# Práctica 2

**De la sesión 1 - apartado 3.2**
**HACED EL EJERCICIO 2 Y EL EJERCICIO 3**
**Revisar la sesion 2 (pero esta hecha entera)**
**y si quereis de la sesion 2 hacer el ejercicio**
**9 y 10 (pero en clase dijimos que no**
**la íbamos hacer)**

ugr | Universidad de Granada
Departamento de Ciencias de la Computación e Inteligencia Artificial

DECSAI

# Teoría de la Información y la Codificación
## Cuaderno de Prácticas 2


| | Nombre y apellidos | e-Mail |
|---|---|---|
| Autor 1 | Samuel Cardenete Rodríguez | samuelcr1995@correo.ugr.es |
| Autor 2 | Pablo Parra Garófano | pabloparra@correo.ugr.es |
| Autor 3 | Gema Correa Fernández | gecorrea@correo.ugr.es |


Curso académico: 17/18

Universidad de Granada
Departamento de Ciencias de la Computación
e Inteligencia Artificial

DECSAI

# Teoría de la Información y la Codificación

## Práctica 2: Canales sin ruido. Implementación de códigos Huffman.

### 1 Requisitos

Para la realización de esta práctica es necesario haber realizado el "*Seminario 2: Plataforma láser para envío y recepción de información con códigos Huffman.*", así como haber terminado con éxito la práctica 1 de la asignatura.

### 2 Contenidos

Este documento contiene las preguntas que el alumno debe saber contestar tras la elaboración de las sesiones prácticas correspondientes al seminario 2, referentes a la generación de códigos óptimos en canales sin ruido, su transmisión y recepción en el destino.

### 3 Sesión 1: Códigos instantáneos

#### 3.1. Contenidos de la sesión

- Generación de códigos instantáneos.
- Modificación de las bibliotecas <ticcommardu.h> para envío y recepción de datos por láser.
- Implementación de demostrador simple de envío y recepción de códigos instantáneos.

#### 3.2. Cuestiones sobre códigos instantáneos

1. Diseñe un código instantáneo para el alfabeto {'A', 'B', C'}, más un código adicional de señalización de fin de mensaje, asumiendo que todos los símbolos son equiprobables.

- Un código es instantáneo, si ninguna palabra codificada coincide con el comienzo de otra.
- Código instantáneo para A,B,C: M = {A,B,C}; A = {0,1}
  1. M1 = {A,B}; M2 = {C} entonces M1 → 0 ; M2 → 1
  2. M11 = {A}; M12 = {B} entonces M11 → 00; M12 → 01

  **A → 00 ; B → 01 ; C → 1**
- Código de señalización de fin de mensaje '*':

  Haciendo el procedimiento anterior **A → 00; B → 01; C → 10; * → 11**

2. Micro-Proyecto **CodigoInstantaneo**. El objetivo de este proyecto consiste en implementar las nuevas funciones de emisión y recepción de laserbits en Arduino, **sendLaserBit2** y **recvLaserBit2**, con el fin de implementar un código instantáneo simple que permita tanto comprobar su funcionamiento como las mejoras en velocidad de transmisión de símbolos a través del canal.

Reutilice el proyecto final de la práctica 1, y cambie el nombre de los programas **emisorArdu** y **receptorArdu** a **emisorArduInstantaneo** y **receptorArduInstantaneo**. Modifique **emisorArduInstantaneo** para que:

a. Reciba un mensaje a codificar desde USB. Asuma que el mensaje sólo puede tener secuencias de longitud máxima de 100 caracteres con los símbolos 'A', 'B' y/o 'C'.
b. Tras la recepción del mensaje, envíe un *Laser-bit* 1 por el emisor láser, indicando el comienzo de las comunicaciones. Para ello, debe utilizar la nueva función **sendLaserBit2** desarrollada en el Seminario 2.
c. Codifique cada símbolo a *Laser-bits*, y los envíe por láser utilizando la nueva función **sendLaserBit2** desarrollada en el Seminario 2..
d. Al finalizar de enviar el mensaje, que codifique el símbolo de señalización de fin de mensaje a laserbits, y también los envíe por láser.
e. Envíe un mensaje "OK" por USB al emisor del PC, y vuelva a iterar desde el paso 1.

