Hagamos nuestros propios Arduinos

Como hemos visto a lo largo de las clases, Arduino es una placa extraordinariamente difundida gracias a sus múltiples virtudes, que todos conocemos. Si bien no es demasiado costosa en origen, en nuestro país alcanza un precio bastante elevado, por razones impositivas y comerciales.

Tal vez este costo no sea demasiado si lo consideramos como un kit de desarrollo, pero, cuando el destino sea embeber el microcontrolador directamente en una obra, estamos pagando por algunas de las prestaciones que Arduino incluye, muy útiles a la hora del desarrollo (por ejemplo el chip FTDI que le permite comunicarse con el puerto USB de nuestra computadora) pero completamente desaprovechadas en los dispositivos ya en funcionamiento, embebidos en algún proyecto.

Esta desventaja se multiplica cuando necesitamos una cantidad considerable de micros, por ejemplo en proyectos de robótica colaborativa o en aquellos que implican muchos sensores distribuidos en un gran espacio, etc.

La idea, entonces, es poder utilizar un microcontrolador AVR de bajo costo, el Atmega8, micro original del primer Arduino, sin componentes adicionales innecesarios, para poder correr en él los mismos programas que corren en las placas originales, sacrificando en algunos casos alguna prestación y en la mayoría de los casos sin diferencia perceptible alguna con el Arduino “oficial”.

Como ya dijimos, una de las cosas que encarece a las versiones actuales de Arduino es el chip FTDI, responsable de comunicar el puerto serial existente en el micro con el puerto USB de la computadora de desarrollo, principalmente para poder programar el micro desde ésta.

En nuestro proyecto, eliminaremos esta parte, programando los micros por medio de un dispositivo programador.

Si bien puede utilizarse cualquier programador, incluyendo uno muy simple, que puede construirse con sólo un par de resistencias y se conecta al viejo puerto paralelo de las PC, propondremos la creación de uno en particular, que es mi favorito por su velocidad, confiabilidad y, sobre todo, porque puede conectarse al puerto USB tanto de una Mac como de una PC bajo Windows o Linux.

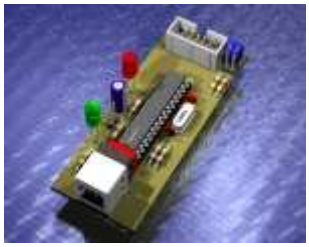
Se trata del USBASP, basado él mismo en un Atmega8 y de diseño Open Hardware. Todo el software que utilizaremos será también, por supuesto, Open Source y/o Freeware.

Cabe aclarar que en el costo mencionado en el título no estamos incluyendo el costo del programador, puesto que se construye una sola vez. De cualquier modo, el valor de sus componentes nos supera los $ 30, cifra que luego seguramente amortizaremos aprovechándolo en muchos proyectos a lo largo del tiempo.

También será necesario, la primera vez, programar el micro incluido en el programador para que éste pueda funcionar, pero esto lo haremos en clase o con un compañero que ya tenga el suyo.

Programador USBASP
Página Oficial: http://www.fischl.de/usbasp/ Hay allí múltiples versiones del mismo proyecto.

La versión que nosotros sugerimos en clase, está en esa misma página:



usbasp_gr.rar
by J.A. de Groot

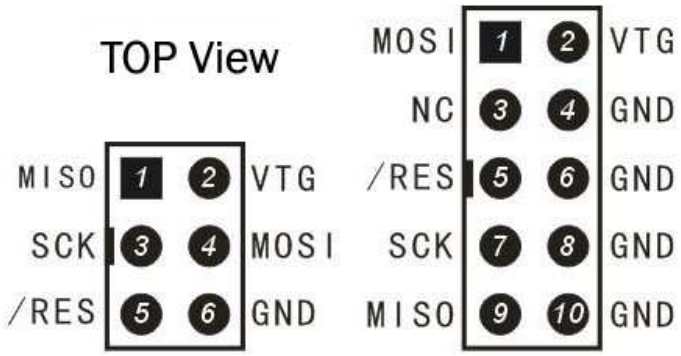
The board is single sided (EAGLE format), measures 3 by 8 cm and uses only regular components.

