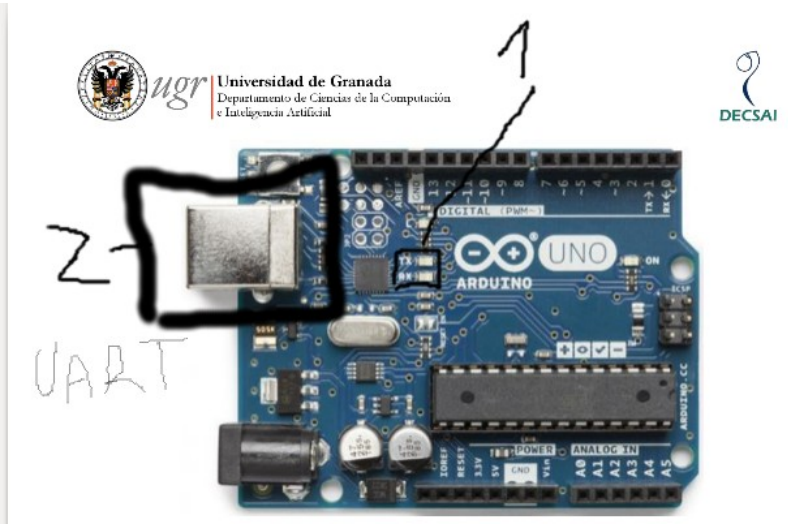
Práctica 1 – Sesión 2

4.2 Cuestiones sobre el puerto UART

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

UART son las siglas en inglés de Universal Asynchronous Receiver-Transmitter. Es el dispositivo que controla los puertos y dispositivos serie y se encuentra integrado en la placa Arduino.
UART permite comunicar con Arduino durante la ejecución del programa y se realiza a través de USB.
Las funciones principales del puerto UART son manejar las interrupciones de los dispositivos conectados al puerto serie y de convertir los datos en formato paralelo, transmitidos al bus del sistema, a datos en formato serie, para que puedan ser transmitidos a través de los puertos y viceversa.

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.



2: puerto gestionado por UART.
1: el led de arriba indica transmisión. El led de abajo indica lectura.

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 USB se gestiona a través de la librería *uart.h*.

Para utilizar esta librería es necesario disponer del código, enlazándolo en el fichero fuente a través de la instrucción *#include <uart.h>*.

Se debe inicializar con la sentencia *uart_init(UART_BAUD_SELECT(x, y))*, siendo x la velocidad en baudios de las comunicaciones serie e y el ciclo de reloj del procesador.

sei() activa las interrupciones hardware para control del puerto serie.

Para enviar datos se utiliza uart_puts(cadena).

Para recibir datos se utiliza uart_getc().

Programa:

```
// Ciclo de reloj del procesador: 16MHz
#define F_CPU 16000000UL

// Velocidad (en baudios) de las comunicaciones serie
#define UART_BAUD_RATE 9600

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

#define BLINK_DELAY_MS 1000

int main(void) {

   // Se establecen como entrada el pin 12
   DDRB |= 0b00010000;

   // Inicialización del puerto UART con la velocidad
   // en baudios del puerto, y la velocidad del procesador
   uart_init(UART_BAUD_SELECT(UART_BAUD_RATE, F_CPU));

   // Activación de las interrupciones hardware para
   // control del puerto serie
   sei();

   while (1) {

         // Recibe los datos del buffer
         char c = uart_getc();
```

```
        // Envía los datos
        uart_putc(data);

    }

    return 0;

}
```

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.

Es una unidad de medida utilizada en telecomunicaciones que representa el número de símbolos por segundo en un medio de transmisión digital.

En comunicaciones serie se utilizan para establecer la velocidad a la que se deben transmistir los datos.

La velocidad básica tradicional es 9600 baudios.

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

Los bps indican la cantidad de bits que se transmiten por segundo. En cambio, los baudios indican los símbolos que se transmiten por segundo. La diferencia se encuentra en que un símbolo puede representar más de un bit.
Este hecho provoca que a velocidades bajas, los bps y baudios son iguales, pero en velocidades altas, por ejemplo, se pueden enviar el doble de bits que de símbolos.

4.3

4.3.9

Practica1/ejercicio4.3.9

- arduLEDcontrol.h / arduLEDcontrol.cpp → controla la respuesta que da el cliente. En este caso enciende un led, lo apaga o no hace nada dependiendo de la orden que reciba.

- pcLEDcontrol.h / pcLEDcontrol.cpp → envía al cliente la orden que quiere realizar, encender o apagar el led.

- ticcommpc.h / ticcommpc.cpp → se encarga de gestionar el envío y recibimiento de datos por parte del servidor.

- uart.h / uart.cpp → se encarga de gestionar las comunicaciones serie entre los dispositivos.

arduLEDcontrol:

- make
- make send

pcLEDcontrol:

- make pcLEDcontrol
- ./bin/pcLEDcontrol [On|Off]

4.3.10

Practica1/ejercicio4.3.10

- arduLEDcontrol.h / arduLEDcontrol.cpp → controla la respuesta que da el cliente. En este caso enciende un led, lo apaga o no hace nada dependiendo de la orden que reciba.

- pcLEDcontrol.h / pcLEDcontrol.cpp → envía al cliente la orden que quiere realizar, encender o apagar el led.

- ticcommpc.h / ticcommpc.cpp → se encarga de gestionar el envío y recibimiento de datos por parte del servidor.

- ticcommardu.h / ticcommardu.cpp → se encarga de gestionar el envío y recibimiento de datos por parte del cliente.

- uart.h / uart.cpp → se encarga de gestionar las comunicaciones serie entre los dispositivos.

arduLEDcontrol:

- make
- make send

pcLEDcontrol:

- make pcLEDcontrol
- ./bin/pcLEDcontrol [On|Off]