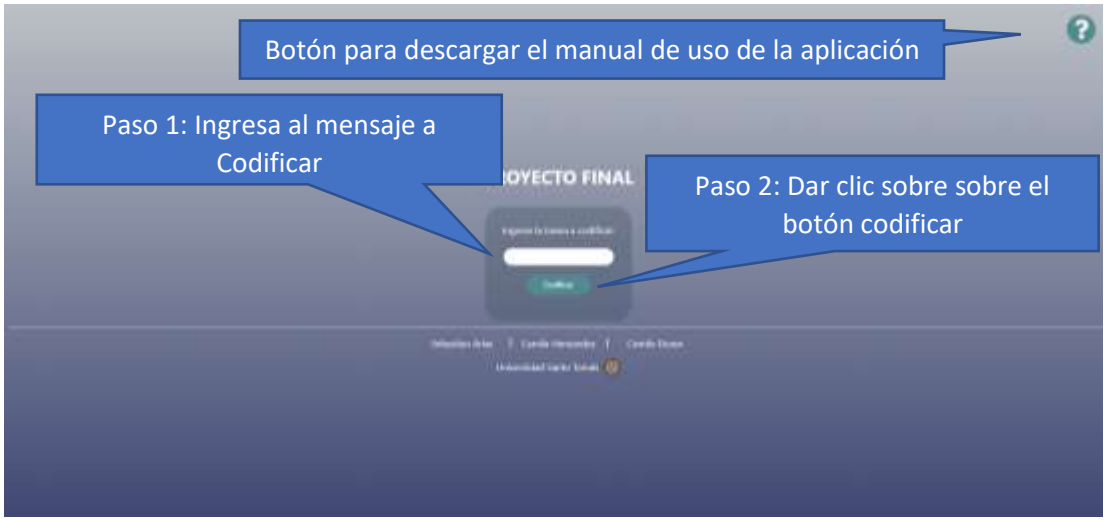
UNIVERSIDAD SANTO TOMÁS
Facultad: Ingeniería Telecomunicaciones
Asignatura: Sistemas de Telecomunicaciones II. Grupo: 6A
Docente: Gustavo Alonso Chica Pedraza
Estudiantes:
- Juan Sebastián Arias Duarte Código: 2219096
- Juan Camilo Duran Rincón. Código: 2228047
- Laura Camila Hernández Moreno Código: 2211829
Fecha: 31/05/2020

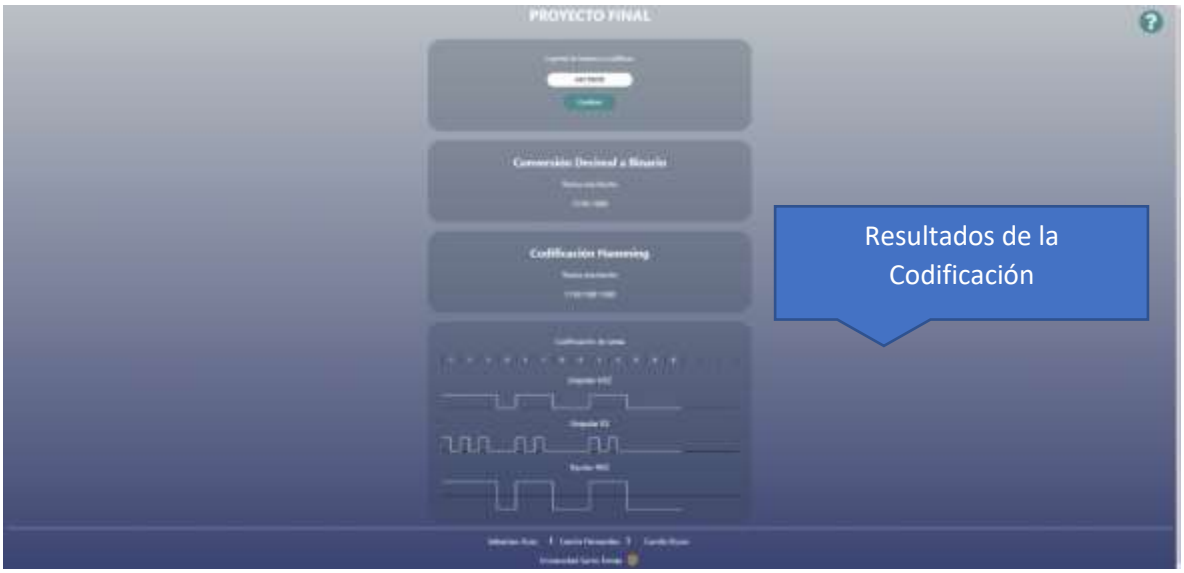
MANUAL DE USUARIO

1. GUÍA DE EJECUCIÓN DEL PROGRAMA

1.1. Pasos Generales para la Ejecución del Programa:

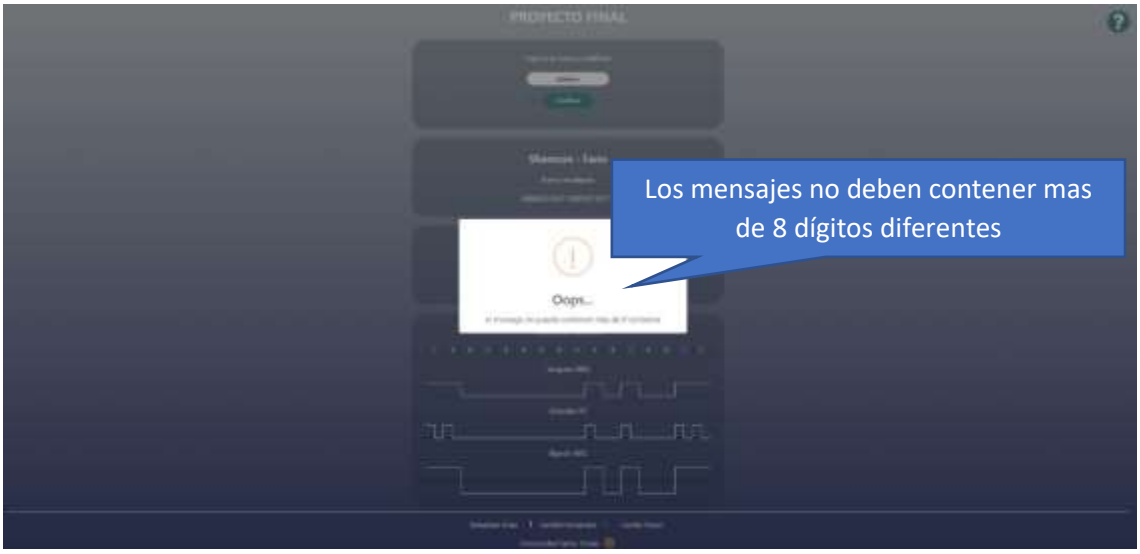


Los mensajes que se ingresen, el programa realizara una conversión inicial en su valor ASCII, el cual es un numero en decimal, este valor se convertirá a binario, luego de obtener la trama binaria de todo el mensaje, se iniciara con la respectiva codificación escogida. Las codificaciones que utilizan esta conversión ASCII son: Conversión Binario – Decimal, RLE y DCPM.