El archivo rar original linkeado arriba contiene el esquemático y el PCB en formato Eagle SIN los diodos Zener de 3.6 V previstos en el circuito original.

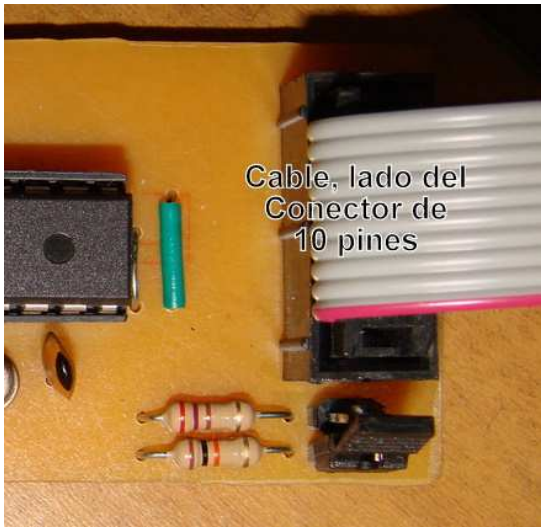
El archivo modificado con el agregado de dos diodos Zener que lo hacen compatible con el puerto USB de notebooks y PCs más nuevas, puede descargarse en: http://miguelgrassi.com.ar/mecatronica/usbasp/Usbasp_conZener.rar e incluye el esquemático y el Board del PCB en formato Eagle y en imágenes ".png" a 600 dpi. El programa para diseño de PCB Eagle, versión Light (Freeware) puede descargarse en http://www.cadsoftusa.com/freeware.htm

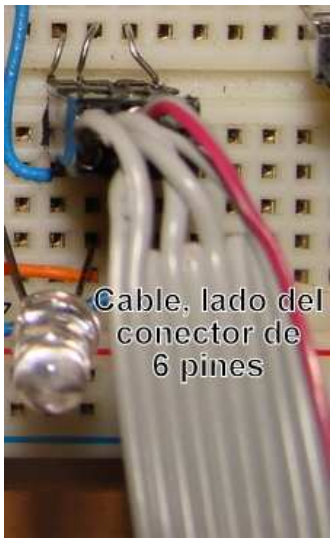
Cable

Pinout del adaptador de 10 pines a 6 pines del ISP de AVR:



Como vimos en clase, se usa un cable plano de 10 conductores de los cuales se toman sólo 6, descartando el pin número 3 (NC-No Conectado) y tres cualesquiera de los cuatro GND.





Jumpers

Ubicación de los tres Jumpers en el programador. El número 2 (Self Programming) se utilizará sólo la primera vez (lo haremos ebn clase) para programar el micro del propio programador. Los otros se explican más abajo

