Configurar AVR5 Studio para *flashear* el código compilado.

Creación de Herramientas Externas y botones.

Por: Valentin

Configurar avrdude para flashear en el chip Atmega el programa, empleando la placa Arduino Uno.

Una vez que el programa compila sin errores, podemos pasar a cargarlo en el chip de la placa. Afortunadamente, se puede hacer con el mismo hardware que se utiliza en el IDE de Arduino.

Arduino emplea la utilidad avrdude para flashear a través de serie, por lo que nosotros vamos a hacer lo mismo. Sólo tenemos que decirle a AVR Studio las opciones que tiene que emplear.

Por otro lado, suponiendo que estamos programando a través de una conexión USB-Serie, como hacen las placas Arduino, **tenemos que apuntar cuál es el puerto COM está empleando el entorno de desarrollo de Arduino**, para emplear nosotros el mismo, en AVR5. Para ello, solo hay que arrancar dicho entorno, y acceder al menú "Tools/Serial Port", para comprobar qué puerto COM está seleccionado.

Construcción de una "External Tool" para cargar el chip de la placa Arduino.

Método General.

Pasamos a construir una "External Tool" que nos permita programar el chip de Arduino. Las "External Tools" se visualizan, por defecto, dentro del menú desplegable "Tools". Es decir, no se visualizan como un botón de una barra de herramientas.
En el siguiente capítulo aprenderemos a crear dicho botón.

Hacemos clic en el menú **"Tools" -> "external Tools ..."**

Y a continuación hacemos clic en el botón "Add" en la ventana que nos aparece.

En el campo "**Comando**", tenemos que poner la ruta de acceso a avrdude.exe en la instalación de Arduino. Por ejemplo:
C:\arduino-1.0\hardware\tools\avr\bin\avrdude.exe

En "**Argumentos**", de manera general, tenemos que poner lo siguiente:

```
-CC:\arduino-1.0\hardware\tools\avr\etc -v -v -patmega328p -cstk500v1
-P\.COM10 -b57600 -D -
Uflash:w:"$(ProjectDir)Debug$(ItemFileName).hex":i
```

Hay que poner la ruta de acceso a la instalación de Arduino de nuestro ordenador y cambiar el puerto COM, el objetivo de chip, y si fuera necesario, la velocidad (ATMega328 necesita de 57600 baudios, mientras que el ATMega168, más antiguo, necesita 19200).

El resto de las *flags* son exactamente los mismos que utiliza el entorno de desarrollo de Arduino. Los flag "-v" se usan para controlar el nivel de detalle de información suministrada – Se puede jugar con este parámetro.

Se pueden incluir hasta cuatro "-v" s (que registra cada byte de serie transmitido!), pero dos flags "-v"proporcionan suficiente información sin ser abrumadora.

También es posible ver la información generada durante el proceso, en una ventana integrada de AVR5. Para ello seleccionaremos la opción "**Use Output Window**". De lo contrario avrdude abrirá una ventana de comandos y la cerrará en el momento en que finalice su trabajo.

Aclaración: El caso de Arduino 1.0 es diferente a todos los anteriores:

Si tengo instalado **Arduino version 0023**, para realizar la carga, los argumentos de avrdude.exe, son los siguientes
avrdude23 -F -v -pm328p -cstk500v1 -P\\.\COM3 -b115200 -D -Uflash:w:"Master.hex":i

Si tengo instalado **Arduino version 1.0**, para realizar la carga, los argumentos de avrdude.exe, son los siguientes
avrdude -F -v -pm328p -carduino -P\\.\COM3 -b115200 -D -Uflash:w:"Master.hex":i

Es decir, cambiar el parámetro -p, pasando de usar un programador "stk500v1" a otro llamado "arduino". Esto es nuevo.