1.2. Excepciones:

El funcionamiento del programa tiene una alerta en el momento en que se detecte que el mensaje tiene más de 8 caracteres diferentes, este mensaje de alerta surge de la necesidad de la utilización de la codificación algebraica, para que esta codificación no defina un numero de base mayor a 8. De esta forma, si se llega hasta una base que sea mayor, este sistema empezaría a tomar bases más complejas a desarrollar

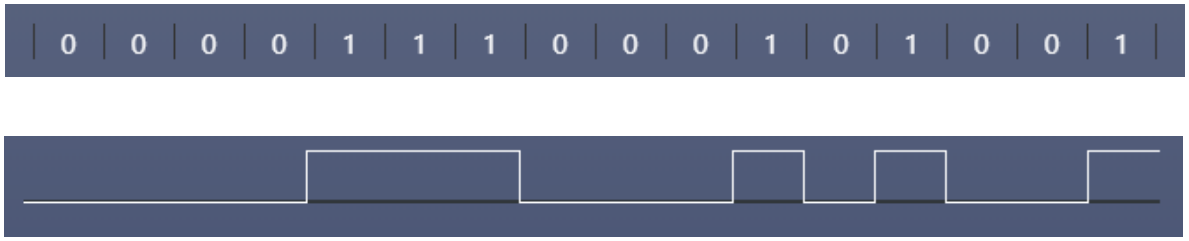


1.3. Explicación de los Códigos de Línea:

Los códigos de línea utilizados en el programa son Unipolar NRZ, Unipolar RZ y Bipolar NRZ. Cada una de las codificaciones que está realizando el programa, conlleva cada una de las codificaciones de línea descritas.

1.3.1. Codificación Unipolar NRZ: No contiene un valor anterior, y utiliza un único valor de nivel. Este valor generalmente es representado entre 1 y 0. El voltaje positivo representa a un binario 1 y un voltaje cero indica un binario 0.

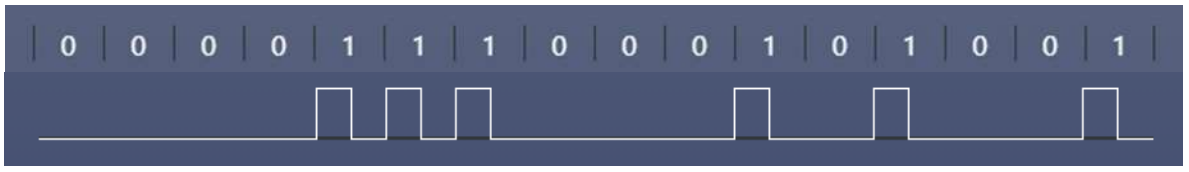
Ejemplo:



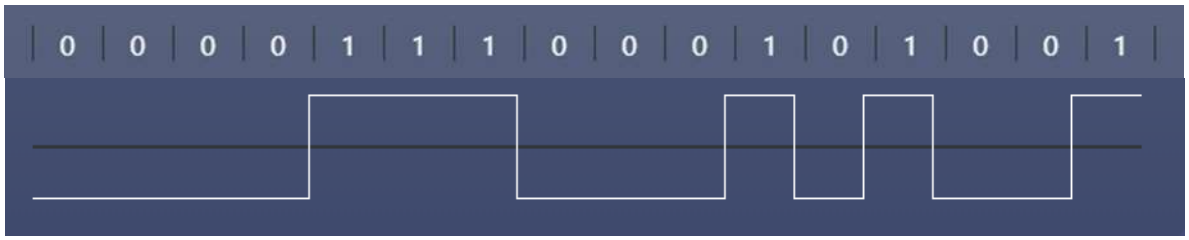
1.3.2. Codificación Unipolar RZ: No contiene un valor anterior, y utiliza un único valor de nivel. Este valor generalmente es representado entre 1 y 0. El voltaje positivo representa a un binario 1, sin embargo, con la particularidad de que su duración será de medio intervalo del bit y un voltaje cero indica un binario 0 con intervalo completo del bit

Ejemplo:

1.3.3. Bipolar NRZ: No contiene un valor anterior. Usa el positivo, negativo, es decir, cuando se aproxime un 1, la gráfica tendera a valores positivos, mientras que, en caso de los ceros, la gráfica tendera a valores negativos.



Ejemplo:



2. Funcionamiento de las Codificaciones:

2.1. Conversión Decimal – Binario

La conversión binaria a decimal es un sistema que permite

Ejemplo:

M=[ORCA]

1. Se convierte el mensaje en su respectivo valor de código ASCII
   M=

O	R	C	A
79	82	67	65

2. Estos valores anteriores, se concatenan y se convierten de uno en uno

O	R	C	A
79	82	67	65
1001111	1010010	1000011	1000001

3. Los valores son concatenados según el orden de izquierda a derecha. Los cuáles serán la trama de la codificación decimal a binario

10011111010010100001110000001

0001

4. Aplicación de codificación Hamming a la trama de la codificación Huffman

4.1. Resultado de la nueva trama de la Codificación Hamming

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	P1	P2	D1	P3	D2	D3	D4	P4	D5	D6	D7	D8	D9	D10	D11	P5	D12	D13	D14	D15	D16	D17	D18
T.F	0	1	1	0	0	0	1	0	1	1	1	1	0	1	0	1	0	1	0	1	0	0	0
FP1	0		1		0		1		1		1		0		0		0		0		0		0
FP2		1	1			0	1			1	1			1	0			1	0			0	0
FP4				0	0	0	1					1	0	1	0					1	0	0	0
FP8								0	1	1	1	1	0	1	0								
FP16																1	0	1	0	1	0	0	0
FP32																							