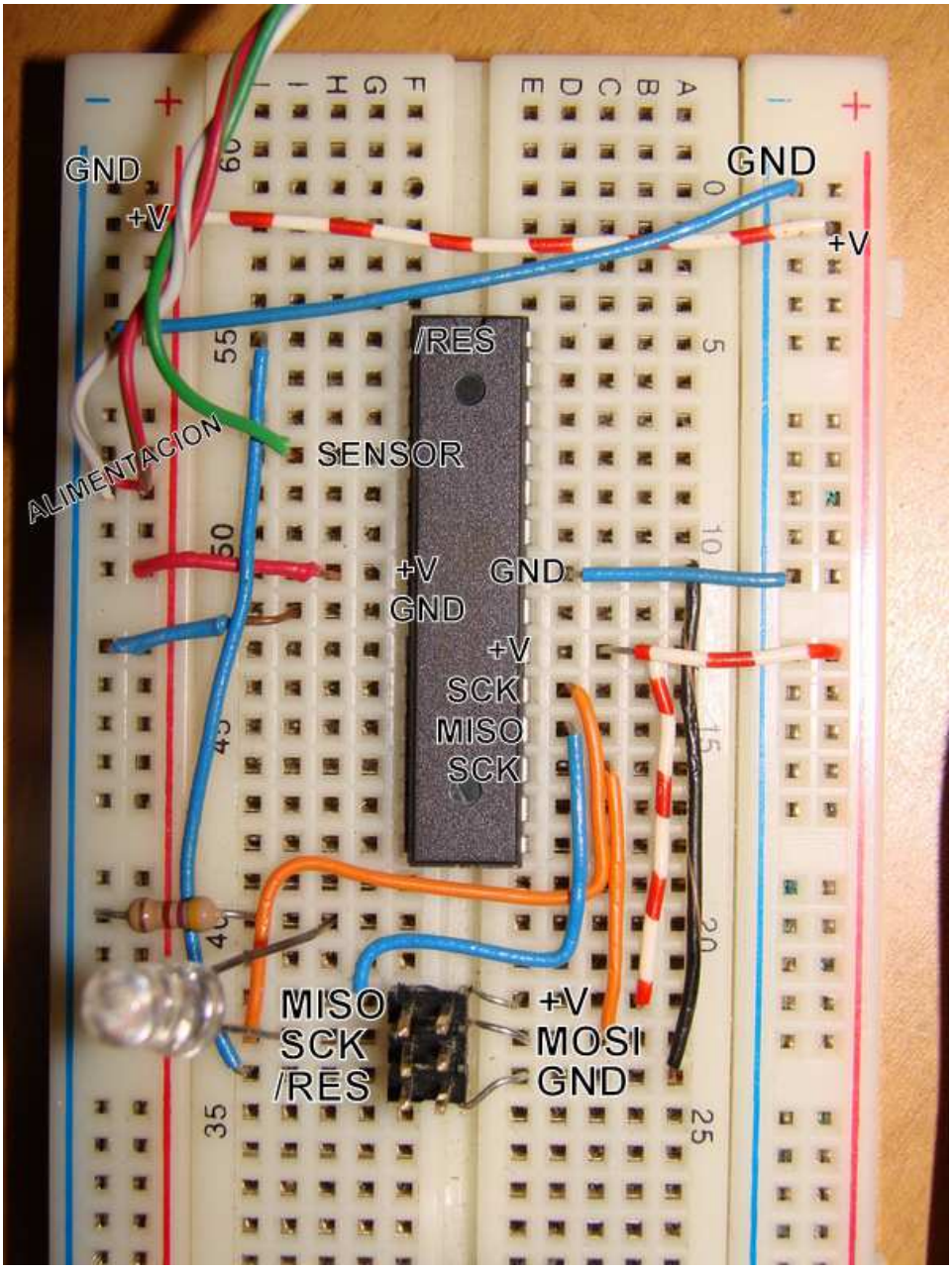
Drivers

Una vez que el programador está armado y se conecta por primera vez a una PC, Windows nos pedirá los drivers, que se deben descargar previamente de la página del USBASP o directamente de http://www.fischl.de/usbasp/usbasp.2009-02-28.tar.gz