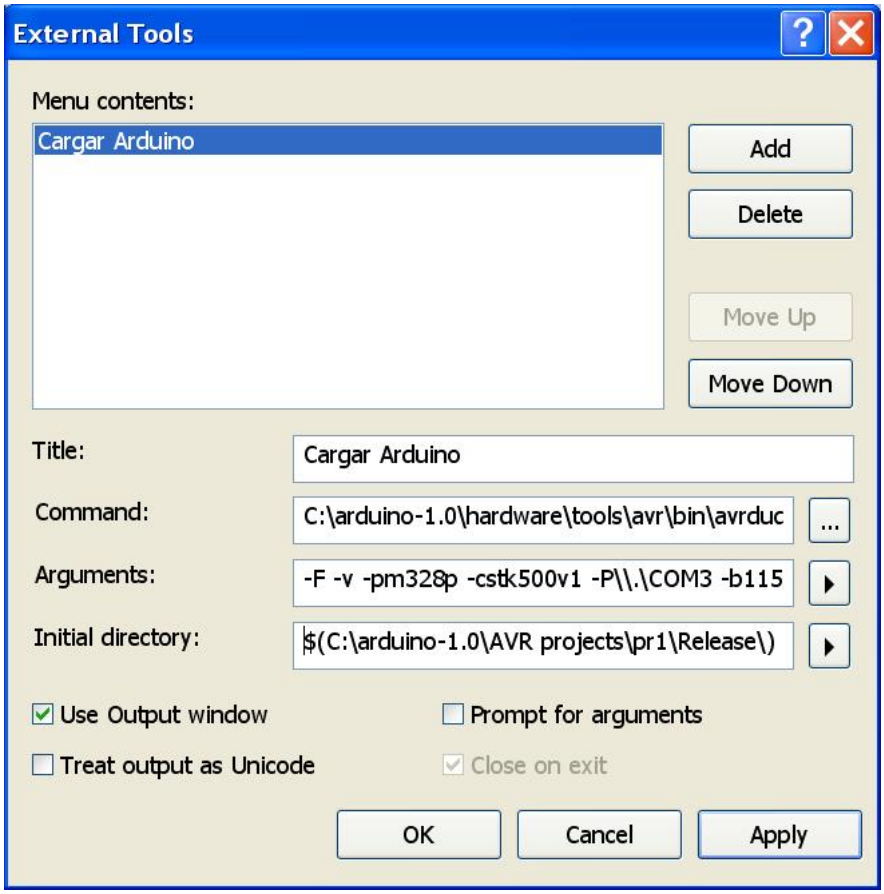
Y claro, en cada caso hay que emplear el arvrdude.config que viene en esa versión de Arduino, en la carpeta

Arduino 1.0:
C:\arduino-1.0\hardware\tools\avr\etc
y en Arduino 0023:
C:\arduino-0023\hardware\tools\avr\etc

Emplear la configuración incluida en Arduino 1,0 me ha dado problemas de funcionamiento (aunque la carga se hace sin errores), así que empleo el avrdude.exe y su archivo config, extraídos de Arduino 0023.

Ejemplo real de una “External Tool” para cargar el chip de la placa Arduino.

En mi caso, estos son los campos antes referidos:



Title:
Cargar Arduino

Commands:
C:\arduino-1.0\hardware\tools\avr\bin\avrdude.exe

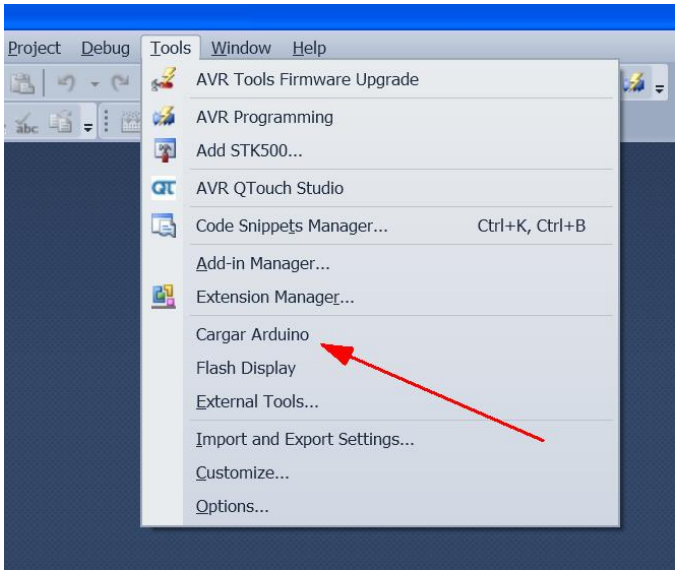
Arguments:
-F -v -pm328p -cstk500v1 -P\\.\COM3 -b115200 -D -Uflash:w:"calculadora.hex":i

Initial Dirctory:
$(C:\arduino-1.0\AVR projects\pr1\Release\)

Como vemos, en el campo “Arguments” he quitado la ruta completa, dejando solo el nombre del fichero de salida “calculadora.hex”.

