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

DECSAI

Teoría de la Información y la Codificación

Cuaderno de Prácticas 1

	Nombre y apellidos	e-Mail
Autor 1		
Autor 2		

Curso académico:

Teoría de la Información y la Codificación

Práctica 1: Estudio y construcción de plataforma para envío y recepción de información por láser.

1 Requisitos

Para la realización de esta práctica es necesario haber realizado el "*Seminario 1: Introducción a Arduino. Diseño y construcción de plataforma para transmisión de datos por láser*".

2 Contenidos

Este documento contiene las preguntas que el alumno debe saber contestar tras la elaboración de las sesiones prácticas correspondientes al seminario 1, referentes a la utilización de la plataforma Arduino Uno, 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.

3 Sesión 1: Primeros pasos

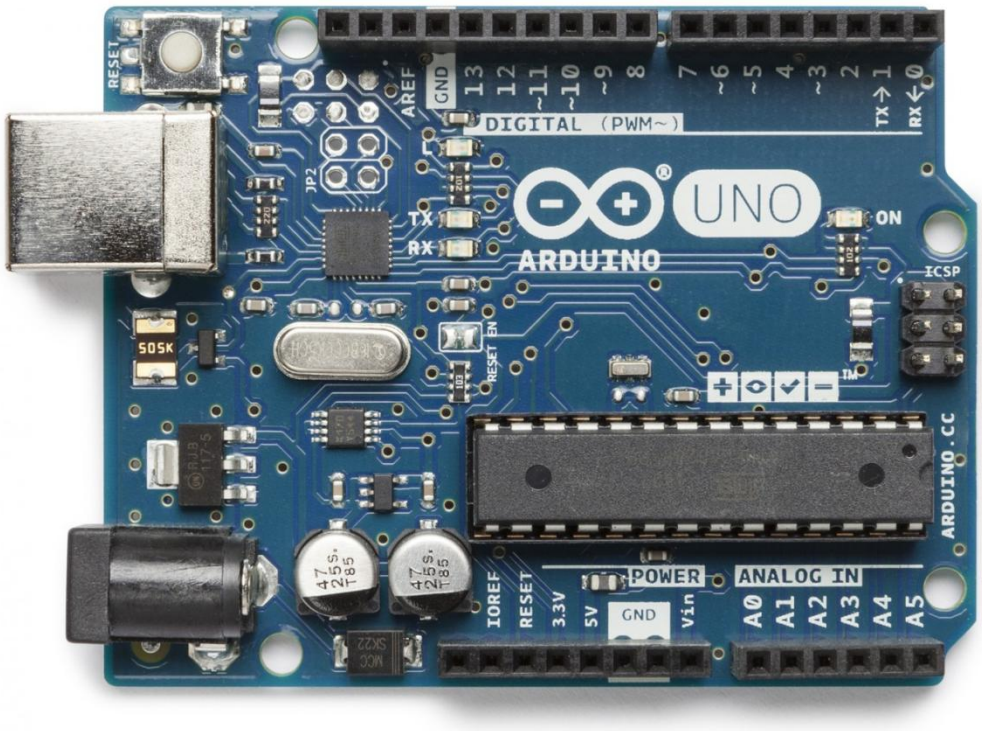
3.1. Contenidos de la sesión

- Arduino Uno: Puertos y pines.
- El microprocesador AVR AtMega328p.
- Compilación y envío de programas.

3.2. Cuestiones sobre Arduino Uno

Estudie y realice el "*Seminario 1: Introducción a Arduino. Diseño y construcción de plataforma para transmisión de datos por láser*". Posteriormente, responda a las siguientes cuestiones **buscando, en los casos en los que sea necesario, ampliación de la información requerida a través de Internet**:

1. En la siguiente figura, describa las distintas componentes de la placa Arduino Uno y para qué se utilizan:



2. ¿Qué es un puerto digital? ¿Y un puerto analógico? ¿En qué se diferencian?

3. ¿Cuántos puertos analógicos tiene la placa Arduino Uno? ¿Cuántos pines en el puerto? ¿cuáles son? ¿Son de entrada, de salida, o de entrada/salida?

4. ¿Cuántos puertos digitales tiene la placa Arduino Uno? ¿Cuántos pines en el puerto? ¿cuáles son? ¿Son de entrada, de salida, o de entrada/salida?

5. Basándose en el dibujo de la pregunta 1 de este apartado, ¿Cuántas tomas de tierra tiene la placa Arduino Uno? ¿Para qué sirve una toma de tierra?

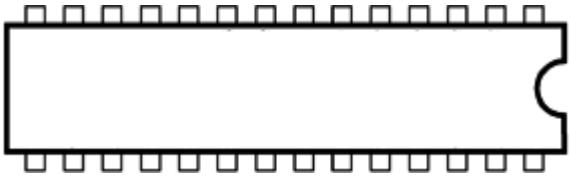
6. Describa 3 formas diferentes de alimentar con corriente a la placa Arduino Uno. Indique cuál es el voltaje máximo recomendado de alimentación por Vin.

7. Describa las precauciones básicas necesarias para asegurar un manejo adecuado de la placa Arduino Uno.

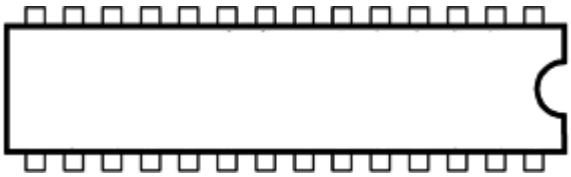
3.3. Cuestiones sobre el microprocesador AVR AtMega328p

1. Busque por Intenet la hoja de especificaciones del microprocesador AVR AtMega328p (*AtMega328p datasheet*). Indique sus especificaciones más relevantes (como mínimo: voltaje de funcionamiento, frecuencia de trabajo del microprocesador, número de pines, número de puertos y su tipo).