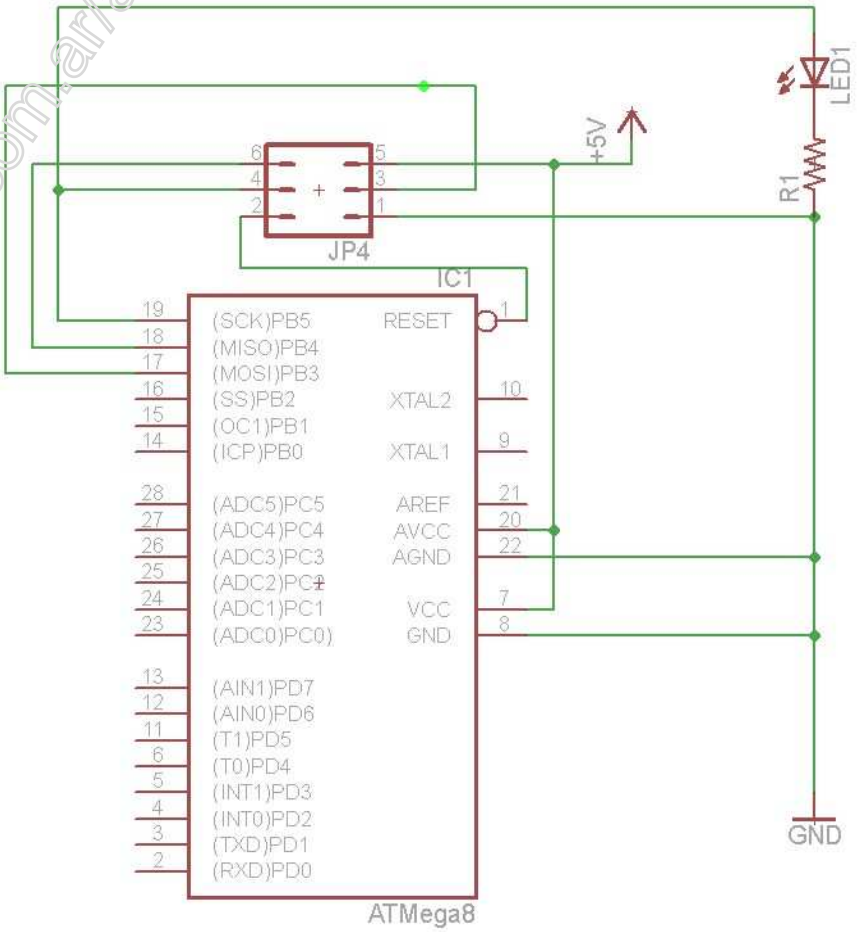
Una vez descomprimido el archivo, los drivers estarán en la carpeta donde lo expadamos, bajo el subdirectorio *usbasp.2009-02-28\bin\win-driver\libusb_0.1.12.1* Esto es necesario sólo Windows, ya que para Mac o Linux no se necesitan descargar drivers

