MICROCONTROLADORES Y
ELECTRÓNICA DE POTENCIA

INSTRUCTIVO
ATMELSTUDIO Y AVR

Tabla de contenido

1. Pasos de instalación y configuración

1) Instalar el programa AtmelStudio (ahora MicrochipStudio) del siguiente enlace:

https://www.microchip.com/en-us/tools-resources/develop/microchip-studio#Downloads

2) Descargar la carpeta ***AVRDUDESS*** del siguiente link: del siguiente link:

https://drive.google.com/drive/folders/0B-0P_4m3NjGYNWl6aGIsdHEtZUk

Este programa nos permitirá en un principio cargar el archivo ejecutable (.hex), comunicado por puerto serie al bootloader del microcontrolador, a través de la placa Arduino y desde AtmelStudio. Luego veremos más posibilidades que nos brinda de lectura/escritura de memorias FLASH, EEPROM, fusibles de configuración, etc., y desde distintos dispositivos programadores.

3) Conectar la placa *Arduino* a utilizar con cable USB a la computadora. Buscar en: Administrador de dispositivos → Puertos COM y LPT (ejemplo: **COM3** para *Arduino UNO*).

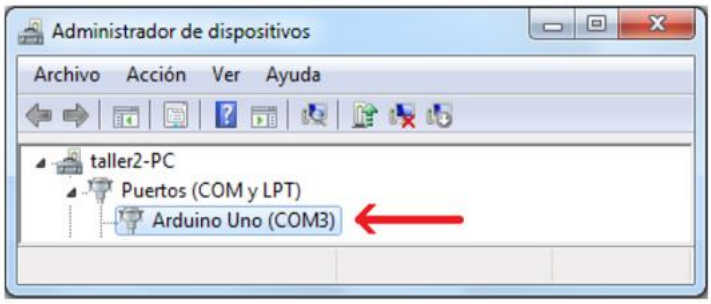


Figura 1. *Administrador de dispositivos para ver puerto COM vinculada a placa Arduino conectada.*

4) Configurar herramientas externas

Tools → External Tools → Add

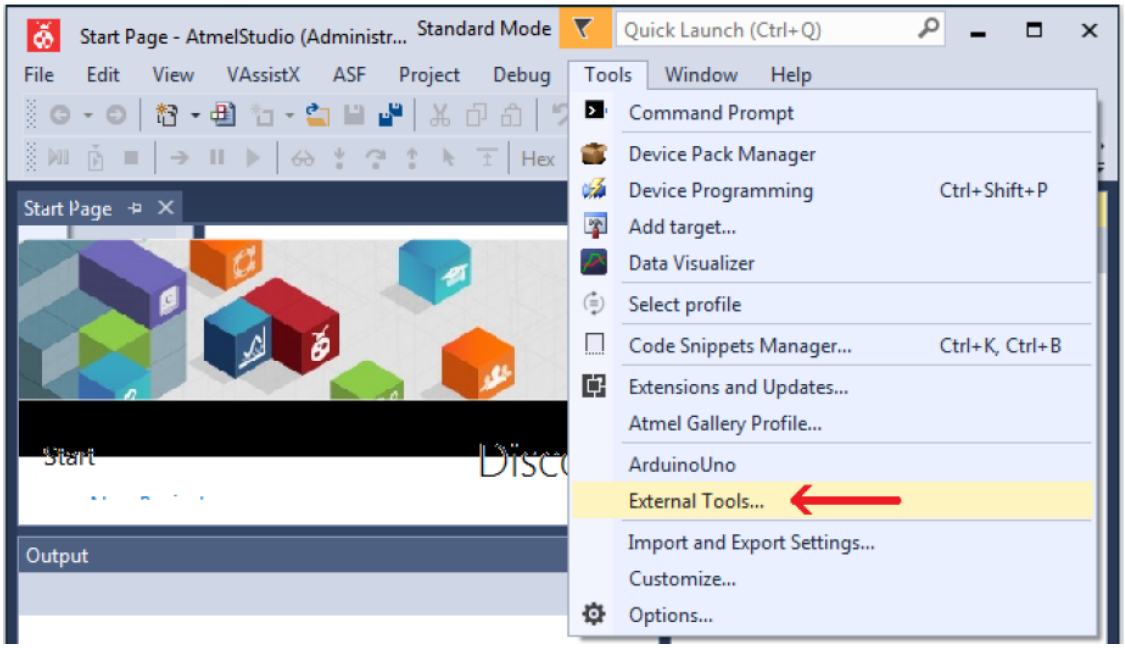


Figura 2. *External tools en AtmelStudio.*

- En **Title** poner nombre de la placa (éste quedará en la pestaña Tools para luego grabar)
- En **Command** poner la ubicación del archivo avrdude.exe descargado (ejemplo: C:\Users\Yo\Desktop\AVRDUDESS\avrdude.exe)
- En **Arguments** escribir (con el puerto COM correspondiente):