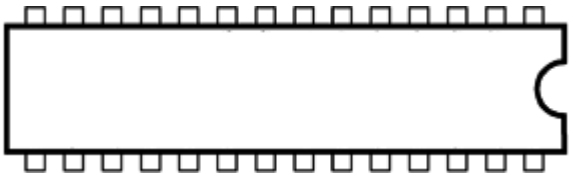
2. Indique cómo se realiza la numeración de las patillas del microprocesador (ver su hoja de especificación). En la siguiente figura, indicar dónde se encuentran las patillas 1, 2, 13, 14, 15, 16, 27 y 28.



3. En la siguiente figura, indique cuáles son las patillas del microprocesador dedicadas a tierra y al puerto de comunicaciones digital B. Identifique los pines PB0 a PB5.



4. Indique, en la siguiente figura, cuál es la asociación entre los pines (patillas) del microprocesador AVR AtMega328p y los pines de los puertos digitales de la placa Arduino Uno.



5. ¿Qué tipos de memoria tiene Arduino Uno? ¿Qué capacidad de memoria tiene cada tipo?

6. ¿Cómo se almacenan los programas en Arduino Uno (disco duro, tarjeta SD, memoria flash…? ¿Dónde se encuentra esta memoria dentro de la placa?

7. ¿Para qué sirve la SRAM y la EEPROM en la placa Arduino Uno?

8. Liste diferentes tipos de placas Arduino y qué microprocesador contiene cada uno. ¿Sabría encontrar información sobre alguna otra placa (distinta de Arduino Uno y no necesariamente Arduino), que utilice el microprocesador AtMega328p?

3.4. Cuestiones sobre la compilación de programas y su copia a Arduino Uno

1. ¿Qué es el programa avr-gcc? ¿para qué sirve?

2. ¿Para qué sirve la opción –Os del programa avr-gcc?

3. ¿Para qué sirve la opción –mmcu del programa avr-gcc? ¿Qué efecto tiene compilar con la opción del programa -mmcu=atmega328p?

4. Explique qué hace la siguiente orden:

   avr-gcc -Os -mmcu=atmega328p -c -o main.o main.cpp

5. Explique qué hace la siguiente orden:

   avr-gcc -mmcu=atmega328p main.o -o main.bin

6. Explique qué hace la siguiente orden, y para qué sirve cada uno de los argumentos utilizados en la misma:

```
avr-objcopy -O ihex -R .eeprom main.bin main.hex
```

7. Suponiendo que trabajamos en sistema Linux y que tenemos una placa conectada al PC mediante el puerto USB, indique cómo podemos conocer cuál dispositivo (qué fichero de la carpeta */dev* del sistema) es el dispositivo asociado a Arduino Uno.

8. Explique qué hace la siguiente orden, y para qué sirve cada uno de los argumentos utilizados en la misma:

```
avrdude -F -V -c arduino -p ATMEGA328P -P /dev/ttyACM0 -b 115200 -U flash:w:main.hex
```

3.5. Cuestiones sobre la construcción básica de programas

1. ¿Cuál es la biblioteca básica para realizar operaciones de E/S en procesadores AVR?

2. ¿Qué es la macro **F_CPU**? ¿Para qué sirve? A la hora de construir un programa en C para Arduino Uno, ¿dónde debe definirse esta macro? Exponga un ejemplo de su definición.

3. ¿Cuál es el valor más apropiado para la macro **F_CPU** (el que proporciona resultados más realistas) en la placa Arduino Uno?

4. En el siguiente programa, indique qué es la variable global **DDRB** y para qué se utiliza dentro de un programa:

```
// Utilizado para el cálculo de ms en _delay_ms
#define F_CPU 1000000UL

#include <avr/io.h>
#include <util/delay.h>


#define BLINK_DELAY_MS 1000

int main (void)
{
 /* Pin 0 del puerto B del micro puesto como salida */
 DDRB |= _BV(DDB0);

 while(1) {
  /* Mandamos señal de voltaje alto al pin 0 del puerto B */
  PORTB |= _BV(PORTB0);
  _delay_ms(BLINK_DELAY_MS);

  /* Mandamos señal de voltaje bajo al pin 0 del puerto B */
  PORTB &= ~_BV(PORTB0); // ~ es el NOT lógico a nivel de bits
  _delay_ms(BLINK_DELAY_MS);
 }
}
```

5. En el programa del ejercicio anterior, indique qué es la macro **DDB0** y para qué se utiliza dentro de un programa.

6. ¿Cuáles son las macros para los pines PB1 al PB5 del microprocesador AVR AtMega328, en un programa en C como el mostrado en el ejercicio 4 de este apartado? ¿Con qué pines de la placa Arduino se corresponden?

7. Indique qué significado tendría ejecutar la sentencia del programa principal en el código siguiente:

```
#define F_CPU 20000000UL

#include <avr/io.h>
#include <util/delay.h>

int main (void)
{
 DDRB = 0x0F;

}
```

8. En el programa del ejercicio 4 de este apartado, indique qué es la variable **PORTB** y para qué se utiliza dentro de un programa.

9. Indique cuál es la diferencia principal entre las variables globales **DDRB** y **PORTB**.

10. En el programa del ejercicio 4 de este apartado, indique qué es la macro **PORTB0** y para qué se utiliza dentro de un programa.

11. Razone si la primera línea del programa del ejercicio 4 de este apartado equivale a escribir:

DDRB= 0x01;

En caso afirmativo, indicar porqué. En caso negativo, indicar cuáles serían las diferencias existentes entre ambas.

12. Razone si la primera línea del programa del ejercicio 4 de este apartado equivale a escribir:

DDRB= DDRB | 0x01;

En caso afirmativo, indicar porqué. En caso negativo, indicar cuáles serían las diferencias existentes entre ambas.

13. Razone si la línea del programa **"PORTB |= _BV(PORTB0)"** del ejercicio 4 de este apartado equivale a escribir:

PORTB |= 0x01;

En caso afirmativo, indicar porqué. En caso negativo, indicar cuáles serían las diferencias existentes entre ambas.

14. Si sustituyésemos la línea **"PORTB |= _BV(PORTB0)"** del programa del ejercicio 4 de este apartado por **"PORTB = 0x01;"** , estaríamos cometiendo un error muy grave que podría dañar la placa Arduino Uno, ¿porqué?.

15. ¿Qué hace la línea del programa **"PORTB &= ~_BV(PORTB0);"** ?

16. Para poner el voltaje de la patilla PB0 del microprocesador AVR AtMega328p a 0V, realizamos la orden **"PORTB &= ~_BV(PORTB0);".** ¿Porqué no hacemos directamente **"PORTB= 0x00;"**? ¿Qué problemas pueden surgir si hacemos esto y cómo podríamos dañar la placa?

17. ¿Se dañaría la placa o existiría algún error si se ejecutase la siguiente secuencia de instrucciones? Razone su respuesta.

```
#define F_CPU 20000000UL

#include <avr/io.h>
#include <util/delay.h>

int main (void)
{
 DDRB = 0x0F;
 PORTB= 0xF0;
}
```

18. Supongamos el siguiente programa:

```
#define F_CPU 20000000UL

#include <avr/io.h>
#include <util/delay.h>


#define BLINK_DELAY_MS 1000

int main (void)
{
 DDRB = 0x0F;

  while(1) {
  PORTB |= 0x07;
  _delay_ms(BLINK_DELAY_MS);
 }
}
```

¿Existe algún error grave en el código?

Si conectásemos el cátodo de un LED a un pin GND de la placa Arduino Uno y el ánodo al pin 8 del puerto digital de la placa, ¿qué ocurriría? ¿y si lo conectásemos al pin 9 de la placa? ¿y si lo conectásemos al pin 10?. Razone su respuesta.

19. En el programa del ejercicio 18 de este apartado, ¿qué sentencia se debería escribir para enviar un 0 (voltaje bajo a 0V) por los pines de la placa Arduino 8 y 9, después de la sentencia **"_delay_ms(BLINK_DELAY_MS);"**? Razone su respuesta.

4 Sesión 2: Comunicación serie

4.1. Contenidos de la sesión

- El puerto UART.
- Comunicación serie bidireccional entre un PC y Arduino Uno.

4.2. Cuestiones sobre el puerto UART

1. ¿Qué es un puerto UART? Describa con detalle lo que conozca de este puerto.

2. ¿Qué puertos son gestionados por UART en la placa Arduino Uno? Señale en la siguiente figura dónde se encuentran esos puertos y los LEDs incrustados en la placa que indican cuándo se producen comunicaciones de entrada o de salida.

3. ¿Cómo se gestiona el puerto USB desde Arduino Uno? ¿Qué definiciones de datos y/o sentencias es necesario incluir en el código de un programa Arduino para utilizar el puerto USB? Indicarlo suponiendo que se utiliza la biblioteca **<uart.h>** utilizada en el Seminario 1. Escriba un programa básico que realice estas acciones.

4. ¿Qué son los baudios? ¿Para qué se utilizan en comunicaciones serie? Indique también cuál es la velocidad básica tradicional en baudios para un canal de comunicaciones serie.

5. El término **bps** se refiere a ***bits por segundo*** en telecomunicaciones. Explique qué relación existe entre los baudios y los bps.

4.3. Cuestiones sobre comunicaciones serie

1. Escriba las funciones de la biblioteca **<uart.h>** que conozca para recepción de datos indicando, para cada una de ellas, cuántos argumentos tiene y qué es cada uno de ellos. Exponga un ejemplo de uso de cada una de ellas.

2. Escriba las funciones de la biblioteca **<uart.h>** que conozca para envío de datos indicando, para cada una de ellas, cuántos argumentos tiene y qué es cada uno de ellos. Exponga un ejemplo de uso de cada una de ellas.

3. Explique qué acciones son necesarias llevar a cabo en un programa en C/C++ para gestión de comunicaciones por USB, compilando el programa para Linux.

4. Explique qué acciones son necesarias llevar a cabo en un programa en C/C++ para gestión de comunicaciones por USB, compilando el programa para Arduino Uno.

5. Explique cómo se utiliza y qué acciones realiza la función ***int InicializarUSB(const char *portname)*** creada en la biblioteca **<ticcommpc.h>** del Seminario 1, y exponga un programa de ejemplo que abra y cierre el puerto USB para comunicación con Arduino Uno.

6. Exponga los códigos de error incluidos en la biblioteca **<uart.h>** para gestión de las comunicaciones USB por UART en Arduino Uno, explicando cada uno de ellos.

