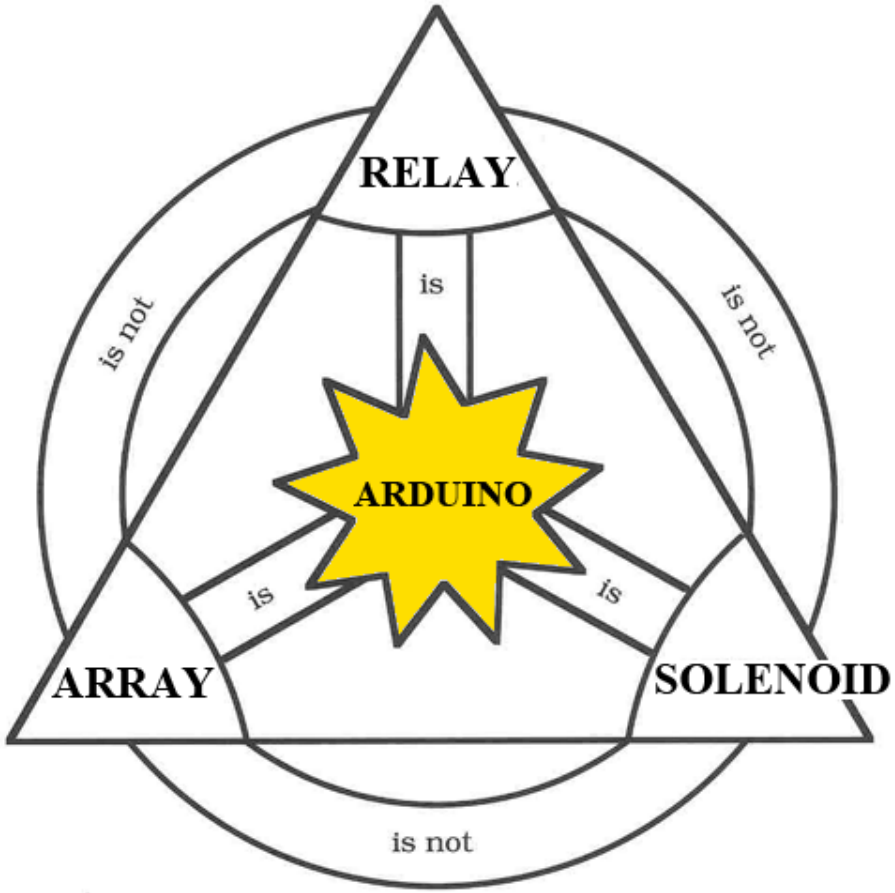


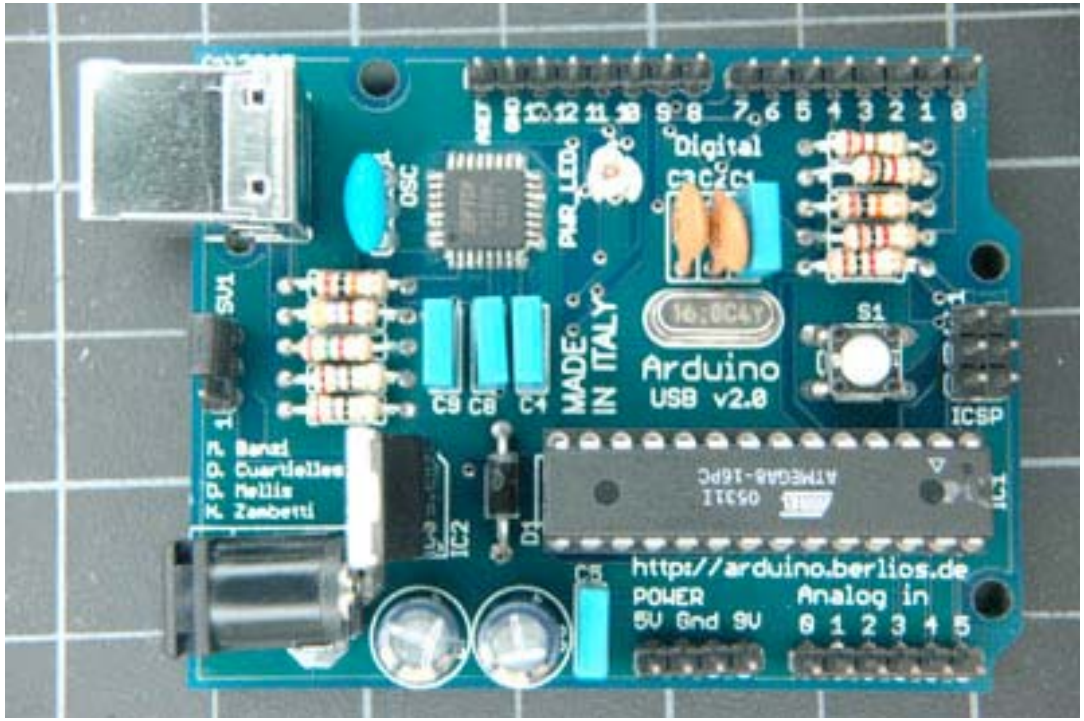
Arduino

OPEN HARDWARE - David Cuartielles, Máximo Banti

http://www.arduino.cc/es/

Hardware

http://www.arduino.cc/es/

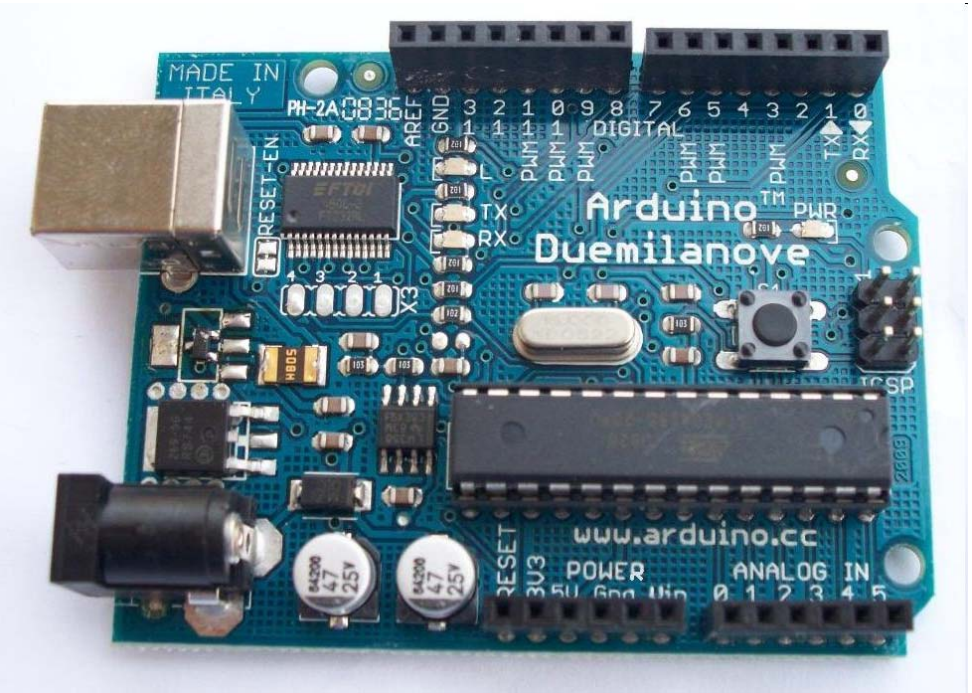


Arduino es una plataforma open-hardware basada en una sencilla placa con entradas y salidas (E/S), **analógicas y digitales**, y en un entorno de desarrollo que implementa el lenguaje Processing/Wiring.

Su corazón es el chip **Atmega8 (antiguas placas),** un chip sencillo y de bajo coste que permite el desarrollo de múltiples diseños.

Hardware

http://www.arduino.cc



Modelo Duemilanove - 2009

Microcontroller: **ATmega168**

Operating Voltage: **5V**

Input Voltage (recomendado) **7-12V** Alimentación externa

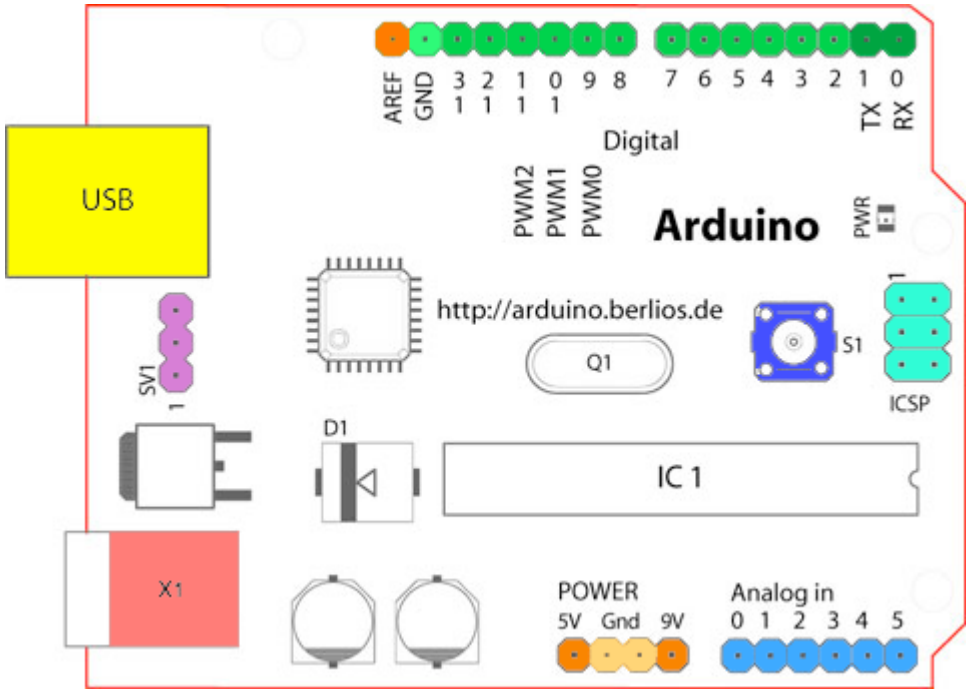
Input Voltage (limite): 6-20V

Digital I/O Pins: 14 (de los cuales 6 pueden ser PWM output [3, 5, 6, 9, 10, y 11])