    - Para ***Arduino UNO*** (microcontrolador *Atmega328p*):

-c arduino -p m328p -P COM3 -b 115200 -U flash:w:"$(ProjectDir)\Debug\$(TargetName).hex":i

    - Para ***Arduino NANO*** (microcontrolador *Atmega328p*):

-c arduino -p m328p -P COM3 -b 57600 -U flash:w:"$(ProjectDir)\Debug\$(TargetName).hex":i

    - Para ***Arduino MEGA*** (microcontrolador *Atmega2560*):

Opción 1:

-c wiring -p m2560 -P COM3 -b 115200 -U flash:w:"$(ProjectDir)\Debug\$(TargetName).hex":i

Opción 2:

-C "C:\Users\AVRDUDESS\avrdude.conf" -v -patmega2560 -cwiring -P COM3 -b 115200 -D -U flash:w:"$(ProjectDir)\Debug\$(TargetName).hex":i

5) Tildar "Use Output window" → Apply → OK

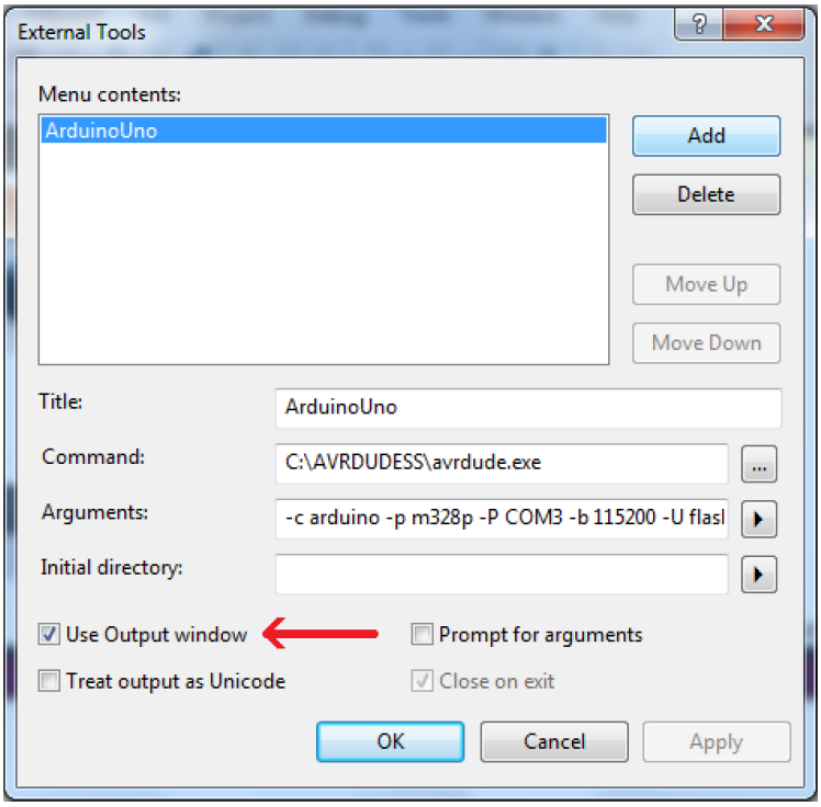


Figura 3. *Configuración de External tools en AtmelStudio.*

2. Parámetros para los Argumentos de External Tools

Para ver las opciones de escritura de **parámetros** en la pestaña ***Arguments*** de las herramientas externas de *AtmelStudio*, podemos ir a la ventana Símbolo de sistema y escribir avrdude en el directorio donde está instalado *AVRDUDESS*. Por ejemplo:

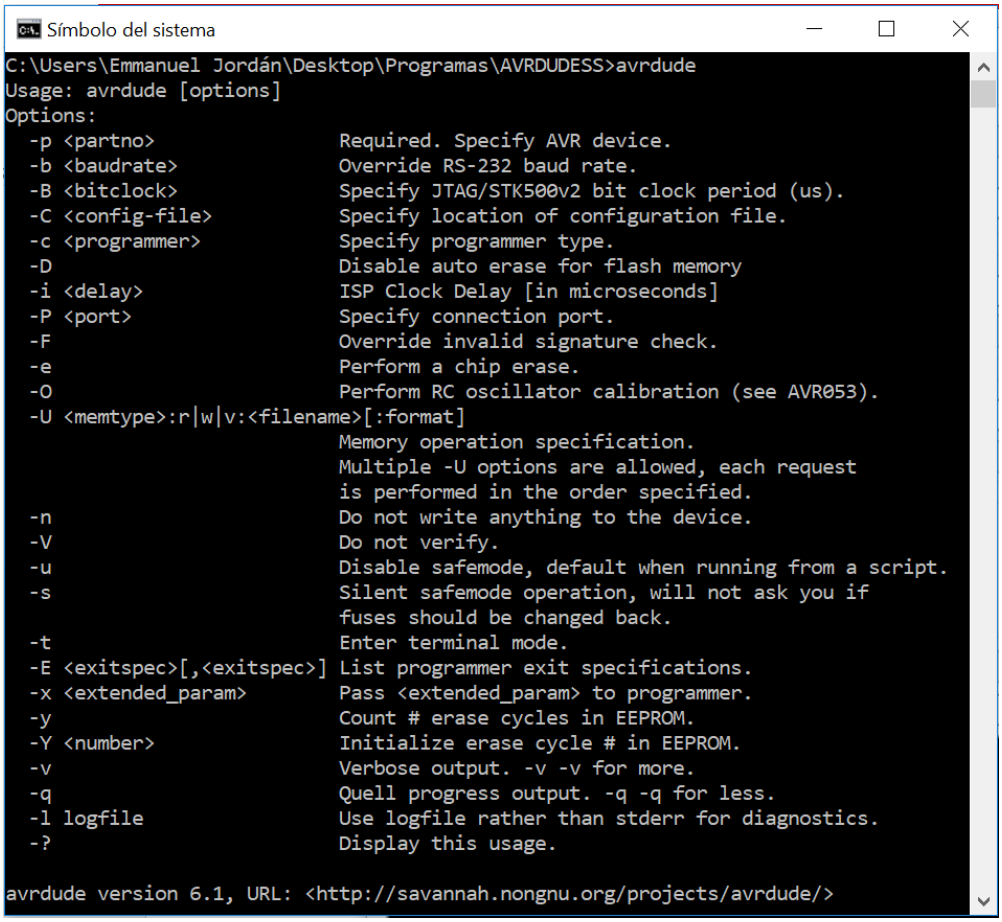


Figura 4. *Símbolo de Sistema para ver parámetros para Argumentos.*

Por ejemplo, para el caso de *Arduino UNO*, escribimos:

-c arduino -p m328p -P COM3 -b 115200 -U flash:w:"$(ProjectDir) \Debug\$(TargetName).hex":i

Por lo que estamos especificando:

- -c arduino tipo de programador *Arduino*
- -p m328p dispositivo AVR: *Atmega328p*
- -P COM3 puerto COM al que está conectada la placa *Arduino*
- -b 115200 baudrate rs232
  (Ej. 115200 para *Arduinos UNO* y *MEGA*, 57600 para *Arduino NANO*)
- -U flash:w:"$(ProjectDir)Debug\$(TargetName).hex":i
  tipo de memoria Flash, sólo escritura,
  (r: read, w: write, v: verify) (i: Intel hex, a: auto-wiring)

Los detalles de cada uno de los parámetros y las opciones de escritura de los mismos se puede ver en: http://www.nongnu.org/avrdude/user-manual/avrdude_4.html

También se pueden escribir los **FUSE-bits** añadiendo los Argumentos parámetros de la forma (detalles más adelante):

-U lfuse:w:0xXX:m -U hfuse:w:0xXX:m -U efuse:w:0xXX:m

3. Creación de proyecto en AtmelStudio

1) File → New → Project

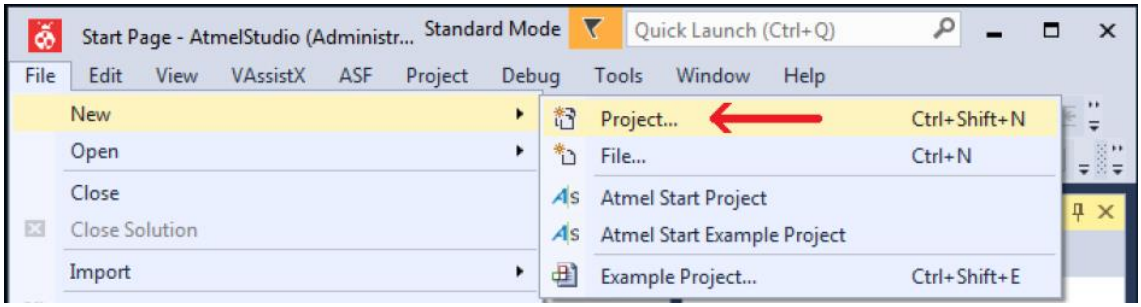


Figura 5. Creación de proyecto en AtmelStudio.

