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

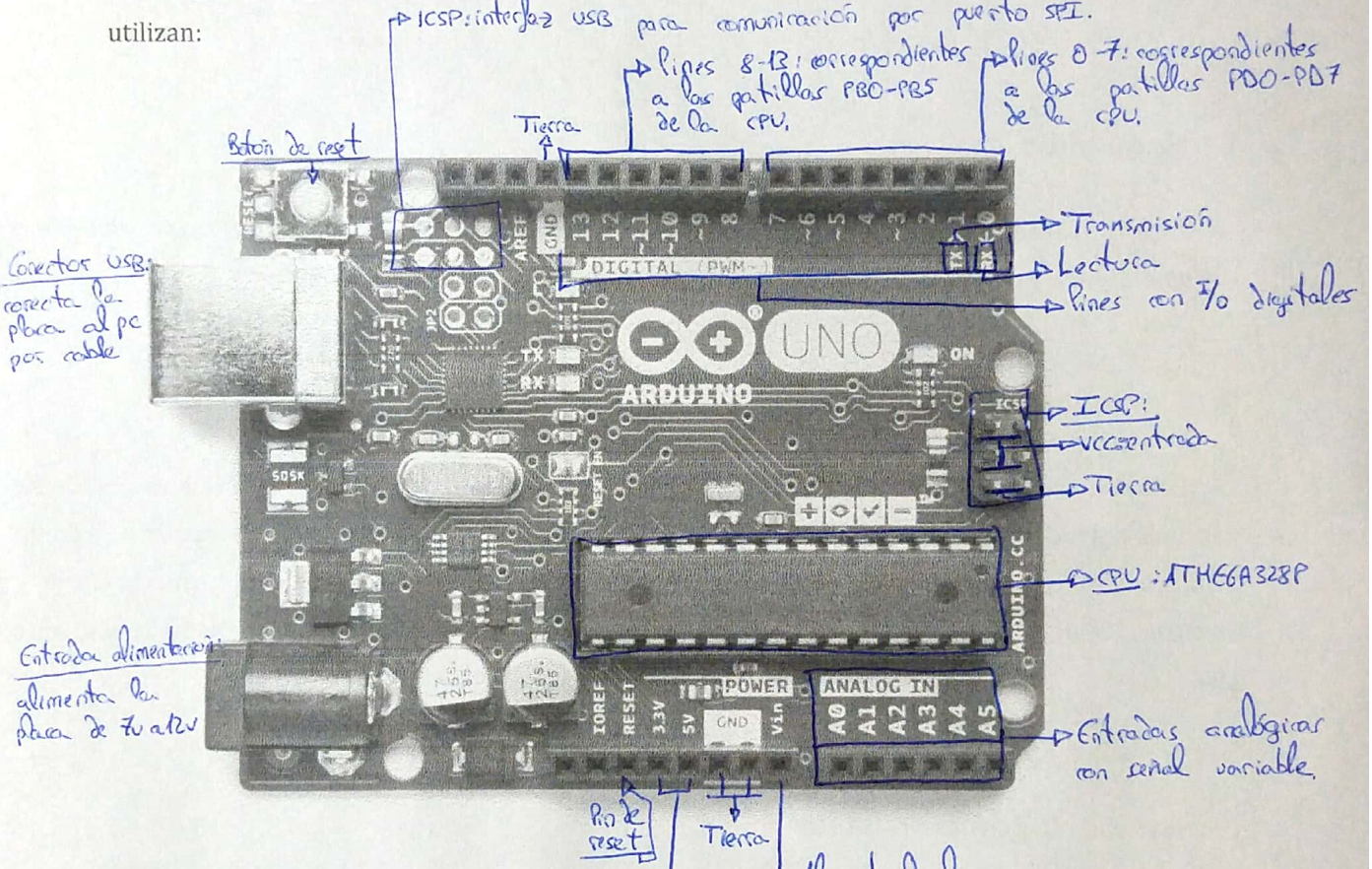
3.1. Contenidos de la sesión

- 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:



Reset: resetea la placa a través del pin o con un botón físico.