Analog Input Pins: 6

Flash Memory: **16 KB** (ATmega168) o **32 KB** (ATmega328)

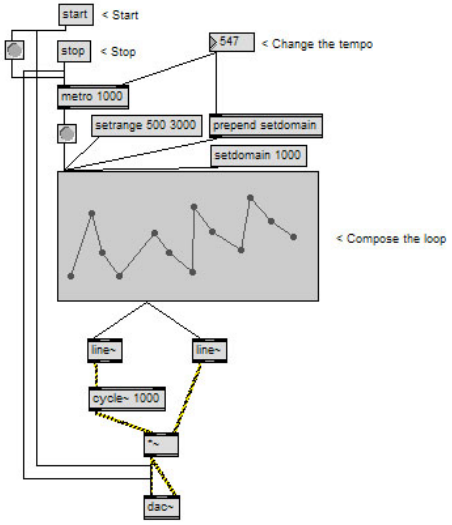
http://www.arduino.cc/es/



- Pin de referencia analógica (naranja)
- Señal de tierra digital (verde claro)
- Pines digitales 3-13 (verde)
- Pines digitales 1-2 / entrada y salida del puerto serie: TX/RX (verde oscuro)
- Botón de reset (azul oscuro)
- Entrada del circuito del programador serie (azul turquesa)
- Pines de entrada analógica 0-5 (azul claro)
- Pines de alimentación y tierra (fuerza: naranja, tierra: naranja claro)
- Entrada de la fuente de alimentación externa (9-12V DC) – X1 (rosa)
- Conmuta entre fuente de alimentación externa o alimentación a través del puerto USB – SV1 (violeta)
- Puerto USB (amarillo)

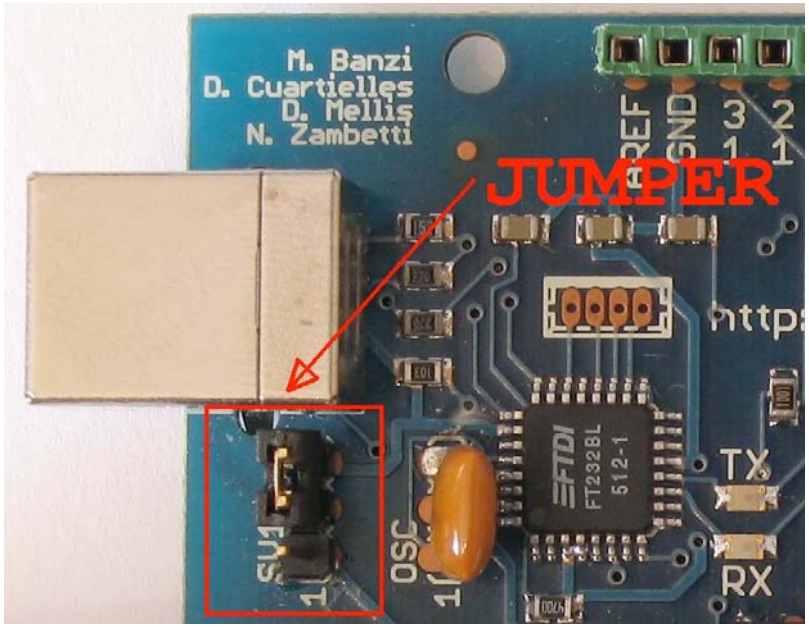
¿Para qué puedo utilizar Arduino?

Arduino puede utilizarse en el desarrollo de objetos interactivos autónomos o puede conectarse a un PC a través del puerto serie utilizando lenguajes como **Flash, Processing, MaxMSP, PureData,** etc ... Las posibilidades de realizar desarrollos basados en Arduino tienen como límite la imaginación.
Asimismo, su sencillez y su bajo coste, recomiendan su uso como elemento de aprendizaje e iniciación en el mundo de la electrónica digital.

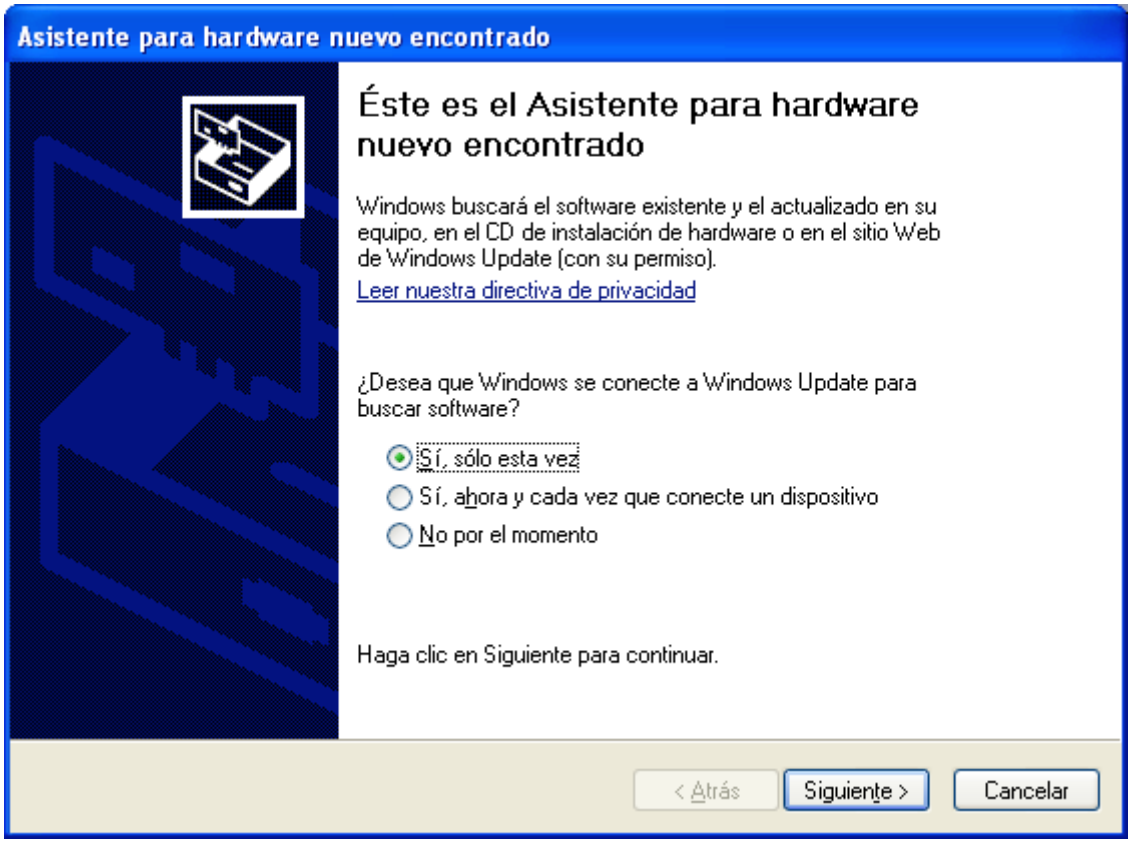


El cable USB debe ser tal y como se muestra en la imagen inferior. Con un conector tipo A (para conectar al PC) y otro tipo B (para conectar a la placa) en sus extremos No hay que equivocarlo con el cable mini-USB que habitualmente se utiliza con dispositivos más pequeños como cámaras de fotos y lectores de tarjetas.

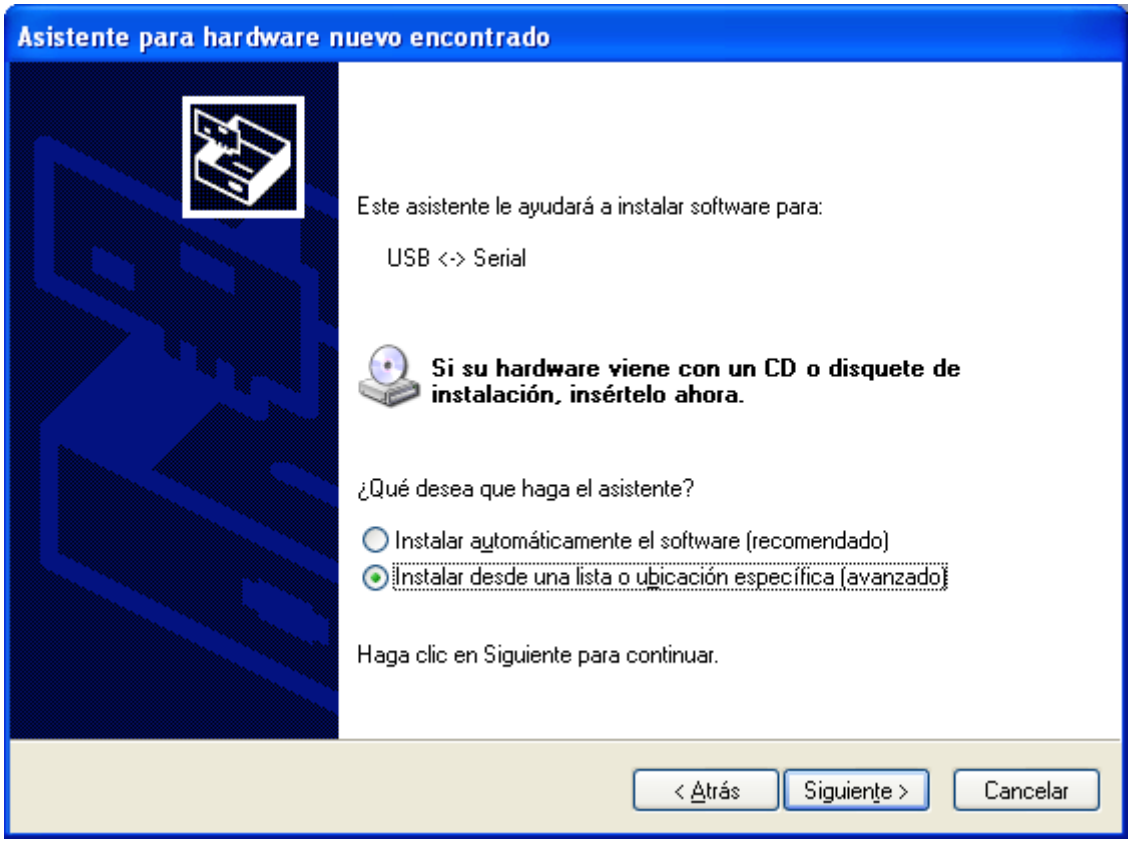
Un tema muy importante a tener en cuenta es que en la placa USB se nos ofrece la posibilidad de alimentar la placa a través de una fuente de alimentación externa. En la imagen siguiente se muestra la posición en la que debe estar el “jumper” para que la alimentación de la placa se realice desde el cable USB. Si se coloca de en la otra posición posible la placa tomará la alimentación de la fuente externa.