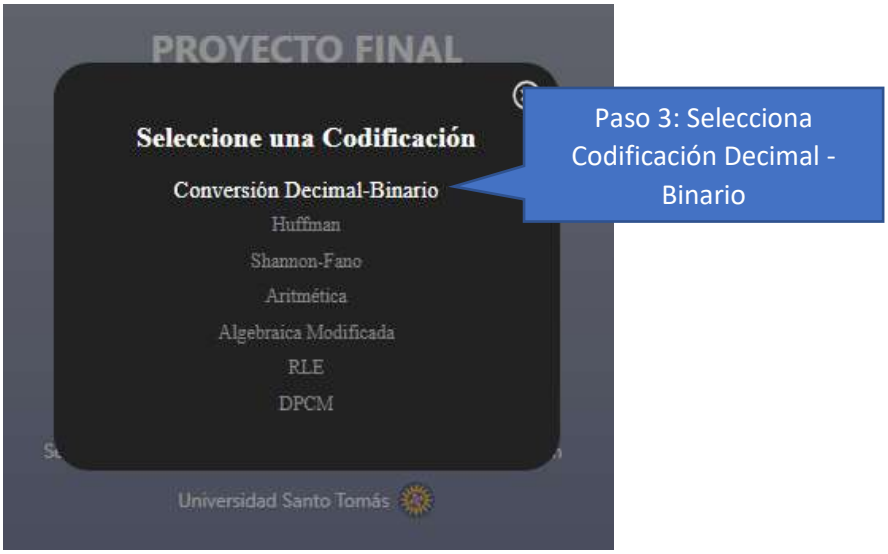
	24	25	26	27	28	29	30	31	32	33	34
	D19	D20	D21	D22	D23	D24	D25	D26	P6	D27	D28
…	0	1	1	1	0	0	0	0	1	0	1
FP1		1		1		0		0		0	
FP2			1	1			0	0			1
FP4					0	0	0	0			
FP8	0	1	1	1	0	0	0	0			
FP16	0	1	1	1	0	0	0	0			
FP32									1	0	1

4.2. Trama de Hamming:

01100010111101010101000011100008 01

5. Comprobación de datos con el programa

Se realizan los dos pasos anteriores descritos en el apartado “pasoso generales para la ejecución del programa” de la página 1, y se prosigue con el paso número 3.



2.2. Conversión Binario – Decimal

Esta codificación se encarga de realizar la respectiva conversión de valores en base 2, a su representación en base 10. Sin embargo, también se mantiene su trama para la respectiva codificación de Hamming y su representación en las codificaciones de línea. Pero si el mensaje ingresado no es una trama binaria, entonces el programa procederá a convertir cada carácter a su valor numérico en ASCII. Esta representación ASCII en decimal, será la que sufrirá una conversión a binario, para luego proceder a realizar la conversión binario-decimal

Ejemplo:

M=[101011]

1. Dependiendo de las posiciones de las bases en 2^n, se realizará su respectiva conversión a base 10
   M=

32	16	8	4	2	1
1	0	1	0	1	1

2. De la tabla anterior, se sumarán únicamente los decimales en donde el binario sea 1. Este valor, corresponderá al valor de la conversión:
   32+8+2+1=43
3. La trama anterior binaria, la cual correspondería a nuestro valor decimal de conversión, se le realizara la respectiva codificación Hamming

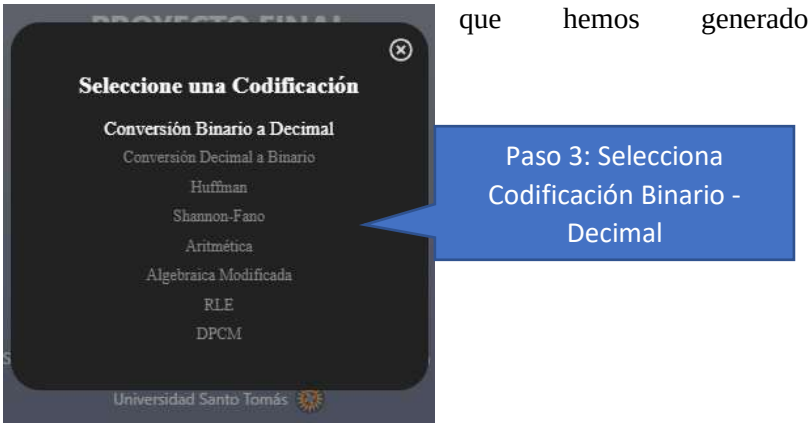
	1	2	3	4	5	6	7	8	9	10
	P1	P2	D1	P3	D2	D3	D4	P4	D5	D6
T.F	0	1	1	1	0	1	0	0	1	1
FP1	0		1		0		0		1	
FP2		1	1			1	0			1
FP4				1	0	1	0			
FP8								0	1	1

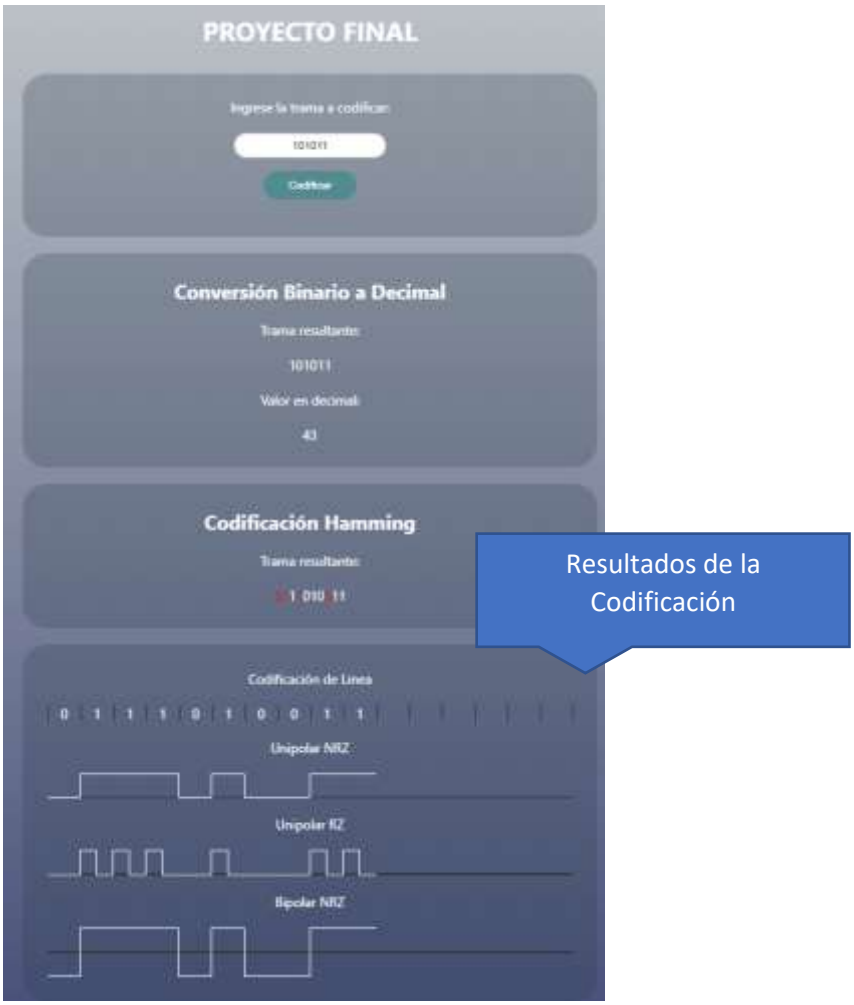
4. TRAMA DE HAMMING

   0111010011

5. Comprobación de datos con el programa

Se realizan los dos pasos anteriores descritos en el apartado “pasoso generales para la ejecución del programa” de la página 1, y se prosigue con el paso número 3. Siguiendo con los pasos generales del apartado anterior, luego de escribir el mensaje a codificar, se presiona en el botón codificar y se selecciona Conversión Binario a Decimal. Los resultados son los que aparecen en la imagen actual, y representan los resultados que hemos generado manualmente en el ejercicio.