Modifique **receptorArduInstantaneo** para que:

a. Se inicialice en estado "En espera", y se quede esperando a que el fotorreceptor detecte excitación lumínica.
b. Acto seguido, que reciba el *laserbit* 1 utilizando la nueva función **recvLaserBit2** desarrollada en el Seminario 2, indicando el comienzo de las comunicaciones, y que pase al estado "recibiendo".
c. Mientras esté en estado recibiendo:

a. Lea un *laserbit* por el fotorreceptor, utilizando la nueva función **recvLaserBit2** desarrollada en el Seminario 2.
b. Intente decodificar el símbolo actual con el bit recibido. Si es posible, almacenar el símbolo en el buffer. Si no es posible, que vuelva al paso anterior.
c. Si se ha podido decodificar el símbolo y este es el símbolo de señalización de fin de mensaje, guarde un '\0' al final del buffer, envíe el buffer por USB al receptor del PC y que pase al estado "En espera".

d. Ejecute todo el código anterior en un bucle infinito.

**ENTREGA DEL EJERCICIO:**

- A continuación, indique en qué carpeta de la entrega de la práctica se encuentra el proyecto software de solución del ejercicio (por ejemplo, "Practica2/ProyectoCodigoInstantaneo"). Incluya en la carpeta un fichero AUTORES.txt con el nombre y apellidos del autor o los autores de la práctica:

- Indique el esquema de compilación y qué contiene cada uno de los ficheros:

- Indique las instrucciones para compilar, ejecutar y enviar (en su caso) cada uno de los programas:

3. Comparación del método desarrollado en el proyecto **CodigoInstantaneo** con el codificador Morse de la práctica 1.
    a. Suponga que se desea transmitir la trama "ABBCCA", con los bits codificados en el ejercicio1. ¿Cuántos *laserbits* de información hubiera sido necesario transmitir con el esquema de comunicaciones de la práctica 1 para transmitir dichos códigos? ¿Cuántos *laserbits* de información es necesario transmitir con el esquema de comunicaciones del proyecto **CodigoInstantaneo**? ¿Existe alguna ganancia en la **tasa de información**? Razone su respuesta.

    b. ¿Existe alguna diferencia en la **velocidad de señalización** o en la **velocidad máxima posible** entre ambos esquemas? Razone su respuesta.

c. Suponga que consideramos ahora también los laserbits transmitidos en la práctica 1 para señalización de fin de símbolo y fin de mensaje (este último en ambas prácticas). ¿Cuántos bits, en promedio, son necesarios para codificar un símbolo utilizando el esquema de comunicaciones de la práctica 1? ¿Y utilizando el proyecto **CodigoInstantaneo**?. Explique si existen diferencias entre ambos métodos y cuál es el motivo de las mismas, indicando qué esquema de comunicaciones es mejor.

d. Supongamos que emitimos tramas de 100 códigos. Cuál es la esperanza del tiempo que se tardaría en enviar una trama completa con el sistema de la práctica 1? ¿Y con el sistema del proyecto **CodigoInstantaneo**?

e. Calcule e indique cuál es la capacidad del canal utilizando el mecanismo de transmisión de la práctica 1 y del proyecto **CodigoInstantaneo**.

# 4 Sesión 2: Códigos óptimos

## 4.1. Contenidos de la sesión

- Generación de códigos Huffman.
- Envío y recepción de árboles de codificación Huffman entre un PC y Arduino.
- Envío y recepción de datos codificados con un código Huffman por láser.

## 4.2. Cuestiones sobre códigos Huffman

1. Micro-Proyecto **PruebaProbabilidades.** El objetivo de este proyecto consiste en la creación de un alfabeto de una fuente y el cálculo de las probabilidades de cada símbolo del mismo, como paso previo necesario a la implementación de un código óptimo.
   En el material de la práctica proporcionado en el portal docente de la asignatura, se proporciona una carpeta **"data"** con un fichero de texto **"quijote.txt"**, el cual contiene un fragmento de la obra **"Don Quijote de La Mancha"**, conteniendo todos los caracteres del alfabeto castellano excluyendo la ñ, con los símbolos de puntuación ".", "," y ";" y el carácter " ".

Se han eliminado los demás símbolos de puntuación y las tildes, aunque se han mantenido los caracteres de salto de línea ("\n") por legibilidad del texto. En la práctica, asumiremos que el tamaño del texto proporcionado puede ser representativo del idioma castellano en cuanto al uso de cada símbolo del alfabeto y su frecuencia de aparición en un texto.