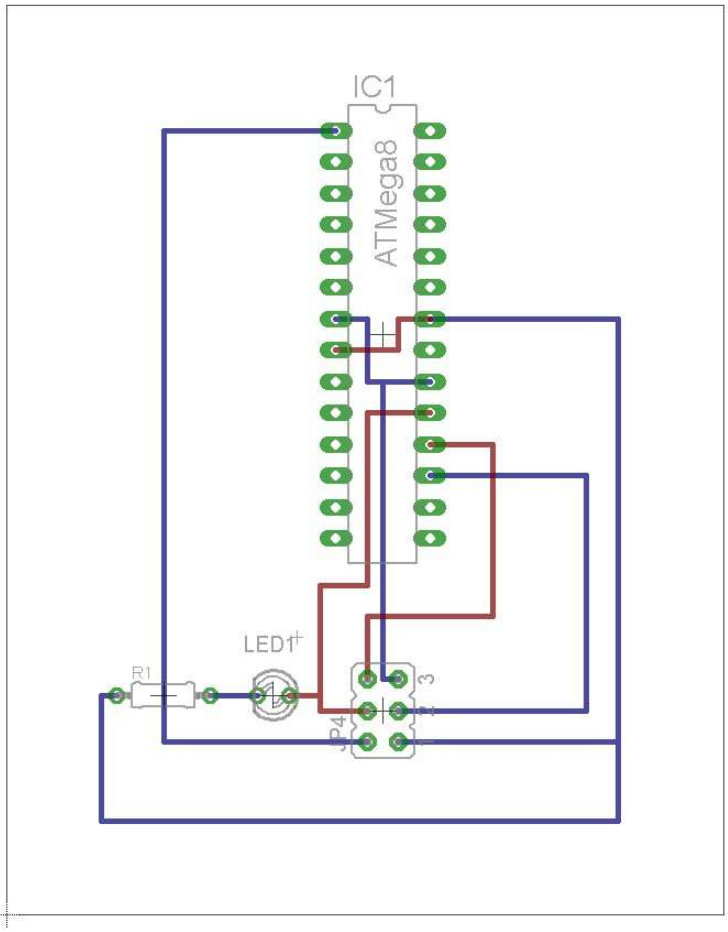
Circuito del Micro a programar

Conexión minimalista de un ATMega8 en Protoboard, con el conector ISP de 6 pines para programarlo. Se ha agregado un LED, conectado al Pin D13, tal como en el Arduino original, una resistencia limitadora de corriente para dicho LED y un cable adicional (el de color verde) a un sensor externo (no mostrado en la imagen) que no intervendrá en este ejemplo



El Protoboard que se muestra corresponde al siguiente Esquema:





Programar el ATmega8 desde el entorno de Arduino

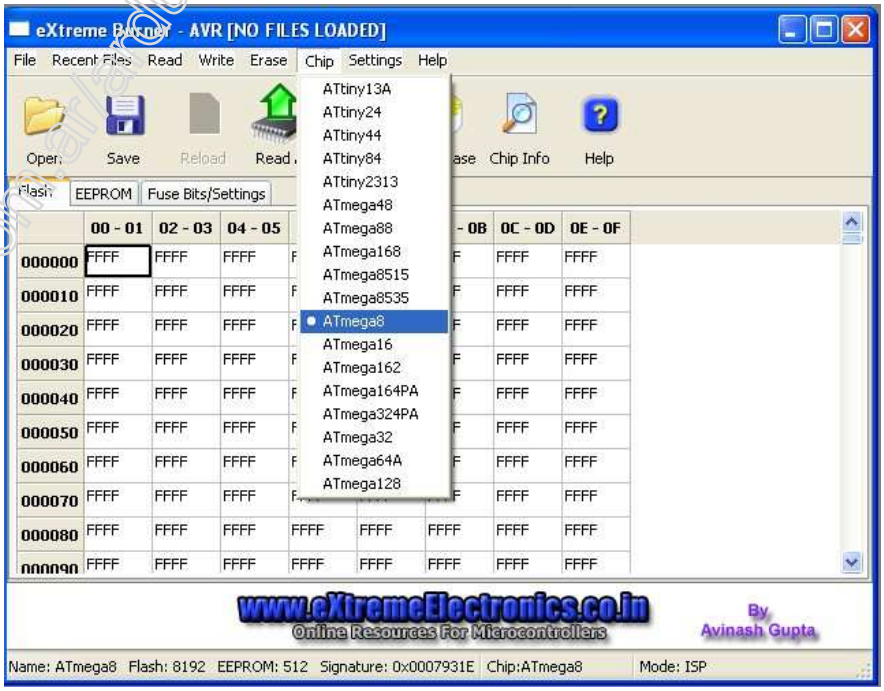
1. Escribir el programa en el entorno de Arduino:

- Abrir un programa ya escrito o escribir normalmente uno nuevo, tal como si fuera para la placa de Arduino
- Seleccionar en el menú Tools – Board la última opción de la lista: *"Arduino NG or older w/Atmega8"*
- Verificar que el programa compila adecuadamente con el botón que tiene el símbolo de "Play" (primero de la izquierda)
- Seleccionar el botón para descargar el programa al Arduino. Dará un error porque no hay ningún Arduino conectado a la computadora, pero **habrá generado el archivo ".hex" que necesitamos** para programar nuestro micro

NOTA IMPORTANTE: En las versiones 0016 y anteriores de arduino el archivo .hex estará en la misma carpeta que el applet de Arduino. En las versiones 0017 en adelante, lo hará en una carpeta del directorio temporal de Windows. El directorio temporal de Windows puede accederse directamente ejecutando el comando "%temp%" (sin las comillas) en la ventana de ejecución de Windows XP, Vista o 7. Buscar allí una carpeta con el nombre de nuestra aplicación.