2) GCC C Executable Project → OK

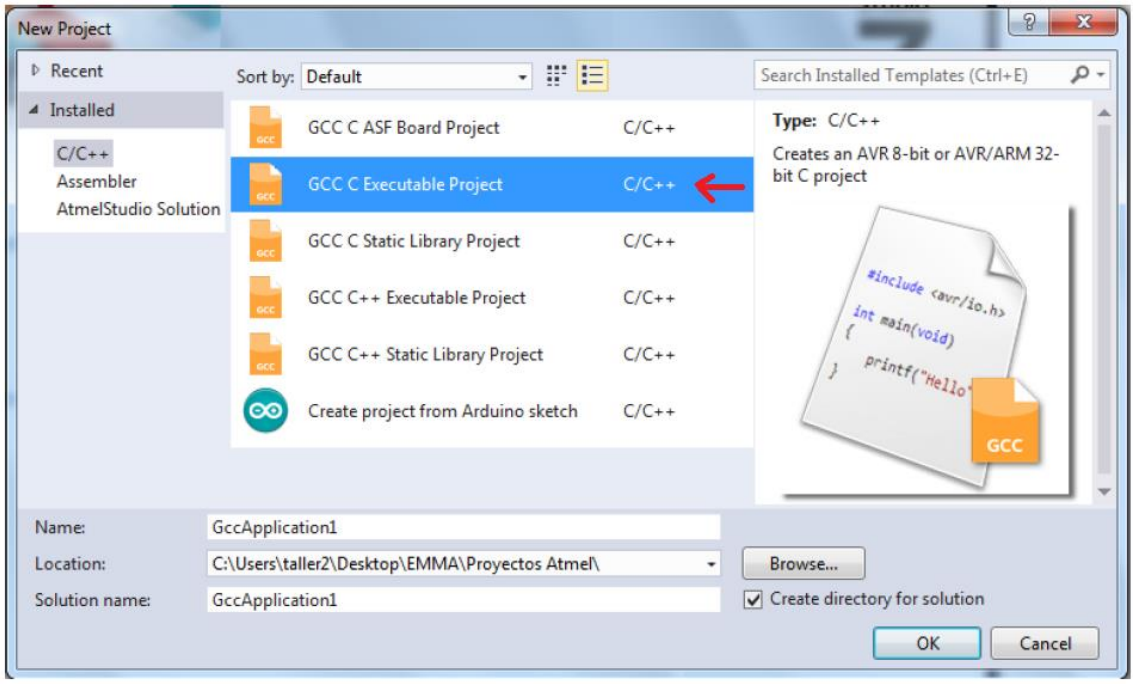


Figura 6. Elección de tipo de proyecto en AtmelStudio.

3) Buscar microcontrolador a grabar → OK (ejemplo *Atmega328p* para *Arduino UNO*)

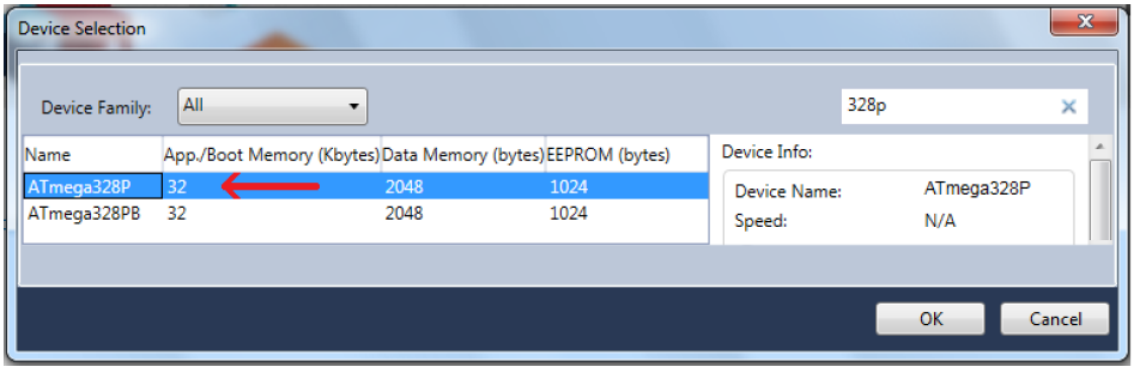


Figura 7. Elección de microcontrolador en AtmelStudio.

Se crea el archivo principal main con biblioteca básica de E/S, en la carpeta solución:

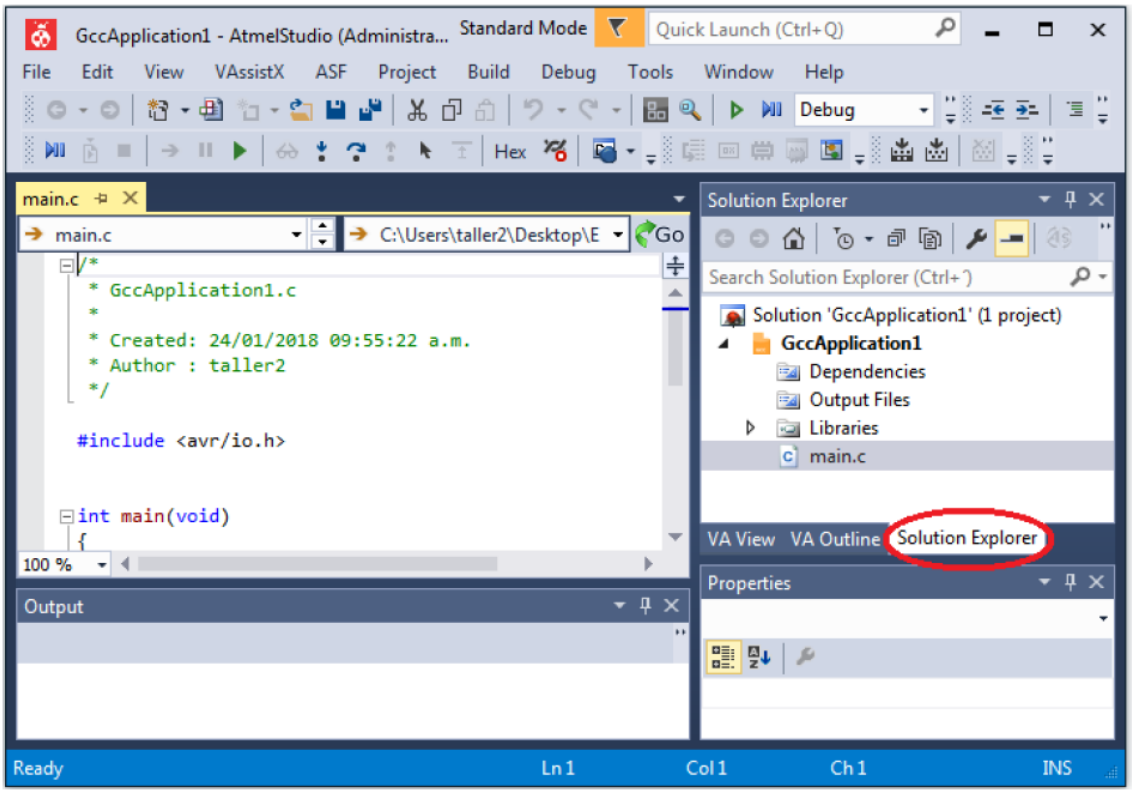


Figura 8. Proyecto creado en AtmelStudio.

Una vez realizado el programa, para grabarlo en el microcontrolador:

- Build → Clean (nombre del proyecto). Opcional.
- Build → Build (nombre del proyecto). Se crea archivo .hex para grabar en microcontrolador.
- Tools → Nombre de placa conectada (ej. *Arduino UNO*)

4. Cómo grabar nuestro programa en el microcontrolador

- ***A través de AtmelStudio:***

Una vez realizado el programa en *AtmelStudio*, lo compilamos desde *Build* → *Build (nombre proyecto)* y lo grabamos desde *Tools* → *nombre de la placa*.

- ***A través de AVRDUDESS:***

Una vez realizado el programa en *AtmelStudio*, lo compilamos desde *Build* → *Build (nombre proyecto)* y luego abrimos el programa *AVRDUDESS*. Elegimos el programador *Arduino*, la velocidad de baudrate adecuada (ejemplo 115200 para *Arduinos UNO* y *MEGA*, 57600 para *Arduino NANO*). Con la placa *Arduino* conectada por cable USB, elegimos *Detect* (debe detectar el microcontrolador a grabar). Buscamos el archivo .hex generado y lo grabamos en la memoria Flash del microcontrolador con *Go*.

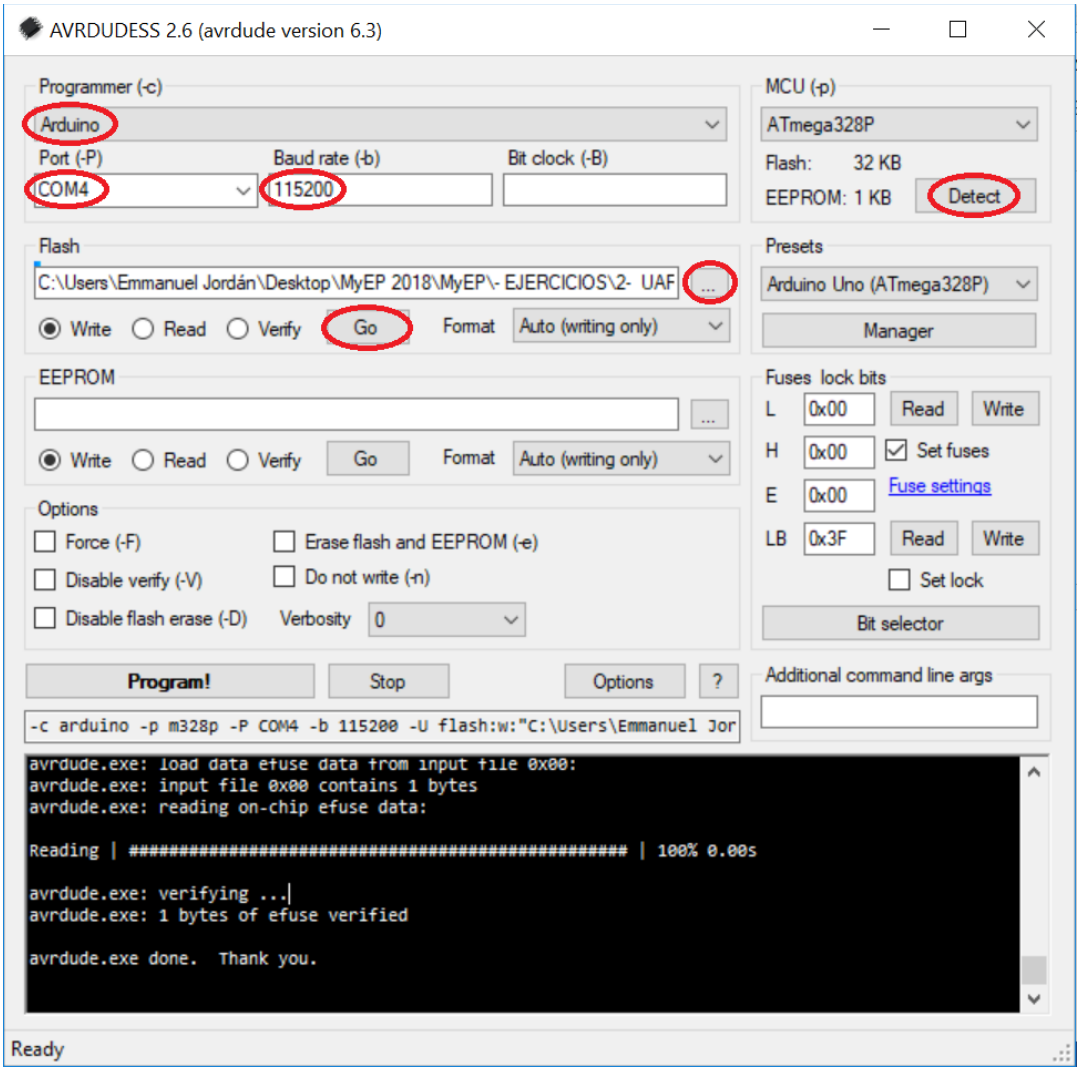


Figura 9. *Grabación de programa desde AVRDUDESS.*

5. Cómo "quemar" el bootloader en un microcontrolador

El bootloader es un pequeño programa (gestor de arranque) que se sitúa en una sección de la memoria del microcontrolador. Éste hace de puente a través de un protocolo serie (UART, I2C, SPI,...) brindando acceso a la memoria Flash del microcontrolador, en la cual se graba el archivo .hex (generado al compilar el código fuente). Para quemar el bootloader necesitamos un programador externo (ejemplo USBasp) o una placa Arduino funcional, como se explica a continuación.

Nota: Las placas *Arduino* deben traer el bootloader cargado en el microcontrolador que contienen para poder grabarlos por el cable USB. Algunas veces esto no se cumple. Si lo tienen, al quemarlo, sobre-escribiremos el anterior y continuará funcionando.

- ***A través de una placa Arduino (placa GRABADORA)***

1) Abrimos *Arduino IDE* y buscamos el ejemplo *ArduinoISP*.

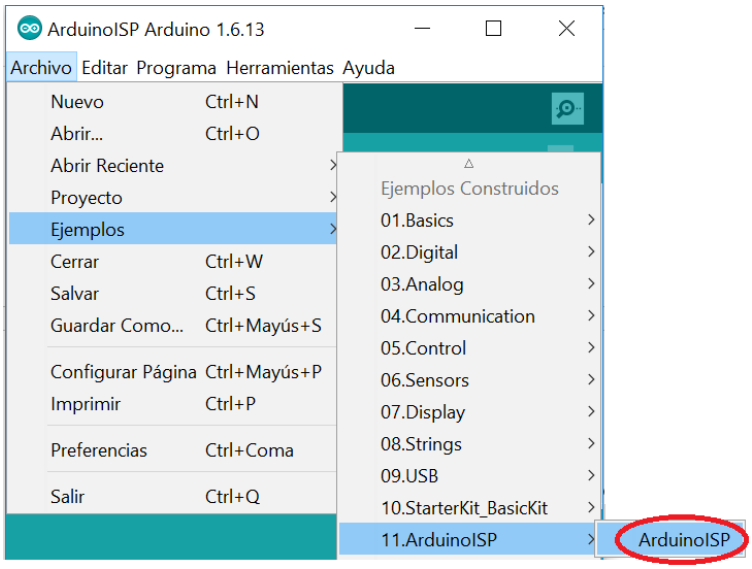


Figura 10. *Quemando bootloader en microcontrolador, paso 1.*

2) Con la placa grabadora conectada por cable *USB*, en herramientas seleccionamos la placa *Arduino* (grabadora) y el puerto COM adecuado y el programador *ArduinoISP*.

