UNIVERSIDAD DE GRANADA

Teoría de la Información y la Codificación

4º Grado en Ingeniería Informática

Guión de prácticas
Práctica 2

Canales sin ruido.
Implementación de códigos Huffman.

© Prof. Manuel Pegalajar Cuéllar
Dpto. Ciencias de la Computación e I. A.
Universidad de Granada

DECSAI

Departamento de Ciencias de la Computación e Inteligencia Artificial

Canales sin ruido. Implementación de códigos Huffman.

1. Requisitos

Para la realización de esta práctica es necesario:

- Haber finalizado la práctica 1.
- Haber realizado el "*Seminario 2: Implementación de códigos Huffman*".

2. Contenido

Este documento contiene los trabajos a realizar en las sesiones de laboratorio correspondientes a la práctica 2, junto con las preguntas que el alumno debe saber contestar tras la finalización de las sesiones. En este guión de prácticas se desarrollan habilidades relacionadas con la creación de códigos Huffman y su implementación en dispositivos microcontroladores de bajas capacidades de memoria y cómputo, tales como la plataforma Arduino Uno. Se continúan desarrollando habilidades introducidas en la práctica 1, tales como la construcción, compilación y envío de programas, transmisión de datos por puerto serie y construcción del hardware y la base software de la plataforma de envío y recepción de datos mediante láser.

Cada una de las secciones siguientes se corresponde con las diferentes sesiones de trabajo en el laboratorio (1 sesión= 2 horas presenciales), y guiarán al alumno en la construcción de códigos Huffman, su implementación en un programa Arduino y su utilización para codificación en una plataforma de envío y recepción de datos mediante láser. Finalmente, las últimas secciones contienen un conjunto de preguntas que el alumno deberá entregar al profesor, junto con la descripción del método de evaluación.

3. Metodología de trabajo

Las prácticas deberán ser realizadas por parejas de estudiantes. Con carácter previo a la realización de este cuaderno de prácticas en el laboratorio, cada estudiante deberá haber estudiado **antes de asistir a clase** el Seminario asociado al cuaderno de prácticas y, preferiblemente, también las explicaciones disponibles en este documento para la sesión de prácticas correspondiente. Si es posible, también es recomendable haber realizado previamente esquemas, diseños e incluso desarrollos que favorezcan realizar las prácticas de laboratorio de forma más fluida. **No es recomendable asistir a clase de prácticas sin haber estudiado previamente el presente cuaderno de prácticas ni el seminario asociado**, pues se corre el riesgo de no poder terminar a tiempo los trabajos requeridos. Con carácter general, **el estudiante debe tener todo el material preparado para poder dedicar las clases de prácticas íntegramente a implementación de los programas en la plataforma Arduino**.

Se recomienda seguir la planificación que se ilustra a continuación para asegurar el éxito en la realización de la práctica:

TAREAS A REALIZAR PARA LA ELABORACIÓN DE LA PRÁCTICA	
Antes de la Sesión 1 en el laboratorio	• Estudiar la teoría de códigos óptimos en canales sin ruido. En especial, la codificación Huffman. • Estudiar el Seminario 2, apartados a) Representación de árboles de codificación y 2) Obtención de códigos Huffman • Estudiar la Sección 4 de este documento. • Realizar pseudocódigo, diagramas o diseños y, si es posible, código fuente, de la tarea final de la sesión.
Sesión 1	• Realización de un programa de lectura de un texto desde fichero y creación de una función para generar el árbol de codificación Huffman asociado a las probabilidades de cada símbolo. • Implementación del árbol en un programa Arduino.
Antes de la Sesión 2 en el laboratorio	• Asegurarse del correcto funcionamiento de las funciones ***sendBit*** y ***recvBit*** desarrolladas en la práctica 1. • Estudiar el Seminario 1, apartado 3) Codificación y decodificación • Estudiar la Sección 4.2 y la sección 5 de este documento. • Realizar pseudocódigo, diagramas o diseños y, si es posible, código fuente, de la tarea final de la sesión. Verificarlos con el profesor.
Sesión 2	• Realizar los montajes requeridos en el trabajo final de sesión e implementar la funcionalidad requerida. Pruebas de funcionamiento.
Antes de la finalización de la entrega de la práctica	• Defensa ante el profesor. • Estudio del tema 3 de teoría. • Elaborar las respuestas al cuestionario final de la práctica.