7. Indique para qué sirve, qué argumentos tienen y qué salidas proporcionan las funciones **write** y **read** de la biblioteca **<unistd.h>** para comunicaciones serie en un PC.

8. Indique cuál es el efecto que produce la función **tcflush** de la biblioteca **<unistd.h>**. ¿Qué acción realiza la orden ***"tcflush(fd, TCIFLUSH);"*** del programa principal del ejemplo del Seminario 1? ¿Qué ocurriría si eliminásemos esta sentencia del programa principal?

9. Proyecto **LEDControl.** Conecte un LED al pin 12 y a tierra de la placa Arduino Uno tal y como muestra la siguiente figura **(ponga cuidado al conectar los polos positivo y negativo del LED donde correspondan)**:

Con esta configuración hardware, modifique el programa de ejemplo de comunicación serie del Seminario 1 para que el cliente (antiguo ***pcecho***) envíe una orden al servidor (antiguo ***arduecho***) de encendido o de apagado del LED. Llame a estos programas **pcLEDcontrol** y **arduLEDcontrol**, respectivamente:

9.1. El programa cliente **pcLEDcontrol** (antiguo ***pcecho***) debe tomar un argumento por línea de comandos, consistente en una cadena que sólo podrá tomar los valores "On" y "Off" y enviará, por puerto serie USB, el mensaje al servidor.

9.2. El programa servidor **arduLEDcontrol** (antiguo ***arduecho***) recibirá mensajes desde el puerto USB. Si se recibe un mensaje:

9.2.1. Si el mensaje es "On", enviará la señal al microprocesador para que active la salida de voltaje alto al pin 12 de la placa Arduino Uno. Tras esta operación, el servidor devolverá por el puerto serie el mensaje "LED encendido.".

9.2.2. Si el mensaje es “Off”, enviará la señal al microprocesador para que active la salida de voltaje bajo (0V) al pin 12 de la placa Arduino Uno. Tras esta operación, el servidor devolverá por el puerto serie el mensaje “LED apagado.”.

9.2.3. Si es otro mensaje, devolverá el mensaje “No entiendo la orden.” por el puerto serie USB.

En cualquier caso, si no se recibe mensaje no se ejecutará ninguna acción.

9.3. El programa cliente **pcLEDcontrol** mostrará el mensaje recibido por el servidor, y terminará el programa.

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, “Practica1/ejercicio4.3.8”). 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 del cliente y del servidor y qué contiene cada uno de los ficheros:

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

NOTA: Cada fichero de código fuente deberá incluir el nombre y los apellidos de sus autores.

PISTAS PARA REALIZAR EL EJERCICIO:

- Se debe crear un buffer para guardar datos de entrada en ambos extremos (**pcLEDcontrol** y **arduLEDcontrol**). Por ejemplo, buffer de 128 bytes útiles como máximo (+1 por el carácter de finalización de cadena).
- Como se desconoce el tamaño del mensaje a enviar, se debe establecer un protocolo para conocer cuándo ha terminado el mensaje (por ejemplo, recibir un '\0' o un '\n' podría servir para indicar que el mensaje ha finalizado).
- En **arduLEDcontrol**:
    - Tener un buffer de caracteres de un tamaño máximo fijo (recomendable el mism tamaño que en **arduLEDcontrol**).
    - Tener un contador para indicar qué posición del buffer de llegada se debe rellenar (inicialmente con valor a 0).
    - Si hay datos en el puerto:
        - Leer el carácter,
        - Introducirlo en el buffer en la posición actual del contador.
        - Si el carácter leído es el carácter de finalización (el que se haya puesto, sea un '\0' o un '\n'):
            - Poner un '\0' en la posición del buffer del contador,
            - Procesar el mensaje, y
            - Finalizar poniendo el contador de nuevo a 0, para volver a recibir un nuevo mensaje.
- En **pcLEDcontrol**:
    - No hay problema para el envío de varios datos con write. Simplemente hay que asegurarse de que, al enviar datos, también se envía al final el carácter de finalización del mensaje.
    - Para la recepción:
        - Tener un buffer de caracteres de un tamaño máximo fijo.

- Tener un contador para indicar qué posición del buffer de llegada se debe rellenar (inicialmente con valor a 0).
- Llamar a read mientras no se haya recibido el carácter de fin de mensaje (salir también si hay error –read devuelve -1 –).
    - Rellenar el buffer desde la posición del contador, tantos bytes como se hayan leído con read.
    - Actualizar el contador sumándole el número de bytes leídos con read.
- Poner un '\0' Al final del mensaje recibido, sustituyendo al carácter de finalización.

10. **(NOTA: Este ejercicio tiene especial relevancia para futuros desarrollos. Ponga especial esfuerzo en realizar las operaciones de forma eficiente, con control de errores y realice múltiples pruebas para asegurar el correcto funcionamiento de las funciones requeridas).**
    Los protocolos establecidos en el apartado anterior para envío y recepción de datos serán necesarios en el futuro para comunicaciones serie entre el sistema de comunicaciones láser construido en las prácticas y un PC. Por ello, se pide:

    10.1. En la biblioteca creada en el seminario **<ticcommpc.h>**, crear las dos funciones siguientes:
    - **bool sendUSB(int &fd, const char *data);** , para enviar una cadena por el puerto USB abierto en ***fd*** (nótese que esta cadena no puede contener en su interior el carácter de parada escogido en el ejercicio anterior). La función devolverá true si hubo éxito al enviar los datos, y false en caso contrario.
    - **bool receiveUSB (int &fd, char *data);** , para recibir un mensaje de texto por el puerto USB abierto en ***fd***, y devolverlo en data (nótese que ***data*** debe tener suficiente memoria reservada como para albergar cualquier mensaje que se pueda recibir desde USB). Esta cadena terminará en '\0', independientemente del carácter de parada escogido para las comunicaciones. La función devolverá true si el mensaje se recibió correctamente (en tal caso, se devolverá por referencia en ***data***), y false en caso contrario.

