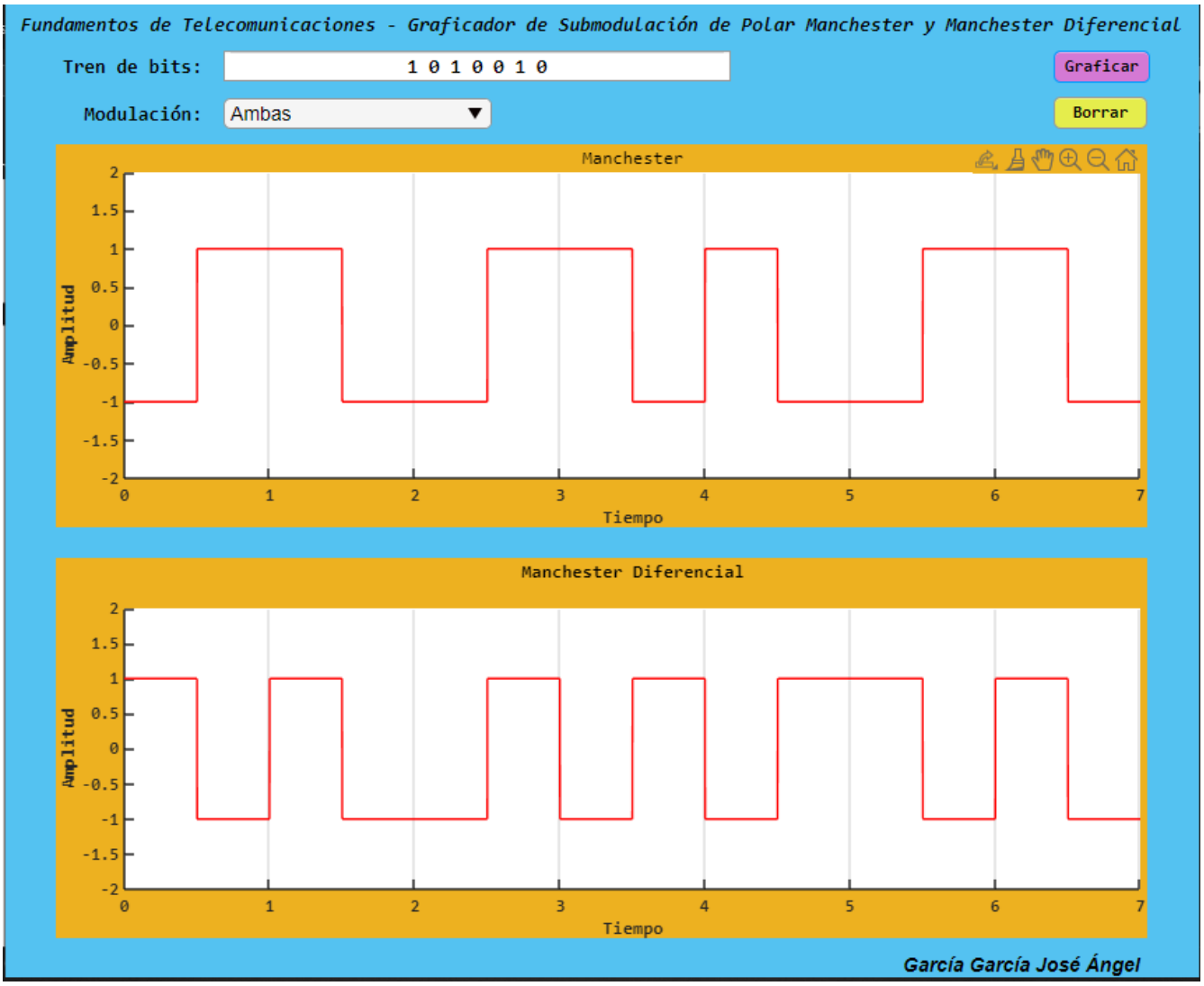


*Ilustración 16 - Graficas de modulaciones de 1 0 1 0 0 1 0*

Para un tren de bits de **1 1 1 0 0 1 1**

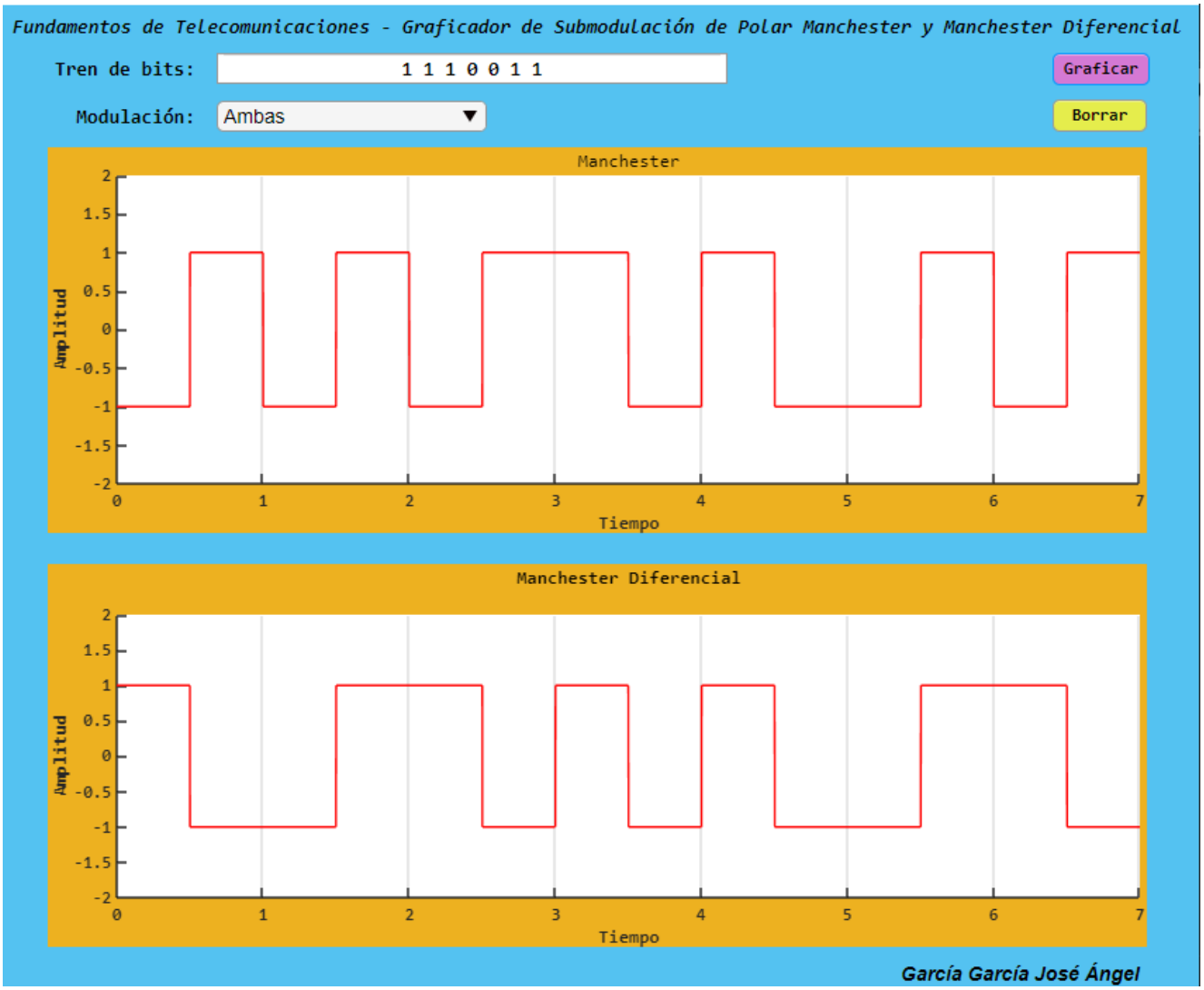


*Ilustración 17 - Graficas de modulaciones de 1 1 1 0 0 1 1*

Para un tren de bits de **1 0 0 1 1 1 1 1 0 0 1 1 1 1**

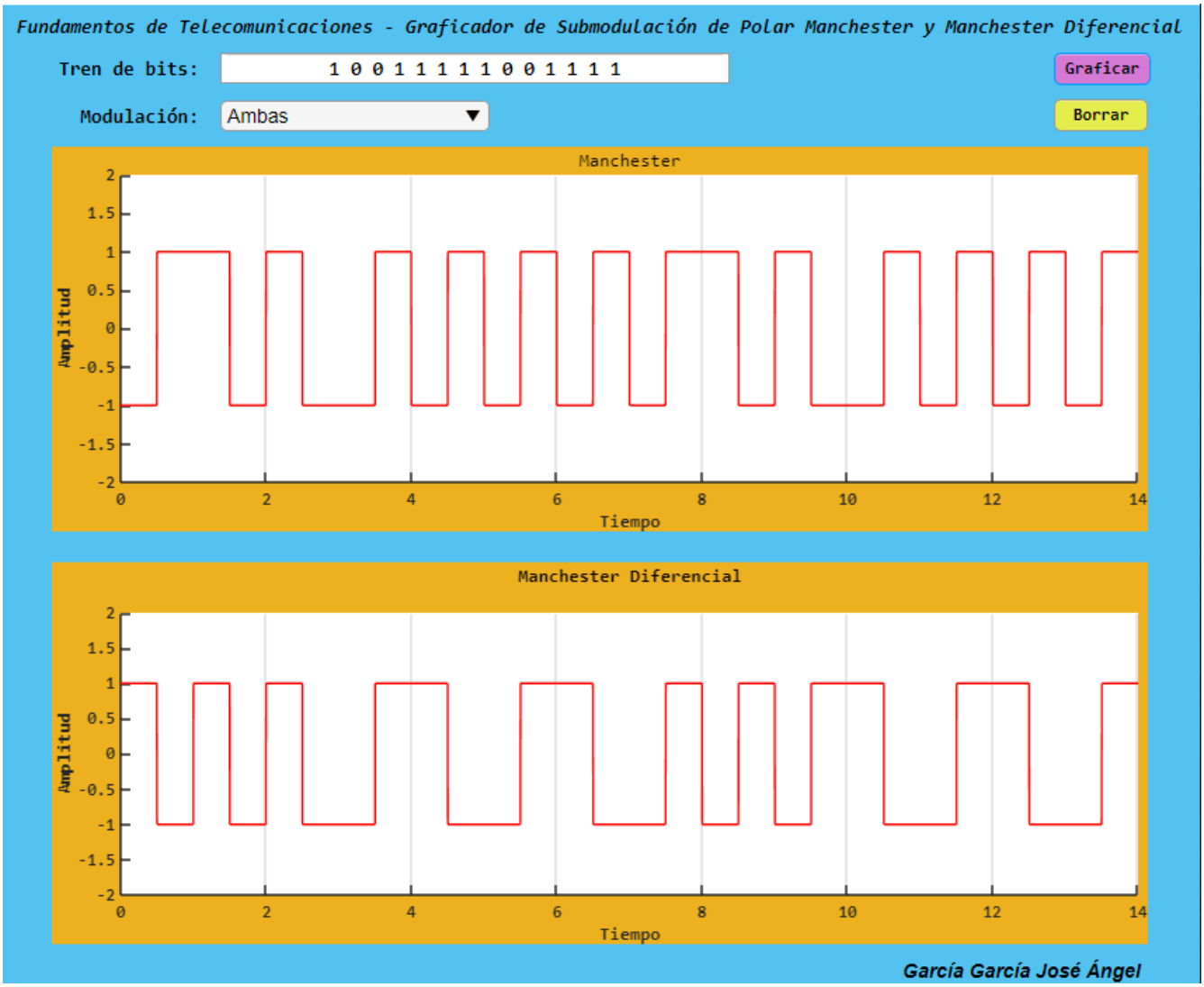


*Ilustración 18 - Graficas de modulaciones de 1 0 0 1 1 1 1 1 0 0 1 1 1 1*

# ENLACE DEL VÍDEO

El vídeo en el que se explica a detalle cada una de las fases del programa desde su desarrollo hasta su implementación lo puede visualizar en el siguiente enlace:

https://youtu.be/PmfMvEPwfjQ

Le dejo una tabla para que pueda revisar de forma más rápida el vídeo, ya que es algo extenso, siempre se me pasa el tiempo ya que trato de explicar a detalle lo que realicé en mi trabajo.

| Minuto | Acción |
|---|---|
| **1:05** | Explicación de la interfaz |
| **2:22** | Prueba en base al libro |
| **7:40** | Prueba en base al foro |
| **13:45** | Explicación del código |
| **14:35** | Explicación de la función Manchester |
| **19:07** | Explicación de la función Manchester Diferencial |