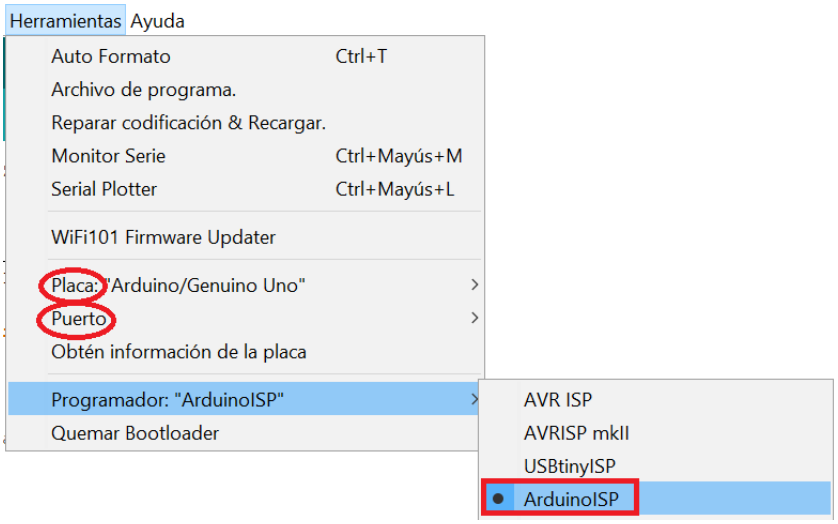


Figura 11. *Quemando bootloader en microcontrolador, paso 2.*

3) En el código fuente descomentamos la definición *USE_OLD_STYLE_WIRING* y verificamos los pines de RESET, SCK, MOSI y MISO (la numeración corresponde a la placa *Arduino*). Seleccionamos además en este código la velocidad adecuada de baudrate de grabación (ejemplo 19200 para *Arduino UNO*).

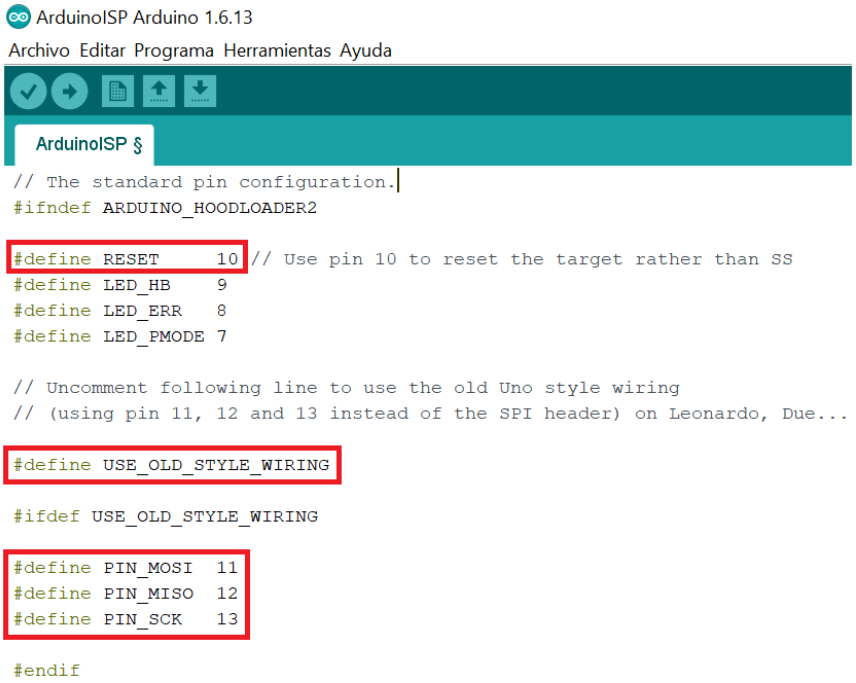


Figura 12. *Quemando bootloader en microcontrolador, paso 3.*

4) Verificamos y grabamos el programa.

5) Conectamos los pines de *GND*, *5V*, *SCK*, *MOSI* y *MISO* de la placa grabadora con los correspondientes del microcontrolador a grabar, así como también el pin definido como *RESET* de la placa grabadora en el pin *reset* del microcontrolador a grabar. En la siguiente imagen se ve un ejemplo de conexionado para cargar el bootloader en un microcontrolador *Atmega328p* a través de una placa *Arduino UNO*.