Se pide:

a. Crear una nueva biblioteca **<utilsPC.h>**. Implementar las funciones **leerFichero** y **calcularProbabilidades** indicadas en el Seminario 2.

b. Implementar un programa principal que haga uso de estas dos funciones para calcular las probabilidades de cada símbolo del alfabeto y los símbolos de puntuación ".", ",", ";" y " ", junto con otro símbolo adicional de señalización de fin de mensaje (**NOTA**: Este símbolo se considera que aparece una única vez, al finalizar el contenido del fichero de texto). El programa deberá mostrar por pantalla los símbolos ordenados por probabilidades.

c. Responda a la siguiente pregunta: Si tuviésemos un texto escrito con una fuente cuyos símbolos son del mismo tamaño y seleccionamos un punto del texto al azar, ¿qué carácter o caracteres podría/n ser con un 50% de probabilidad de acierto? ¿En qué se basa para dar su respuesta?

Los simbolos que aparecen más veces en el texto, me baso en el resultado obtenido en la salida de calcularProbabilidades() con quijote.txt

Los caracteres serían: "espacio", "E", "A", "O", "S". Ya que el 50% del texto, está escrito con esas caracteres.

**ENTREGA DEL EJERCICIO:**

- A continuación, indique en qué carpeta de la entrega de la práctica se encuentra el proyecto software de solución del ejercicio (por ejemplo, "Practica2/ProyectoPruebaProbabilidades"). Incluya en la carpeta un fichero AUTORES.txt con el nombre y apellidos del autor o los autores de la práctica:

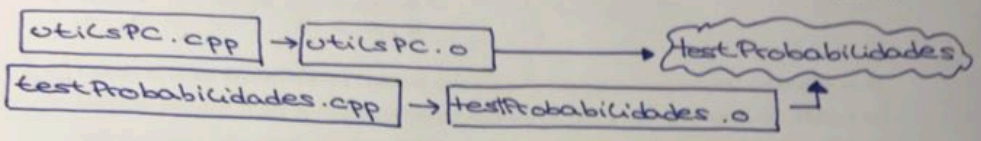
"Practica2/src/testProbabilidades.cpp"

"Practica2/bin/testProbabilidades"

- Indique el esquema de compilación y qué contiene cada uno de los ficheros.

Ficheros usados { utilsPC.h
utilsPC.cpp
testProbabilidades.cpp

ugr | Universidad de Granada
Departamento de Ciencias de la Computación e Inteligencia Artificial
DECSAI




- Indique las instrucciones para compilar y ejecutar:

```
make
./bin/testProbabilidades
```

2. Considere un alfabeto con los símbolos del mensaje "Ata la jaca a la estaca", más un símbolo adicional de señalización de fin de mensaje (seleccione un símbolo cualquiera que no pertenezca al alfabeto anterior). Asumiendo que todos los símbolos fuesen equiprobables, ¿cuál es el mínimo número de bits que serían necesarios para almacenar cada símbolo?. Razone su respuesta.

- Alfabeto = {A, T, □ (espacio), L, J, E, S, C, * (símbolo fin de mensaje)}
- Nuestro alfabeto tiene 9 elementos, usaremos un alfabeto binario {0,1} para codificar cada uno. Por lo que nuestro mínimo número de bits serían 4, ya que:

$$8 = 2^3 < 9 < 2^4 = 16$$

3. Calcule las probabilidades de ocurrencia de los símbolos del alfabeto del ejercicio anterior, contando con el símbolo de señalización de fin de mensaje como parte del alfabeto.

- A → 0'375 (9)
- □ → 0'2083 (5)
- C → 0'083 (2)
- L → 0'083 (2)
- T → 0'083 (2)
- E → 0'04167 (1)
- J → 0'04167 (1)
- S → 0'04167 (1)
- * → 0'04167 (1)

* En los siguientes ejercicios usaré, el número de apariciones de cada símbolo, en vez de la probabilidad, ya que ahí faltan números decimales. Pero de todas maneras obtendré el mismo resultado.

ugr | Universidad de Granada
Departamento de Ciencias de la Computación e Inteligencia Artificial

DECSAI

[ a , 5 , 2 , 2 , 2 , 1 , 1 , 1 , 1 ]
A ▫ C L T E J S *

**4.** Calcule el árbol de codificación Huffman que surgiría con las probabilidades calculadas en el ejercicio anterior. ¿Cuál es la esperanza del número de bits a usar por símbolo? ¿Mejora en algo al mínimo número de bits que serían necesarios para almacenar cada símbolo calculado en el ejercicio 1 de este apartado?