    10.2. Crear una nueva biblioteca **<ticcommardu.h>**, para ser utilizada en programas a cargar en la placa Arduino Uno, con las dos funciones siguientes:
    - **bool arduSendUSB(const char *data);** , para enviar una cadena por el puerto USB (nótese que esta cadena no puede contener en su interior el carácter de parada escogido en el ejercicio anterior). La función devolverá true si hubo éxito al enviar los datos, y false en caso contrario.

- **bool arduReceiveUSB (char *data);** , para recibir un mensaje de texto por el puerto USB en data (nótese que ***data*** debe tener suficiente memoria reservada como para albergar cualquier mensaje). Esta cadena terminará en '\0', independientemente del carácter de parada escogido para las comunicaciones. La función devolverá true si el mensaje se recibió correctamente (en tal caso, se devolverá por referencia en ***data***), y false en caso contrario.

10.3. Modifique el programa del ejercicio anterior para que haga uso de estas nuevas bibliotecas y sus funcionalidades.

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, "Practica1/ejercicio4.3.9"). 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 del cliente y del servidor y qué contiene cada uno de los ficheros:

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

NOTA: Cada fichero de código fuente deberá incluir el nombre y los apellidos de sus autores.

5 Sesión 3: Sensores y emisores de láser

5.1. Contenidos de la sesión

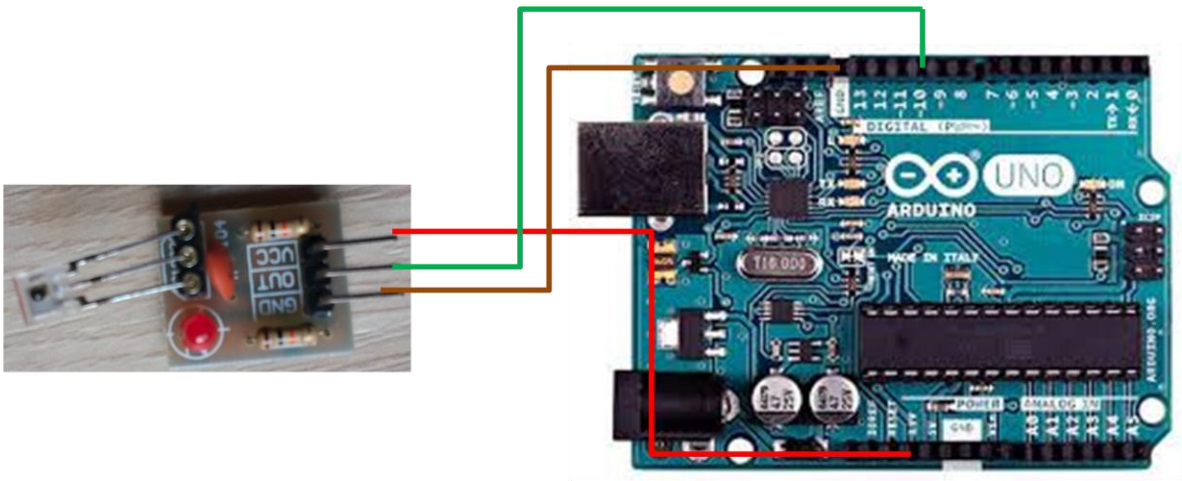
- Fotorreceptores. El fotorreceptor 2PCS láser no modulador sensor Tubo.
- Emisores láser. El módulo KY-008 de 650nm.

5.2. Cuestiones sobre fotorreceptores

1. ¿Qué es un fotorreceptor? ¿para qué sirve?

2. Explique los fundamentos físicos de un fotorreceptor (su funcionamiento a nivel de electrónica).

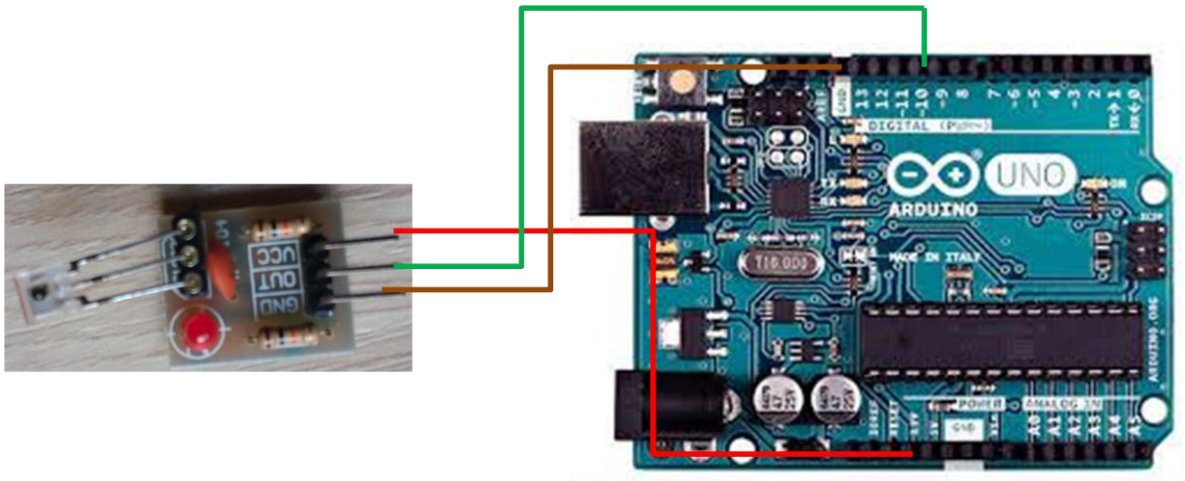
3. Suponiendo que colocamos el fotorreceptor **Módulo receptor 2PCS láser no modulador sensor Tubo** como indica la siguiente figura:



Explique, apoyándose con el uso de código fuente, cómo habría que inicializar los puertos para que se pudiese recibir información correctamente, y cómo se leería la información desde el Pin digital 10 seleccionado en la imagen, de la placa Arduino.