2.3. Codificación Huffman:

Es un método estadístico que permite asignar un código binario a cada carácter o símbolos a comprimir del mensaje. Este método está diseñado para minimizar el numero bits innecesarios a transmitir.

Ejemplo:

M = Mensaje; LM = Longitud del Mensaje; A= Alfabeto; n= Longitud DE A

Codificar el mensaje “CORREDOR”

6. Trama de la codificación Huffman
7. Se obtiene la longitud del mensaje. LM = 8
8. Se obtienen los símbolos sin repetir que contenga el mensaje y su longitud.
   A = {CORED} n = 5
9. Se obtiene la probabilidad de cada símbolo en el mensaje según su repetición

   C = 1/8

   O = 2/8

   R = 3/8

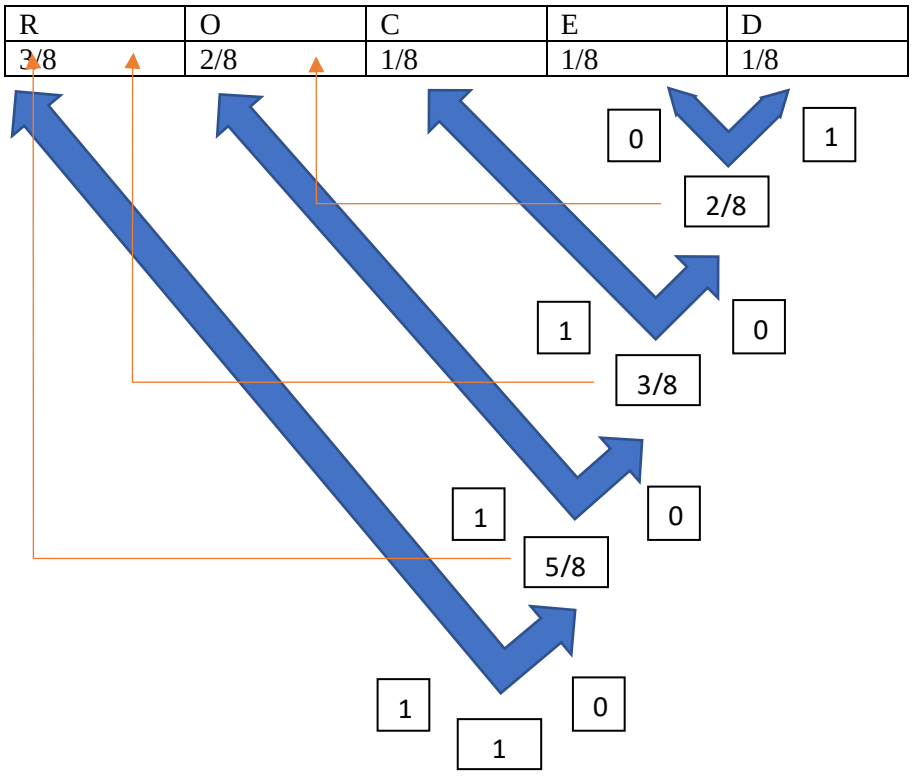
   E = 1/8

   D = 1/8

10. Se organizan los símbolos de mayor a menor según las probabilidades obtenidas, en caso de que las probabilidades de los símbolos sean iguales, se utilizara un ordenamiento secundario, el cual permite ordenarlos según el primero que aparezca en el mensaje

3/8	2/8	1/8	1/8	1/8
R	O	C	E	D

11. Luego de ordenar, se suman las dos últimas probabilidades y se reordena si es necesario. Siempre se le asigna un 0 a la rama mayor. Esta suma siempre se realizará entre los siguientes dos números.



12. La trama binaria de cada símbolo se lee de abajo hacia arriba para este caso, y se sigue la dirección de las ramas hasta llegar al símbolo.

R	O	C	E	D
3/8	2/8	1/8	1/8	1/8
1	01	001	0000	0001

13. Se completa la trama del mensaje

C	O	R	R	E	D	O	R
001	01	1	1	0000	0001	01	1

14. Aplicación de codificación Hamming a la trama de la codificación Huffman a cada bit de la trama se enumera

D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11
0	0	1	0	1	1	1	0	0	0	0
D12	D13	D14	D15	D16	D17	D18				
0	0	0	1	0	1	1				

15. Las posiciones de los bits que representan potencias de 2^n, deben ser reescritas mediante el prefijo P_N

1	2	3	4	5	6	7	8	9	10	11	12
P1	P2	D1	P3	D2	D3	D4	P4	D5	D6	D7	D8
1	1	0	0	0	1	0	1	1	1	1	0
13	14	15	16	17	18	19	20	21	22	23	
D9	D10	D11	P5	D12	D13	D14	D15	D16	D17	D18	
0	0	0	1	0	0	0	1	0	1	1	