Lo he hecho para reducir la longitud del campo “Arguments”. La ruta la he puesto en el campo “Initial Directory”, para que de forma que sea más fácil modificarla, comprobarla, etc.

Ya tenemos la herramienta,



Pero es más cómodo utilizar un botón integrado en el entorno de desarrollo….

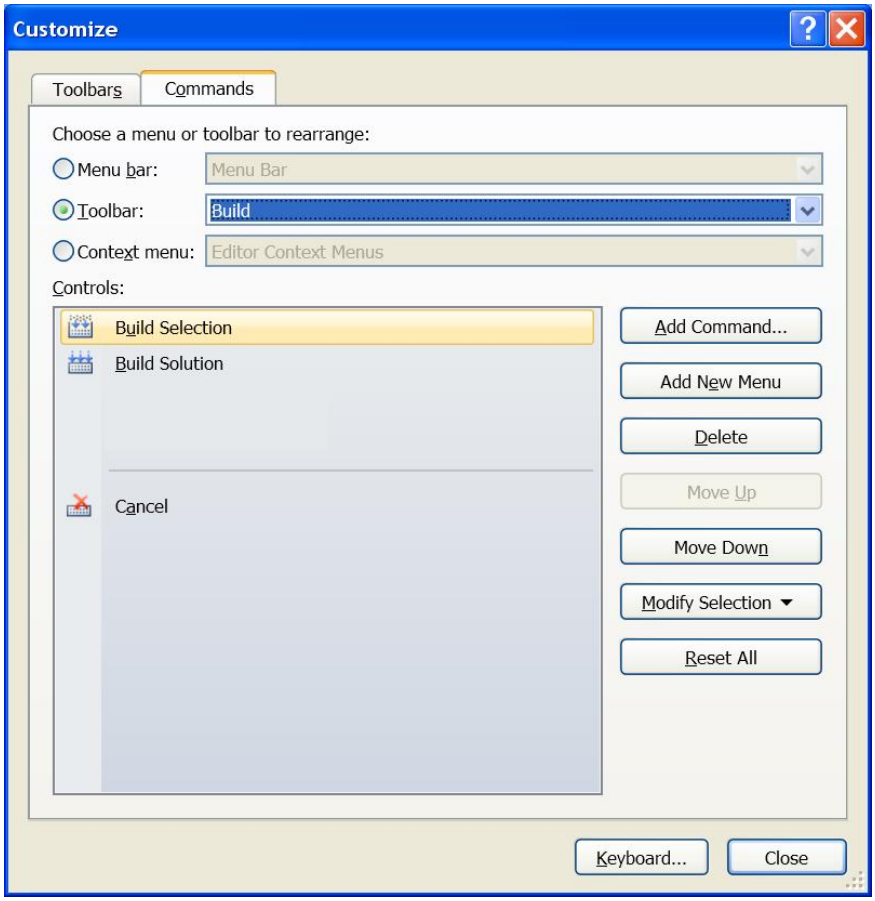
Creación del Botón “Cargar Arduino” en una barra de Herramientas.

Vamos a crear un botón, en la barra de herramientas “Build” que esté asociado a la herramienta de carga que acabamos de crear.