4. Sesión 1: Representación de árboles de codificación en estructuras matriciales estáticas

4.1. Prerrequisitos

Para elaborar esta sesión es necesario que el alumno haya estudiado previamente el ***Seminario 2: Implementación de códigos Huffman***, apartados:

1. Representación de árboles de codificación
2. Obtención de códigos Huffman

Se asume también que el estudiante ha instalado el entorno IDE de Arduino en su PC y ha finalizado la práctica 1 con éxito.

4.2. Árboles de codificación

Todo código instantáneo puede implementarse en una estructura de datos de árbol. Este árbol se representaría de la siguiente forma:

- Un nodo raíz.
- Por cada nodo, tantos hijos como símbolos tenga el alfabeto del código (salvo los nodos hoja). Cada arco que une un nodo padre con un nodo hijo tendrá asociado unívocamente un símbolo del alfabeto del código.
- Todos los nodos del árbol se identifican con la cadena vacía, salvo los nodos terminales, que se identifican con un símbolo del alfabeto de la fuente cada uno.
- En los nodos terminales, un símbolo del alfabeto de la fuente asociado a la codificación.

La Figura 1 muestra un ejemplo de árbol de código que codifica en cadenas binarias el alfabeto de la fuente {A, B, C, D, E, F}, de modo que A->000, B->001, C->01, D->100, E->101, F->11.

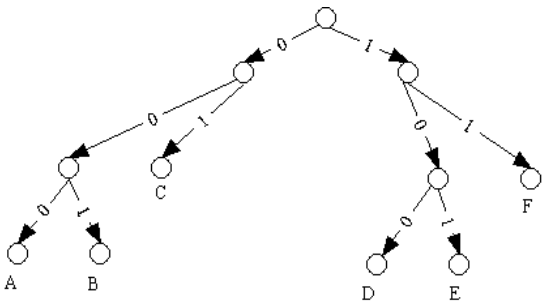

Figura 1: Ejemplo de árbol de codificación

La codificación de un símbolo ***a*** del alfabeto de la fuente en el árbol del código se realiza como sigue:

1. Pila **p**=∅ (vacío)
2. Identificar **n**=nodo hoja que se corresponde con el símbolo ***a***
3. Mientras (**n** no sea la raíz del árbol), hacer:
   1. Añadir a **p** el símbolo del arco que une **Padre(n)** con **n**
   2. Actualizar **n**= **Padre(n)**
4. **código**= vector de **|p|** componentes
5. Mientras (**p** no vacía), hacer:
   1. Insertar al final de **código** el primer elemento de **p**
   2. Sacar el primer elemento de **p**.
6. Devolver **código**

Por ejemplo, para codificar el símbolo **'D'**, el algoritmo anterior devolvería **código="100"**.

La codificación de cadenas de símbolos se efectúa ejecutando el algoritmo anterior para cada símbolo existente en la cadena.

La decodificación de un símbolo se efectúa comenzando desde la raíz, y viajando por el árbol hasta llegar a un nodo hoja. El algoritmo siguiente muestra cómo se realizaría esta operación:

1. Lista de símbolos a decodificar **L**=palabra del código de entrada
2. **N**= nodo raíz, **i**=0
3. Mientras (**N** no es nodo hoja), hacer:
   1. **h**=encontrar el **Hijo(N)** tal que la arista que une **N**->**h** está asociada con el símbolo del alfabeto del código **L[i]**
   2. **N**= **h**, **i**= **i**+1
4. **s**= símbolo asociado al nodo **N**
5. Devolver **s**

La decodificación de cadenas de palabras del código se efectúa ejecutando el algoritmo anterior para cada palabra existente en la cadena. Así, por ejemplo, para la entrada 01100, se devolverán los símbolos del alfabeto de la fuente en la cadena **"CD"**.