16. Resultado de la trama de la Codificación Hamming

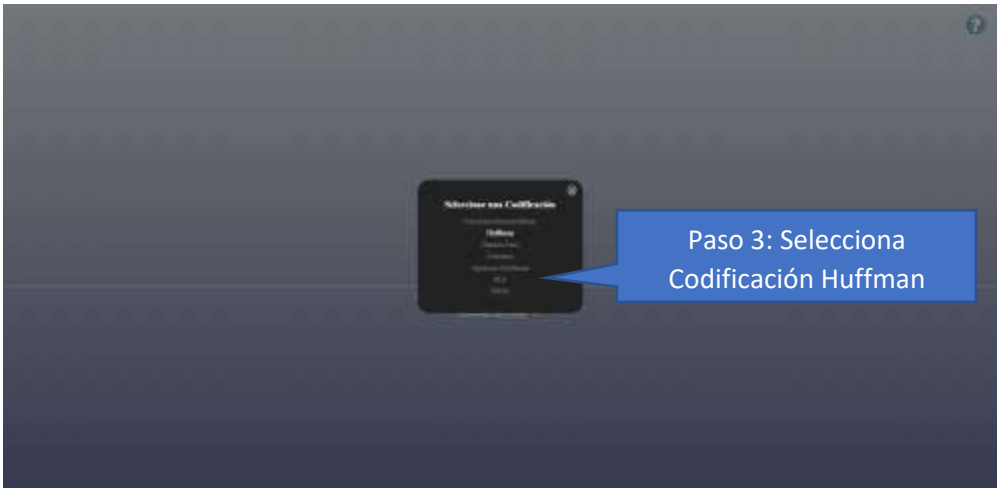
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	P1	P2	D1	P3	D2	D3	D4	P4	D5	D6	D7	D8	D9	D10	D11	P5	D12	D13	D14	D15	D16	D17	D18
T.F	1	1	0	0	0	1	0	1	1	1	1	0	0	0	0	1	0	0	0	1	0	1	1
FP1	1		0		0		0		1		1		0		0		0		0		0		1
FP2		1	0			1	0			1	1			0	0			0	0			1	1
FP4				0	0	1	0					0	0	0	0					1	0	1	1
FP8								1	1	1	1	0	0	0	0								
FP16																1	0	0	0	1	0	1	1

17. Trama de Hamming:

11001010111100001000 1011

18. Comprobación de datos con el programa

Se realizan los dos pasos anteriores descritos en el apartado “pasoso generales para la ejecución del programa” de la página 1, y se prosigue con el paso número 3. Siguiendo con los pasos generales del apartado anterior, luego de escribir el mensaje a codificar, se presiona en el botón codificar y se selecciona Codificación Huffman. Los resultados son los que aparecen en la imagen actual, y representan los resultados que hemos generado manualmente en el ejercicio.



2.4. Codificación Shannon-Fano

Es una técnica de compresión de datos que permite encontrar la probabilidad de aparición de cada símbolo de un mensaje. En términos generales:

- Se calcula las frecuencias relativas de los símbolos de la fuente
- se ordena dichas frecuencias en orden ascendente
- se usa el principio de divide-y-vencerás para crear sucesivamente los símbolos usados en la codificación: en cada paso se divide los símbolos (ordenados) en dos intervalos con más o menos la misma suma de probabilidades, a un lado se añade el bit 0 al otro el bit 1, los bits se acumulan a lo largo de la recursión

Ejemplo:

M = Mensaje; LM = Longitud del Mensaje; A= Alfabeto; n= Longitud DE A

Codificar el mensaje “ANITALAVALATINA”

1. Se obtiene la longitud del mensaje. LM = 15
2. Se obtienen los símbolos sin repetir que contenga el mensaje y su longitud.
   A = {ANITLV} n = 6

3. Se obtiene la probabilidad de cada símbolo en el mensaje según su repetición

   A = 6/15

   N = 2/15

   I = 2/15

   T = 2/15

   L = 2/15

   V = 1/15

4. Ordeno de mayor a menor y ejecuto divisiones y paso una línea sobre el más cercano a ella. Se cuentan los caracteres y se dividen entre dos sucesivamente.

A	N	I	T	L	V
6	2	2	2	2	1
0	0	1	1	1	1
0	1	0	0	1	1
		0	1	0	1

5. Se completa la trama del mensaje

A	N	I	T	A	L	A	V	A	L	A	T	I	N	A
00	01	100	101	00	110	00	111	00	110	00	101	100	01	00

6. Aplicación de codificación Hamming a la trama de la codificación Huffman

6.1. Resultado de la trama de la Codificación Hamming

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	P1	P2	D1	P3	D2	D3	D4	P4	D5	D6	D7	D8	D9	D10	D11	P5	D12	D13	D14	D15	D16	D17	D18
T.F	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	1	0	1	1	0	0	0	1
FP1	0		0		0		1		1				0		0		0		1		0		1
FP2		0	0			0	1			0	0			1	0			1	1			0	1
FP4				0	0	0	1					1	0	1	0					0	0	0	1
FP8								0	1	0	0	1	0	1	0								
FP16																1	0	1	1	0	0	0	1
FP32																							

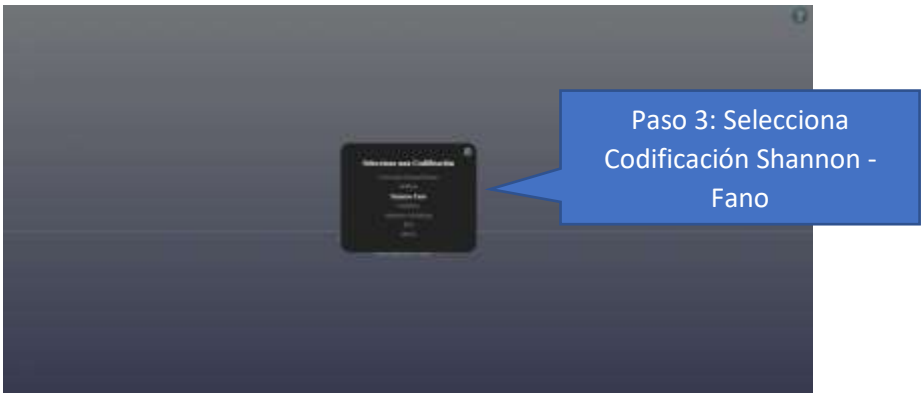
	24	25	26	27	28	29	30	31	32	33	34	35	36	37	24	25	26	27	28	29			
	D19	D20	D21	D22	D23	D24	D25	D26	P6	D27	D28	D29	D30	D31	D32	D33	D34	D35	D36	D37			
…	1	1	0	0	1	1	0	0	0	0	1	0	1	1	0	0	0	1	0	0			
FP1		1		0		1		0		0		0		1		0		1		0			
FP2			0	0			0	0			1	0			0	0			0	0			
FP4					1	1	0	0					1	1	0	0							
FP8	1	1	0	0	1	1	0	0									0	1	0	0			
FP16	1	1	0	0	1	1	0	0															
FP32										0	1	0	1	1	0	0	0	1	0	0			

6.2. Trama de Hamming:

00000010100101010110001110011000010110001000

7. Comprobación de datos con el programa

Se realizan los dos pasos anteriores descritos en el apartado “pasoso generales para la ejecución del programa” de la página 1, y se prosigue con el paso número 3. Siguiendo con los pasos generales del apartado anterior, luego de escribir el mensaje a codificar, se presiona en el botón codificar y se selecciona Codificación Shannon - Fano. Los resultados son los que aparecen en la imagen actual, y representan los resultados que hemos generado manualmente en el ejercicio.