Dentro del menú “Tools” vamos al apartado “Customize…”

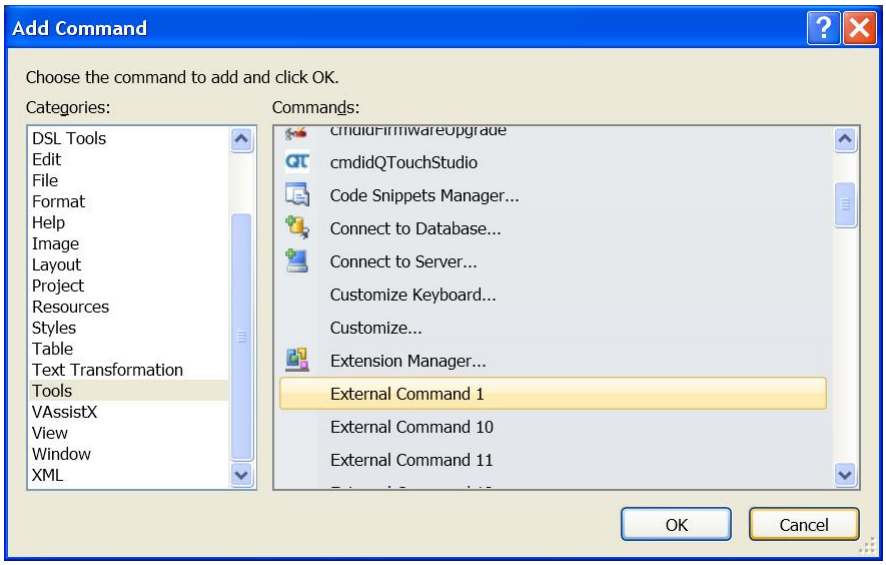
Seleccionamos la pestaña “Commands”, marcamos la opción “Toolbar:” para que lo que vamos a hacer, se cree en una barra de herramientas (y no en un menú, etc.)

Dentro de la lista desplegable, elegimos “Build” para que se cree en esa barra de herramientas.



Ahora pulsamos el botón “Add command...”

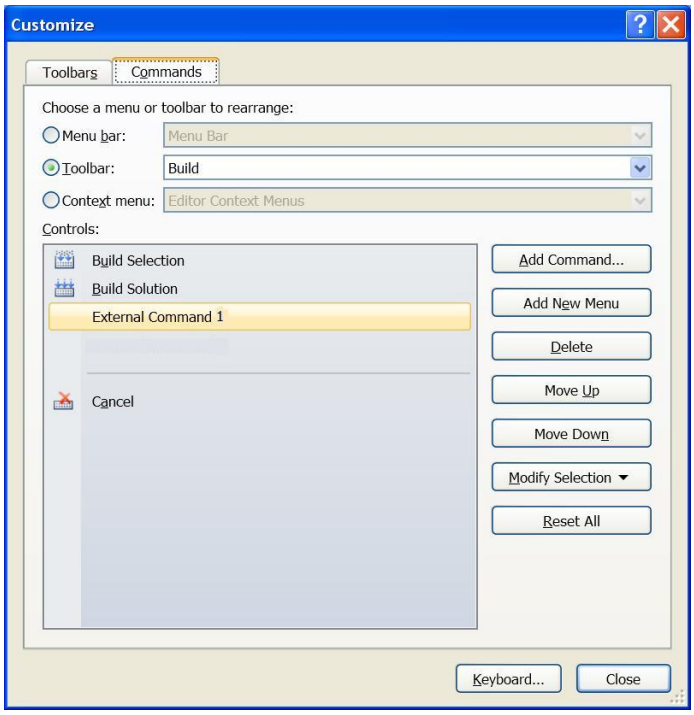
Nos sale otra pantalla:



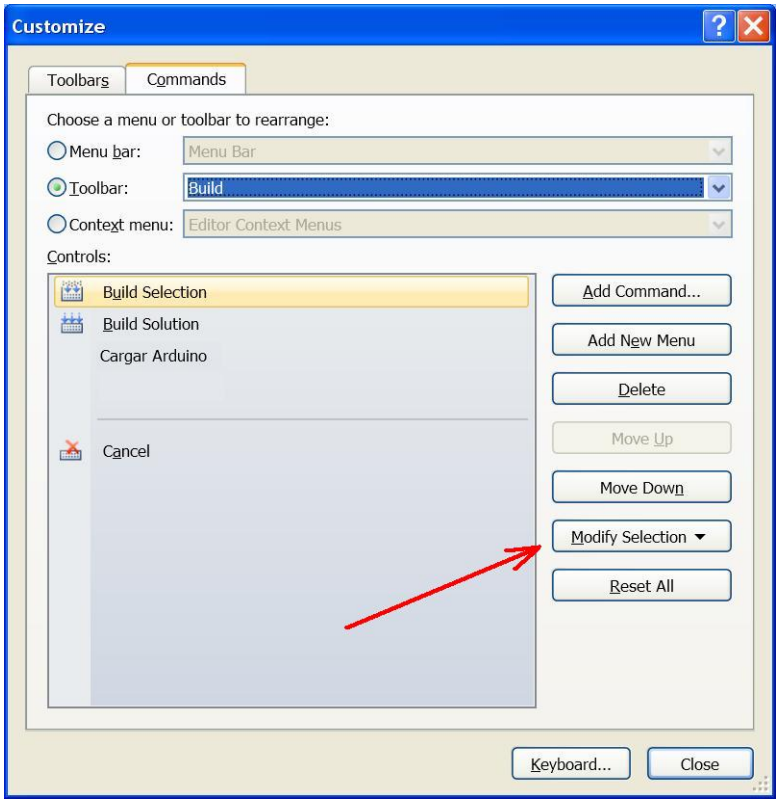
En ella vamos a el cuadro de la izquierda “Categories”, y selecciono “Tools”.

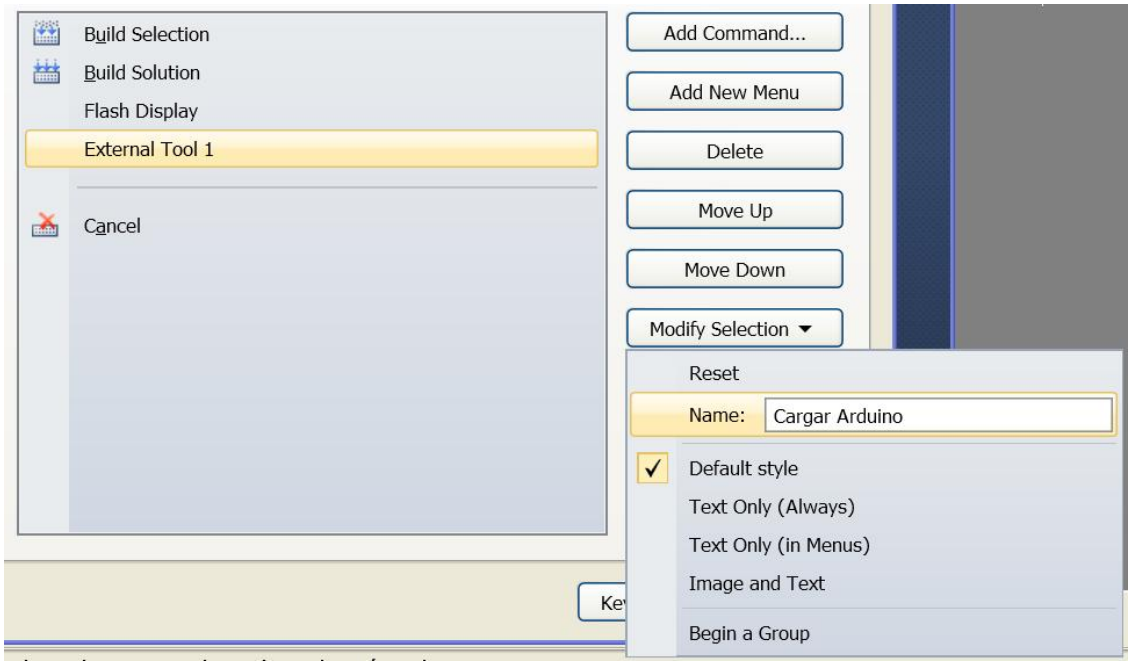
Ahora en el cuadro de la derecha, bajando con la barra de desplazamiento, selecciono la opción “External Command 1” y pulso “OK”

El “External Command 1” que hemos seleccionado se corresponde con la herramienta de carga que creamos en el apartado anterior. Más adelante la cambiaremos de nombre, para que nos sea más fácil identificarla entre tantas opciones….

