Práctica 2
Codificación de canal

1 Código de repetición

1. Generaremos en primer lugar una función en Matlab que, dado un array de dimensiones $1 \times N$ con bits para transmitir, genere un array $1 \times (nN)$ con las palabras código obtenidas al utilizar un código de repetición con n repeticiones por bit.

   La función debe tener el siguiente encabezado:

   ```
   function codigo = CodificadorRepeticion(bits, n)
   ```

2. A continuación generaremos la función correspondiente al decodificador, que dado un vector $1 \times (nN)$, compruebe cada una de las palabras código y corrija aquellas en las que se detecte algún error tal y como hemos visto.

   ```
   function bits = DecodificadorRepeticion(codigo, n)
   ```

3. Por último, generar una función que simule un canal que introduce errores en los bits con probabilidad p.

   ```
   function codigoRx = Canal(codigoTx, p)
   ```

4. Para probar el sistema construido, simularemos un vector con 10.000 bits aleatorios, que haremos pasar por el codificador, canal y decodificador configurados con $n = 3$ y $p = 0.02$:

   ```
   bitsTx = round(rand(1,10000));
   codigoTx = CodificadorRepeticion(bitsTx, 3);
   codigoRx = Canal(codigoTx, 0.02);
   bitsRx = DecodificadorRepeticion(codigoRx, 3);
   ```

   ¿Cuál es la probabilidad de error final obtenida? Comprobar que coincide con lo visto en teoría.

2 Código de paridad

1. Generaremos en primer lugar una función en Matlab que, dado un array de dimensiones $1 \times N$ con bits para transmitir, genere un array $1 \times (N\frac{n}{n-1})$ con las palabras código obtenidas al utilizar un código de paridad par con palabras de n bits, siendo el último el de paridad.

   La función debe tener el siguiente encabezado:

   ```
   function codigo = CodificadorParidad(bits, n)
   ```

2. A continuación generaremos la función correspondiente al decodificador, que dado un vector $1 \times (N\frac{n}{n-1})$, compruebe cada una de las palabras código y marque aquellas en las que se haya detectado algún error, devolviendo un vector $1 \times N$.

   ```
   function bits = DecodificadorParidad(codigo, n)
   ```

3. Para probar el sistema construido, vamos a volver a utilizar un vector con 10.000 bits aleatorios, que haremos pasar por el codificador, canal y decodificador configurados con $n = 3$ y $p = 0.02$:

   ```
   bitsTx = round(rand(1,10000));
   codigoTx = CodificadorParidad(bitsTx, 3);
   codigoRx = Canal(codigoTx, 0.02);
   bitsRx = DecodificadorParidad(codigoRx, 3);
   ```

   ¿Cuál es la probabilidad de que no se haya detectado algún error?

3 Código Hamming

1. Generaremos una función en Matlab que, dado un valor de q devuelva las matrices de generación y de comprobación de paridad de un Código Hamming:

   ```
   function [G, H] = MatricesHamming(q)
   ```

2. Crearemos ahora una función en Matlab que, dado un array de dimensiones $1 \times N$ con bits para transmitir, genere un array $1 \times N\frac{n}{k}$ con las palabras código obtenidas al utilizar un código Hamming matriz generadora G.

   La función debe tener el siguiente encabezado:

   ```
   function codigo = CodificadorHamming(bits, G)
   ```

3. A continuación generaremos la función correspondiente al decodificador, que dado un vector $1 \times n \cdot N$, compruebe cada una de las palabras código y corrija aquellos errores que detecte.

   ```
   function bits = DecodificadorHamming(codigo, H)
   ```

   La función deberá, en primer lugar, generar la tabla de síndromes. Posteriormente se utiliza la matriz H para obtener el vector con los síndromes de decodificación y se compara cada uno de ellos con la entrada correspondiente de la tabla para obtener el vector corregido.

   Para generar la tabla con los síndromes asociados a los errores más comunes vamos a considerar los $2^q - 1$ vectores de error $1 \times n$ con menor número de unos.

4. Para probar el sistema construido, vamos a volver a utilizar un vector con 10.000 bits aleatorios, que haremos pasar por el codificador, canal y decodificador configurados con $q = 3$ y $p = 0.02$:

   ```
   bitsTx = round(rand(1,10000));
   codigoTx = CodificadorHamming(bitsTx, 3);
   codigoRx = Canal(codigoTx, 0.02);
   bitsRx = DecodificadorHamming(codigoRx, 3);
   ```

   ¿Cuál es la probabilidad de error obtenida tras la decodificación?

5. Podemos comprobar también este codificador utilizando como entrada un mensaje de texto. Representad tanto el mensaje original, como el recibido (sin corregir los errores) y el corregido.

4 ¿Qué entregar?

- Todas las funciones creadas.
- Script del primer ejemplo de Matlab (códigos de repetición).
- Script del segundo ejemplo de Matlab (códigos de paridad).
- Script del tercer ejemplo de Matlab (códigos bloque), tanto con una señal aleatoria como con el texto.

5 Apartado opcional

Con todas las funciones que habéis generado, podéis probar cómo funcionarían para el caso de un fichero de sonido.

Cargad el fichero de audio Brahms_mono.wav. Utilizad para ello la función `audioread`, con el parámetro extra `'native'` para que se carguen las muestras en formato entero de 16 bits, que es el original.

Ahora vamos a transmitir esta señal por un canal con una probabilidad de error de bit de 0.0001. Podéis escuchar el resultado utilizando las funciones `audioplayer` y `play`.

Repetid ahora el proceso anterior pero introduciendo un codificador Hamming con $q = 3$. Comprobad si se nota la mejoría en la calidad del sonido recibido.