ICSP: sirve para programas el bootloader del microcontrolados ATMega y cargar programas directamente.

Tierra: sirve para tener como referencia el voltaje de la placa.

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

El puerto digital se corresponde con los pines del 1 al 13.
Solo entienden dos niveles de señal: LOW con valores cercanos a 0V y HIGH con valores cercanos a 5V. Engloban tanto las entradas como las salidas en el mismo puerto.
El puerto analógico se corresponde con los pines de A0 a A5. Solo contienen las entradas y leen valores de tensión de 0 a 5 voltios con una resolución de 10 bits.
La diferencia entre ambos puertos es que el analógico es solo de entrada y el digital es de entrada y salida. Además el analógico lee en un rango de valores y el digital solo lee dos valores (o tiene dos estados).

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?

Tiene 1 puerto analógico.
El puerto tiene 6 pines.
Se identifican del puerto A0 al A5.
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?

Tiene 1 puerto digital.
Tiene 14 pines en el puerto, de los cuales 6 proveen una salida PWM.
Se identifican del puerto 0 al 13.
Los puertos son de entrada y 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?

Tiene tres tomas de tierra (junto al puerto digital y los pines de Power). Sirve para regular la tensión que existe en la placa. Cuando se alimenta con alguna tensión, necesita otra como referencia y permite mantener el voltaje dentro de límites correctos cuando existe algún fallo.

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.

Se puede alimentar conectándola por USB, por la entrada de corriente estándar de 7v a 12v y mediante el pin Vin que se encuentra en el puesto POWER.
El voltaje recomendado oscila entre 7v y 12v.

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

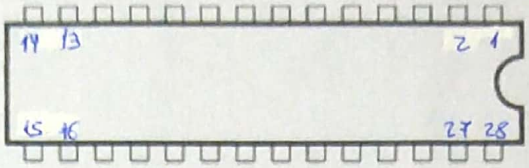
- No conectar dos o más pines entre sí.
- No aplicar un voltaje superior 5'5v a cualquier pin digital.
- No invertir la corriente al utilizar el pin Vin.
- No conectar más voltaje a los pines de 5v o 3'3v.
- No conectar el voltaje a tierra.
- No alimentar la placa desde dos entradas de corriente diferentes.
- No aplicar más de 13v al pin RESET.
- No superar los 200 mAh como corriente total en la placa.
- No tocar con los dedos las patillas de los elementos que se conecten a los pines.

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).

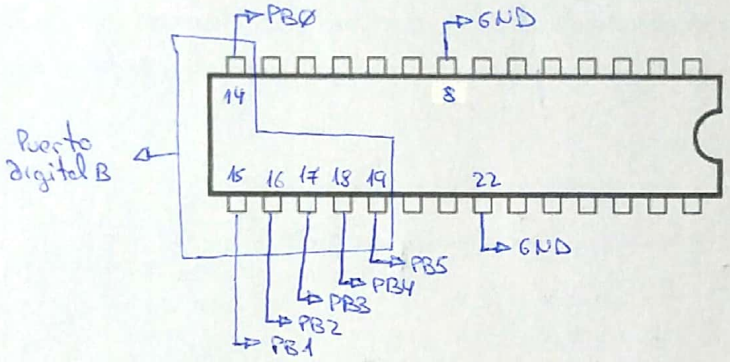
- Voltaje de funcionamiento: 1'8V - 5'5V
- Fiecuencia de trabajo del microprocesador:
  - Ø - 4 MHz : 1'8V - 5'5V
  - Ø - 10 MHz : 2'7V - 5'5V
  - Ø - 20 MHz : 4'5V - 5'5V
- Número de pines: 28-pin PDIP
- Número de puertos y su tipo:
  - 3 puertos.
  - B: pines digitales del 8 al 13.
  - C: entradas analógicas.
  - D: pines digitales del Ø al 7.

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.

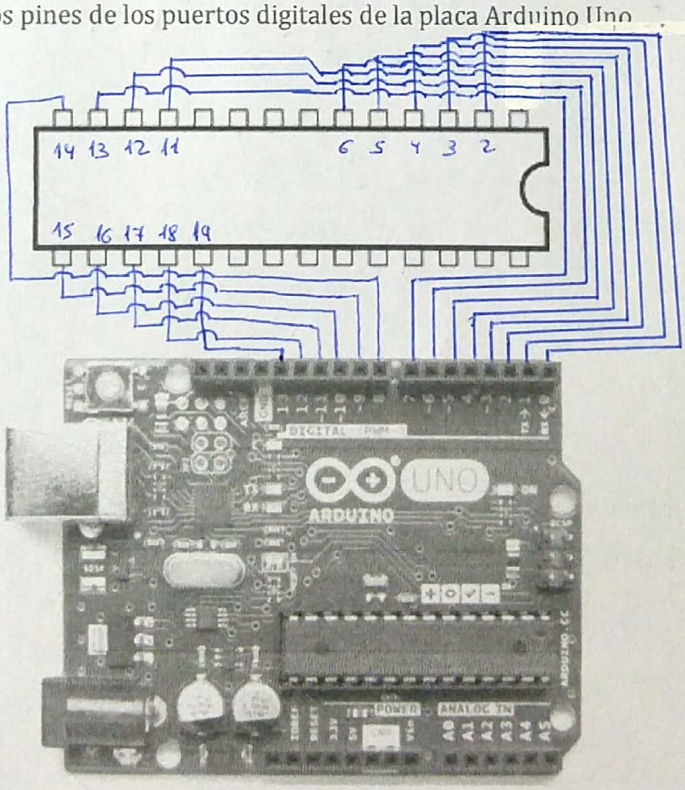


Se coloca la CPU en vertical con la muesca hacia arriba, numerando los pines desde la esquina superior izquierda hasta la esquina inferior izquierda, continuando por la esquina inferior derecha hasta la esquina superior derecha.

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?

Tiene tres tipos de memoria:
- 32KBytes de memoria flash para programas.
- 1KByte EEPROM.
- 2KBytes SRAM Interna

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

Los programas se guardan en la memoria flash.
Se encuentra junto a la conexión por USB.

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

SRAM: similar a la RAM en un pc. Memoria volátil donde se guardan las variables y los datos parciales.

EEPROM: memoria no volátil que mantiene los datos después de desconectar la placa de la corriente. Tiene un número de usos limitados.

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 UNO: ATmega328p
- Arduino Mega: ATmega256
- Arduino Ethernet: ATmega328p
- Arduino DUE: Atmel SAM3X8E ARM Cortex-M3
- Arduino Leonardo: ATmega32u4
- Arduino Micro: ATmega32u4
- Arduino Mini: ATMega328

Otra placa que utiliza el microprocesador ATmega328p es GEEKCREIT UNO.

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 el compilador/enlazador que permite crear un archivo ejecutable para Arduino desde un fichero escrito en C y así poder enviar el programa desarrollado para su ejecución.

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

   Arduino cuenta con poco espacio en memoria y esta opción permite optimizar nuestro programa para que pese lo menos posible.

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

   En Arduino se debe indicar qué microprocesador se está utilizando y se hace a través de esta opción.
   Esta opción es la que permite indicar el set de instrucciones de la arquitectura de la CPU.

4. Explique qué hace la siguiente orden:

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

   A partir de un programa escrito en C (main.cpp) permite generar un fichero objeto, con el nombre main.o, optimizando en espacio e indicando la arquitectura de la CPU.

5. Explique qué hace la siguiente orden:

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

   A partir del fichero objeto anterior, crea un ejecutable.

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

Genera un fichero en hexadecimal a partir del ejecutable anterior.
-O ihex indica que el fichero de salida será en hexadecimal
-R elimina una sección que no es necesaria, en este caso .eeprom.
Los nombres de los ficheros indican el fichero fuente y la salida

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.

lsusb
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

Envía el programa a Arduino para su ejecución.
-F: comprueba que la firma del dispositivo es correcta para poder continuar.
-V: deshabilita el chequeo automático de verificación cuando se envían los ficheros.
-c arduino: indica el id del programa que va a ser usado.
-p ATMEGA328p: indica la CPU conectada.
-P /dev/ttyACM0: puerto que identifica al dispositivo.
-b 115200: indica la frecuencia de ~~rafy~~ de la CPU.
-Uflash:w:main.hex: indica el lugar al que se envía el fichero.

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

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 indica el tiempo en el que se refresca el tick del procesador.
   Se define al comienzo del programa, donde se realizan los DEFINE

   Ejemplo:
   #define F_CPU 16000000UL

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

   16000000UL

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

   DDRB es el registro de direccionamiento de datos del puerto B del microprocesador (PB0 - PB5).
   Se utiliza para indicar como se quiere que funcionen los pines del puerto B.

```
// 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.

DDB0 indica el pin 0 del puerto B.
Se utilizar para establecer ese pin como entrada o salida.

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.
Se corresponden con los pines del 9 al 13 del puerto digital.

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;

}
```

Se está asignando en la macro DDRB (pines del 8 al 13) los cuatro primeros como HIGH (salida) y el resto como LOW (entrada).

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.

La variable PORTB es la salida del puerto B. Contiene los datos de salida existentes en los pines PB0-PB7 de la CPU.
Se puede comparar con "cout" de C++.
En este caso manda una señal de voltaje alto al Pin 0 del puerto B.

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

DDRB es el puerto en general, tanto entradas como salidas.
PORTB es la salida de ese puerto.

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.

Similar a la pregunta 8 pero refiriéndose al pin 8.

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.

La asignación de este ejercicio establece todos los pines a 0, excepto el primero que lo hace a 1.
La instrucción del ejercicio 4 mantiene los pines como estuvieran, excepto el primero que lo cambia a 1 (si era 0).

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.

En este caso, la instrucción sí mantiene todos los pines como estuvieran anteriormente, excepto el primero, que siempre le asigna 1.
Las líneas son equivalentes.

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.

Es equivalente porque el resto de pines se quedan como estaban, excepto el primero que siempre valdrá 1.

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?.

Se estarían asignando todos los pines como entrada, excepto el primero como salida. No se sabe qué elementos se encuentran conectados a los pines, por lo que si se establece como entrada y un elemento es de salida (o al contrario) podrían encontrarse fallos.

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

Mantiene el resto de pines como estén, excepto el primero que cambia su configuración. Si se encuentra como salida, de este modo pasa a ser entrada.

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?

No se utiliza esta instrucción por lo comentado 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 = 0x0F;
 PORTB= 0xF0;
}
```

La primera instrucción establece los cuatro primeros pines como salida y el resto como entrada. La segunda instrucción manda voltaje bajo a los cuatro primeros y alto al resto, por lo que esta inconsistencia podría dañar la placa

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?

Los pines que se han configurado como salida son los cuatro primeros, los únicos que tienen voltaje alto son los tres primeros, por lo que el cuarto no estaría correctamente configurado.

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.

Con esta configuración, el led se encendería siempre, ya que es a partir del pin 11 (incluido) cuando no funcionaría.

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.

Como se quiere cambiar solo los pines 8 y 9 de la placa y estos con el primero y segundo del puerto B, lo que se debe hacer es PORTB &= 0x4.
Si se tienen los pines configurados como 000111 y se quiere que el resultado sea 000100 haciendo la operación &, se tiene que el valor necesario es 000100, que en hexadecimal es 4.