4.3. Representación matricial estática de árboles de codificación

La representación de árbol será una matriz de **n** * 4 componentes, donde **n** es el número máximo de nodos que podrá tener el árbol. Suponiendo que el alfabeto de la fuente tiene $\mathbf{2^k}$ símbolos y que la codificación a usar es binaria, entonces podemos estimar $\mathbf{n=2^{k+1}-1}$. Si tuviésemos, por ejemplo, 30 símbolos en el alfabeto de la fuente, entonces **k=5**, y tendríamos un total de $\mathbf{n=2^6-1=63}$ nodos en el árbol como cota superior.

Por tanto, podremos almacenar con seguridad cualquier árbol de Huffman que codifique un total de 30-32 símbolos en un código binario con una matriz **Huffman[63][4]**. Cada componente **i** en la matriz **Huffman[i][j]** tendrá asociado un nodo del árbol, y cada componente **j** contendrá lo siguiente:

- **j=0: Huffman[i][j]** contendrá el símbolo del alfabeto de la fuente asociado al nodo **i**, si **i** es nodo hoja, o el símbolo vacío **'\0'** si no es nodo hoja.

- **j=1: Huffman[i][j]** contendrá el índice de la matriz que contiene al hijo izquierda del nodo **i**, o valor **-1** si **i** es un nodo hoja. De este modo, **Huffman[j][]** es hijo izquierda del nodo **i** si **Huffman[i][1]=j**.
- **j=2: Huffman[i][j]** contendrá el índice de la matriz que contiene al hijo derecha del nodo **i**, o valor **-1** si **i** es un nodo hoja. De este modo, **Huffman[j][]** es hijo derecha del nodo **i** si **Huffman[i][2]=j**.
- **j=3: Huffman[i][j]** contendrá el índice de la matriz que contiene al padre del nodo **i**, o valor **-1** si **i** es el nodo raíz. De este modo, **Huffman[j][]** es padre del nodo **i** si **Huffman[i][3]=j**.

A modo de ejemplo, la Figura 3 muestra la matriz estática que codifica el árbol de la Figura 2, remarcando en rojo el nodo raíz y sus hijos.

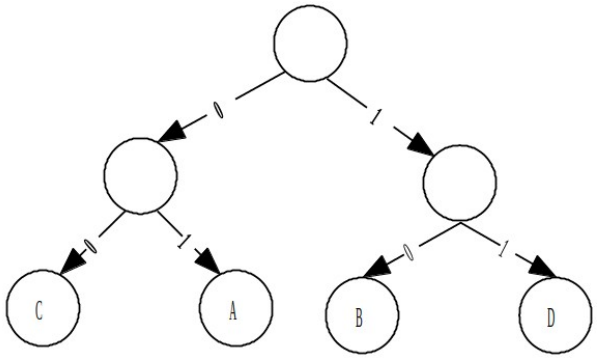

Figura 2: Ejemplo de árbol de codificación Huffman

0	1	2	3	4	5	6
A	B	C	D	- -	- -	- -
- 1	- 1	- 1	- 1	1	2	5
- 1	- 1	- 1	- 1	3	0	4
5	4	5	4	6	6	- 1

Figura 3: Ejemplo de matriz que codifica un árbol Huffman

4.4. Creación del árbol de codificación

Suponemos que disponemos de partida de un vector de **s** componentes **alfabeto[0..s-1]** con los **s** símbolos del alfabeto de la fuente, junto con las probabilidades de ocurrencia de cada símbolo **prob[0..s-1]**. Asignaremos cada símbolo **i=0..s-1** las componentes **0..s-1** de la matriz de codificación Huffman. De este modo, el tamaño de la matriz será **mSize**:

1. Para cada símbolo **i**=0..**s**-1, hacer:
   1. **matriz**[**i**][0]= **Alfabeto**[i] // Asignación del símbolo del alfabeto al nodo i
   2. **matriz**[**i**][1]= **Matriz**[**i**][2]= -1; // El nodo i es hoja, no tiene hijos

3. **matriz**[**i**][3]= -1 // No se conoce el padre
2. **mSize**= **s** // Tamaño actual de la matriz

A continuación, deberemos disponer de un vector de **Nodos[0..s-1]** y las probabilidades de ocurrencia del símbolo asociado a cada nodo **p[0..s-1]**.