2.5. Codificación Aritmética

Es una forma de codificación entrópica que utiliza el método estadístico, fundamentándose en el conocimiento a priori de las probabilidades de cada símbolo. La codificación aritmética no produce un código simple para cada símbolo. Produce un código para un mensaje entero Cada símbolo añadido al mensaje progresivamente modifica el código de salida El efecto final de cada símbolo de la entrada sobre el código de salida puede ser un número fraccionario de bits en lugar de un número entero.

Formula de codificación:

$$Lim\ Inferior = \inf(i-1) + [\inf(i) * \lambda(i-1)]$$

$$Lim\ Superior = \inf(i-1) + [\sup(i) * \lambda(i-1)]$$

Ejemplo:

S= BIC

P=0.25, 0.5, 0.25

M=[BICI]

1. Se calcula los rangos y longitudes de cada carácter del mensaje

Símbolo	Probabilidad	Rango	Longitud
B	0.25	[0.0 , 0.25)	$\lambda = 0.25$
I	0.5	[0.25 , 0.75)	$\lambda = 0.5$
C	0.25	[0.75 , 1.0)	$\lambda = 0.25$

2. Se codifica según el orden del mensaje de izquierda a derecha, utilizando las ecuaciones de codificación aritmética.

- B= [0.0 , 0.25)
- BI=
  $Lim\ Inferior = 0 + [0.25 * 0.25] = 0.0625$
  $Lim\ Superior = 0 + [0.75 * 0.25] = 0.1875$
  $Nuevo\ Lim. = [0.0625 , 0.1875)\ \lambda = 0.125$
- BIC
  $Lim\ Inferior = 0.0625 + [0.75 * 0.125] = 0.15625$
  $Lim\ Superior = 0.0625 + [1.0 * 0.125] = 0.1875$
  $Nuevo\ Lim. = [0.15625 , 0.1875)\ \ y\ \lambda = 0.03125$
- BICI
  $Lim\ Inferior = 0.15625 + [0.25 * 0.03125] = 0.1640625$
  $Lim\ Superior = 0.15625 + [0.75 * 0.03125] = 0.1796875$
  $Nuevo\ Lim. = [0.1640625, 0.1796875)\ y\ \lambda = 0.0156625$

El valor para codificar es 0.1640625, cuya representación binaria es: 0.0010101

3. Aplicación de codificación Hamming a la trama de la codificación Aritmética

3.1. Resultado de la trama de la Codificación Hamming

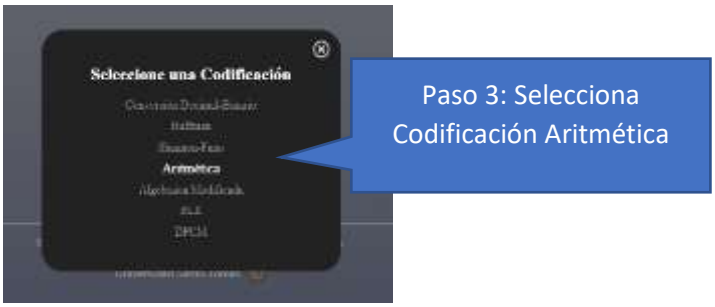
	1	2	3	4	5	6	7	8	9	10	11
	P1	P2	D1	P3	D2	D3	D4	P4	D5	D6	D7
T.F	0	0	0	1	0	1	0	0	1	0	1
FP1	0		0		0		0		1		1
FP2		0	0			1	0			0	1
FP4				1	0	1	0				
FP8								0	1	0	1

3.2. Trama de Hamming:

00010100101

4 Comprobación de datos con el programa

Se realizan los dos pasos anteriores descritos en el apartado “pasoso generales para la ejecución del programa” de la página 1, y se prosigue con el paso número 3. Siguiendo con los pasos generales del apartado anterior, luego de escribir el mensaje a codificar, se presiona en el botón codificar y se selecciona Codificación Aritmética. Los resultados son los que aparecen en la imagen actual, y representan los resultados que hemos generado manualmente en el ejercicio.



2.6. Codificación Algebraica modificada

Ejemplo:

Se desea codificar el siguiente mensaje

M=[TOTTO]

1. Se crea una nueva matriz, la cual contiene los caracteres únicos del mensaje, teniendo en cuenta que estos deben ubicarse en el mismo orden de izquierda a derecha

A=[TO]

2. A cada carácter se le asigna un número que empieza desde '0' y se eleva realizando la suma de 1 para cada posición de izquierda a derecha

   T= 0

   O=1

3. Al número mayor de la lista anterior, se le suma 1, y este nuevo resultado se convertirá en la base numérica a trabajar para la codificación
   (1+1) = 2; la base de codificación será 2.
4. Reemplazamos en el mensaje los caracteres por los números asignados en el paso 2.

M =

T	O	T	T	O
0	1	0	0	1

5. Se escribe un '0' y en seguida la serie de números del paso anterior y se le asigna la base encontrada en el punto 3.
   $0.01001_{[2]}$

6. Se crea una constante de longitud 'L', cuyo valor está determinado por la cantidad de numero después del punto
   L=5
7. La codificación se da inicio con la conversión del número del paso 5 a decimal, es decir, para este caso, se realiza la conversión de base 2 a base 10
   $0.01001_{[2]} = 0.28125_{[10]}$
8. El resultado de la conversión anterior debe pasarse a binario
   $0.28125_{[10]} = 0.01001_{[2]}$

Esta conversión es el resultado de conversión sucesiva de conversión,

$0.28125_{[10]} * 2 = 0.5625$
$0.5625_{[10]} * 2 = 1.125$
$0.125_{[10]} * 2 = 0.25$
$0.25_{[10]} * 2 = 0.5$
$0.5_{[10]} * 2 = 1$

Las conversiones se realizan tantas veces como sea necesario

Resultado Final: 01001, lo cual corresponde a nuestro mensaje a codificar

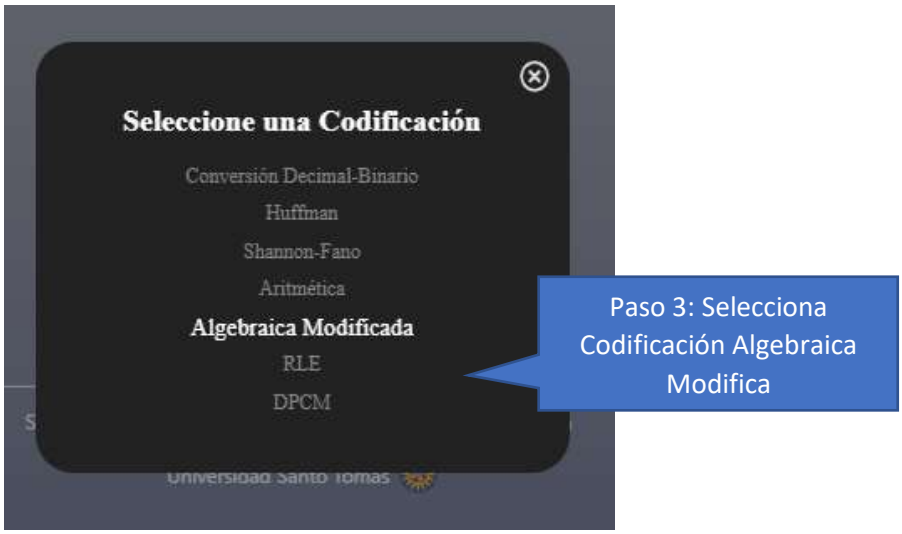
9. Aplicación de la codificación Hamming