| **31:22** | Explicación de la función Graficar |
| **37:16** | Explicación de la función del botón Graficar |

# CONCLUSIÓN

La evidencia que presenté anteriormente demuestra que en esta práctica se cumplió con el objetivo, que era crear un programa que fuera capaz de realizar la submodulación de Polar Manchester y Manchester Diferencial, bueno, el poderlo graficarlos era lo más importante. Justamente la información presentada en el apartado de resultados obtenidos demuestra las pruebas de ejecución del programa y verifica exitosamente que funciona correctamente, además que se presenta un enlace a un vídeo en el que se explica cada uno de los detalles del programa, haciendo así que la explicación del funcionamiento del programa sea más fácil de entender y que se recalque que funciona correctamente haciendo lo solicitado.

Realizar el programa en Matlab fue relativamente sencillo, ya que ya se había tenido experiencia previa con este lenguaje, justamente porque el programa de la primera unidad se realizó ahí y se practicó lo suficiente. La interfaz gráfica que se presenta es algo sencilla a mi parecer, sólo tiene algo de colores para hacerla más atractiva, pero es clara y precisa para presentar la información solicitada.

Todas las actividades solicitadas fueron realmente buenas, especialmente que se haya solicitado estas submodulaciones de Polar Manchester y Manchester Diferencial, ya que en una práctica anterior fueron justo estos los más complicados de realizar y los que mayor problemas nos dieron para poder graficarlas, entonces, ya con más conocimiento sobre la misma, ahora puedo decir que si está fácil cómo usted mencionaba, pero eso sí, al principio me enredaba mucho el poder comprenderlas pero gracias a esta práctica y a las anteriores ahora he comprendido muy bien estas modulaciones.

Ahora bien, quedando satisfecho con los resultados obtenidos, el trabajo presentado y reforzando el conocimiento en cuanto al entorno de desarrollo en Matlab y al conocimiento de la asignatura, espero poder seguir haciéndolo de la misma forma durante el transcurso de las clases, aunque sean de forma virtual, me está pareciendo muy entretenida y adecuada la forma de aprender en esta asignatura.