Árbol de codificación Huffman

1. [ a , 5 , 2 , 2 , 2 , 1 , 1 , 1 , 1 ]
A ▫ C L T E J S *
[S* → 0 ; EJ → 1 ; [S → 0 ; * → 1
[E → 0 ; J → 1

2. [ a , 5 , 2 , 2 , 2 , 2 , 2 ]
A ▫ C L T → EJ
S*

3. [ a , 5 , 2 , 2 , 2 , 4 ]
A ▫ C L T → S*EJ
[L → 0 ; T → 1

4. [ a , 5 , 4 , 2 , 4 ] ⇨ [ a , 5 , 4 , 4 , 2 ]
A ▫ C → LT → S*EJ
A ▫ S*EJ → LT → C
[LT → 0 ; C → 1

5. [ a , 6 , 5 , 4 ] [A , LTC , ▫ , EJS* ] [ ▫ → 0 ; S*EJ → 1

6. [ a , a , 6 ] [A , ▫S*EJ , LTC ] [ ▫S*EJ → 0 ; LTC → 1

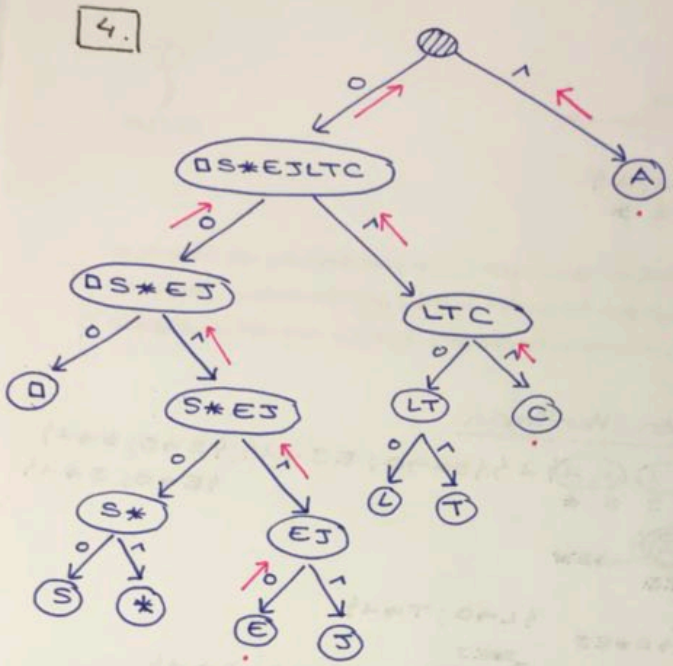
7. [ 15 , a ] [ ▫S*EJLTC , A ] [ ▫S*EJLTC → 0 ; A → 1

**5.** Codifique el mensaje "ACE" utilizando el código Huffman anterior, y proporcione la cadena de bits resultante.

- Mirar proceso atrás en rosa.
- Mensaje ACE: 1 011 00110
  * Si incluiríamos al final del mensaje el '*': 00101

ETS Ing. Informática y de Telecomunicación. C/ Daniel Saucedo Aranda s/n. 18071. Granada (Spain). Tlf: +34 958 244019. Fax: +34 958 243317

4.




* Ir al nodo hoja, y subir hasta la raíz, por esos nodos intermedios. La secuencia obtenida, se le da la vuelta, es decir:
  - Nodo C a Nodo Raíz: "110"
  - Darle la vuelta: "011"

- ¿Cuál es la esperanza del número de bits a usar por símbolo? ¿Mejora? (TODOS SÍMBOLOS EQUIPROBABLES)
  * Esperanza = $4'5 \leq 5$ $\left(\sum_{i=1}^{9} p(S=s_i) * I(S=s_i)\right)$
  * No mejora

ugr | Universidad de Granada
Departamento de Ciencias de la Computación e Inteligencia Artificial

DECSAI

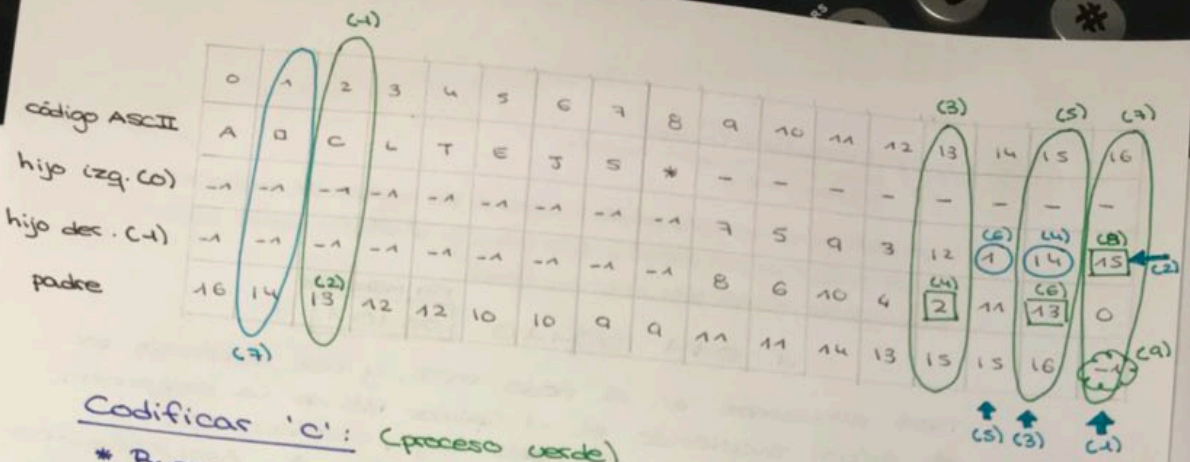
6. Suponiendo que se tiene como entrada la cadena de bits resultante del ejercicio anterior, indique un algoritmo para decodificar dicha cadena en los símbolos iniciales. Explique el proceso a seguir utilizando dicha cadena como ejemplo.