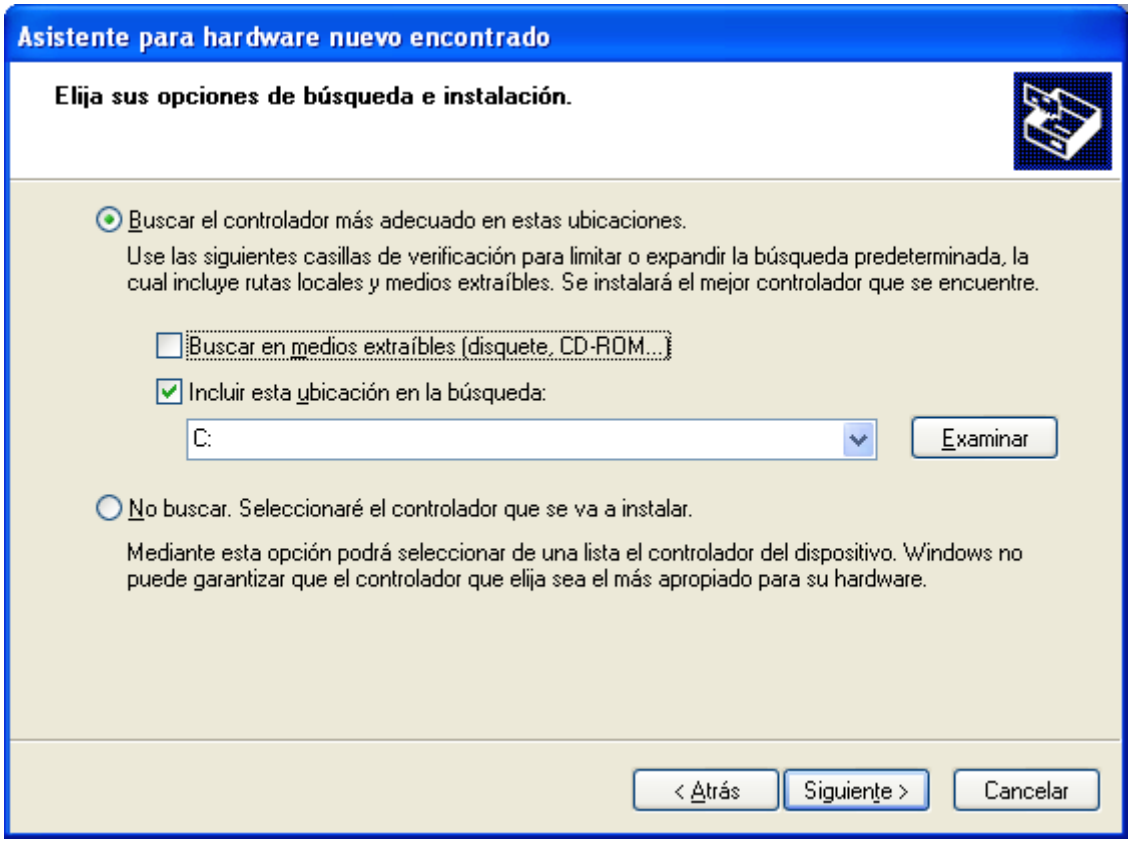
Instalación USB



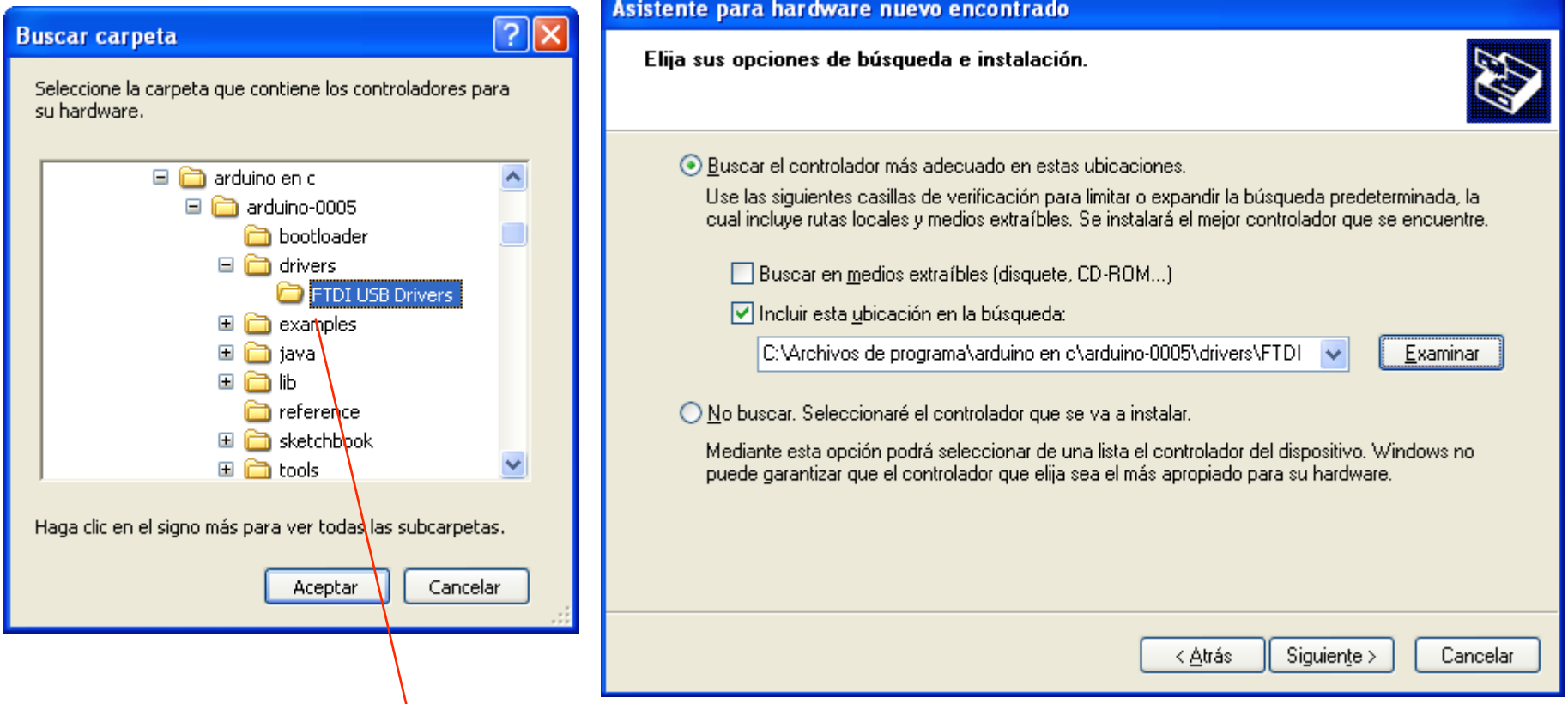
Instalación USB



Instalación USB

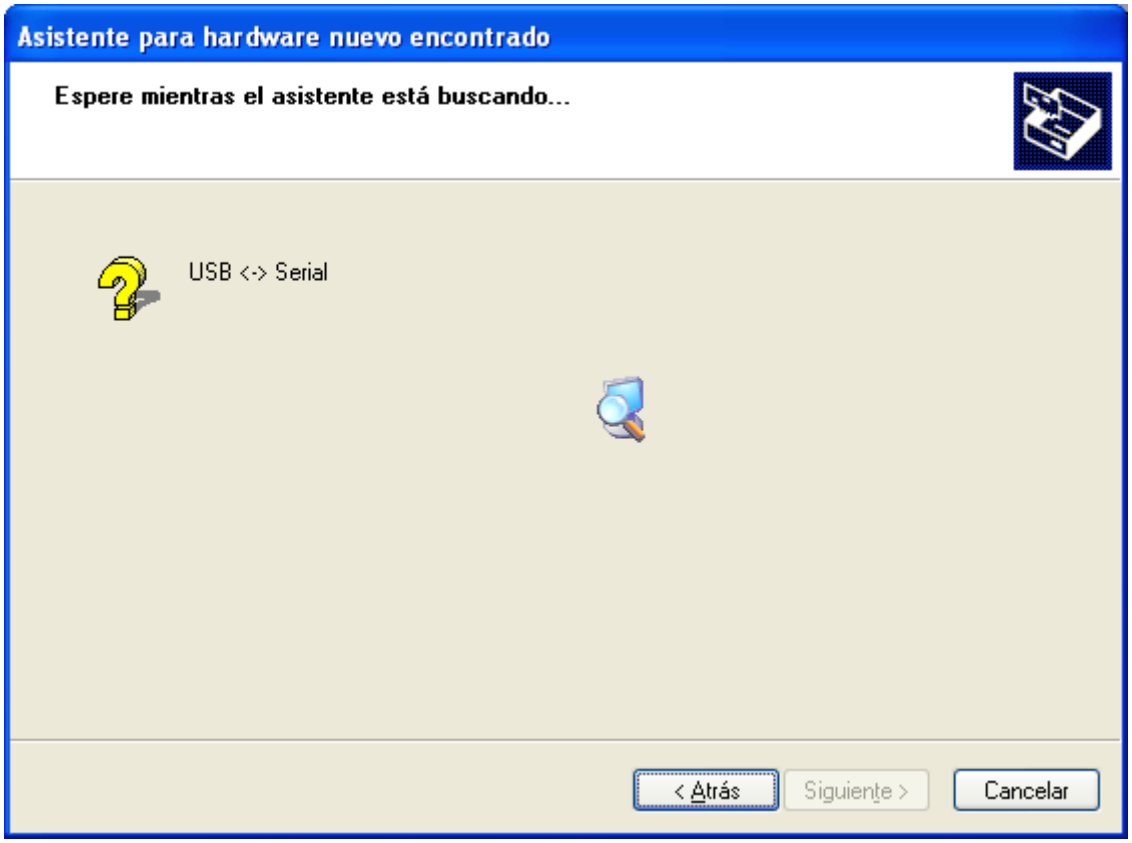


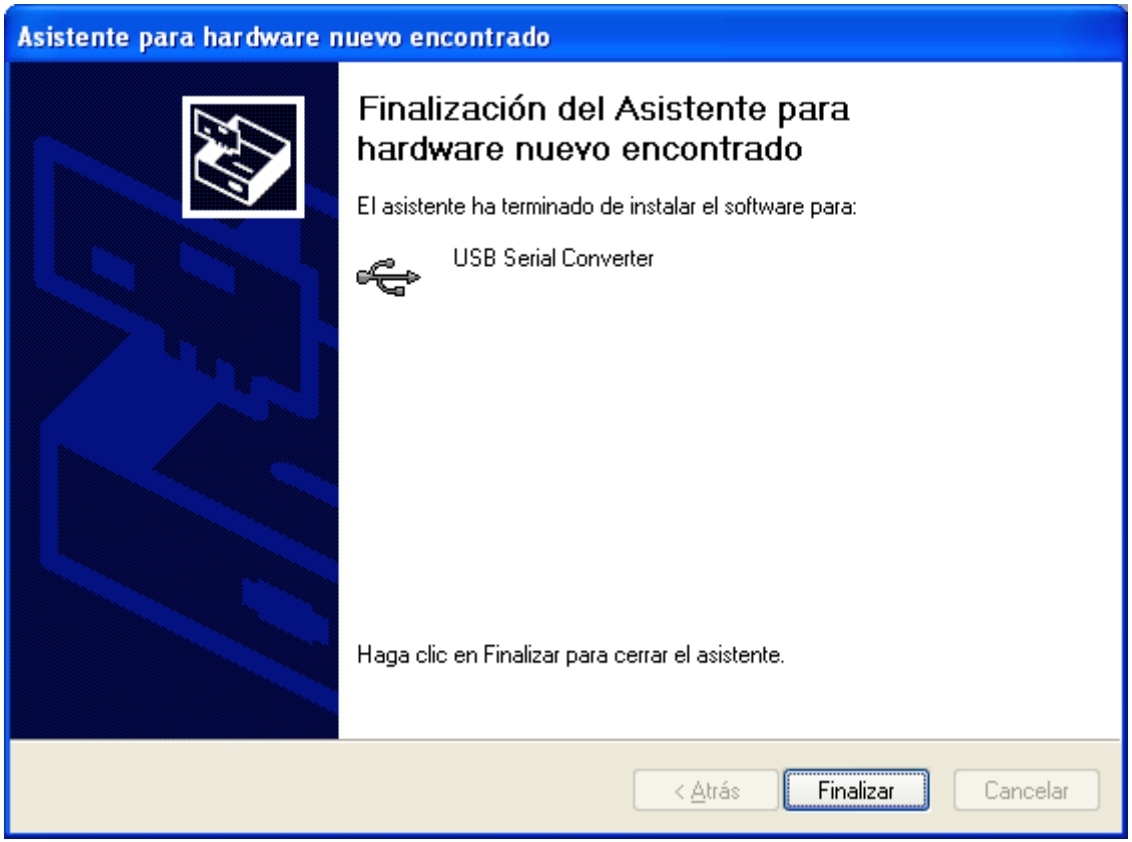
Instalación USB



Estará en Arduino en C\archivos de programa\arduino\drivers\FTDI

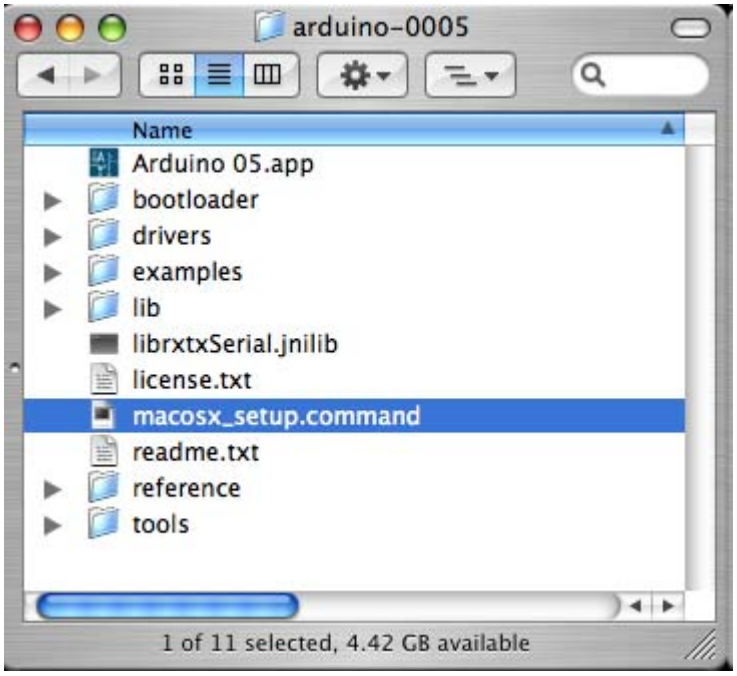
Instalación USB





Instalación del USB Serial converter terminado