1. Para cada símbolo **i**=0..**s**-1, hacer:
   1. **Nodos**[**i**]= **i**;
   2. **p**[**i**]= **prob**[**i**];

A continuación, necesitaremos ordenar los vectores de Nodos y de probabilidades p por orden decreciente de probabilidad, usando cualquier algoritmo de ordenación:

1. Desde **i**=0..**s**-2
   1. Desde **j**=**i**+1..**s**-1
      1. Si (**p**[**i**]<**p**[**j**])
         1. Intercambiar **p**[**i**] y **p**[**j**]
         2. Intercambiar **Nodos**[**i**] y **Nodos**[**j**]

Nos referiremos a este procedimiento como **Ordenar(Nodos, p, s)**, para ordenar los **s** nodos junto con sus probabilidades de forma decreciente.

Acto seguido, el procedimiento para crear el árbol de Huffman consiste en fusionar los dos nodos de menor probabilidad en un nodo nuevo a crear, que sería padre de ambos nodos, y repetir este procedimiento hasta que sólo quede un único nodo en el árbol. Este procedimiento puede resumirse de la siguiente forma:

1. **tam=s**
2. Mientras (**tam**>1), Hacer:
   1. Ordenar(**Nodos**, **p**, **tam**)
   2. **Nodo1= Nodos**[**tam**-1]**, Nodo2= Nodos**[**tam**-2] // Cogemos los nodos de menor prob.
   3. **matriz**[**mSize**][0]= '\0' // Creación del nuevo nodo, que no es hoja
   4. **matriz**[**mSize**][1]= **Nodo1**; // Asignación del hijo izquierda
   5. **matriz**[**mSize**][2]= **Nodo2**; // Asignación del hijo derecha
   6. **matriz**[**Nodo1**][3]= **mSize**; // Asignación del padre del hijo izquierda
   7. **matriz**[**Nodo2**][3]= **mSize**; // Asignación del padre del hijo derecha
   8. **matriz**[**mSize**][3]= -1; // Padre del nodo actual recién creado: Desconocido
   9. **p[tam**-2**]= p[tam**-2**]+p[tam**-1**];** // Fusionamos probabilidades
   10. **tam**= **tam**-1; **Nodos**[**tam**-1]= **mSize**; // Fusionamos los dos nodos en el último
   11. **mSize**= **mSize**+1; // Se incrementa el tamaño de la matriz con el nodo creado
3. **raíz**= **mSize**-1;

Aplicando el procedimiento anterior, se van fusionando en cada iteración los dos nodos de menos probabilidad, creando un nodo padre de ambos, que es un nodo intermedio del árbol. El procedimiento se aplica hasta que no quedan nodos por fusionar y, por tanto, el último que se creó será el nodo raíz del árbol.

Se puede verificar fácilmente cómo la matriz de la Figura 3 se puede obtener para el alfabeto {A, B, C, D}, con las probabilidades P(A)= 0.3, P(C)= 0.3, P(B)= 0.2, P(D)= 0.2.

4.5. Trabajo final de sesión

4.5.1. Material necesario

- PC con compilador de C/C++ instalado
- IDE Arduino instalado en el PC.
- Fichero ***quijote.txt*** con el fragmento de *Don Quijote de la Mancha* proporcionado en la práctica.

4.5.2. Tarea a realizar

Escriba un programa en C/C++ que realice lo siguiente:

- Leer el fichero ***quijote.txt***.
- Crear dos vectores ***alfabeto*** y ***prob***. El vector ***alfabeto*** deberá contener, en mayúsculas, todos los símbolos que aparecen en el fichero ***quijote.txt,*** **salvo el caracter de fin de línea '\n'**. Cada componente ***prob[i]*** deberá contener la probabilidad de ocurrencia del símbolo ***alfabeto[i]*** en el fichero. Esta probabilidad puede calcularse como:

  v= número de veces que el caracter en mayúsculas ***alfabeto[i]*** aparece en el fichero.
  t= número total de caracteres que hay en el fichero.
  prob[i]= v/t

- Crear la matriz que representa el árbol de codificación Huffman según se ha explicado en los apartados 4.3 y 4.4 de este documento.
- Muestre la matriz por pantalla.