FIN MENSAJE

1 011 00110 00101

Nos situamos en el nodo raíz, y nos movemos por el árbol siguiendo el 1 (primer bit de la secuencia), hemos llegado a un nodo hoja, codificamos y guardamos en un buffer la letra A. Actualizamos Nodo Actual = Raíz, y hacemos el mismo procedimiento: leemos bit de la secuencia y bajamos por el árbol, hasta que no lleguemos a un nodo hoja, no paramos de leer (011), llegamos al nodo hoja C, guardamos en el buffer la letra y volvemos a poner el nodo actual a la raíz. Volvemos hacer el mismo procedimiento. Terminaremos de leer nuestro mensaje con el símbolo de finalización (00101), y devolvemos el buffer (cadena decodificada)

7. Atendiendo a la representación de árbol vista en clase (utilizando una matriz), codifique el árbol de Huffman obtenido en el ejercicion 4 de este apartado dentro de una matriz. Explique cómo codificar un símbolo y como decodificar una cadena de bits. Utilice, para ello, el mensaje "ACE" de los ejercicios anteriores.




| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| código ASCII | A | □ | C | L | T | E | J | S | * | – | – | – | – | – | – | – | – |
| hijo izq. (0) | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 7 | 5 | 9 | 3 | 12 | (6) 1 | (4) 14 | (8) 15 | (2) |
| hijo der. (1) | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 8 | 6 | 10 | 4 | (4) 2 | 11 | (6) 13 | 0 | |
| padre | 16 | 14 | (2) 13 | 12 | 12 | 10 | 10 | 9 | 9 | 11 | 11 | 14 | 13 | 15 | 15 | 16 | (9) -1 |

## Codificar 'C': (proceso verde)

* Buscamos el nodo hoja que contiene el símbolo desde el comienzo de la tabla (4), cogemos su padre (2) y visitamos el nodo (3), comprobamos qué hijo es el nodo anterior (4), es el hijo derecha, ponemos un 1, visitamos el padre del nodo actual (5) y vemos que el nodo anterior es hijo derecha (6) así que ponemos un 1, visitamos el padre de ese nodo (7) y comprobamos que es hijo derecha (8), por eso que guardamos un 0, vamos al padre de ese nodo (9) y como tiene -1, nos dice que ya hemos llegado a la raíz. Hemos obtenido la secuencia 110, debemos darle la vuelta para que esté de la raíz al nodo hoja **011**

## Decodificar '000': (proceso azul)

* Partimos del nodo raíz (1), decodificamos el 0, como es 0, nos movemos al hijo izquierda (nodo 15) (2) (3), como es nodo intermedio seguimos decodificando ahora el 0, nos movemos al hijo izquierda (4) (5), como es nodo intermedio, seguimos decodificando, ahora el 0, nos movemos al hijo izquierda (6) y vamos a ese nodo (7), como es nodo hoja, decodificamos **'□'** y reiniciaizamos el árbol.

8. Micro-Proyecto **HuffmanTree**. El objetivo de este proyecto consiste en implementar el algoritmo de Huffman para cálculo de códigos óptimos, utilizando los métodos para adquisición del alfabeto de entrada y sus probabilidades elaborados previamente en el proyecto **PruebaProbabilidades**, así como comprobar su correcto funcionamiento. El árbol que representa la codificación Huffman deberá ser compartido por el PC (dado que este deberá generar el árbol a partir del alfabeto de la fuente) como por Arduino (porque será necesario utilizar el árbol para codificación y decodificación de los símbolos). Debemos prestar especial atención a utilizar únicamente C/C++ estándar compatible entre el PC y Arduino.

Para ello, crearemos una nueva biblioteca **<Huffman.h>** (y su correspondiente fichero de implementación **Huffman.cpp**), que será compilada por separado para PC (utilizando gcc/g++) y para Arduino (utilizando arv-gcc). El contenido de esta de esta biblioteca deberá ser:

- Creación de la estructura **ArduTree** explicada en el Seminario 2, para almacenar el árbol con un máximo de 63 nodos.
- Declaración de la función **Huffman** para generar un árbol de codificación Huffman partiendo de un alfabeto de entrada y sus probabilidades.

- Declaración de la función **HuffmanEncoder** que, dado un árbol de codificación Huffman y un código ASCII de entrada, proporcione el código Huffman asociado al mismo, junto con el número de bits de dicho código.
- Declaración de la función **InitHuffmanDecoder**, para inicializar un árbol de Huffman para decodificación.
- Declaración de la función **HuffmanDecodeBit**, para tratar de decodificar un bit en el árbol que representa una codificación Huffman y, en caso de alcanzar la decodificación de un símbolo, devolver dicho símbolo.

La estructura **ArduTree** y la declaración de las funciones deberá realizarse en el fichero **Huffman.h**. Las funciones deberán ser implementadas en el fichero **Huffman.cpp**. Tras implementar esta nueva biblioteca, se pide:

a. Elabore un programa, utilizando la biblioteca anterior, que calcule el alfabeto (+ un nuevo símbolo de señalización de fin de mensaje) para los caracteres de la frase "Ata la jaca a la estaca" y la probabilidad de cada símbolo. Con la función **Huffman**, genere el árbol de codificación y muéstrelo por pantalla. Verifique que la matriz del árbol se corresponde con el árbol generado a mano en el ejercicio 4 de este apartado. ✓
b. Utilizando el codificador Huffman (función **HuffmanEncoder**), genere la secuencia de bits del mensaje "ACE" (**no olvidar el símbolo de finalización**) y muestre el resultado por pantalla. Compare los resultados con la matriz del árbol generado y con los resultados calculados a mano en el ejercicio 5 de este apartado. ✓
c. Asumiendo la secuencia de bits de salida del apartado anterior, utilice las funciones **InitHuffmanDecoder** y **HuffmanDecodeBit** para decodificar la secuencia de bits en los caracteres ASCII correspondientes y muestre el resultado por pantalla. ✓

**ENTREGA DEL EJERCICIO:**

- A continuación, indique en qué carpeta de la entrega de la práctica se encuentra el proyecto software de solución del ejercicio (por ejemplo, "Practica2/ProyectoHuffmanTree"). Incluya en la carpeta un fichero AUTORES.txt con el nombre y apellidos del autor o los autores de la práctica:

  "Practica2/src/testHuffman.cpp"
  "Practica2/bin/testHuffman"

- Indique el esquema de compilación y qué contiene cada uno de los ficheros.

  Ficheros usados:
  - utilsPC.cpp / utilsPC.h
  - Huffman.cpp / Huffman.h
  - testHuffman.cpp

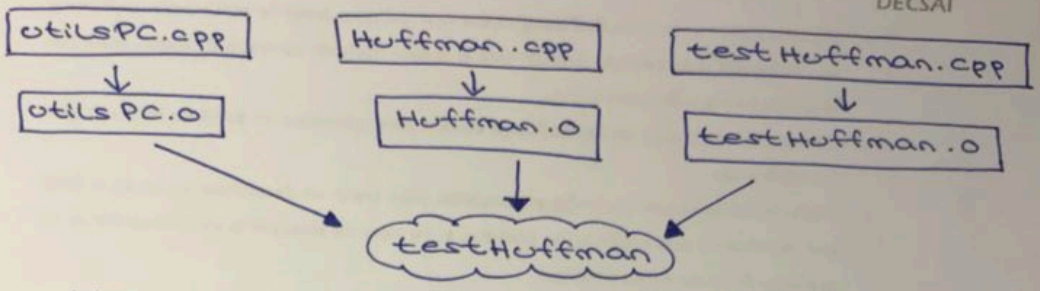
código AS

hijo izq

hijo der

padre






- Indique las instrucciones para compilar y ejecutar:

```
make
./bin/testHuffman
```

X NO HECHO Micro-Proyecto **ShareTree**. El objetivo de este proyecto consiste en implementar las funciones para enviar y recibir el árbol de codificación Huffman por USB hacia y desde PC o Arduino, así como comprobar su correcto funcionamiento. Se pide:

a. Añadir las funciones **sendHuffmanUSB(int &pd, const ArduTree &tree)** y **receiveHuffmanUSB(int &pd, ArduTree &tree)**, descritas en el Seminario 2, en la biblioteca de comunicaciones para PC **<ticcommpc.h>**, y su correspondiente implementación en el fichero **ticcommpc.cpp**.

b. Añadir las funciones **sendHuffmanUSB(const ArduTree &tree)** y **receiveHuffmanUSB(ArduTree &tree)**, descritas en el Seminario 2, en la biblioteca de comunicaciones para Arduino **<ticcommardu.h>**, y su correspondiente implementación en el fichero **ticcommardu.cpp**.

c. Elabore el programa **ShareTreeArdu** para Arduino. Este programa deberá inicializar el puerto UART y el servicio de interrupciones para utilizar el USB, y declarar una variable con la estructura de tipo **ArduTree**. Posteriormente, en un bucle infinito, deberá recibir por USB un árbol de codificación Huffman, guardarlo en la variable declarada, y volver a enviarlo al PC.

d. Elabore el programa **ShareTreePC** para PC. Este programa deberá generar el árbol de codificación Huffman del proyecto **HuffmanTree** anterior, mostrarlo por pantalla, enviarlo a Arduino por USB y esperar a recibirlo de nuevo por USB. Se mostrará este

árbol por pantalla, junto con un mensaje indicando si el árbol es el mismo que el enviado por USB o si existe alguna diferencia.

**ENTREGA DEL EJERCICIO:**

- A continuación, indique en qué carpeta de la entrega de la práctica se encuentra el proyecto software de solución del ejercicio (por ejemplo, "Practica2/ProyectoShareTree"). Incluya en la carpeta un fichero AUTORES.txt con el nombre y apellidos del autor o los autores de la práctica:

- Indique el esquema de compilación de cada programa y qué contiene cada uno de los ficheros.

- Indique las instrucciones para compilar y ejecutar cada uno de los programas (o para enviar a Arduino, en su caso).

10. **Proyecto final de la práctica**: **HuffmanProject**. Todos los micro-proyectos presentados en este Cuaderno de Prácticas han estado orientados a crear las infraestructuras software necesarias para poder desarrollar este proyecto final con simplicidad:
    a. El micro-proyecto **CodigoInstantaneo** nos ha permitido implementar las funciones **sendLaserBit2** y **recvLaserBit2** y probar con éxito su funcionamiento.

b. El micro-proyecto **PruebaProbabilidades** nos a ayudado a adquirir un alfabeto de la fuente a partir de un texto escrito, y a calcular las probabilidades de cada símbolo del alfabeto.
c. En el micro-proyecto **HuffmanTree** hemos desarrollado los métodos para construir un árbol de forma eficiente, crear un árbol de codificación Huffman, codificar un símbolo ASCII en un código Huffman, y decodificar una secuencia de bits de un código Huffman a un código ASCII.
d. El micro-proyecto **ShareTree** nos ha permitido ampliar las bibliotecas **ticcommpc** y **ticcommardu** para enviar y recibir un árbol de codificación Huffman entre un PC y Arduino, y probar que la transmisión se realiza con éxito.

Este proyecto agrupa todos los desarrollos previos, y su objetivo consiste en crear un sistema de comunicaciones por láser utilizando un código Huffman para la transmisión. El proyecto, con respecto a su estructura, es similar al proyecto final de la práctica 1 dado que se compone de 4 programas:

- **emisorPC**. Encargado de:
  - Leer el fichero con el fragmento de ***Don Quijote de La Mancha***,
  - Calcular el alfabeto de la fuente y las probabilidades de cada uno de sus símbolos,
  - Crear el árbol de Huffman asociado al alfabeto y sus probabilidades,
  - enviarlo al Arduino emisor, y
  - quedar a la espera de que el usuario introduzca mensajes por teclado. Estos mensajes se enviarán a Arduino emisor (**¡sólo si son válidos!**), el cual los codificará y enviará por láser.

- **receptorPC**. Encargado de:
  - Leer el fichero con el fragmento de ***Don Quijote de La Mancha***,
  - Calcular el alfabeto de la fuente y las probabilidades de cada uno de sus símbolos,
  - Crear el árbol de Huffman asociado al alfabeto y sus probabilidades,
  - enviarlo al Arduino emisor, y
  - quedar a la espera de que el Arduino receptor le envíe cadenas ASCII por USB. Cada vez que se reciba una cadena, se deberá mostrar por pantalla.
  - El programa sólo terminará cuando se pulse la secuencia de teclas **CTRL-C**, la cual deberá controlarse por interrupciones para liberar los recursos reservados en el programa (cierre de puertos USB, liberación de memoria dinámica…).

- **emisorArdu.** Encargado de:

- Leer por USB el árbol de Huffman que le envía el emisor PC.
- En un bucle infinito, quedar a la espera de que el emisor PC le envíe tramas de mensajes (no superiores a 100 Bytes). Tras recibir por USB el mensaje a enviar:
    - Enviará por láser un BIT **LASER_HIGH**, para indicar el comienzo de las comunicaciones.
    - Para cada símbolo del mensaje:
        - Lo codificará con el código de Huffman dado en el árbol.
        - Enviará los bits del código por láser.
        - Pasará a procesar el siguiente símbolo del mensaje.
    - Al finalizar, enviará el símbolo de señalización de fin de mensaje y volverá a quedar a la espera de una nueva trama de datos por USB.

- **receptorArdu**. Encargado de:
    - Leer por USB el árbol de Huffman que le envía el emisor PC. Acto seguido, en un entrará en modo **"En espera"** y comenzará un bucle infinito que realizará las siguientes acciones:
        - Comprobar la entrada del fotorreceptor. Mientras que la entrada no indique detección de luz, no hará nada.
        - Cuando detecte una señal de luz por el fotorreceptor, leerá el bit de comienzo de transmisión de datos y pasará al estado **"Recibiendo"**. También inicializará el decodificador Huffman.
        - Mientras esté en estado **"Recibiendo"**:
            - Leerá un bit recibido por láser.
            - Intentará decodificar el bit.
            - Si se consigue decodificar un símbolo, se añadirá a un buffer y se reinicializará el decodificador.
            - Si el símbolo era la señalización de finalización, se enviará el buffer por USB al recepto PC y se pasará al estado **"En espera**".

**ENTREGA DEL EJERCICIO:**

- A continuación, indique en qué carpeta de la entrega de la práctica se encuentra el proyecto software de solución del ejercicio (por ejemplo, "Practica2/ProyectoHuffmanProject"). Incluya en la carpeta un fichero AUTORES.txt con el nombre y apellidos del autor o los autores de la práctica:

- Indique el esquema de compilación de cada programa y qué contiene cada uno de los ficheros.

- Indique las instrucciones para compilar y ejecutar cada uno de los programas (o para enviar a Arduino, en su caso).

# 5 Evaluación

La evaluación del cuaderno se regirá por las siguientes normas de calificación:

- La nota final del cuaderno será numérica de 0 a 10.
- La sesión 1 se evaluará hasta un máximo de 1,5 puntos:

| Ejercicio | Puntuación |
|---|---|
| **3.2.1** | 0,3 |
| **3.2.2** | 0,7 |
| **3.2.3** | 0,5 |

- La sesión 2 se evaluará hasta un máximo de 8,5 puntos:

| Ejercicio | Puntuación |
|---|---|
| **4.2.1** | 0,6 |
| **4.2.2** | 0,3 |
| **4.2.3** | 0,3 |
| **4.2.4** | 1 |
| **4.2.5** | 0,5 |
| **4.2.6** | 0,5 |
| **4.2.7** | 0,5 |
| **4.2.8** | 1 |
| **4.2.9** | 0,8 |
| **4.2.10** | 3 |

Para aquellos ejercicios que tengan más de un apartado, cada apartado se valorará con igual puntuación salvo el proyecto final de la práctica (**4.2.10**), donde el **emisorPC** y el **receptorPC** no se incluyen en la evaluación porque pueden reutilizarse en gran medida desde el código fuente desarrollado en la práctica 1 y en los ejercicios 4.2.1, 4.2.8 y 4.2.9 de este cuaderno.

Independientemente de la sesión a la que pertenezcan, el código fuente de los programas deberá funcionar para que la parte de desarrollo práctico de su sesión sea evaluada. En caso contrario, la calificación de esa parte será de 0 puntos. Con estas pautas, los ejercicios se evaluarán atendiendo a los siguientes criterios:

- **Calificación NO APTO:** El ejercicio no está resuelto, no se responde a todo lo que se requiere o está mayormente incompleto.
- **Calificación APTO:** El ejercicio está resuelto, pero contiene fallos o no profundiza en la respuesta a la pregunta con el detalle requerido.
- **Calificación DESTACA:** El ejercicio está resuelto con la profundidad requerida y no tiene errores, o contiene mínimos fallos menores (tipográficos, etc.).

Una calificación **NO APTO** supone que el ejercicio se evalúa con 0 puntos. La calificación **APTO** indica que el ejercicio se evalúa con la mitad de su valor. Por último, la calificación **DESTACA** otorga la máxima calificación del ejercicio.

**La detección de copia en algún ejercicio supondrá la calificación de 0 en todos los ejercicios entregados.**