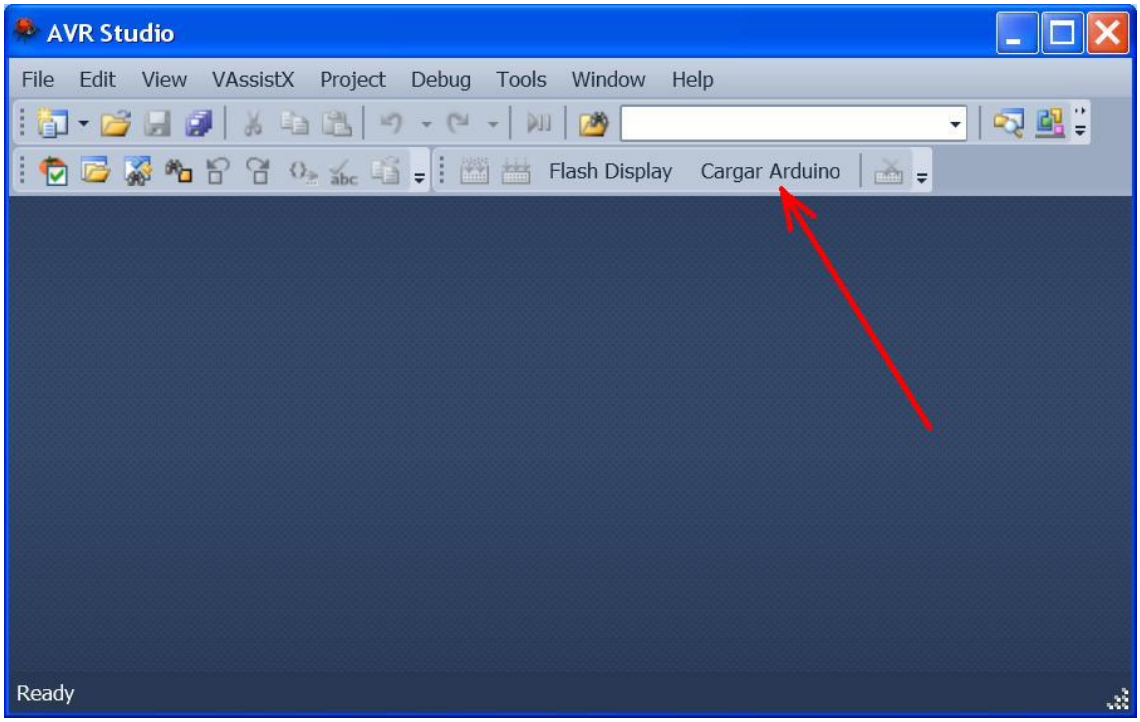


Ya tenemos un botón en la barra de herramientas “Build”, pero mejor, le vamos a cambiar el nombre: Pulso “Modify Selection”





¡Ya está creado el botón….!



A partir de este momento, es posible cargar nuestro programa en el chip Atmega de la placa Arduino, de la misma forma que se hace en el entorno Arduino.

Mejoras en el uso de la “External Tool”: Directorio auxiliar de salida, común a todos los proyectos.

La principal limitación que presenta lo que acabamos de explicar, es que la **“External Tool” está fijada para un determinado proyecto**, ya que la ruta del fichero de salida *. hex queda fijada al configurar la herramienta.

Esto nos obliga a actualizar el directorio utilizado en la “External Tool” cada vez que cambiamos de proyecto.
Para mitigar de alguna manera el problema, se puede simplificar la herramienta, haciendo que:

1. El entorno de desarrollo AVR5 Studio copie el archivo *.hex a un directorio de salida, común para todos los proyectos. En este ejemplo, se llamará C:\compilaciones.
2. La “External Tool” llame al programa cargador avrdude.exe en este directorio.

De esta forma, **si todos los proyectos usan el mismo nombre para el archivo de salida *.hex, podremos cambiar de un proyecto a otro, sin modificar nada.**

La mejora se basa en aprovechar que al crear el archivo de salida *.hex, el directorio activo es el que contiene dicho fichero. Esto nos permite copiarlo a otro directorio común sin tener que especificar la ruta del proyecto (que es diferente en cada uno de ellos). Así el comando de copia es siempre el mismo.
Una vez ahí, ejecutamos el programa cargador sobre ese archivo *.hex.