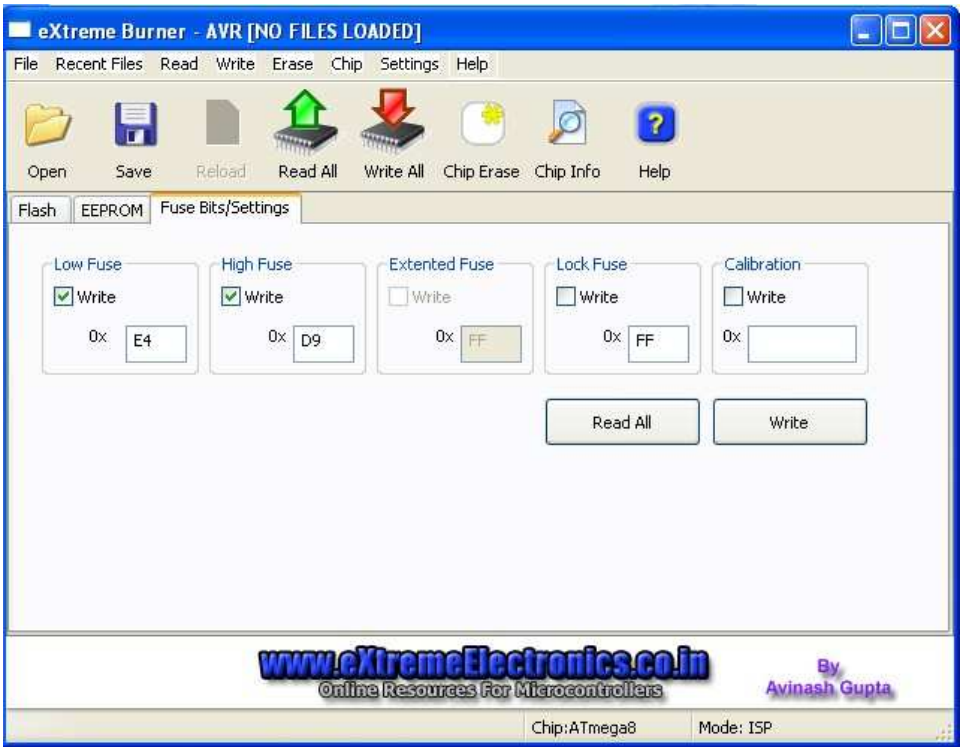
2. Quemar el programa en nuestro micro:

- Montar en un protoboard un micro Atmega8, una tira de 6 pines (3x2), un LED y una resistencia de 300 a 600 Ohms. Conectar las partes según se muestra en la foto de más arriba, y en el correspondiente esquema. Este será nuestro micro "target", es decir el que contendrá finalmente el programa que deseamos crear y ejecutar.
- Conectar el programador a nuestro micro target con el cable de programación de 10 pines a 6 pines ya explicado (ver fotos).
- Colocar o quitar el jumper de "Target Supply" (para alimentar o no el micro del protoboard desde el propio programador) según corresponda. Si el micro está alimentado en forma externa por otra fuente RETIRAR este jumper. Si todo el conjunto se alimentará desde el puerto USB, colocarlo.
- Si nuestro micro target no dispone de un cristal (como es el caso de nuestro ejemplo) colocar el jumper "Slow Clock" en el programador.
- El jumper restante (Self Prog) debe ser removido y no volverá a usarse
- Conectar a nuestra computadora el programador USBASP con los drivers correctamente instalados, como se explica más arriba. La computadora debería detectar el dispositivo como "USBASP"
- Abrir la aplicación Freeware llamada ***eXtreme Burner-AVR*** que previamente se debe haber instalado y que puede descargarse de http://extremeelectronics.co.in/software/BurnerAVR/Setup.exe
- Indicar en la opción "Chip" del menú del eXtreme Burner el micro Atmega8 que estamos por programar (Ver gráfico en página siguiente)