Tras la creación del programa, realice la siguiente acción:

- Crear dos programas en Arduino **Emisor** y **Receptor**, e implemente la matriz resultante manualmente como una matriz constante global.

5. Sesión 2: Codificación y decodificación

5.1. Prerrequisitos

Para elaborar esta sesión es necesario que el alumno haya estudiado previamente el ***Seminario 2: Implementación de códigos Huffman***, apartados:

1. Codificación y decodificación

Se debe haber finalizado con éxito la sesión 2 de la práctica 1.

Se asume también que el estudiante ha finalizado con éxito la sesión 1 de prácticas.

5.2. Material necesario

- 2 PCs con IDE Arduino instalado y puerto USB (PC emisor y PC receptor).
- 2 placas Arduino Uno (1 placa para el PC emisor, 1 placa para el PC receptor).
- 2 cables USB de conexión Arduino – PC (1 cable para cada Arduino, emisor y receptor).
- 1 Módulo receptor 2PCS láser no modulador sensor Tubo.
- 1 Módulo emisor láser KY-008.
- 6 Cables conectores macho/hembra.
- 1 kit de soporte para el emisor y receptor.

5.3. Montaje del soporte para el emisor y receptor

Al finalizar la práctica, se deberá poder enviar información a través de láser entre 2 plataformas Arduino diferentes. El sistema de comunicaciones deberá seguir el esquema se que muestra en la Figura 4: Un Arduino emisor codificará información recibida por puerto serie desde el PC emisor, y transmitirá los datos por láser. Por otra parte, otro Arduino receptor muestreará el canal con el sensor fotorreceptor, y captará la secuencia de palabras del código generado, las decodificará, y las enviará a PC por el puerto serie.

Para poder construir con éxito todo este sistema, en primer lugar es necesario disponer de un método fiable para transferir y recibir datos por el canal. Esto conlleva que:

- El láser emisor y el fotorreceptor estén correctamente alineados.

- Exista un protocolo de comunicaciones válido implementado coherentemente tanto en el Arduino emisor como en el Arduino receptor.

En esta sesión de prácticas nos centramos en estos aspectos del sistema, implementando el método para transmitir datos por el canal.

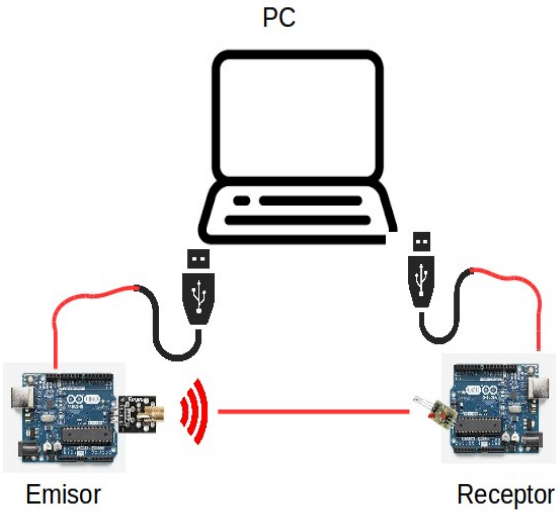

Figura 4: Esquema de montaje del emisor y del receptor

El soporte del montaje para el envío de datos entre el Arduino emisor y el Arduino receptor permitirá que el láser y el fotorreceptor estén correctamente alineados. Este soporte está compuesto por 3 piezas:

- **Base** (Figura 5). Es una placa de plástico con pestañas a cada lado, para sujetar los soportes del emisor y del receptor de modo que estén alineados.
- **Soporte emisor** (Figura 6). Es una pieza de plástico con hendiduras para introducir el emisor láser, permitiendo que la luz pueda ser enviada a través del orificio de salida.
- **Soporte receptor** (Figura 7). Es una pieza de plástico con hendiduras para introducir el fotorreceptor, permitiendo que la luz direccional del láser pueda entrar al interior e incidir en el fotorreceptor.