Para automatizar la copia del archivo de programa *.hex al directorio “C:\compilaciones” podemos emplear los eventos programables para cuando se termina de construir el fichero de salida *.hex, y definidos en las opciones del proyecto.

Para ejecutar el fichero por lotes *. bat podemos emplear la “External Tool” directamente.

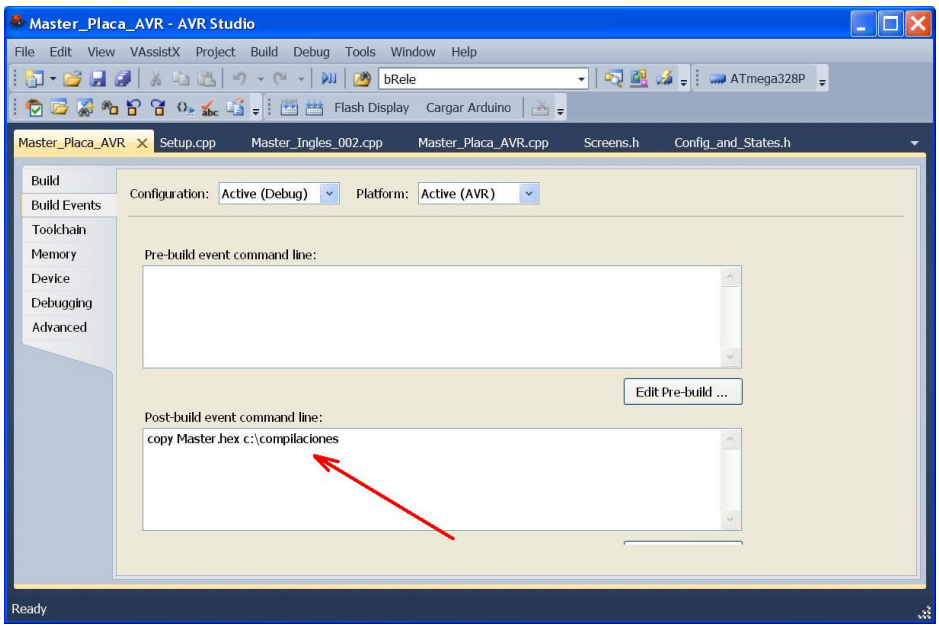
1ª Modificación de la “External Tool” para emplear un directorio auxiliar común a todos los proyectos.

Vamos al menú “Project → Project Propperties” (en el segundo aparecerá el nombre de nuestro proyecto, en vez de la palabra Project)

Vamos a Build Events, y añadimos, en “Post-Build event Command Line” la siguiente línea:

copy Master.hex c:\compilaciones

(Asumiendo que el archivo de programa se llama Master.hex y que el directorio de destino queremos que sea c:\compilaciones)



El contenido de la “External Tool” será:

Title:
Cargar Arduino
Commands:
C:\arduino-1.0\hardware\tools\avr\bin\avrdude.exe
Arguments:
-F -v -pm328p -cstk500v1 -P\\.\COM3 -b115200 -D -Uflash:w:"Master.hex":i
Initial Dirctory:
$(C:\compilaciones\)

De esta forma, cada vez que compilamos un proyecto, el archivo de programa *.hex es copiado al directorio c:\compilaciones.

A continuación, pulsamos el botón “Cargar Arduino” que creamos anteriormente, y esto hará que se ejecute el programa cargador avrdude.exe, para *flashear* nuestro chip en la placa Arduino.

2ª Modificación del uso de la “External Tool” para añadir el uso de un fichero de ejecución por lotes *.bat.

