Práctica 2
Codificación de canal

1 Introducción

Además de los ficheros de la Práctica 1, en esta práctica utilizaremos algunos archivos adicionales del directorio Python:

- `Practica2.py`: Se trata del archivo principal que debemos ejecutar para desarrollar la práctica. Se trata de un ejemplo de codificación de canal con un archivo de texto como mensaje de entrada.
- `CodCanal.py`. Archivo con una serie de funciones relacionadas con la codificación de canal
  - **paridad_cod**
  - **paridad_dec**
  - **matrices_hamming**
  - **hamming_cod**
  - **hamming_dec**
- `Canal.py`. Archivo con las funciones necesarias para simular un canal de transmisión.
  - **canalAWGN**

Dentro del archivo *FuncionesP2.py* veréis que faltan por desarrollar algunas funciones que se han dejado en blanco:

- **numErrores**: Esta función debe tomar a su entrada los mensajes a la entrada y a la salida del sistema en formato *string* y devolver a su salida el número de errores detectados entre ambos:
- **repeticion_cod**: Esta función debe tomar a su entrada un mensaje en formato binario (*String*), y el parámetro k del codificador de repetición, y devolver a su salida el mensaje codificado.
- **repeticion_dec**: Esta es la función inversa a la anterior. Dado un mensaje recibido del que sabemos que ha sido codificado con un código de repetición con parámetro k, debe devolver el mensaje binario decodificado, con los posibles errores corregidos.

2 ¿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.

3 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.

Una vez hayáis completado estas funciones ya podéis abrir el fichero `Practica2.py` y ejecutarlo. Comprobad que el fichero de salida (salida.txt) coincida con la entrada, y el número de errores producidos al final. Probad a variar la cantidad de ruido en el canal así como a comparar el resultado con otros codificadores de canal que tenéis disponibles en el código (paridad y Huffman).

- El archivo `FuncionesP2.py`.
- Un documento de texto en el que se responda a lo siguiente:
  - ¿Qué diferencias aprecias entre los distintos codificadores de canal en función de ruido en el canal?