Figura 5: Base del soporte

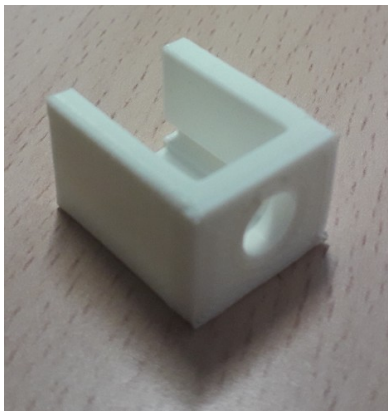

Figura 6: Soporte del emisor

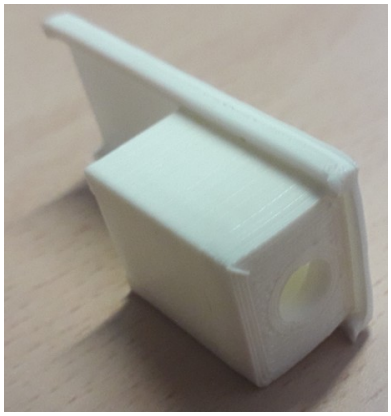

Figura 7: Soporte del receptor

Se deberá montar el soporte, junto con los módulos emisor y receptor de láser según se indica en la Figura 8. **IMPORTANTE: Debemos asegurarnos de que la luz del láser incide correctamente en el cristal del módulo fotorreceptor**.

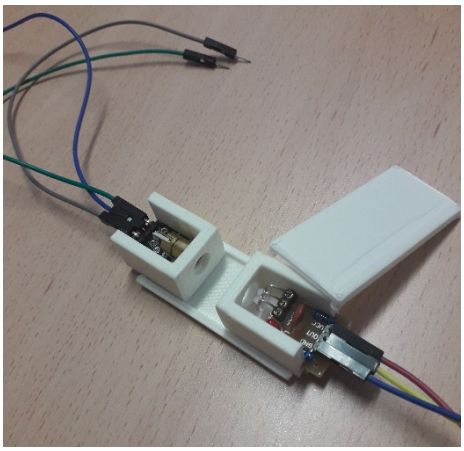

Figura 8: Montaje del soporte

5.4. Conexiones

Una vez realizado el montaje del soporte para la transmisión, se realizarán las conexiones del emisor láser y del fotorreceptor según se indica a continuación:

- **Emisor láser:**
  - Pin **S** del láser al PIN DIGITAL 8 del Arduino emisor.
  - Pin **–** del láser a algún PIN GND del Arduino emisor.
  - El pin restante del láser al PIN de 5V de salida del Arduino emisor.
- **Fotorreceptor:**
  - Pin **GND** del fotorreceptor a algún PIN GND del Arduino receptor.
  - Pin **VCC** del fotorreceptor al PIN de 3.3V de salida del Arduino receptor.
  - Pin **OUT** del fotorreceptor al PIN DIGITAL 7 del Arduino receptor.

El esquema de las conexiones a realizar se muestra en la Figura 9.

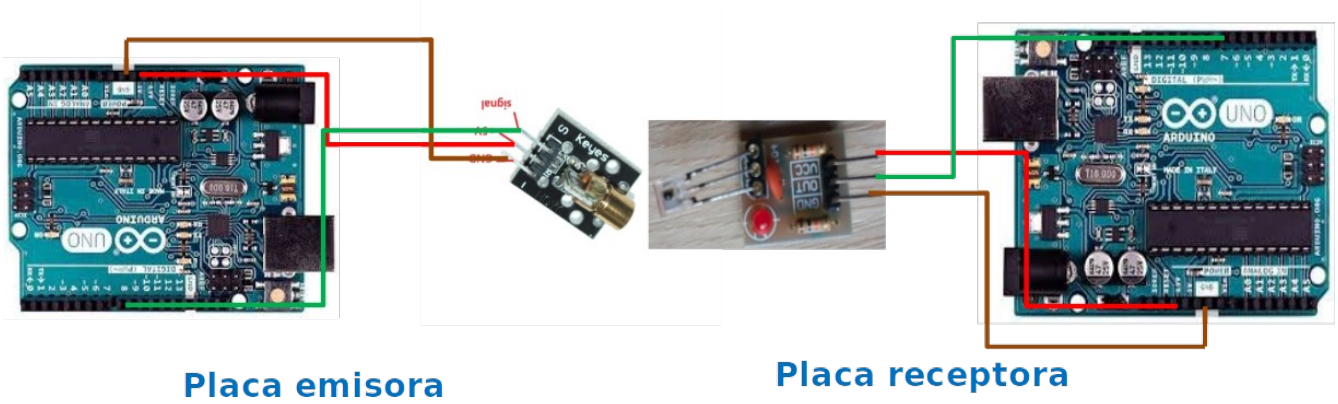

Figura 9: Conexiones de las placas emisora y receptora

5.5. Envío y recepción de bits

Se reutilizarán las funciones ***sendBit*** y ***recvBit*** elaboradas en la práctica 1.