Si además queremos emplear diferentes nombres para cada archivo de salida de cada proyecto, hay que hacer más cosas:

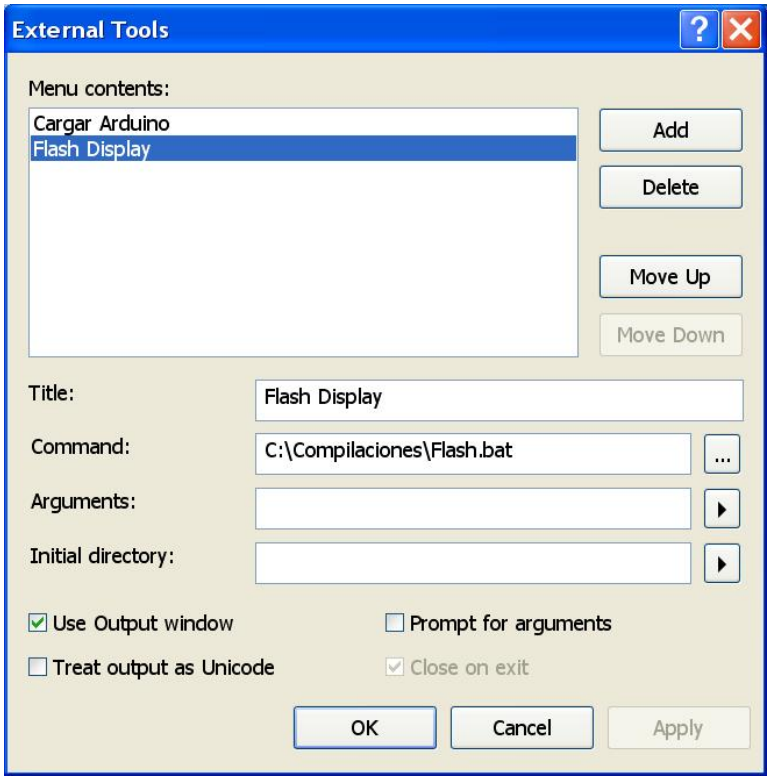
Necesitamos eliminar del entono AVR5 Studio casi todo lo que es susceptible de ser modificado en cada proyecto, cada versión, etc.

Para ello, haremos que la “External Tool” pasa a ejecutar un archivo de lotes *.bat, que será el que llame al programa cargador avrdude.exe, para cada nombre de fichero.

El proceso queda como sigue:

1. El entorno de desarrollo AVR5 Studio copia el archivo *.hex a un directorio de salida, común para todos los proyectos. En este ejemplo, se llamará C:\compilaciones.
2. La “External Tool” lanza un archivo*.bat definido por nosotros, el cual llama al programa cargador avrdude.exe. En este ejemplo se llamará Flash.bat.

Como ejemplo. en esta otra herramienta mostrada a continuación, llamada “Flash Display” vemos en el campo “Arguments”que simplemente se llama a un fichero Flash.bat, el cual hace todo el trabajo.



El contenido del fichero *.bat para el display, es:

```
CLS
@echo off
echo Flasheamos el display con la nueva version.
avrdude -c usbtiny -F 1 -p ATMEGA328p -U flash:w:main.hex
echo Finalizado, revise el resultado.
```

Y en el caso de una placa Arduino (suponiendo que el archivo de programa se llama Master.hex) sería algo como:

```
CLS
@echo off
echo Flasheamos el display con la nueva version.
-F -v -pm328p -cstk500v1 -P\\.\COM3 -b115200 -D -Uflash:w:"Master.hex":i
echo Finalizado, revise el resultado.
```

De esta forma, cada vez que compilamos un proyecto, el archivo de programa *.hex es copiado al directorio c:\compilaciones.
A continuación, pulsamos el botón “Cargar Arduino” que creamos anteriormente, y esto hará que se ejecute el fichero bat.
El fichero bat ejecutará el programa cargador avrdude.exe, y de esta forma *flashseará* nuestro chip en la placa Arduino.

Si ahora abrimos otro proyecto, en el que el archivo de salida *.hex tiene otro nombre, solo es necesario editar el archivo por lotes *.bat, y cambiarlo. Pero la configuración de la “External Tool” permanece inalterado.

Agradecimientos.

El texto referido a los parámetros de carga del programa, en el chip, empleando el cargador avrdude.exe se basan en gran medida en el contenido de esta web:
http://www.engblaze.com/tutorial-using-avr-studio-5-with-arduino-projects/
y en mis propias experiencias.