4. Explique la diferencia entre **PORTB** y **PINB**. ¿Qué efecto produce **PINB & 0x04**? ¿Y **PINB & 0x03**?

5. Proyecto **SerialFotorresistor.** Conecte el **Módulo receptor 2PCS láser no modulador sensor Tubo** como indica la siguiente figura:



Elabore dos programas dentro de este proyecto, haciendo uso de las bibliotecas **<ticcommpc.h>** y **<ticcommardu .h>** según sea el programa (PC o Arduino), para transmisión de datos por el puerto USB:

a. **fotoPC**: Un programa que se ejecutará en un PC para Linux, conectándose con Arduino Uno a través de USB. El programa ejecutará un bucle infinito y, en cada iteración del bucle, recibirá cadenas con valores "0" o "1" desde la placa Arduino Uno. Si se recibe una cadena "0", se mostrará el mensaje "Sin luz, enciende las velas.". Por el contrario, si se recibe la cadena "1" se mostrará "Hace sol, vamos a salir.". Para la elaboración de esta práctica, haga uso de la función **bool receiveUSB (int &fd, char *data);** desarrollada en la sesión anterior e incluida en la biblioteca **<ticcommpc.h>**.

b. **fotoArdu**: Un programa que se ejecutará en una placa Arduino con el esquema mostrado en la figura anterior. Ejecutará un bucle infinito y, en cada iteración del bucle, se comprobará cuál es el valor de luz medido por el fotorresistor. Si se detecta luz, se enviará la cadena "1" por puerto USB. En caso contrario, se enviará la cadena "0". Al final del bucle impondremos que la hebra duerma por 125us, provocando de esta forma que se envíen 8.000 datos/segundo al PC. Haga uso de la función **bool arduSendUSB(const char *data);** implementada e incluida en la biblioteca **<ticcommardu.h>** en la sesión anterior.

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, "Practica1/ejercicio5.2.6"). 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 del cliente y del servidor y qué contiene cada uno de los ficheros:

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

5.3. Cuestiones sobre emisores láser

1. Busque por Internet diferentes emisores láser que puedan ser utilizados en Arduino, y compare los modelos en cuanto a características, funcionalidades y precio.

6 Sesiones 4 y 5: Prototipo emisor-receptor láser

6.1. Contenidos de la sesión

- Codificación y decodificación de mensajes.
- Coordinación de las componentes del sistema de emisión/recepción.
- Implementación de los programas **emisorPC**, **receptorPC**, **emisorArdu** y **receptorArdu**.

6.1. Cuestiones teóricas sobre el prototipo

1. Describa 2 alternativas diferentes para elaborar la transmisión de información por láser, a nivel físico, en el prototipo, indicando sus ventajas e inconvenientes.

2. Indique, según la nomenclatura que hemos adoptado en el Seminario 1, qué es un *codeBit* y qué es un *laserBit*, indicando cómo se relacionan entre sí. Ponga un ejemplo de esta relación.

3. Suponiendo que escogemos enviar la cadena de laser-bits “010110”, dibuje el diagrama del voltaje que se enviará a lo largo del tiempo al pin de emisión de láser de la placa emisora Arduino.

4. Suponiendo que escogemos enviar la cadena de code-bits “01001”, dibuje el diagrama del voltaje que se enviará a lo largo del tiempo al pin de emisión de láser de la placa emisora Arduino.

5. Para el ejemplo de envío de datos del ejercicio anterior, explique cómo la placa receptora deberá saber cuándo ha terminado la recepción de un símbolo y cuándo ha terminado la recepción de todo el mensaje.

6. Busque por Internet cuál es el código Morse para representar caracteres de la A a la Z (evitar la Ñ). Añada 3 símbolos más para representar espacios en blanco, puntos, comas, y punto y comas, y asígnele la secuencia de puntos y rayas que desee. Escriba aquí su código:

Símbolo	Código Morse	Símbolo	Código Morse
A		P	
B		Q	
C		R	
D		S	
E		T	
F		U	
G		V	
H		W	
I		X	
J		Y	
K		Z	
L		.	
M		,	
N		;	
O		(ESPACIO)	

7. Suponiendo que queremos transmitir el mensaje "Hola" en código Morse a través de la plataforma, ¿cómo se codificaría este mensaje en los símbolos de transmisión por láser (*laser-bits*)? En el receptor, ¿cómo se detectaría que se ha recibido el código "H"? ¿Cómo detectaríamos el fin del mensaje "Hola" completo? Dibuje el diagrama de voltajes alto/bajo que se enviaría a lo largo del tiempo por el emisor para enviar el mensaje completo.

8. Enuncie el **Teorema de Muestreo de Nyquist-Shannon**, y explique cómo nos ayudará este teorema en la elaboración de la práctica. ¿Cómo relacionaría este teorema y las constantes **UMBRAL_U** y **SAMPLE_PERIOD** incluidas en la biblioteca **<ticcommardu.h>** en el Seminario 1?

6.2. Biblioteca para codificación y decodificación de datos

9. **Biblioteca <arducodif.h>: Codificación y decodificación**.

   Cree una nueva biblioteca compuesta por un fichero **include/arducodif.h** y un fichero **src/arducodif.cpp.** En el fichero .h, cree los siguientes prototipos de funciones, cuya funcionalidad se encuentra descrita en el Seminario 1:

void codificaSimboloMorse(const char corig, char &ccodif, unsigned char &nUtil);

void codificador(const char *orig, const int nOrig, char *codif, unsigned char *util);

void decodificaSimboloMorse(const char ccodif, const unsigned char nUtils, char &decodif);

void decodificador(const char *codif, const unsigned char *utiles, const int nCodif, char *decodif);

   a. En el fichero **arducodif.cpp**, implemente la función **codificaSimboloMorse** para que, dado un símbolo de la tabla del ejercicio anterior como entrada en el parámetro **corig**, devuelva como salida un byte con el código Morse en binario, asumiendo que los puntos se codificarán con 0 y las rayas con 1. En el parámetro de salida nUtil, se devolverá cuántos bits ocupa el símbolo codificado. Por ejemplo, para codificar el símbolo de entrada **corig='J'**, se devolverá la salida **ccodif=XXXX0111** y **nUtil=4** (el valor de los bits marcados

con X es irrelevante). Otro ejemplo, para codificar el símbolo de entrada **corig='K'**, se devolverá la salida **ccodif=XXXXX101** y **nUtil=3.**

b. En el fichero **arducodif.cpp**, implemente la función **decodificaSimboloMorse** para que, dado un byte como entrada en el parámetro **ccodif**, que codifica un símbolo Morse de la tabla del ejercicio anterior como entrada, y otro parámetro de entrada **nUtils** con el número de bits útiles de **ccodif**, devuelva como salida en **decodif** el símbolo decodificado, asumiendo que los puntos se codifican con 0 y las rayas con 1. Por ejemplo, para decodificar el byte de entrada **ccodif =00010010** con **nUtils=3**, se devolverá la salida **decodif ='R'**. Otro ejemplo, para decodificar el símbolo de entrada **ccodif =01110011** con **nUtils=2**, se devolverá la salida **decodif ='M'.**

c. En el fichero **arducodif.cpp**, implemente la función **codificador**, que tenga como entrada una cadena de caracteres **orig** con longitud **nOrig**, y devuelve en cada componente de **codif** y **util** cada símbolo codificado y el número de bits útiles de cada símbolo. Por ejemplo, para la entrada **orig="RM"** con **nOrig=2**, se devolverá **codif={XXXXX010 , XXXXXX11 }** y **utils= {3, 2}.** Otro ejemplo: para la entrada **orig="JKM"** con **nOrig=3**, se devolverá **codif={ XXXX0111, XXXXX101, XXXXXX11 }** y **utils= {4, 3, 2}**.

d. En el fichero **arducodif.cpp**, implemente la función **decodificador**, que tenga como entrada una cadena de caracteres **orig** con longitud **nOrig**, y devuelve en cada componente de **codif** y **util** cada símbolo codificado y el número de bits útiles de cada símbolo. Por ejemplo, para la entrada **codif={XXXXX010 , XXXXXX11 } , utiles= {3, 2}. Y nCodif=2,** se devolverá la cadena **decodif ="RM"**. Otro ejemplo: para la entrada devolverá **codif={ XXXX0111, XXXXX101, XXXXXX11 }** y **utiles= {4, 3, 2}, nCodif=3,** la salida devolverá **decodif ="JKM"**.

e. Escriba un ejemplo de traza (cómo se ejecutaría paso a paso) del codificador, para codificar la cadena "Bye".

f. Escriba un ejemplo de traza (cómo se ejecutaría paso a paso) del decodificador, para codificar la secuencia de símbolos {00000000, 00000011, 00000000}, con número de bits útiles por cada símbolo igual a {3, 2, 3}.

6.3. Aplicación para envío y recepción de datos mediante láser

1. **Aplicación emisorPC.** Cree un programa para Linux en C/C++ que realice lo siguiente:
   - Pida desde la entrada estándar hasta un máximo de 100 caracteres.
   - Si no se ha introducido nada (cadena vacía), salir del programa.
   - Si se ha escrito algo (hasta 100 caracteres), pasar los caracteres leídos a mayúsculas y enviar la cadena por USB a una placa Arduino Uno. Si alguno de los caracteres no se encuentra en la tabla del ejercicio 1 de este apartado, no se enviarán por USB y se mostrará un error.
   - Si se envió el mensaje por USB, entonces el programa esperará hasta recibir el mensaje "OK" por USB desde la placa Arduino. Seguidamente, se volverá al paso 1.

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, "Practica1/ProyectoFinal"). 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 del programa y qué contiene cada uno de los ficheros:

- Indique las instrucciones para compilar y ejecutar el programa:

NOTA: Cada fichero de código fuente deberá incluir el nombre y los apellidos de sus autores.

2. **Aplicación receptorPC**. Cree un programa para Linux en C/C++ que realice lo siguiente en un bucle infinito:
   - Lea desde USB cadenas de caracteres que se envían desde una placa Arduino.
   - Muestre la cadena por consola.
   - Busque por Internet cómo capturar la interrupción de teclado CTRL-C, y escriba la función para cerrar el descriptor de fichero del puerto y salir del programa si durante la ejecución del programa se pulsa esta combinación de teclas.

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, "Practica1/ProyectoFinal"). 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 del programa y qué contiene cada uno de los ficheros:

- Indique las instrucciones para compilar y ejecutar el programa:

NOTA: Cada fichero de código fuente deberá incluir el nombre y los apellidos de sus autores.

3. **Aplicación emisorArdu**. Cree un programa en C/C++ para la placa Arduino Uno que realice lo siguiente en un bucle infinito:
   - Lea desde USB cadenas de caracteres que se envían desde un PC. Se asume que el tamaño máximo de cada cadena recibida será de 100 bytes útiles.
   - Codifique la cadena en código Morse, utilizando las funciones de la biblioteca previamente desarrollada **<arducodif.h>**.
   - Envíe cada símbolo codificado por láser, utilizando la biblioteca previamente desarrollada **<ticcommardu.h>**.

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, "Practica1/ProyectoFinal"). 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 del programa y qué contiene cada uno de los ficheros:

- Indique las instrucciones para compilar y enviar a Arduino Uno el programa:

NOTA: Cada fichero de código fuente deberá incluir el nombre y los apellidos de sus autores.

4. **Aplicación receptorArdu**. Cree un programa en C/C++ para la placa Arduino Uno que realice lo siguiente:
   - Debe tener una variable de estado **estado**, capaz de diferenciar entre el estado "Recibiendo" y "En espera" (ver Seminario 1). Inicialmente, se debe encontrar "En espera". También debe tener declarados dos buffers, uno para recibir datos codificados desde el receptor láser (llamémosle **Laser**), y otro buffer **decodificado** para almacenar la información recibida por el receptor láser ya decodificada. Ambos buffers deben estar inicializados a vacío. A continuación, en un bucle infinito se debe realizar lo siguiente:
     - Leer dato del fotorreceptor láser, mientras no se encuentre en estado "Recibiendo".
     - Leer un mensaje desde el receptor láser (máximo de 100 bytes útiles), hasta que se vuelva a pasar a estado "En espera" (ver Seminario 1).
     - Decodificar el mensaje.
     - Enviarlo por USB a un PC.

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, "Practica1/ProyectoFinal"). 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 del programa y qué contiene cada uno de los ficheros:

- Indique las instrucciones para compilar y enviar a Arduino Uno el programa:

NOTA: Cada fichero de código fuente deberá incluir el nombre y los apellidos de sus autores.

6.4. Cuestiones teóricas sobre el prototipo implementado

1. ¿Cuál es el periodo de envío de un *laserBit* por el canal de comunicaciones? ¿Y el periodo de la señal de fin de símbolo? ¿Y el periodo de la señal de fin de mensaje? ¿Cómo los ha calculado?

2. ¿Cuál sería la frecuencia (en Hz) a la que trabaja el sistema de emisión por láser?. Explique cómo la ha calculado.

3. ¿A qué velocidad, como mucho, debería muestrear el fotorreceptor para poder reproducir fielmente la señal, considerando el periodo de envío de un *laserBit* por el emisor? ¿porqué?

4. ¿Cuál es la velocidad de señalización del sistema de emisión/recepción por láser? ¿y la velocidad máxima posible?

5. ¿Cuál es la capacidad del canal de comunicaciones por láser?

6. Suponiendo que la emisión de cualquier símbolo “A”, “B”, “C”, etc. es equiprobable, explique cómo se calcularía la tasa de información promedia del sistema suponiendo tramas de 100 símbolos.

7. Suponiendo que la emisión de cualquier símbolo “A”, “B”, “C”, etc. es equiprobable, razone, calcule y explique: ¿Cuál es la información aportada por la emisión de un **LASER_DOT**? ¿Y de un **LASER_DASH**? ¿Cuál es la entropía de la fuente? Según el valor de esta entropía, ¿cuántos bits de datos serían necesarios, en este caso, para codificar todos los símbolos del alfabeto utilizado en las prácticas?

8. Suponiendo que el fotorreceptor haya recibido iluminación por el láser en un momento dado, ¿cuál es la probabilidad de que la siguiente señal que se reciba sea también iluminación? ¿Desde el punto de vista del receptor, ¿qué información aporta recibir iluminación? ¿y no recibir iluminación? ¿Cuál es la entropía del emisor?

9. Explique cómo calcularía la información mutua entre lo que se recibe por el receptor y lo que emite el emisor. ¿Cuánto vale la información mutua en este sistema? Explique porqué proporciona dicho valor, justificándolo en términos de canales con y sin ruido.

7 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 punto. Todos los ejercicios tienen la misma puntuación.
- La sesión 2 se evaluará hasta 2 puntos. Todos los ejercicios tienen la misma puntuación.
- La sesión 3 se evaluará hasta 3 puntos. Todos los ejercicios tienen la misma puntuación.
- Las sesiones 4 y 5 se evaluará hasta 4 puntos. Entre ellos:
    - Los ejercicios teóricos valen hasta 0,5 puntos. Todos los ejercicios tienen la misma puntuación.
    - Los ejercicios para desarrollo de la biblioteca **<arducodif.h>** valen hasta 1 punto. Todos los ejercicios tienen la misma puntuación.
    - Los ejercicios **emisorPC** y **receptorPC** valor 0,5 puntos, evaluados conjuntamente.
    - Los ejercicios **emisorArdu** y **receptorArdu** valor 2 puntos, evaluados conjuntamente.
    -

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.