5.6. Trabajo final de sesión

Se realizará un sistema de emisión y recepción de datos codificados con códigos Huffman siguiendo la estructura de emisión/recepción mostrada en el apartado 5.3 de este documento. El sistema tendrá dos subsistemas: Emisor y receptor. La funcionalidad específica de cada subsistema se explica en los siguientes apartados.

5.6.1. Tarea a realizar: Codificación de mensajes

El envío de mensajes por el subsistema de Arduino emisor se realizará en un programa **Emisor** de la siguiente forma:

1. El usuario introducirá una cadena válida, con símbolos del alfabeto de la fuente, a través del ***Monitor Serie*** de Arduino (alfabeto castellano salvo la ñ en mayúscula, signos de puntuación punto (.), coma (,), punto y coma (;), y espacio ( ).
2. El sistema Arduino emisor recibirá la cadena, de máximo 60 caracteres, por el puerto serie.
3. El sistema Arduino emisor codificará la cadena del alfabeto de la fuente, símbolo a símbolo, en palabras del código binario creado con la codificación Huffman, generando una única cadena compuesta por la concatenación de las palabras del código resultantes de la codificación.
4. El sistema Arduino emisor enviará cada uno de los símbolos "0"/"1" resultantes de la codificación por el canal, haciendo uso de la función **sendBit**.
5. El comportamiento anterior deberá poder ejecutarse múltiples veces sin necesidad de reiniciar la placa Arduino.

Para llevar a cabo el punto 3 anterior, se implementará una función ***codifica***, que tenga como entrada una cadena de símbolos de la fuente y, como salida, una cadena con la codificación de todos los símbolos. La codificación de cada símbolo del alfabeto de la fuente, dentro de la función ***codifica***, deberá realizarse siguiendo la guía mostrada en el apartado 4.2 de este documento. Se permite una implementación libre de la pila descrita en el algoritmo de codificación (por ejemplo, cadena de caracteres, array de caracteres, etc.), de modo que la codificación de la cadena ***s*** del alfabeto de la fuente en una cadena de salida ***o*** se realice como sigue:

1. Inicializar **o** a vacío
2. Para cada símbolo **a** en la cadena de entrada ***s***, hacer:
    1. Pila **p**=∅ (vacío)
    2. Identificar **n**=nodo hoja que se corresponde con el símbolo ***a***
    3. Mientras (**n** no sea la raíz del árbol), hacer:
        1. Si **n** es hijo izquierda de **Padre(n)**, añadir bit 0 a **p**
        2. En otro caso, si **n** es hijo derecha de **Padre(n)**, añadir bit 1 a **p**
        3. Actualizar **n**= **Padre(n)**
    4. Mientras (**p** no vacía), hacer:
        1. Insertar al final de **o** el primer elemento de **p**
        2. Sacar el primer elemento de **p**.
3. Devolver **o**

5.6.2. Tarea a realizar: Decodificación de mensajes

La recepción de mensajes por el subsistema de Arduino receptor se realizará en un programa **Receptor** de la siguiente forma:

1. El programa Arduino tendrá un buffer de caracteres, inicializado a vacío.
2. El programa Arduino estará comprobando constantemente si se están recibiendo datos por el láser, usando la función implementada ***recvBit***. Mientras no se reciban datos, no hará nada.
3. Cuando el programa Arduino detecte que el emisor está enviando datos por láser, recogerá los bits existentes en el canal mediante la función ***recvBit*** implementada, y guardará los datos el el buffer.
4. Cuando finalmente el programa Arduino detecte que el emisor ha parado de enviar datos por láser, decodificará la secuencia de bits en una secuencia de símbolos con la función ***decodifica***. Los datos decodificados serán enviados por el puerto serie a PC, como una cadena de caracteres.
5. El usuario podrá visualizar el monitor serie de Arduino, y comprobará que la cadena recibida es la misma que la introducida por el usuario en el PC emisor.
6. Este comportamiento deberá poder ser repetido múltiples veces sin necesidad de reinicializar la placa Arduino.