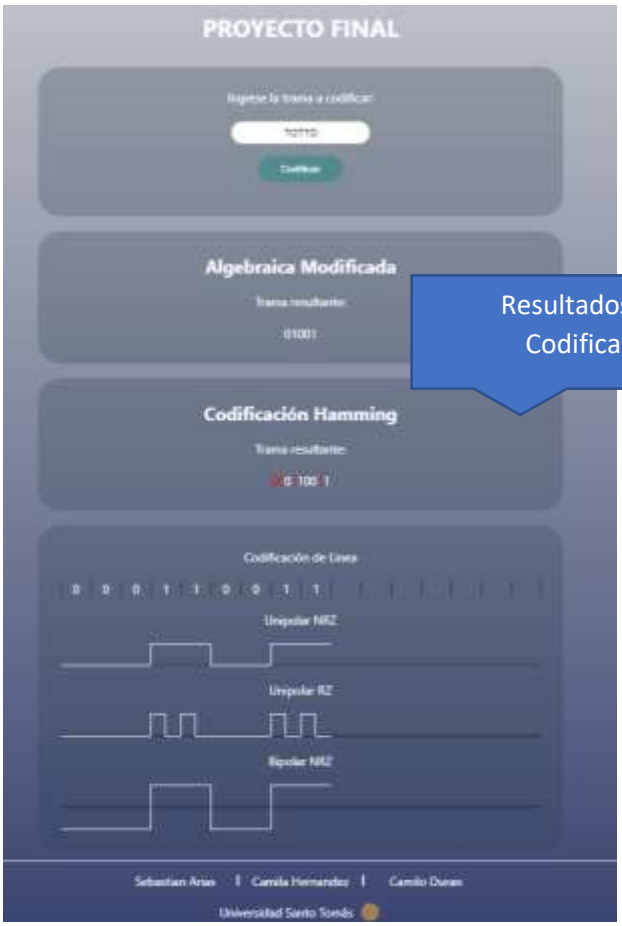
	1	2	3	4	5	6	7	8	9
	P1	P2	D1	P3	D2	D3	D4	P4	D5
T.F	0	0	0	1	1	0	0	1	1
FP1	0		0		1		0		1
FP2		0	0			0	0		
FP4				1	1	0	0		
FP8								1	1

10. Trama de Hamming:
    000110011

11. Comprobación de datos con el programa

Se realizan los dos pasos anteriores descritos en el apartado “pasoso generales para la ejecución del programa” de la página 1, y se prosigue con el paso número 3.





2.6. Codificación RLE

Es un método de compresión de datos y se basa en la repetición de elementos consecutivos. El principio fundamental consiste en codificar un primer elemento al dar el número de repeticiones de un valor y después el valor que va a repetirse. Por lo tanto, es muy util cuando los datos contienen muchas repeticiones.

Ejemplo:

Se desea codificar el siguiente mensaje:

M=[OXXO]

1. Se convierten los caracteres únicos en su respectivo valor en ASCII

M=

O	X	X	O
79	88	88	79

2. Luego de determinar el código ASCII de los caracteres, se procede a contabilizar la cantidad de números requeridos en el código obtenido, el cual, para este caso, la suma seria de 8. Esta suma determinara el tamaño de la matriz.
3. Con la suma anterior, se procederá a establecer el tamaño de la matriz, determinando sus filas y columnas de tal forma que estos sean iguales, y cuyo tamaño estará determinado por la cantidad de datos, el cual, para este caso, sería de 8. Para una suma de 8 números dentro de la matriz, quiere decir que se necesita de un tamaño de matriz de 3 x 3. Porque de esta forma, en la matriz podrán estar los 8 datos, e incluso los sobrantes se llenarán con ceros. Para este caso, la matriz se llenará mediante una lectura en horizontal.

7	9	8
8	8	7
8	9	0

4. Con la matriz anterior, hay que realizar una lectura en zigzag de tal forma que se puedan agrupar sus repeticiones y se escogen la repetición más grande de caracteres y repeticiones, y se encuentran los bits necesarios para representarlos

   7:1

   9:1

   8:3

   7:1

   8:1

   9:1

   0:1

   Número más grande en caracteres: $9 = 1001_{[2]}$ = se necesitan de 4 bits

   Número más grande en repeticiones: $3 = 11_{[2]}$ =se necesita de 2 bits

5. Con los tamaños anteriores, se me obliga a representar los valores de caracteres y repeticiones para cada una de las encontradas. Por ejemplo: en el caso de 7:1, quiere decir que el valor 7 en binario se representara en forma de caracteres, mientras que el valor de repeticiones se representa por dos bits, que para este caso serial el 1.

7:1	9:1	8:3	7:1	8:1	9:1	0:1
011101	100101	100011	011101	100001	100101	000001

   La trama resultante seria la siguiente:

   01110110010110001101110110000110010100000 1

6. Aplicación de codificación Hamming a la trama de la codificación Huffman

6.1. Resultado de la trama de la Codificación Hamming

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	P1	P2	D1	P3	D2	D3	D4	P4	D5	D6	D7	D8	D9	D10	D11	P5	D12	D13	D14	D15	D16	D17	D18
T.F	1	0	0	0	1	1	1	0	0	1	1	0	0	1	0	0	1	1	0	0	0	1	1
FP1	1		0		1		1		0		1		0		0		1		0		0		1
FP2		0	0			1	1			1	1			1	0			1	0			1	1
FP4				0	1	1	1					0	0	1	0					0	0	1	1
FP8								0	0	1	1	0	0	1	0								
FP16																0	1	1	0	0	0	1	1
FP32																							