- Hacer Clic en el botón “Read All” (flecha verde) para verificar que el chip está correctamente conectado. De ser así el programa deberá leer nuestro target e indicarlo

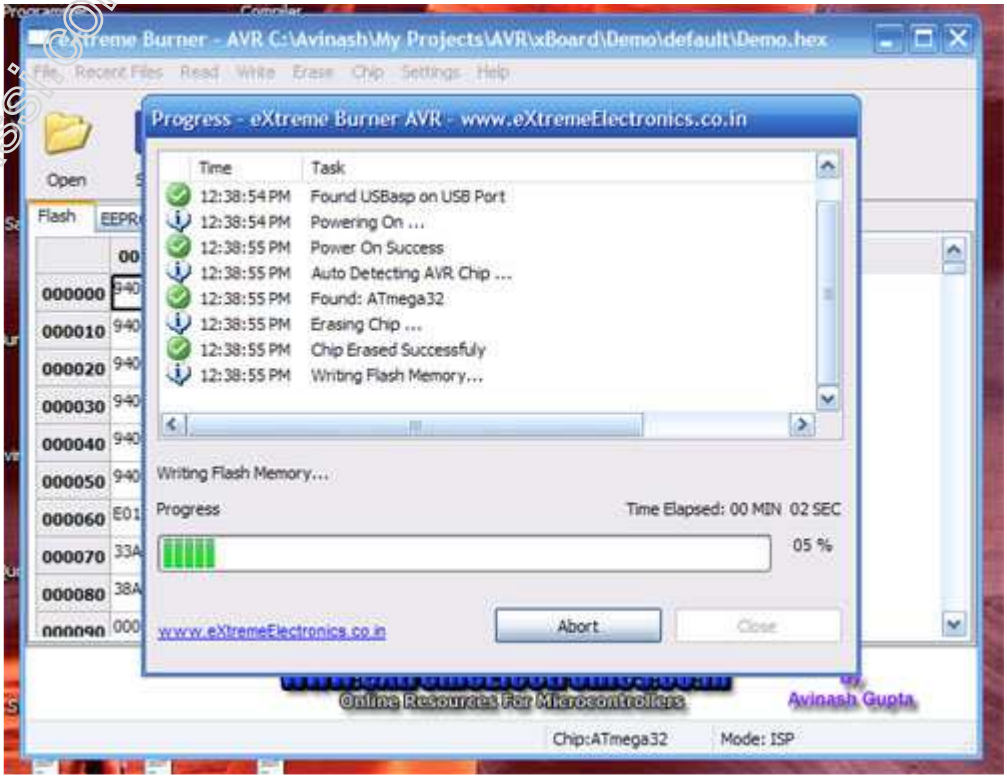
- Si todo funcionó correctamente, seleccionar la solapa “Fuse Bits/Setting” del eXtreme y escribir los valores E4 y D9 en los fusesLow y High respectivamente, marcando los checks box “Write” correspondientes



- Clickear el botón Write para grabar los Fuses en nuestro micro. Aguardar la confirmación y una vez grabados, desmarcar los check boxes para evitar errores futuros.

- Seleccionar la solapa “Flash” del eXtreme y, usando clickeando el botón Open, abrir el archivo .hex que ha generado el entorno de Arduino y que previamente hemos localizado según las instrucciones de arriba. En la ventana Flash veremos el contenido (en valores hexadecimales) de nuestro programa

- Grabar el programa al micro con la opción Write All (Flecha Roja)



- Nuestro programa debería cargarse al micro y comenzar a ejecutarse inmediatamente

Compatibilidad de las conexiones con Arduino

Para conocer como conectar distintos elementos correctamente a nuestro micro y que se mantenga la nomenclatura de Arduino en los programas es necesario utilizar la siguiente equivalencia entre los pines: