

Práctica 1

Sesión 1

**Supuestamente está todo,
pero si queréis echarle un
vistazo, por si hay algo mal**



ugr

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



Teoría de la Información y la Codificación

Cuaderno de Prácticas 1

Nombre y apellidos

e-Mail

Autor 1	Samuel Cadenete Rodríguez	samuelcr1995@correo.ugr.es.
Autor 2	Pablo Parra Baró fons	pablo.parra@correo.ugr.es

Curso académico:

17/18

Autor 3	Gema Corea Fernández	gcorea@correo.ugr.es
---------	----------------------	----------------------

TO DA LA PRÁCTICA 1, SE ENCUENTRA
EN LA MISMA CARPETA



ugr

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



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.



ugr

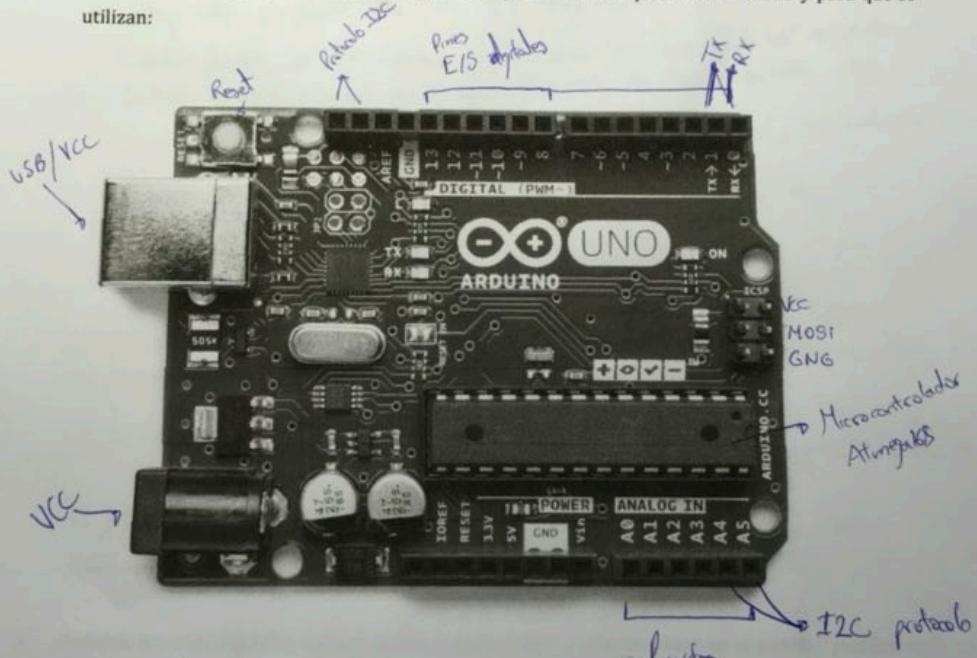
Universidad de Granada
Departamento de Ciencias de la Computación
e Inteligencia Artificial



3.2. Cuestiones sobre Arduino Uno

Jueves - Quijote
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:



- Reset: Reinicia la ejecución del programa.
- Pines digitales E/S:
 - 1 Entrada: Adquirir datos desde sensores Analógicos o dispositivos externos.
 - 2 Salida: Envío de datos a dispositivos externos o actuadores.
 - 3 Tx (transmisión) y Rx (lectura).

- Entrada USB/VCC: Comunicación mediante puerto serie USB para enviar o recibir datos.
- Puertos Analógicos: Envío y Recepción de datos analógicos en otros dispositivos.



ugr

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



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

- Puerto Digital: Es aquel puerto que se usa para el envío ^{y recepción} de datos digitales, es decir, alto 5V, bajo 0V.
- Puerto Analógico: Es aquel puerto que se usa para la recepción de datos analógicos, es decir a diferencia de los digitales cuyo rango ~~es~~ de valores es binario (Alto o bajo), en analógico tenemos un rango de valores entre 0V y 5V.

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?

Arduino Uno posee 6 pines Analógicos y 1 puerto analógico (C)

Los pines analógicos son el A0, A1, A2, A3, A4, A5 y son únicamente de entrada.

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?

Posee 2 puertos digitales, con 6 pines en el puerto B y 8 en el puerto D. Los pines son PB0 - PB5 y PD0 - PD7 y son 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? 5 tomas de tierra.



ugr

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



DECSAI

Las formas de tierra suelen para llamar a "tierra" cualquier conexión indebida de la corriente eléctrica.

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.

- ↳ Mediante el puerto de conexión USB
- ↳ Mediante el conector de alimentación
- ↳ Mediante el pin V_{in} de la placa

El voltaje permitido está entre 5V y 12V

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

- No conectar 2 pines entre si
- No aplicar un voltaje superior a 5.5V a cualquier pin de entrada/salida.
- Alimentarlo por Vin pero sin insertar la corriente (conectar el positivo con negativo y viceversa).
- ~~No conectar un voltaje superior al requerido en los pines de voltaje~~
- No conectar voltaje directamente a tierra.
- No aplicar dos entradas de voltaje diferente para alimentar la placa.
(Entrada externa Vin y entrada externa por el conector de alimentación).
- ~~No aplicar un voltaje superior a 13V en el pin RESET.~~
- No incluir en la placa una corriente superior a la soportada
(la suma total de toda la corriente incluida en los pines de E/S no puede ser superior a los 20mA).



UGR

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



3.3. Cuestiones sobre el microprocesador AVR AtMega328p

1. Busque por Internet 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).

- Voltaje de funcionamiento: 1,8V - 5,5V

• Frecuencia de trabajo: 0-20 MHz

• Número de pines: 32

• Posee tres puertos, dos digitales y uno analógico

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.

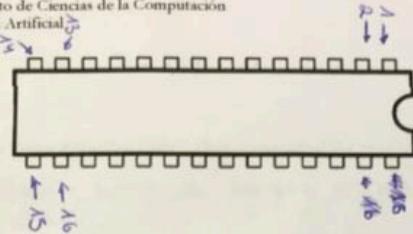


UGR

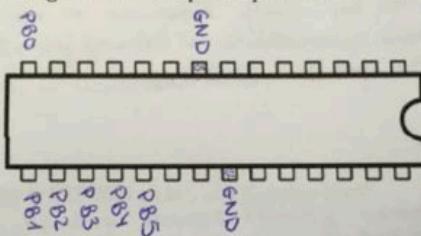
Universidad de Granada
Departamento de Ciencias de la Computación
e Inteligencia Artificial



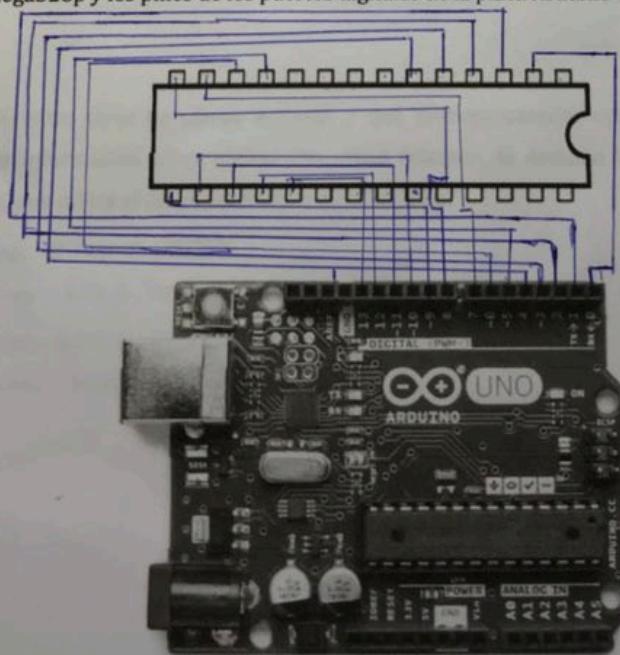
DECSAI



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.





UGR

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



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

- Memoria Flash: 32KB, de los que 0.5KB son para el arranque.
- SRAM: 2KB.
- EEPROM: 1KB.

6. ¿Cómo se almacenan los programas en Arduino Uno (disco duro, tarjeta SD, memoria flash...?)

¿Dónde se encuentra esta memoria dentro de la placa?

- Se almacenan en la memoria flash, por esta razón las programaciones no pueden ser muy grandes, ni utilizaremos memoria dinámica.
- Se encuentra en el microcontrolador ATmega328p

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

- SRAM: Static Random Access Memory, memoria estática de acceso aleatorio, mantiene los datos mientras sigue alimentada, donde se crean y manipulan variables.
- EEPROM: Síntesis de los programadores pueden usar para guardar información a largo plazo.

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?

- Arduino Ethernet
- Arduino MKR FOX 1200
- Arduino MKR ZERO
- Arduino Mega 2560



UGR

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



3.5. Cuestiones sobre la construcción básica de programas

1. ¿Cuál es la biblioteca básica para realizar operaciones de E/S en procesadores AVR?
<avr/io.h> Que contiene las definiciones de variables para direcciones puestos.
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.
F_CPU es una variable para establecer cada cuánto tiempo se refresca el tick del procesador.
#define F_CPU 1600000UL
3. ¿Cuál es el valor más apropiado para la macro **F_CPU** (el que proporciona resultados más realistas) en la placa Arduino Uno?
El valor más apropiado será 1600000UL ya que su clock speed es de 16MHz.
4. En el siguiente programa, indique qué es la variable global **DDRB** y para qué se utiliza dentro de un programa:



UGR

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



avr-objcopy -O ihex -R .eprom main.bin main.hex

-objcopy: copia el contenido de un fichero a otro,
en este caso

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.

Con el comando `ls /dev/serial/by-id -l`

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`

Sirve para enviar el programa a Arduino (almacenando en la memoria flash).

Argumentos:

- p → para indicarle el microcontrolador
- P → puerto donde se encuentra.
- b → velocidad de transmisión.
- U → indica donde se desea (W) escribir el programa.
- F → Para anular la comprobación de verificar la firma del dispositivo
- V → Desactivar la verificación de datos
- c → Especifica el programa que se va a utilizar



Ugr

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



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

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

Es un compilador. Traduce el código C para generar un programa.

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

Para optimizar el tamaño del programa.

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

Sirve para indicar en qué microcontrolador "correrá" el programa.

La opción -mmcu=atmega328p indica que el microcontrolador será el ATMEGA 328P

4. Explique qué hace la siguiente orden:

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

Para crear el archivo objeto main.o a partir del main.cpp.

5. Explique qué hace la siguiente orden:

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

Crear el archivo binario main.bin a partir del main.o

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



ugr

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



```
// 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);
    }
}
```

DDRB es el registro de direcciónamiento de datos del puerto B del microprocesador (PB0 - PB7)

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

DDB0 es la dirección del PIN 0 del puerto B del microprocesador (el pin PB0)

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?

DDB1, DDB2, DDB3, DDB4, DDB5

↓ ↓ ↓ ↓ ↓
pin 9 , 10 , 11 , 12 y 13

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.

PORTB es un byte que contiene los datos de salida existentes en los pines PB0 - PB7 del microprocesador. Los utilizamos para indicar al microprocesador que envíe alto o bajo a uno o varios pines del puerto B.

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

DDRB es el Registro de direccionamiento de datos del puerto B mientras que PORTB contiene los datos de salida existentes en los pines PB0 - PB7.

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.

Es una macro que hace referencia al pin 0 del puerto B, es decir, en realidad es un Byte tal que así: "0000 0001". En el programa se emplea para mandar como dato de salida al pin 0 del puerto B señales de voltaje.

11. Razona 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.

No es equivalente puesto que de esta forma sobreescrituimos con zeros el valor del resto de los pines, mientras que como viene en el programa únicamente modificamos como salida el pin 0.



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

$$\text{DDRB} = \text{DDRB} | 0x01;$$

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

Si sería equivalente puesto que $0x00 = 0x01$ y ademas empleamos el operador 'or' a nivel de bit.

13. Razona si la línea del programa " $\text{PORTB} |= \text{_BV(PORTB0)}$ " del ejercicio 4 de este apartado equivale a escribir:

$$\text{PORTB} |= 0x01;$$

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

En ambos casos estamos ~~indicando~~ al micro que envíe voltaje alto al pin 0 del puerto B, puesto que $\text{_BV(PORTB0)} = 0x01$

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

Porque machacaríamos el valor en el resto de los puertos por 0 y si existe algún dispositivo escuchando como por ejemplo un micrófono podría producirse un cortocircuito.

15. ¿Qué hace la línea del programa " $\text{PORTB} \&= \sim\text{_BV(PORTB0)}$;"?



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?

Porque de esta forma estariamos sobrescribiendo ~~los~~ a cero las salidas que habíamos antes en el resto de pines lo cual dañaría la placa tal como indicamos en el ejercicio 14.

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 = 0xF0;
    PORTB= 0xF0;
}
```

Sí, puesto que estamos enviando datos a pines que no hemos configurado como salida.

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

¿Existe algún error grave en el código?

Estamos marcando alto como salida y no existe PB6

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.

Hemos configurado todos los pines de DDRB como salida y además hemos ~~señal~~ enviado 'alto' tanto a PB0, PB1 y PB2 por lo que en los tres casos el led permanecería encendido.

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.

$\text{PORTB} \leftarrow 0b00000011;$

Sesión 2

**Revisar todas las preguntas que
he contestado, puede que alguna
esté mal, revisar los ejercicios
9 y 10 del apartado 4.3, por
si falta algo en la contestación**



UGR

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



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.

- Es el puerto mediante el cual se realizan las comunicaciones en serie, como por ejemplo el Bus USB ("Bus universal en serie") del cual emplea Arduino para dicha comunicación.
- UART (Universal Asynchronous Receiver-Transmitter)
- El puerto UART es gestionado a través de interrupciones hardware.
- Para usar este puerto utilizaremos la biblioteca `<uart.h>` que nos facilita la tarea de enviar y recibir datos por UART. Esta biblioteca proporciona los datos leídos desde el puerto UART en bytes junto con otro byte de comprobación de errores.

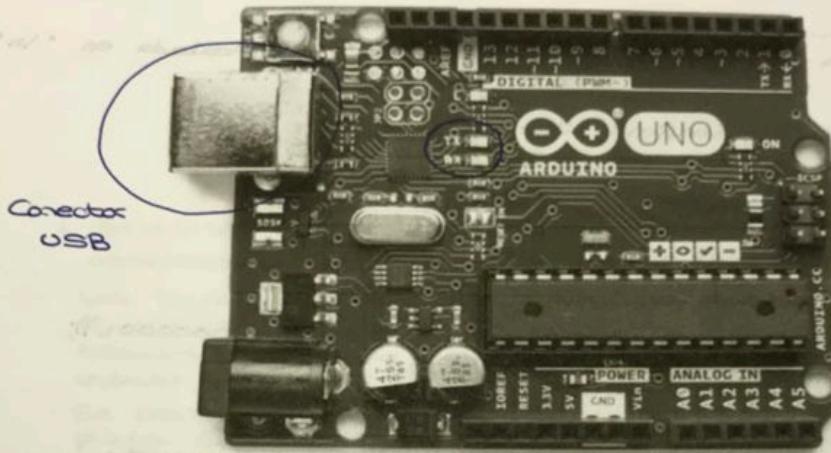
2. ¿Qué puertos son gestionados por UART en la placa Arduino Uno? Señale en la siguiente figura dónde se encuentran esos puertos y los LEDs incrustados en la placa que indican cuándo se producen comunicaciones de entrada o de salida.

En Arduino, la comunicación por USB se realiza a través del puerto serie UART. El conector utilizado en la placa es un USB Tipo B.



ugr

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



Los LED's instalados en la placa:

1. tx : transmisión en la comunicación (la placa envía datos)
2. rx : lectura (la placa recibe datos)

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.

• El puerto UART es gestionado a través de interrupciones hardware.

1. `uart_init(UART_BAUD_SELECT(UART_BAUD_RATE, F_CPU))`;
Inicialización del puerto UART con la velocidad en baudios del puerto y la velocidad del procesador.
2. `sei();`
Activación de las interrupciones hardware para control del puerto serie.

3.

3. `c = uart-getc();`
Devuelve un entero (2 bytes) B.O : byte leido por UART
B.I : control de errores
4. `uart-puts-P ("Error recibiendo trasmisión")`
Envía una cadena de bytes por UART. Acabada en "\n" para enviar los datos del buffer UART.
5. `uart-putc (unsigned char) c;`
Envía un byte por UART.

```
int main (void) {  
    unsigned int c;  
    uart-init (UART-BAUD-SELECT (9600, 1600000L));  
    sei();  
    while (1) {  
        c = uart-getc(); // Recogemos byte desde puerto  
        if (c & UART-NO-DATA) { // Dicmosmos  
            -delay-ms(1);  
        } else if (c & (UART-FRAME-ERROR || UART-OVERRUN-ERROR  
            || UART-BUFFER-OVERFLOW)) {  
            uart-puts-P ("Error\n"); // Error recepción ...  
        } else {  
            uart-putc (unsigned char) c); // Todo OK  
        }  
    }  
    return 0;  
}
```

PPC

Ctra.
Tel.
Tel.:

Fa:
Adr
Inc
Pv
E



ugr

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

DECSAI

• (bueno) adaptaciones de baudios a

• Tanto en la velocidad como en el tipo de transmisión.

• La velocidad es medida en baudios.

• Los baudios se miden en segundos.

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. $\Rightarrow 9600$ baud

• Los baudios son una unidad de medida que representa el número de símbolos por segundo en un medio de transmisión, tanto analógico como digital. Es decir, indican la velocidad del puerto serie.

• Se utilizan para transmitir información desde la placa a otro dispositivo (PC) o recibir información. Como la comunicación es en serie, la información viene por un solo canal, una a continuación de otra.

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

Los baudios sólo son iguales a bits por segundo, cuando la señal representa un único bit. Cuando tenemos una transmisión binaria (valores 0 y 1), un baudio equivale a un bit por segundo

• Velocidad de transmisión en serie (V-bps): se mide en bps

• Velocidad en símbolos (V-s): se mide en baudios

$$V\text{-bps} = n \cdot V\text{-s} \quad (n = \text{nº de bits por cada símbolo})$$

<http://www.ate.us.es/personal/maromero/docs/arc1/Hema3-arc1.pdf>

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.

* unsigned int uart_getc (void);
• Obtener el byte recibido desde el puerto UART.
• Devuelve un entero (2 bytes):
- Byte 0: Dato leído por UART.
- Byte 1: Control de errores.
- UART-NO-DATA se devuelve cuando no hay datos disponibles.

```
bool arduReceiveUSB (char * data) {  
    int posbuf = 0;  
    data [0] = '\n';  
    int c;  
    while (data [posbuf - 1] != '\n' &amp; posbuf == 0) {  
        // Recogemos byte desde puerto UART  
        c = uart_getc ();  
        if (c &amp; UART_FRAME_ERROR || c &amp; UART_OVERRUN -  
            ERROR || c &amp; UART_BUFFER_OVERFLOW) {  
            return false;  
        } else if (! (c &amp; UART_NO_DATA)) {  
            data [posbuf] = (char) c;  
            posbuf++;  
            // Todo ok, insertamos el carácter en el buffer  
        }  
    }  
    return true;
```

PPG

PPG
Ctra. S
Tel. S
Tel. S

Fa
Adm
Inc
R
F



ugr

Universidad de Granada

Departamento de Ciencias de la Computación
e Inteligencia Artificial



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.



* void uart_putc (unsigned char data) : Envía un byte por UART.

· Parámetros:

-data: byte para ser transmitido.

· Return : NONE

* void uart_puts (const char *s) : Envía un string por UART

· Parámetros:

-s: string a ser transmitido

· Return : NONE

* void uart_puts_p (const char *s) : Envía una cadena de bytes por UART. Acabada en "\n" para enviar los datos del buffer UART.

· Parámetro :

-s: cadena de bytes a ser transmitida

· Return : NONE

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.

2.

```
bool arduSendUSB (const char *data) {
    int nData = strlen (data + 1); // Tamaño buffer
    for (int i = 0; i < nData; i++) {
        portPutc (data [i]);
    }
    return true;
}
```



ugr

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



3.

Linux ve las comunicaciones como 'archivos'. Normalmente, Arduino se suele conectar a "/dev/ttyS0", luego podemos abrirlo como un archivo así como leer y escribir en este:

```
std::string str;
fstream f;
f.open ("/dev/ttyS0");
while (f >> str) cout << str;
```

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.

```
int main() {
    int data[] = {10, 5, 13};
    FILE *file;
    file = fopen ("dev/ttyUSB0", "w");
    for (int i=0; i<3; i++) {
        fprintf (file, "%d", data[i]);
        fprintf (file, "%c", ',');
    }
    fclose (file);
}
```

5. Explique cómo se utiliza y qué acciones realiza la función `int InicializarUSB(const char *portname)` creada en la biblioteca `<ticommpch.h>` del Seminario 1, y exponga un programa de ejemplo que abra y cierre el puerto USB para comunicación con Arduino Uno. (ejercicio 3 y 4)

• Función para inicializar el puerto USB

- Entradas: Una cadena de caracteres por el puerto USB
(ejemplo: "/dev/ttyACM0", "/dev/ttyUSB0", etc)

- Salidas: un descriptor de puerto correspondiente al puerto de comunicaciones si hubo éxito (valor 0 o mayor). Un código de error (menor que 0) en caso contrario:

- Valor ≥ 0 : si todo se inicializó correctamente. Se corresponde con el código del puerto abierto.
- Valor -1: si hubo error al abrir el puerto.
- Valor -2: sino se conoce el puerto de entrada dado en "portname"

```
int InicializarUSB (const char * portname) {
    int fd; //Descriptor de fichero para abrir el puerto
    //Abrimos puerto para E/S, sin bloques del puerto
    fd = open (portname, O_RDWR | O_NOCTTY);
    if (fd=-1) return -1;
    [...]
```

PP

Ctr
Te'
Te



ugr

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



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.

En el byte más significativo encontramos los códigos de errores siguientes:

- UART-NO-DATA: No hay datos en el puerto UART para leer.
- UART-FRAME-ERROR: Error en la recepción del paquete de datos.
- UART-OVERRUN-ERROR: La interrupción hardware está configurada lenta: se ha recibido un dato antes de leer el que había en el puerto UART.
- UART-BUFFER-OVERFLOW: El buffer del puerto UART está lleno. Nos están enviando datos con mayor velocidad de la que los estamos procesando.

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.

#include <unistd.h>
ssize_t write (const int fd, char *buf, size_t count)

La escritura de datos que un proceso puede realizar en un archivo se lleva a cabo mediante la llamada al sistema write:

- *fd: descriptor del fichero en el que se quiere escribir.
 - *buf: puntero que apunta al buffer donde se encuentran los datos a escribir.
 - *count: indica el número de bytes que contiene el buffer.
- Esta llamada puede retornar:
- * Si se ha ejecutado correctamente: n: bytes que fueron transferidos.
 - * Si ha habido algún error: -1
- Si ocurre algún error, dicho error se almacenará en var. errno.

7.

```
#include <unistd.h>
ssize_t (signed int fd, char *buf, size_t count)
    ↑
    read
```

- La llamada al sistema read se realiza desde un proceso de usuario y permite la lectura de datos de un archivo. La llamada al sistema read devuelve el número de bytes que se han podido leer del archivo y en caso de error devuelve -1. Si esto último ocurre, se almacena el tipo de error en la variable errno.

* fd: descriptor de fichero, creado en una llamada previa 'open', del cual se leerán los datos.

* buf: puntero al buffer de caracteres donde se recibirán las datos.

* count: número máximo de bytes que se van a almacenar en el buffer de caracteres.

Referencias: Internet (página explicación)



UGR

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



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?

(mirar en "limmcompc.cpp") (Referencias : INTERNET)

- Forzamos vacío del buffer.
- Descarta datos grabados en el objeto al que se hace referencia "fd" pero que no se transmiten, o datos recibidos pero no leídos, según el valor de queue-selectors:
 - * `TCIFLUSH`: vacía los datos recibidos pero no leídos.
 - * `TCOFLUSH`: vacía los datos escritos pero no transmitidos.
 - * `TCIOFLUSH`: vacía los datos recibidos pero no leídos y los datos escritos pero no transmitidos.

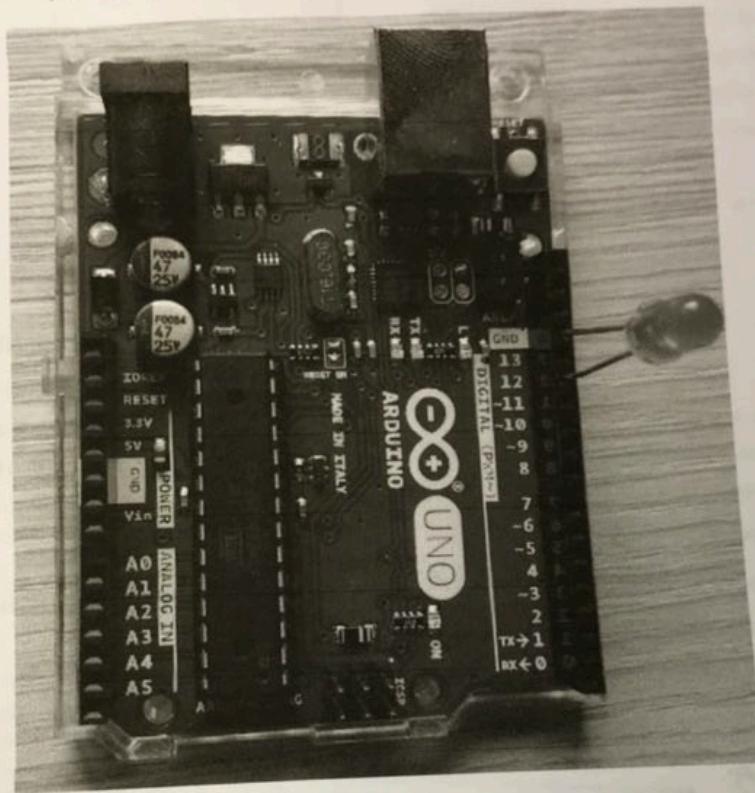
(Limpiamos los buffers del puerto (entrada y salida) antes de empezar a utilizarlo).

- Si la eliminásemos no descharfamos las entradas no leídas y/o datos de salida no enviados.

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 este configuración hardware, modifique el programa de ejemplo de conexión de serie de la parte 1 para que el cliente (segundo puerto) envíe una orden al servidor (tercero puerto) indicando el encendido o de apagado del LED. Llene a estos programas **puertoSerial** y **puertoServidor**, respectivamente.

- 0.1. El programa cliente pc_Ethernet (segundo puerto) debe tener un argumento por consola, especificando en una cadena que tipo de orden enviar al servidor (apagar, encender, o cambiar color). El resultado de la ejecución es el nombre del puerto (SERIAL) de red utilizada.
- 0.2. El programa servidor establece una conexión serial con el puerto serie del puerto SERIAL. Es decir, no necesita un servidor.
- 0.3. Si el mensaje es "On", enciende la señal de control para que el LED esté encendido y permanezca así al salir. Si es "Off", lo pone apagado. Una vez apagado, el LED se encenderá por el puerto serie el comando "On" nuevamente.



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."

programas hechos



PPC

Ctra.
Tel.
Tel.

Fa
Adr
In
R

F

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.

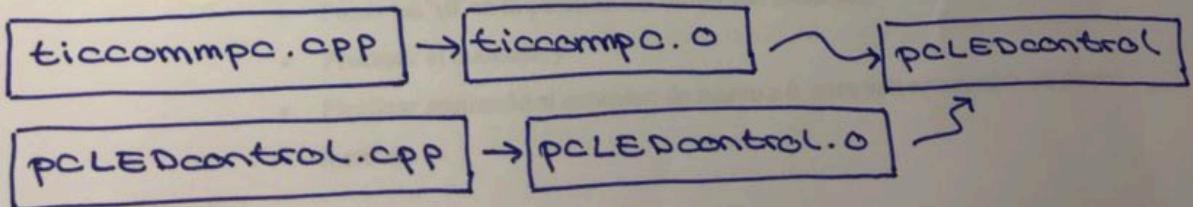
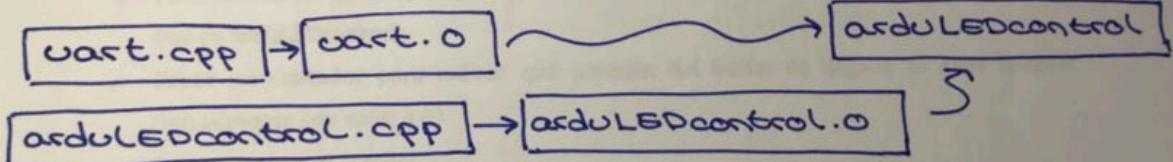
ENTREGA DEL EJERCICIO:

- 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:

"Practica1/src/arduledcontrol.cpp"

"Practica1/src/pcLEDcontrol.cpp"

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



no probado ni puesto como se enviaría ./bin/...

PP

Ctr
Te
Tf



UGR

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



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

`make arduLEDcontrolAPP`

`make pcLEDcontrolAPP`

`make sendLEDcontrol`

no están puestos los nombres

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 mismo tamaño que en `arduLEDcontrol`).
 - Tener un contador para indicar qué posición del buffer de llegada se debe llenar (*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.

a. Gracia M. Teresa, fax 95 75 09
I. 93 586 -
el. SIC Inform

Fax direct
Administración
Industrias 9:
Refinish Trair

ESPAÑA
Oficinas Con
Oficinas C
PPG - Nexr

Fábric
Fábr
Cen
F



ugr

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



- Tener un contador para indicar qué posición del buffer de llegada se debe rellenar (initialmente 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 <ticommpc.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 <ticommardu.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.

PPC

Ctra.
Tel.
Tel. S

Fax
Adr
Inc
R
E



ugr

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



- 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.

programas hechos

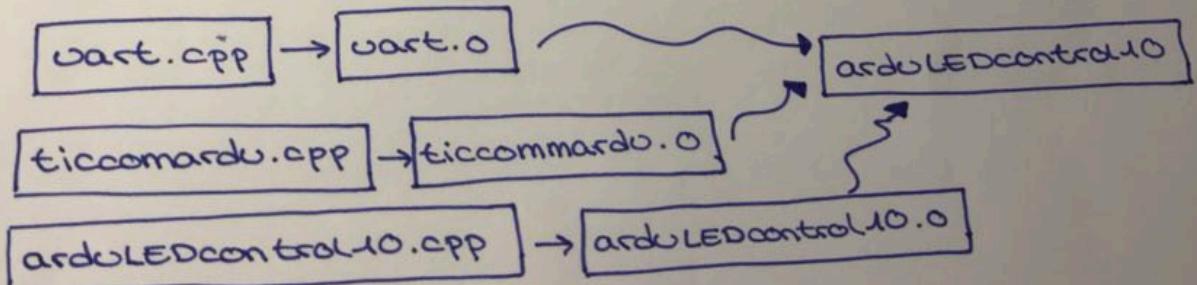
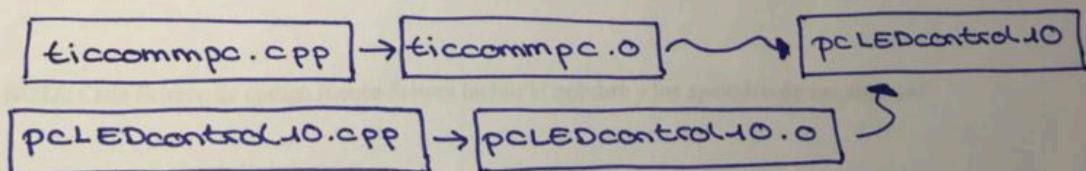
ENTREGA DEL EJERCICIO:

- 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:

"Practica1 /src /arduledcontrol10.cpp"

"Practica1 /src /pcLEDcontrol10.cpp"

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



no probado ni puesto como se enviaría ./bin/...



ugr

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



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

make arduLEDcontrol.cpp -o
make pcLEDcontrol.cpp -o
make sendLEDcontrol -o

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

no están puestos los nombres

Sesión 3

**Revisar toda la sesión, terminar
el ejercicio 5 del apartado 5.2 y
HACER EL EJERCICIO 1 DEL
APARTADO 5.3**



Ugr

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



5 Sesión 3: Sensores y emisores de láser

5.1. Contenidos de la sesión

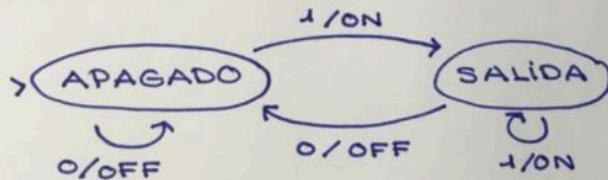
- Fotorreceptores. El fotorreceptor 2PCS láser no modulador sensor Tubo.
- Emisores láser. El módulo KY-008 de 650nm.

5.2. Cuestiones sobre fotorreceptores

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

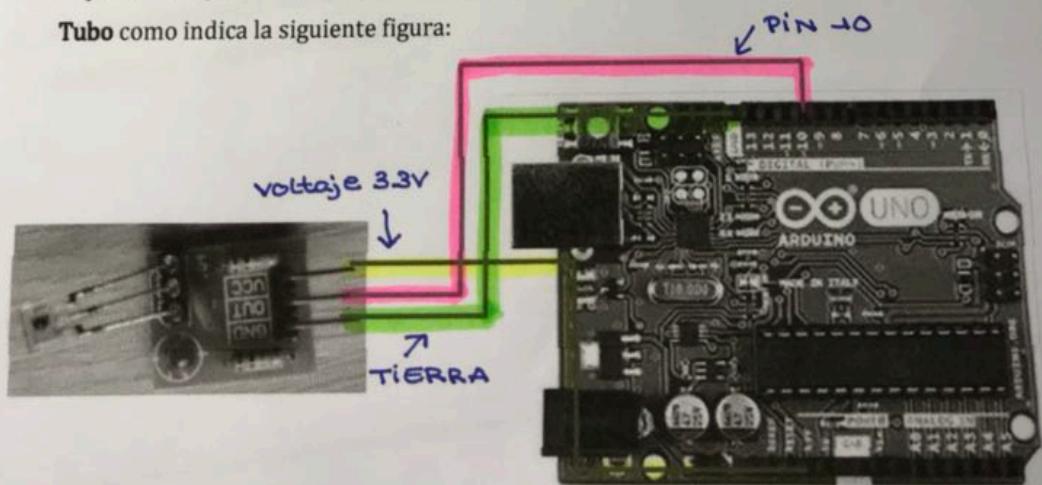
Los fotorreceptores son capaces de captar la luz emitida mediante láser. Los utilizaremos para construir los prototipos de emisión / recepción de información mediante láseres.

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



* Leer el Pin, si es '1' encender el LED, en otro caso '0' apagar el LED.

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





UGR

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



- ① Se recibe luz por el fotoresistor
② No se recibe luz por el fotoresistor

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.

```
/* FOTORRECEPTOR */  
[...]  
// Ponemos PIN 10 de la placa como entrada (DDRB &= ~BV(DRB2)  
[...]  
while(1){  
    //Cogemos que hay en el PIN 10 como entrada  
    dato = PINB & 0x03  
    ① if (dato>0 && ...) { PORTB |= ~BV(PORTB4); ... }  
    ② else if (dato==0 && ...) { PORTB &= ~_BV(PORTB4); ... }  
    [...]
```

LED
PIN 12

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

- PORTB es un byte que contiene los datos de salida existentes en los pines.
- PINB se utiliza para las entradas.
- PINB & 0x04 → Cogemos qué hay en el PIN 11 como entrada.
- PINB & 0x03 → Cogemos qué hay en el PIN 10 como entrada.

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

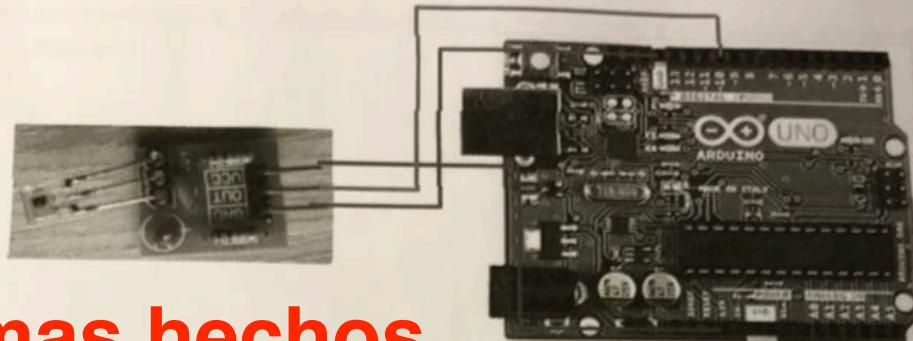


ugr

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



Apartado - 00 - E-mail:
- 93 586 75 04
- 74 40



programas hechos

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:

- 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>.
- 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 fotoresistor. 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:

HACER



UGR

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



DECSAI

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

HACER

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

HACER



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



Apartado 1000-7500 - E-24081
Málaga - Spain

Edificio General 93 580
Málaga - Spain

Fax 96
90 - Fax
20 - Fax

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.

HACER

Sesión 4-5

**HACER EL EJERCICIO 1 DEL
del apartado 6.1,
revisar los demás ejercicios y
HACER EL APARTADO 6.4**



UGR

Universidad de Granada

Departamento de Ciencias de la Computación
e Inteligencia Artificial

DECSAI

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.

- *CodeBit*: un bit de un código a transmitir.
- *LaserBit*: un bit realmente transmitido por el medio físico.

2.

Ejemplo: para transmitir los codeBits 000.11.000 (SMS) se han necesitado un total de 22 easerBits:

código $\begin{cases} 0 = 10 & \text{F. Simbolo} \\ 1 = 110 & 101010\textcircled{0}110110\textcircled{0}101010\textcircled{0}0 \end{cases}$ F. Simbolo F. Simbolo F. Simbolo
F. mensaje

Estos easerBits se han utilizado para:

- Enviar 000 y el fin de símbolo.
(7 easerbits para 3 codeBits)
- Enviar 11 y el fin de símbolo.
(7 easerbits para 2 codeBits)
- Enviar 000 y el fin de símbolo.
(7 easerbits para 3 codeBits)
- Enviar el fin del mensaje (1 easerBit)



UGR

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



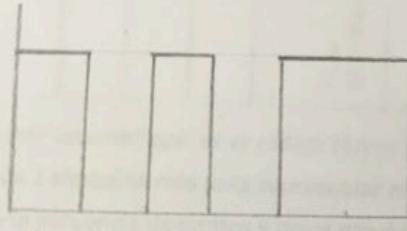
F-mail
22 - D
4 08
5 10
BARCE
da. de
Fax Ce
Fax A
DE PO
GUNA
LVA IC
D PAL
- Fax

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.

Los codeBits serán los que se enviarán indefinidamente mediante el láser en una placa Arduino.

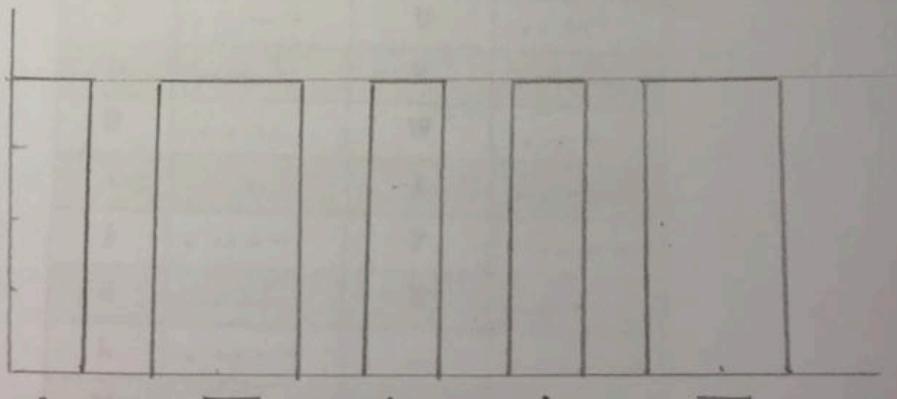
$$\left\{ \begin{array}{l} \text{LASERBITS} = 010110 \\ \text{CODEBITS} = 001 \\ \dots \\ \text{código} \quad \left\{ \begin{array}{l} 01 = 0 \\ 10 = 1 \end{array} \right. \end{array} \right.$$

*Supongo que en esta secuencia no hay bits para señalizar el fin de símbolo ni el fin del mensaje.
(aplico lo mismo para el ejercicio 3 como el ejercicio 4)



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.

$$\text{CODEBITS} = 01001 \\ \dots$$





UGR

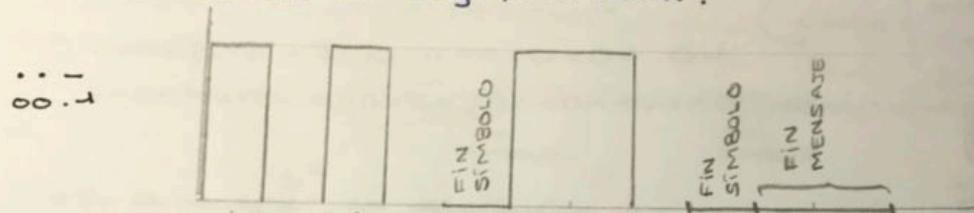
Universidad de Granada
Departamento de Ciencias de la Computación
e Inteligencia Artificial



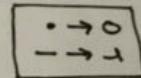
DECSAI

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.

Cuando enciendes un símbolo y queremos señalizar su fin, enviaremos un ciclo a la baja y cuando queremos señalizar el fin del mensaje, enviaremos dos ciclos a baja, es decir:



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 asignele 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	— —	(ESPACIO)	• • • •
O	— — —		



UGR

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

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.

HOLA = ...,-,-.,-,-,-,-

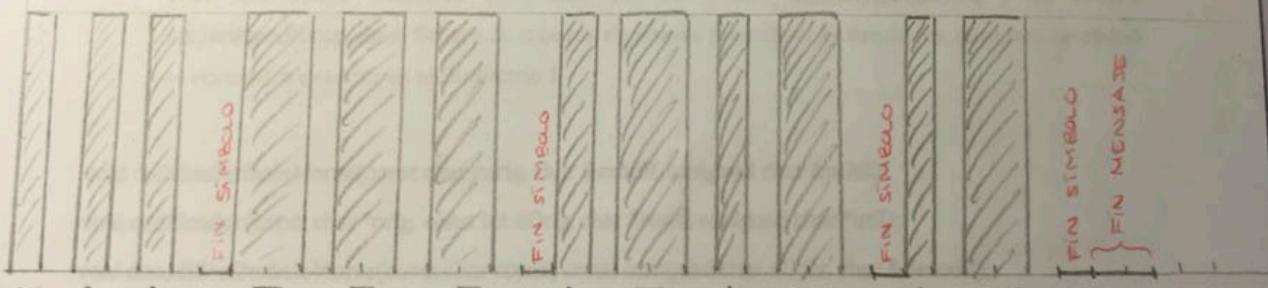
CODEBITS = 000 . -111 . 0101 . 01

LASERBITS = 10101010 110110110 10110101010 10110100

código { 0=10
1=-10

* En el receptor, se detectaría que se ha recibido la "H", debido al FIN DE SIMBOLO.

* Se detecta el fin del mensaje, cuando llega un fin de símbolo, más otro bit de serialización del mensaje.



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?

El teorema de muestreo afirma que para muestrear correctamente una señal de f Hz, se requiere como mínimo una frecuencia de muestreo de $2f$ Hz.

Por lo que necesitamos capturar suficiente información para ser capaces de reconstruir la señal analógica original.

(MUESTREO: medir la amplitud de la señal a intervalos regulares.)

C A

... de Correos
ail: 100

Por consiguiente, hacemos uso de este teorema en las prácticas usando las constantes:

- ① UMBRAL-U 15
- ② SAMPLE-PERIODO UMBRAL-U/3)

Usaremos un umbral suficiente alto, para poder enviar y recibir, así que usaremos $U = 15\text{ms}$ y múltiplo de 3, porque vamos hacer 3 muestras (15×3), es decir cada 5ms voy muestreando:

- ① Tiempo mínimo que dura un ciclo de ráfaga.
- ② La frecuencia de muestreo que utilizaremos para 'mirar' que se detecta en el fotorreceptor (tiempo de muestreo)



UGR

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



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);
```

- 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



UGR

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



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 `decodificaSímboloMorse` 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, XXXXX11}` y `utils={3, 2}`. Otro ejemplo: para la entrada `orig="JKM"` con `nOrig=3`, se devolverá `codif={XXXX0111, XXXX101, XXXXX11}` 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, XXXXX11}`, `utils={3, 2}`. Y `nCodif=2`, se devolverá la cadena `decodif ="RM"`. Otro ejemplo: para la entrada devolverá `codif={ XXXX0111, XXXX101, XXXXX11}` y `utils={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".

→ codificador (cadena, tam-cadena, cadena-codificada,

Bye strlen(Bye) → SALIDA
codificador (cadena, tam-cadena, cadena-codificada, nutils)

for i to tam-cadena :

1. Codificamos símbolo a símbolo (los símbolos de la cadena)
codifica símbolo (cadena[i], símbolo-codificado, util-símbolo)
 - 1.1. Nos vamos a la función "codificaSímbolo" y buscamos el CASE del símbolo (letra) y devolvemos su codificación y su tamaño util.
2. Guardamos todos los símbolos codificados y su tamaño util en dos buffers.



- 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}.
- $\begin{array}{ccccccc} \downarrow & \dots & \dots & \dots & \downarrow & \text{4,2,2,3} & \downarrow \\ \text{decodificadores} & \xrightarrow{\text{cadena_codificada}} & \xrightarrow{\text{nútels}} & \xrightarrow{\text{tam}} & \xrightarrow{\text{decodificada}} & \xrightarrow{\text{(n-elementos_nútels)}} & \text{SALIDA} \end{array}$
- for i:0...tam :
- 1. Decodificamos uno a uno los símbolos de la cadena decodificada. Símbolo (cadena_codificada[i], nútels[i], símbolo_codificado)
 - 1.1. Nos vamos a la función 'decodificaSímbolo', y buscamos el CASE con el mismo 'nútels', luego dentro de ese CASE, buscamos en 'cadena_codificada' y devolvemos el símbolo decodificado.
 - 2. Guardamos todas las cadena decodificadas en un buffer y es devolvemos .

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:
 - o Pida desde la entrada estándar hasta un máximo de 100 caracteres.
 - o Si no se ha introducido nada (cadena vacía), salir del programa.
 - o 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.
 - o 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:

"Practica1 / src / emisorPCSS.cpp"

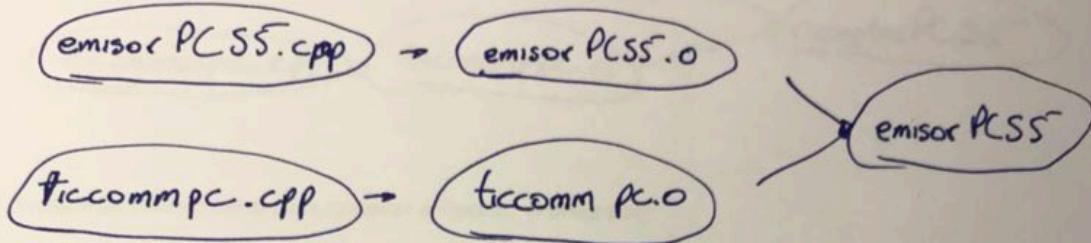


ugr

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



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



- Indique las instrucciones para compilar y ejecutar el programa:

make emisor_PCSS

./bin/emisor_PCSS

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

- 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:

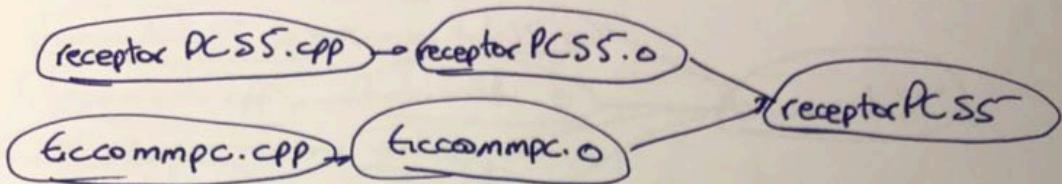
"Practica1/src/receptor_PCSS.cpp"

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



ugr

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



- Indique las instrucciones para compilar y ejecutar el programa:

make receptorPCSS

./bin/receptorPCSS

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:

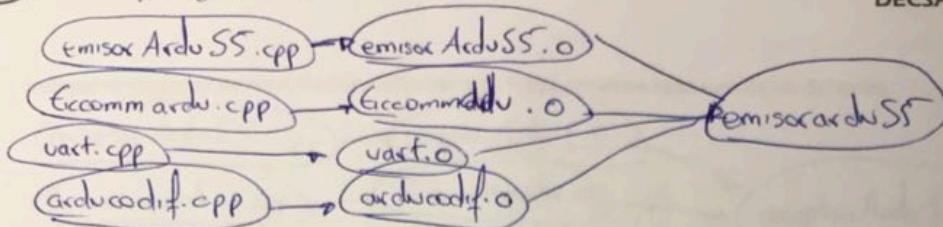
"Practica1/sec/emisorArduSS.cpp"

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



UGR Universidad de Granada

Departamento de Ciencias de la Computación
e Inteligencia Artificial



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

make emisorArduSS

make sendEmisorSS

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 (llámemosle **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:

"Practica1/src/receptorArduSS.cpp"

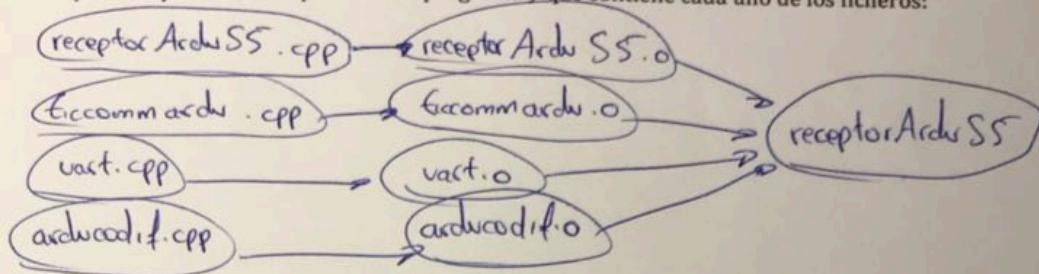


ugr

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



- 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:

make receptorArduSS

make sendReceptorSS

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

6.4. Cuestiones teóricas sobre el prototipo implementado

HACER

- ¿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.



ugr

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

HACER



7. Suponiendo que la emisión de cualquier símbolo “A”, “B”, “C”, etc. es equiprobable, razoné, 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?



ugr

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

HACER



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?



ugr

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

HACER



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.