	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
	D19	D20	D21	D22	D23	D24	D25	D26	P6	D27	D28	D29	D30	D31	D32	D33	D34	D35	D36	D37
…	0	1	1	1	0	1	1	0	1	0	0	0	1	1	0	0	1	0	1	0
FP1		1		1		1		0		0		0		1		0		0		0
FP2			1	1			1	0			0	0			0	0			1	0
FP4					0	1	1	0					1	1	0	0				
FP8	0	1	1	1	0	1	1	0									1	0	1	0
FP16	0	1	1	1	0	1	1	0												
FP32									1	0	1	0	1	1	0	0	0	1	0	0

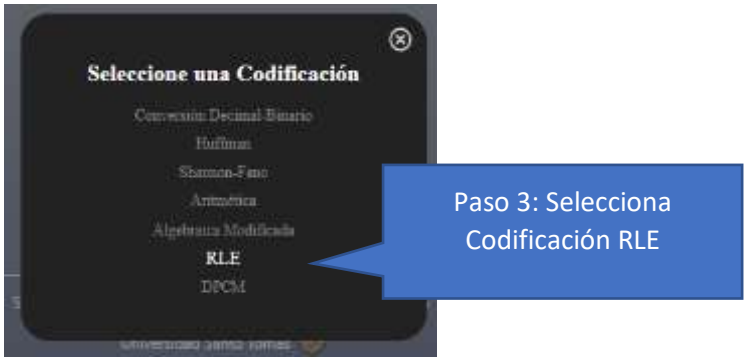
	44	45	46	47	48
	D38	D39	D40	D41	D42
…	0	0	0	0	1
FP1		0		0	
FP2			0	0	
FP4	0	0	0	0	
FP8	0	0	0	0	
FP16					1
FP32	0	0	0	0	1

6.2. Trama de Hamming:

10001110011001001100011011101101000110010100000 1

7 Comprobación de datos con el programa

Se realizan los dos pasos anteriores descritos en el apartado “pasoso generales para la ejecución del programa” de la página 1, y se prosigue con el paso número 3.



2.7. Codificación DPCM

Es un codificador de forma de onda que parte de la base de PCM pero contiene funcionalidades en la predicción de las muestras de la señal.

Ejemplo:

Codificar el siguiente mensaje mediante la utilización de DPCM

M=[LORO]

1. Se convierten los caracteres únicos en su respectivo valor en ASCII

M=

L	O	R	O
76	79	82	79

2. Luego de determinar el código ASCII de los caracteres, se procede a contabilizar la cantidad de números requeridos en el código obtenido, el cual, para este caso, la suma seria de 8. Esta suma determinara el tamaño de la matriz.

3. Con la suma anterior, se procederá a establecer el tamaño de la matriz, determinando sus filas y columnas de tal forma que estos sean iguales, y cuyo tamaño estará determinado por la cantidad de datos, el cual, para este caso, sería de 8. Para una suma de 8 números dentro de la matriz, quiere decir que se necesita de un tamaño de matriz de 3 x 3. Porque de esta forma, en la matriz podrán estar los 8 datos, e incluso los sobrantes se llenarán con ceros. Para este caso, la matriz se llenará mediante una lectura en zigzag

7	6	2
7	8	7
9	9	0

4. Se elabora una nueva matriz entre las diferencias de los valores de las posiciones de la matriz original. La primera posición de esta nueva matriz, debe ser el mismo valor de la matriz original. Estas diferencias deben respetar la lectura inicial, que para este caso fue zigzag, ejemplo: 7-6= -1

7	-1	-6
1	-1	5
2	2	-9

5. En la nueva matriz, se identifica el dato más grande entre los positivos y negativos, es decir, no interesa su signo.
   Dato más grande de la matriz: 9

6. Luego de obtener el número más alto, se procede a representar dicho número a binario; de esta forma, se puede conocer el número de bits que requiere

$$9_{[10]} = 1001_{[2]}$$

   Se requieren de 4 bits para poder representar al 9 en binario

7. Luego de obtener el número de bits requeridos, se procede a sumarle 1 a ese valor encontrado. Esta suma es debido a que hay que agregar el bit de signo a la trama de bits

   4 bits + 1 bit = 5 bits

8. Este nuevo valor de bits será el tamaño que representa a cada trama de datos a codificar, es decir, cada valor será codificada a una trama de 5 bits.
   Ejemplo:

Bit de signo +	Trama de Información			
0	0	0	0	1
Bit de signo -	Trama de Información			
1	0	0	0	1

9. Se procede a convertir cada dato de la matriz de diferencia, a un tamaño binario que cumpla con los requisitos anteriores del paso 8. (Se ha colorado con rojo el bit de signo)

7	-1	1	2	-1	-6	5	2	-9
00111	10001	00001	00010	10001	10110	00101	00010	11001

Trama Final de codificación:

0011110001000010001010001101100010101000101 1001

10. Aplicación de codificación Hamming a la trama de la codificación Huffman

10.1. Resultado de la trama de la Codificación Hamming

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	P1	P2	D1	P3	D2	D3	D4	P4	D5	D6	D7	D8	D9	D10	D11	P5	D12	D13	D14	D15	D16	D17	D18
T.F	1	0	0	1	0	1	1	1	1	1	0	0	0	1	0	1	0	0	0	1	0	0	0
FP1	1		0		0		1		1		0		0		0		0		0		0		0
FP2		0	0			1	1			1	0			1	0			0	0			0	0
FP4				1	0	1	1					0	0	1	0					1	0	0	0
FP8								0	1	1	0	0	0	1	0								
FP16																1	0	0	0	1	0	0	0
FP32																							

	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
	D19	D20	D21	D22	D23	D24	D25	D26	P6	D27	D28	D29	D30	D31	D32	D33	D34	D35	D36	D37
…	1	0	1	0	0	0	1	1	0	0	1	1	0	0	0	1	0	1	0	0
FP1		0		0		0		1		0		1		0		1		1		0
FP2			1	0			1	1			1	1			0	1			0	0
FP4					0	0	1	1					0	0	0	1				
FP8	1	0	1	0	0	0	1	1									0	1	0	0
FP16	1	0	1	0	0	0	1	1												
FP32									0	0	1	1	0	0	0	1	0	1	0	0

	44	45	46	47	48	49	50	51		
	D38	D39	D40	D41	D42	D43	D44	D45		
…	0	1	0	1	1	0	0	1		
FP1		1		1		0		1		
FP2			0	1			0	1		
FP4	0	1	0	1						
FP8	0	1	0	1						
FP16					1	0	0	1		
FP32	0	1	0	1	1	0	0	1		

10.2 Trama de Hamming:

10010110110001010001000101000110011000101000101 1001

11. Comprobación de datos con el programa

Se realizan los dos pasos anteriores descritos en el apartado “pasoso generales para la ejecución del programa” de la página 2, y se prosigue con el paso número 3.