Para llevar a cabo el punto 4 anterior, se implementará una función ***decodifica***, que tenga como entrada una cadena de símbolos del alfabeto del código y, como salida, una cadena con la decodificación de todos los símbolos en el alfabeto de la fuente. La decodificación de cada símbolo del alfabeto de la fuente, dentro de la función ***decodifica***, deberá realizarse siguiendo la guía mostrada en el apartado 4.2 de este documento, de modo que la decodificación de la cadena ***s*** del alfabeto del código en una cadena de salida ***o*** en el alfabeto de la fuente se realice como sigue:

1. Inicializar **o** a vacío
2. **N**= nodo raíz, **i**=0
3. Mientras (**i** distinto de **|s|**) hacer:
    1. Mientras (**N** no es nodo hoja), hacer:
        1. Si **s(i)** es 0, hacer **h**=Hijo Izquierda de **N**
        2. En otro caso, si **s(i)** es 1, hacer **h**= Hijo derecha de **N**
        3. Hacer **N=h**, i**= i**+1
    2. Insertar el símbolo asociado al nodo **N** al final de **o**
    3. **N**= nodo raiz
4. Devolver **o**

Cabe destacar que el comportamiento anterior asume que la secuencia de entrada *s* contiene códigos válidos codificados con Huffman. En caso contrario, este comportamiento podría no dar los resultados esperados.

6. Entrega y evaluación de la práctica

La práctica debe ser resuelta **por parejas, y contribuirá con 2 puntos sobre 5 (prácticas) a la calificación global de la asignatura.** La evaluación principal será continua, y se realizará por sesiones o al finalizar la práctica. Se evaluarán los siguientes items:

- Funcionamiento del ejercicio final de la sesión 1: 3 puntos.
- Funcionamiento del ejercicio final de la sesión 2: 4 puntos.

El funcionamiento de los ejercicios anteriores se evaluará en el laboratorio. Consistirá en una inspección y entrevista del profesor con los estudiantes, donde se tratarán y se deberán resolver las siguientes cuestiones:

- Prueba del correcto funcionamiento esperado del sistema (mitad de la puntuación).
- Explicación de cómo se ha implementado cada parte (mitad de la puntuación).

Adicionalmente, al finalizar la práctica se deberá presentar al profesor, mediante la entrega habilitada en la plataforma docente web de la asignatura, la siguiente documentación:

- Código fuente final de cada una de las sesiones de la práctica. El código fuente se organizará por carpetas, denominadas "S1" (código de la sesión 1), "S2" (código de la sesión 2). **La no entrega del código supondrá la calificación numérica 0 en el apartado correspondiente, independientemente de la defensa realizada en clase de laboratorio.**
- **Entrega del cuestionario del anexo de este documento**, en formato PDF. Todas las preguntas del cuestionario se valorarán con la misma calificación. La calificación total del cuestionario en la práctica es de 3 puntos.

7. Anexo: Cuestionario de evaluación

Nombre, apellidos y email del estudiante 1:	
Nombre, apellidos y email del estudiante 2:	

CUESTIONARIO

Calcule la capacidad del canal creado en la práctica

Calcule la entropía de la fuente, y compárela con la entropía de la fuente calculada en la práctica 1. En base a estos valores, ¿qué código es más eficiente?

Calcule la tasa de información de la fuente

En base a los valores de la tasa de información de la fuente y la capacidad del canal, indique si el canal se está aprovechando al máximo en la práctica.

¿Cuál es la longitud promedio de las palabras del código creado? ¿Cuánto se tardaría, en promedio, en enviarse 4 símbolos por el canal?

¿Cuánto se tardaría, usando el código uniforme de la práctica 1, en enviarse 4 símbolos por el canal? En base a esta respuesta y a la del apartado anterior, ¿qué código sería preferible usar para el sistema de emisión/recepción de datos por láser?