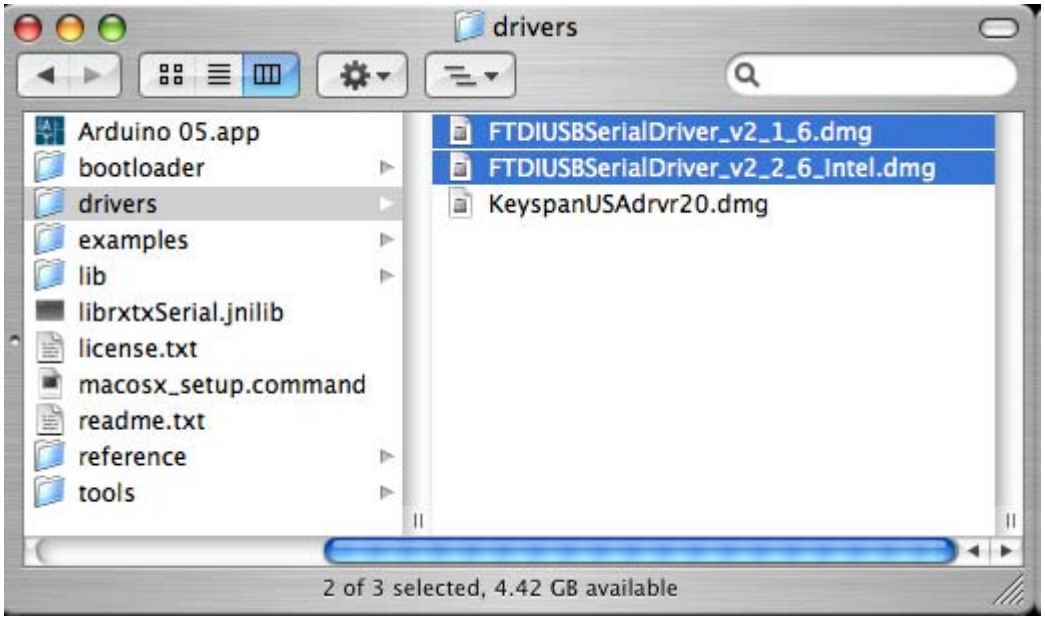
Mac OS X (10.3.9 o posteriores)



Después de descargar el IDE, ejecuta el macosx_setup.command. Corrige el permiso en unos ficheros para su uso con el puerto serie y le pedirá su contraseña de sistema. Puede que tengas que reiniciar el sistema después de ejecutar este script

SOLO USAR ESTO CON ARDUINOS ANTIGUOS, ACTUALMENTE OBSOLETO

Mac OS X (10.3.9 o posteriores)

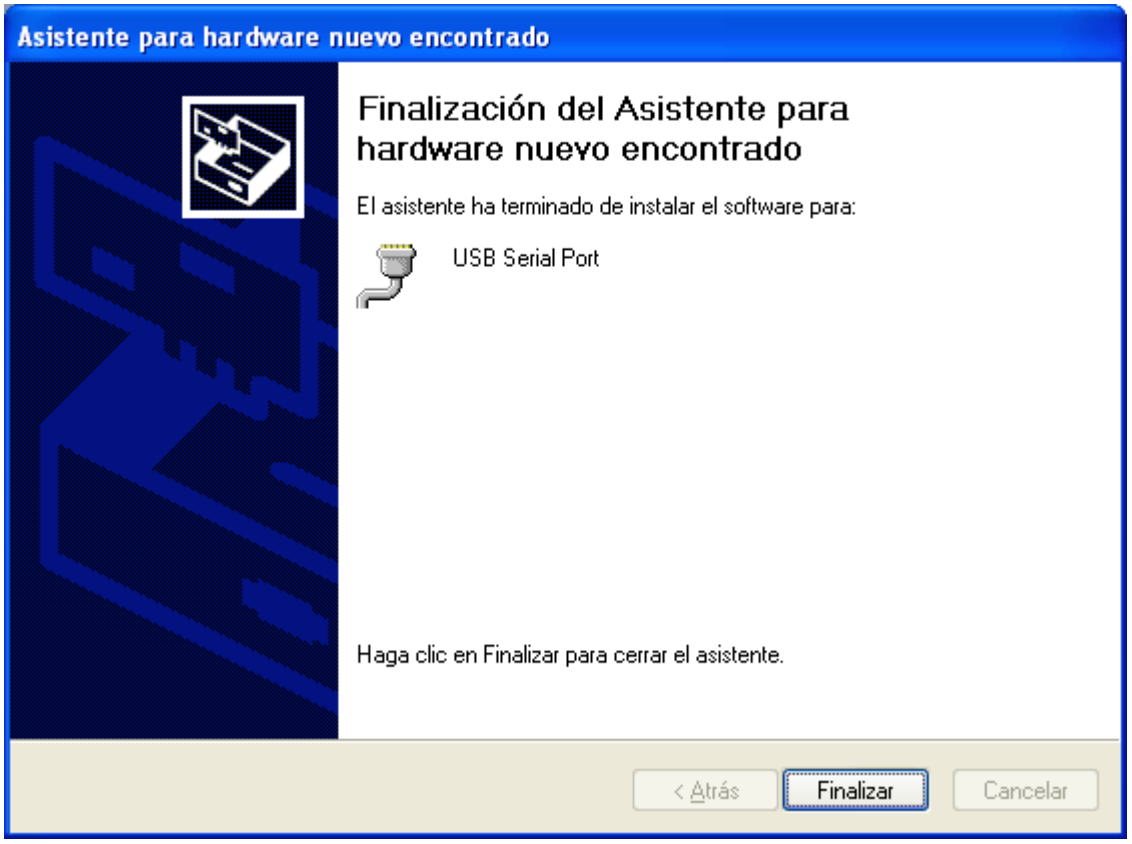


FTDIUSBSerialDriver_v2_1_9.dmg
(PPC) Macs como Powerbook, iBook, G4 or G5

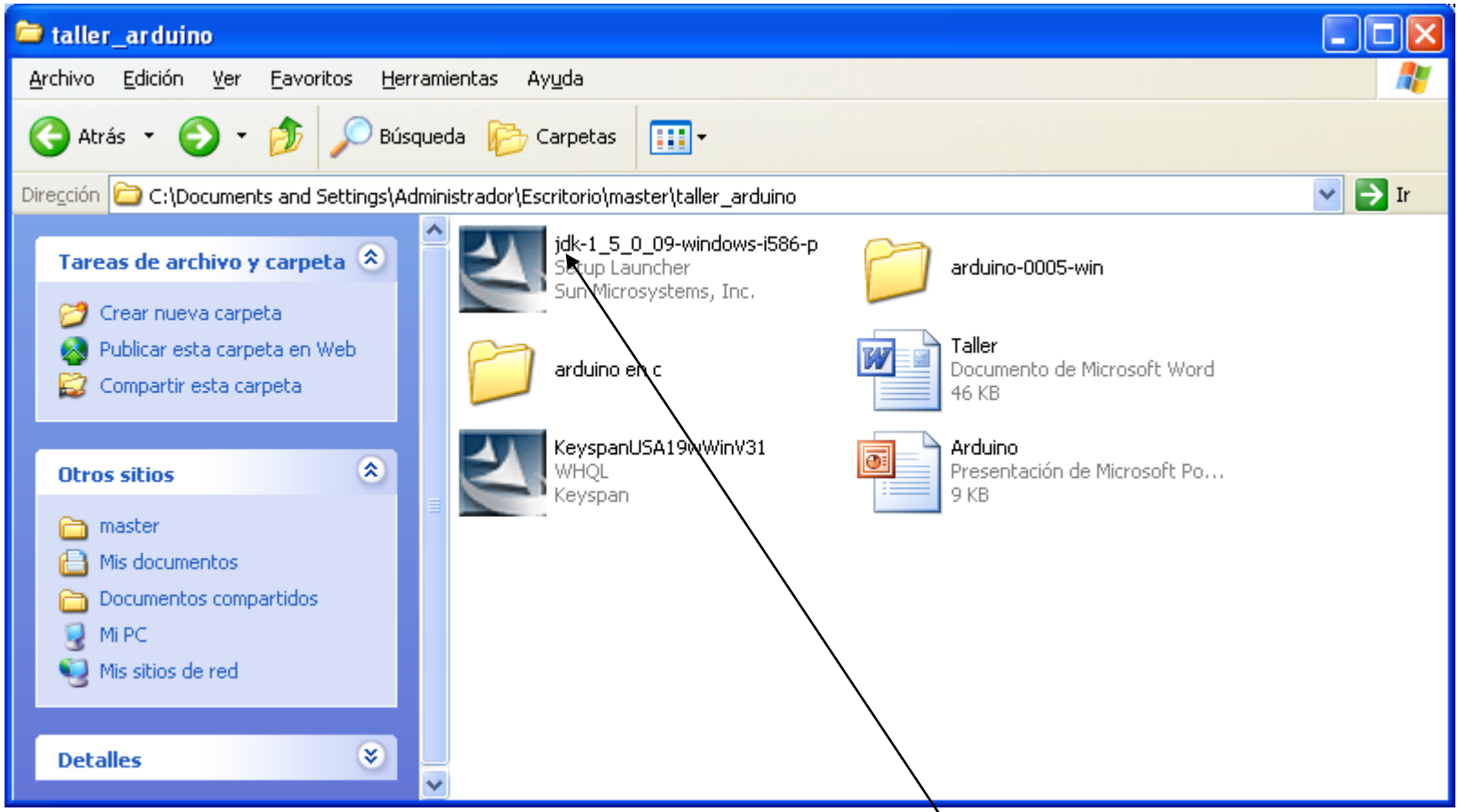
FTDIUSBSerialDriver_v2_2_9_Intel.dmg
(Intel) Macs como MacBook, MacBook Pro, or Mac Pro

Instalación USB

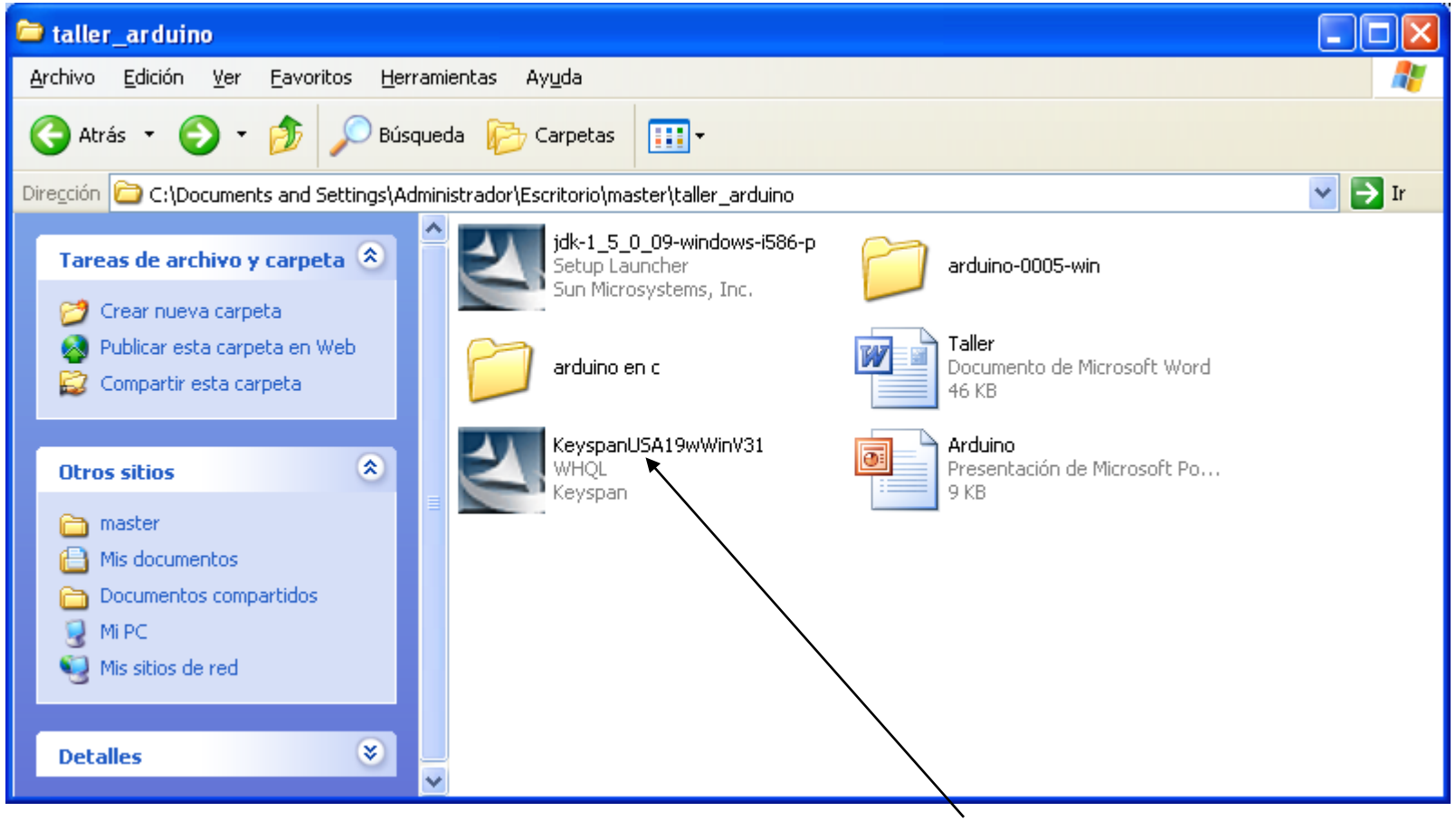
Nos pedirá los mismos pasos que hemos visto para el USB serial PORT



Instalación del USB Serial Port terminado



1º **Instalar el JDK**
Maquina Virtual Java de SUN
(ya que el programa de arduino esta desarrollado en Java y necesita de este



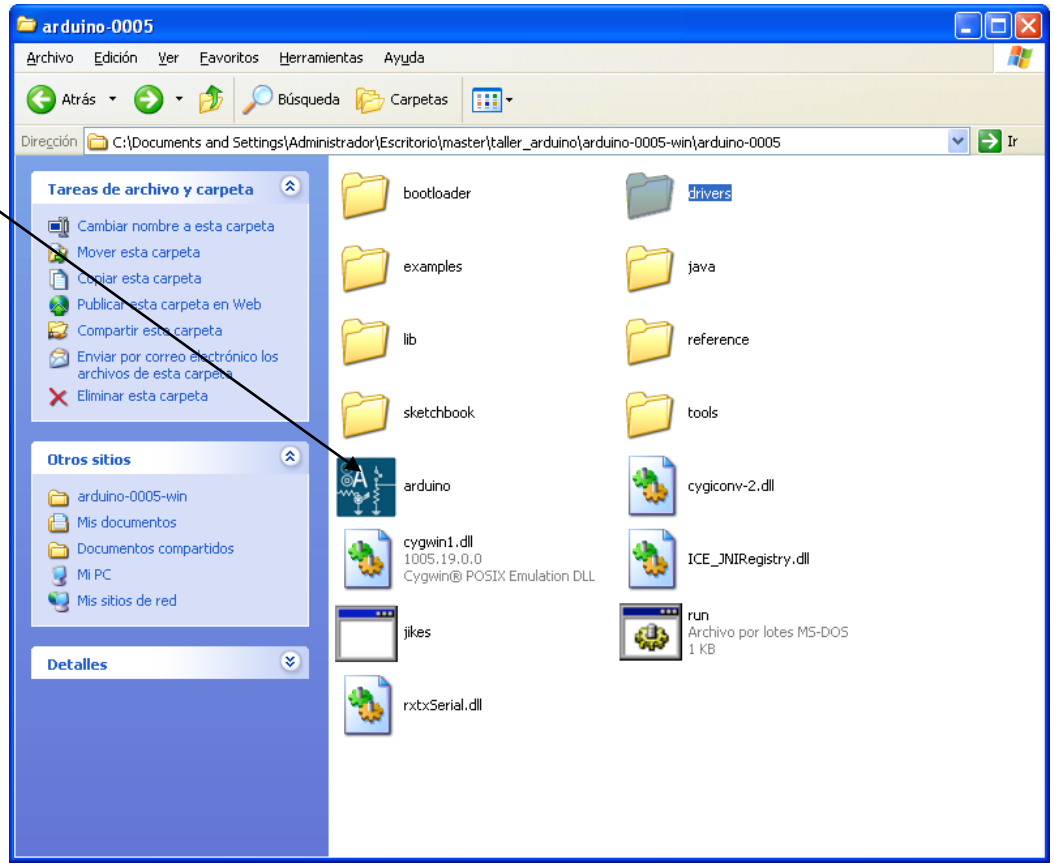
2º **Instalar el KeySpan**
Para que funcione el puerto USB con Arduino

SOFTWARE

INSTALACIÓN DEL ENTORNO DE PROGRAMACIÓN

Una vez descargado, para comenzar a trabajar con el entorno de desarrollo en Windows, tan sólo es necesario descomprimir el contenido del fichero comprimido en una carpeta de nuestro PC. Una vez descomprimido tan sólo es necesario ejecutar el fichero *"Arduino.EXE"*. **(La carpeta arduino del programa debe estar en Archivos de programa para que funcione correctamente)**

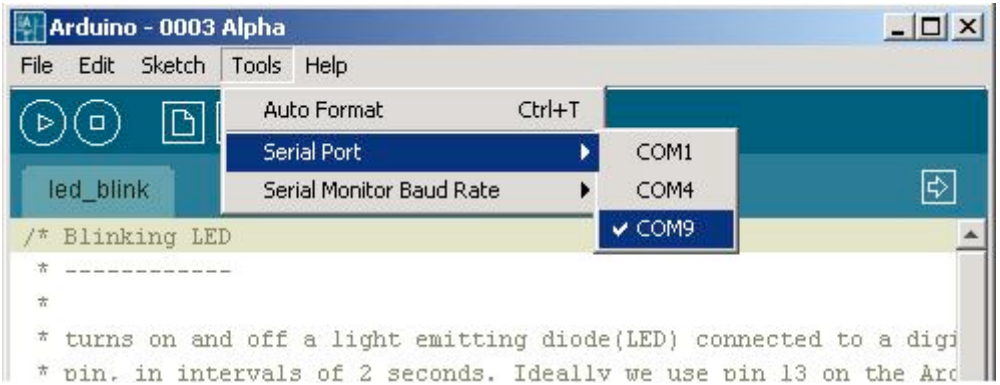
Arduino.EXE



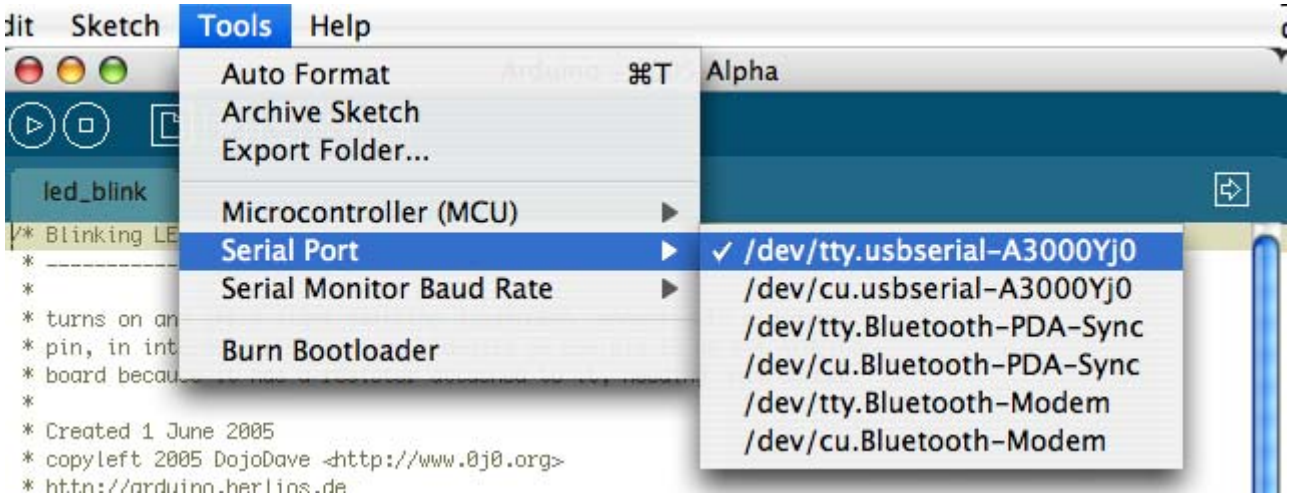
CONFIGURACIÓN DE LAS COMUNICACIONES

Lo primero que tenemos que hacer es configurar las comunicaciones entre la placa Arduino y el PC. Para ello deberemos abrir en el menú *"Tools"* las opciones *"Serial Port"* y *"Serial Monitor Baud Rate"*.

En la primera de las dos opciones deberemos seleccionar el puerto serie al que está conectada nuestra placa. En Windows el puerto será **COM1** o **COM2** para la placa serie, **COM3**, **COM4** … para la placa USB (o para la placa serie conectada mediante un adaptador serie-USB). Si se utiliza un adaptador serie-USB el nombre puede variar.



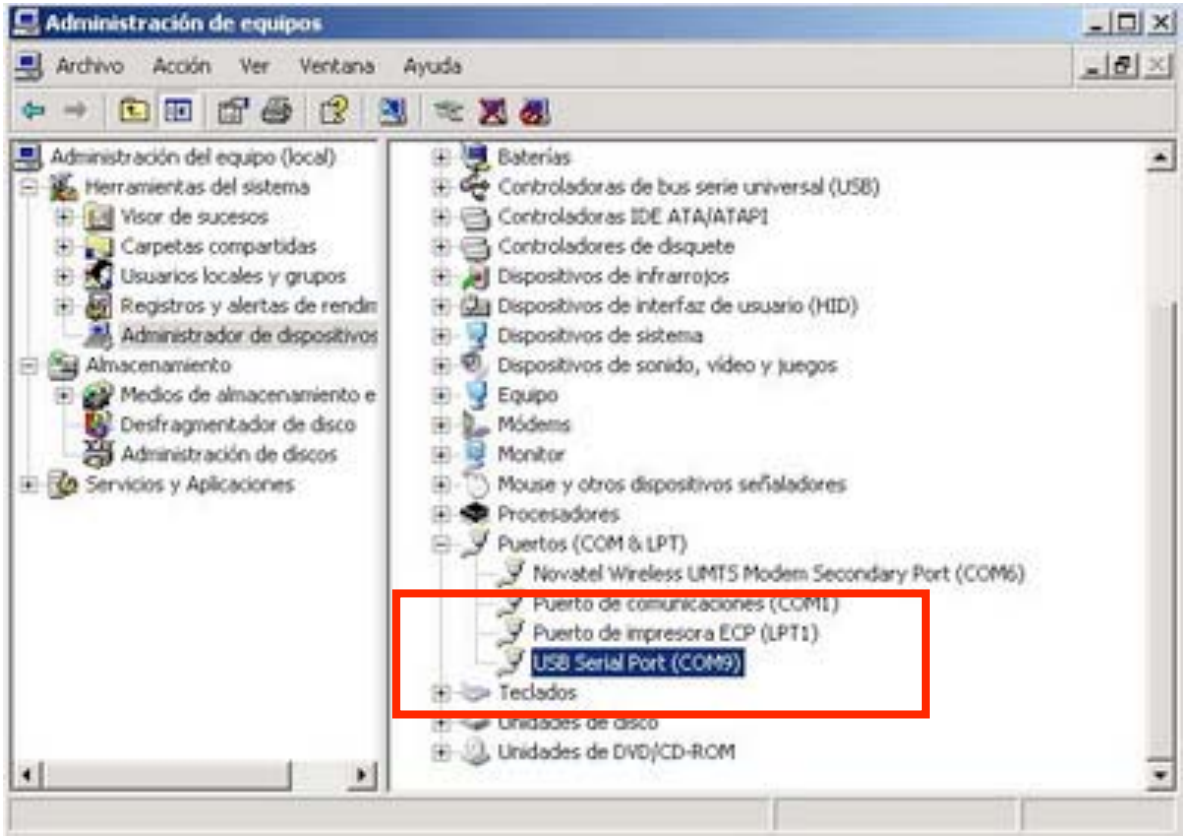
Mac OS X (10.3.9 o posteriores)



En MAC, debería ser algo como **/dev/cu.usbserial-1B1** para USB board, o algo como **/dev/cu.USA19QW1b1P1.1** si estamos utilizando el adaptador Keyspan como adaptador para el puerto de serie.

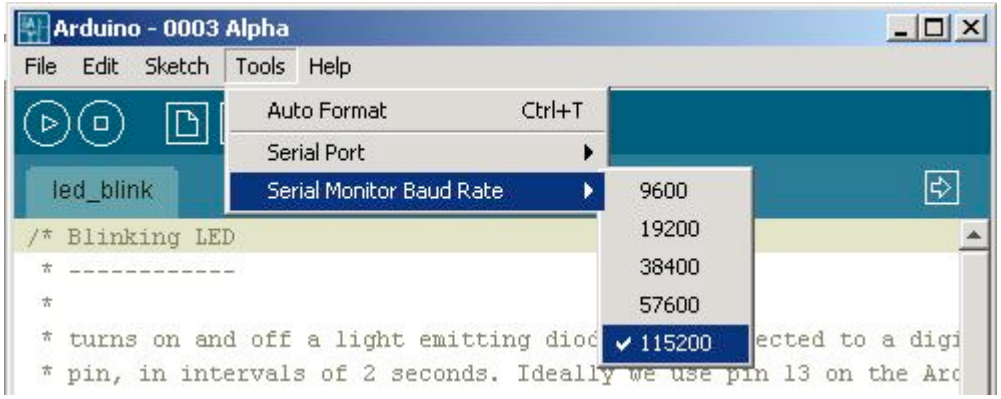
Configuración del puerto serie

En Windows, si desconocemos el puerto al que está conectado nuestra placa podemos descubrirlo a través del “Administrador de dispositivos”.

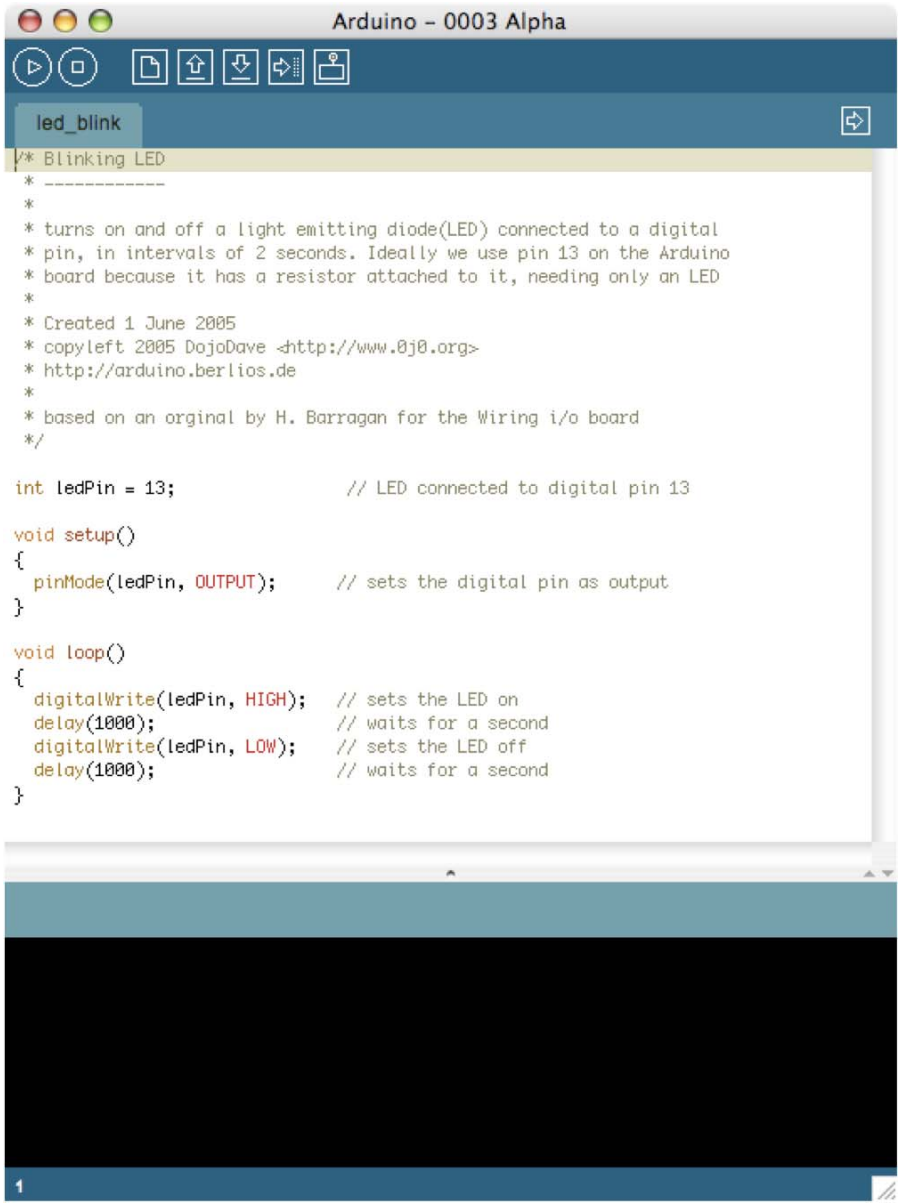


Configuración de la velocidad

También debemos configurar la velocidad a la que la placa y el PC se comunican. Esto lo hacemos desde el menú “Serial Monitor Baud Rate”. El valor por defecto es de 115200 baudios.



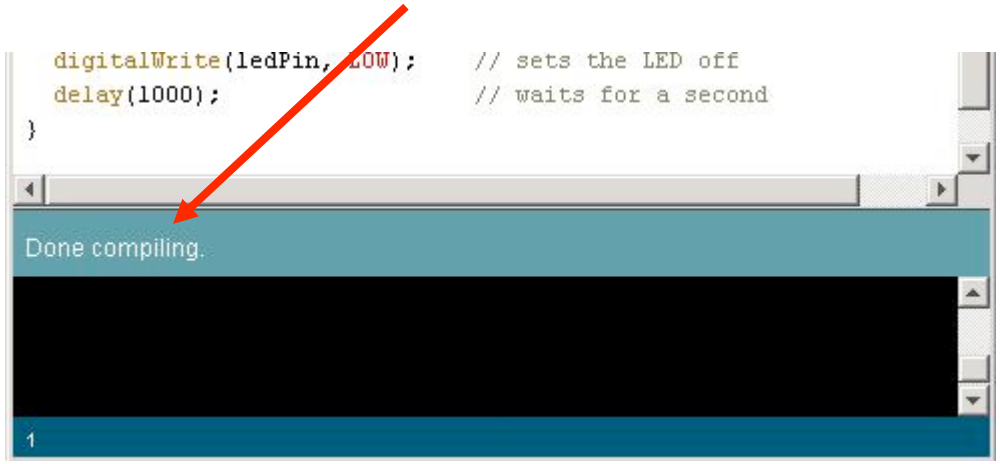
http://www.arduino.cc/en/Main/Software



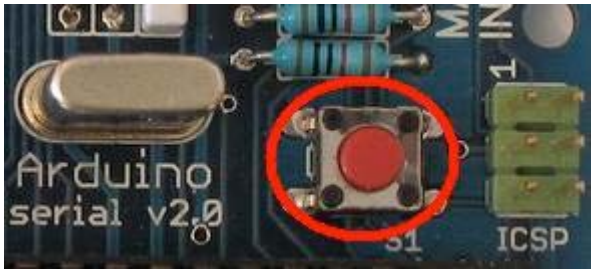
ENVIANDO EL PROGRAMA DE EJEMPLO A LA PLACA ARDUINO

Primero comprobamos que el código fuente es el correcto. Para ello pulsamos el botón de verificación de código que tiene forma de triángulo inclinado 90 grados.

Si todo va bien deberá aparecer un mensaje en la parte inferior de la interfaz indicando “Done compiling”.



Una vez que el código ha sido verificado procederemos a cargarlo en la placa. Para ello tenemos que pulsar el botón de reset de la placa e inmediatamante después pulsar el botón que comienza la carga.

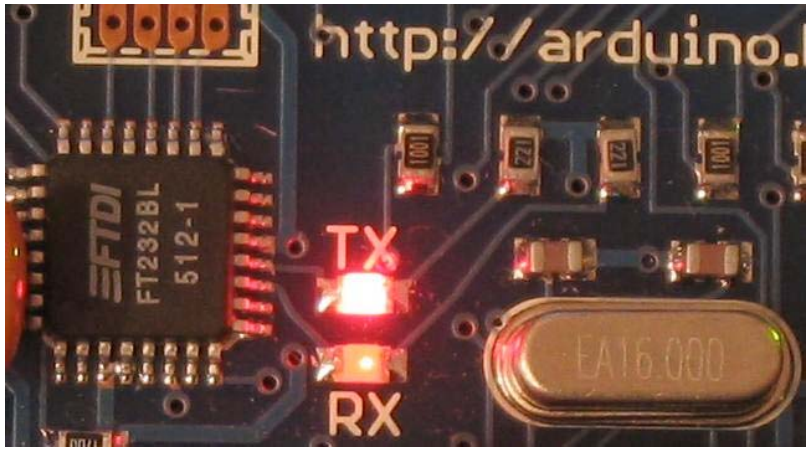


Botón de reset (LIMPIAR EL PROGRAMA ANTERIORMENTE ENVIADO)

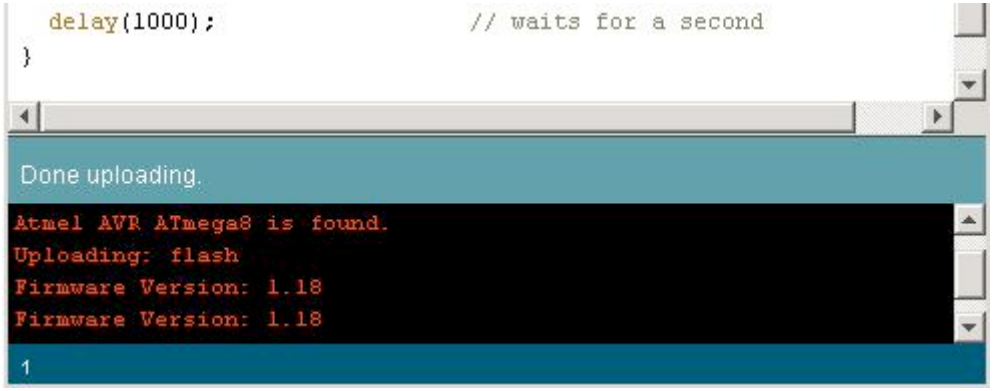
Los modelos **Duemilanove** no hace falta limpiar el programa

ENVIANDO el programa a la placa

Durante la carga del programa, en la placa USB, se encenderán los LED que indican que se están enviando y recibiendo información por el puerto serie: TX/RX.



Si todo se ha realizado correctamente debe aparecer un mensaje como el que se muestra a continuación:



El programa se ha cargado correctamente en la placa

LENGUAJE

Estructura:

Arduino tiene dos estructuras una “setup()” y “loop()”

• La función setup() es llamada justo en el momento en que el programa comienza. Se utiliza para inicializar variables, definir los modos de entrada o salida de los pines, indicar librerías, etc.

•Después de crear la sección setup(), que inicializa y asigna los valores iniciales, la sección loop() hace precisamente lo que su nombre indica en inglés(bucle), y se repite continuamente, permitiendo que tu programa mute y responda. Se usa para controlar de forma activa la tarjeta Arduino. El tiempo del bucle, varía según el número de instrucciones que contenga. Y se puede conocer y controlar con las funciones de temporización (millis(), delay(), etc).

void setup()

void loop()

Estructura: **Ejemplo**

```
int ledPin = 13;          Creo un variable de tipo entero que la denomino
                          ledPin y le asigno el valor 3,

void setup()
{
  pinMode(ledPin, OUTPUT);      // le digo con pinMode, que el pin 13 será una salido en mi placa
}

void loop() // comienza el bucle que no parará
{
  digitalWrite(ledPin, HIGH);   // le digo con digitalWrite que el pin 13 se active = HIGH
  delay(1000);                  // le digo que se espere un segundo con delay + 1000 ( milisegundos)
  digitalWrite(ledPin, LOW); // le digo con digitalWrite que el pin 13 se apague = LOW
  delay(1000); // le digo que se espere un segundo con delay + 1000 ( milisegundos)
}
```

VARIABLES

•char (caracter)
Tipo de datos para definir caracteres (ASCII), símbolos tipográficos tales como A, d, y $. Cada tipo char ocupa un byte (8 bits) de memoria y debe ser delimitado por comillas sencillas

```
char var
char var = 'val'
```

•int (entero corto)
El tipo entero es un tipo de datos que almacena números, y los almacena en campos de 2 byte (16 bits), tomando valores negativos y positivos lo que nos da un rango de -32,768(- 2^15) a 32,767 (2^15 - 1).

```
int ledPin = 13;
```

•long (entero largo)
El tipo de dato Long o tipo de entero largo almacena un número de rango extendido, y los almacena en campos de 4 byte (32 bits), tomando valores negativos y positivos lo que nos da un rango de -2,147,483,648 (-2^31) a 2,147,483,647 (2^31-1).
Sin signo, toma valores de 0 a +4,294,967,295 ¿es así en Aduino?

•boolean (lógica)
Tipo de datos para valores Booleanos de verdadero (true) o falso (false). Es común usar valores de tipo boolean con la estructuras de control y para determinar el flujo o secuenciación de un programa.

```
boolean var = true;
```

•Byte
http://www.arduino.cc/es/Referencia/Byte

•Array (vector)
http://www.arduino.cc/es/Referencia/Vector

•String
Tipo de valor de cadena de caracteres

Constantes

Las constantes son variables predefinidas en el sistema. Son usadas para hacer que los programas sean más fáciles de leer. Las englobaremos en los siguientes grupos.

HIGH | LOW

Cuando se realiza la lectura o escritura sobre un pin digital, sólo hay dos valores posibles que el pin puede tomar o que se le puede asignar: HIGH y LOW.

•HIGH = equivale a 5 Voltios
•LOW = equivale a 0 Voltios

INPUT | OUTPUT

Los pines digitales pueden ser usados tanto como en modo **INPUT (entrada)** o modo **OUTPUT (salida).** Dichos valores, representan precisamente lo que su significado en inglés indica.

Funciones

Pines digitales

•pinMode(pin, mode)
Configura el pin especificado para que se comporte como una entrada (input) o una salida (output).

pinMode(5, OUTPUT); // asigna al pin digital como salida

•digitalWrite(pin, value)
Asigna el valor de salida HIGH o LOW al pin especificado.
digitalWrite(5, HIGH);

•int digitalRead(pin)
Lee o captura el valor de entrada del pin especificado, dará valores HIGH o LOW

```
int ledPin = 13; //
int inPin = 7; // pulsador conectado a pin digital 7
int val = 0; // variable para almacenar el valor de captura o lectura

void setup() {
pinMode(ledPin, OUTPUT); // asigna al pin digital 13 como modo pin de salida
pinMode(inPin, INPUT); // asigna al pin digital 7 como modo pin de entrada
}

void loop() {
val = digitalRead(inPin); // lee o captura el valor de entrada del pulsador
digitalWrite(ledPin, val); // asigna el valor capturado al LED
}
```

Funciones

Pines Analógicos

int analogRead(pin)

Lee o captura el valor de entrada del especificado pin analógico, la tarjeta Arduino realiza una conversión analógica a digital de 10 bits. Esto quiere decir que mapeará los valores de voltage de entrada, entre 0 y 5 voltios, a valores enteros comprendidos entre 0 y 1024.

Ejemplo

```
int ledPin = 13; // LED conectado a pin digital 13
int analogPin = 3; // potentiómetro conectado a pin analógico 3
int val = 0; // variable para almacenar el valor capturado
int threshold = 512; // valor de disparo o umbral (1024/2)

void setup() {
pinMode(ledPin, OUTPUT); // asigna modo salida el pin digital 13
 }

void loop() {
val = analogRead(analogPin); // captura el pin de entrada
if (val >= threshold) {
digitalWrite(ledPin, HIGH); // enciende el LED
} else {
digitalWrite(ledPin, LOW); // apaga el LED
}
 }
```

•analogWrite(pin, value)- PWM **analogWrite** --->Con enlace a PWM **PWM**

Funciones

Pines Analógicos

analogWrite(pin, value)

Escribe o asigna un valor analógico (señal PWM) a pines 9,10 y 11 ¿?. Se puede usar para encender un Led e ir variando la intensidad de su brillo o impulsar un motor a distintas velocidades. Después de invocar o realizar una llamada a la función analogWrite, el pin generará una señal estable hasta la próxima invocación o llamada de la función analogWrite (o una llamada a digitalRead o digitalWrite sobre el mismo pin).

Los pines analógicos, al contrario que los pines digitales, no necesitan ser declarados como modo INPUT(entrada) o OUTPUT (salida).

Ejemplo

```
int ledPin = 9; // LED conectado al pin digital 9
int analogPin = 3; // potenciómetro conectado al pin analógico 3
int val = 0; // variable para almacenar el valor de captura o lectura

void setup() {
pinMode(ledPin, OUTPUT); // asigna el pin como modo salida
 }

void loop() {
val = analogRead(analogPin); // lee o captura el pin de entrada
analogWrite(ledPin, val / 4);

// IMPORTANTE los valores de analogRead van desde 0 a 1023, los valores de analogWrite desde 0 a 255
// por eso los dividimos entre 4
}
```

Temporizadores

delay(ms)

Detiene el programa durante una cantidad de tiempo (en milisegundos) especificado mediante parámetro.

delayMicroseconds (milisegundos);

delayMicroseconds(1136);

```
int ledPin = 13; // LED conectado a pin digital 13

void setup() {
pinMode(ledPin, OUTPUT); // asigna al pin digital como pin de salida
}
void loop() {
digitalWrite(ledPin, HIGH); // enciende el LED
delay(1000); // espera un segundo
digitalWrite(ledPin, LOW); // apaga el LED
delay(1000); // espera un segundo
 }
```

Comunicación:

Serial

Se usa para crear la comunicación entre la placa Arduino y una computadora o otros dispositivos.

Normalmente lo declararemos en el : **VOID SETUP(){ }**

Serial.Begin(velocidad en baudios);

Obtiene el rango de valores en bots por segundo (baudios) para la transmisión con el puerto serie-
USB = 9600 baudios

Serial.print(valor que queremos imprimir, tipo de dato);

Imprime el valor de los datos que están entre paréntesis

Serial.print(valor,DEC);

Imprime decimales como cadena ASCII

Serial.print(valor,HEX);

imprime hexadecimales como cadena ASCII

Serial.print(valor,BIN);

imprime BINRIOS como cadena ASCII

Serial.print("valor de la cadena”)

imprime la cadena de caracteres

Serial.println();

Imprime los datos linea a linea. Es más cómodo para ver los datos

Estructuras de control:

•if

if (condición) Descripción La estructura If comprueba si la condición contenida entre paréntesis () se ha cumplido, como por ejemplo si una entrada supera un cierto número. El formato de la comprobación que se realiza en el if es:

```
if (algunaVariable > 50) {
// hacer algo aquí
}
```

•if...else

La estructura if/else da un mayor control sobre el flujo o secuenciación del código que la estructura de control básica if, permitiendo agrupar múltiples comprobaciones juntas.

```
if (pinFiveInput < 500)
{
//hacer cosa A
}
else
{
//hacer cosa B
}
```

Estructuras de control:

•for

Realiza el control sobre una secuencia de repetición. Se compone de tres partes: init (inicializado de la variable local), test (condición) , y update (actualización del valor la variable local), cada parte debe ser separada por punto y coma ";". El bucle continua hasta que la condición establecida se cumple (es verdad) o no (falsa). Es útil cuando se usa en combinación con vectores y operar sobre grupo de datos/pines.

```
for (int i=1; i <= 8; i++){
digitalWrite(i, HIGH); // declaración usando el valor de la variable local i;
 }
```

•switch case

•Como la estructura de control "If", la estructura de control "switch case" ayuda en el control del flujo o secuenciación de los programas. Permite hacer una lista de "casos" posibles dentro de un bloque delimitado por paréntesis, en el cual arduino encontrará el caso más idóneo y lo ejecutará.

```
•switch (var) {
•case 1: //hacer algo cuando var == 1
• break; case 2: //hacer algo cuando var == 2
•break;
•default: // si ninguna de las anteriores, hacer la parte de default }
```

•While

Realiza un bucle de forma continuada hasta que la expresión contenida dentro de los paréntesis () deja de ser verdadera. Es útil para crear bucles propios, pero asegurando el seguimiento de alguna de las variable usadas para parar o salir del bucle, si esa es la intención.

```
var = 0;
while(var < 200){
//hacer que algo se repita 200 veces var++;
}
```

Comentarios

// (comentarios de línea)

/* */ (comentarios de multi-línea)

Los comentarios son usados para informar sobre la forma en que el programa funciona. No serán compilados, ni serán exportados al procesador. Son útiles para entender lo que cierto programa o informar a otras personas que puedan hacer uso de el.

Contenidos desarrollados por: David Cuartielles

http://www.arduino.cc/es/
http://www.arduino.cc