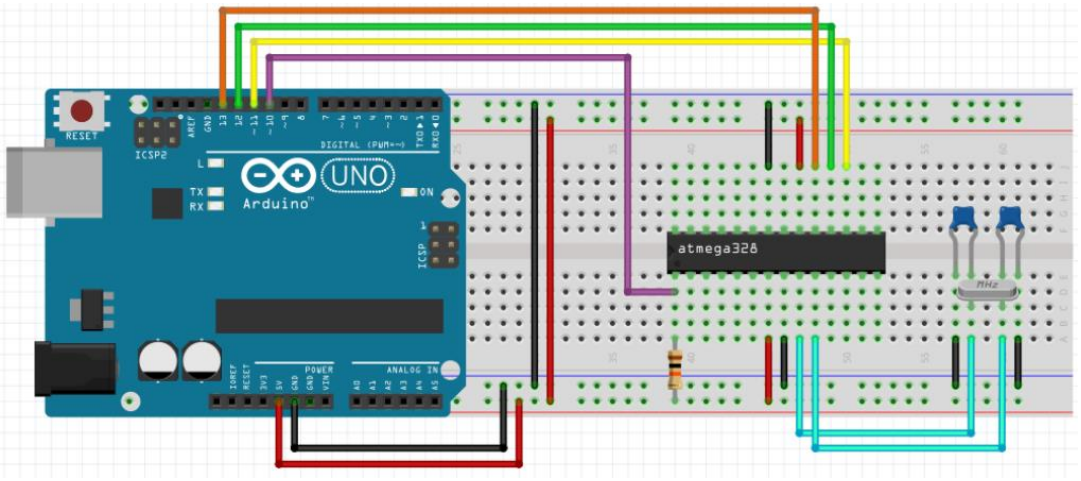


Figura 13. *Quemando bootloader en microcontrolador, paso 5.*

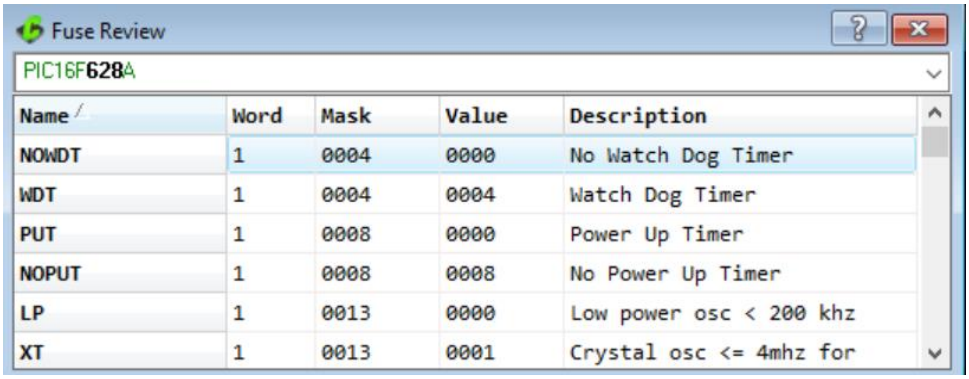
6) En *Arduino IDE* cambiamos en herramientas el programador a *Arduino as ISP*, elegimos el puerto adecuado y la placa *Arduino* a grabar (puede ser distinta a la placa grabadora).

7) Vamos a herramientas → quemar Bootloader.

6. ¿Qué son los FUSE-bits del microcontrolador?

Los FUSE-bits permiten configurar la fuente de oscilación (oscilador externo, cristal externo, cristal interno, etc.), el tiempo de "start-up" (tiempo extra desde reseteo), brown-out reset, watch-dog timer, habilitación de pin de Reset, entre otras cosas, del microcontrolador.

En los microcontroladores **PIC**, los FUSE-bits son modificables al momento de grabar el micro, definiéndolos en el código fuente. Para ver los fusibles disponibles para el PIC que se está utilizando, vamos a *"View"* → *"Config Bits"* (PIC C Compiler) y aparece una ventana como la siguiente:



Name	Word	Mask	Value	Description
NOWDT	1	0004	0000	No Watch Dog Timer
WDT	1	0004	0004	Watch Dog Timer
PUT	1	0008	0000	Power Up Timer
NOPUT	1	0008	0008	No Power Up Timer
LP	1	0013	0000	Low power osc < 200 khz
XT	1	0013	0001	Crystal osc <= 4mhz for

Figura 14. *FUSE-bits en microcontrolador PIC.*

Por ejemplo para trabajar con cristal interno y habilitar el pin MCLR, entre otras opciones, escribimos:

```
#FUSES INTRC          //Oscilador interno
#FUSES MCLR           //pin Master Clear habilitado
#FUSES NOWDT          //Sin timer de Watch Dog
#FUSES NOPUT          //Sin timer Power Up
#FUSES NOPROTECT      //Código no protegido contra lectura
#FUSES NOBROWNOUT     //Sin reseteo Brownout
#FUSES NOLVP          //Sin programación con bajo voltaje B3(PIC16) o B5(PIC18)
#FUSES NOCPD          //Sin protección EE
#FUSES RESERVED       //Usado para setear los FUSE bits reservados
```

Figura 15. *Escritura de FUSE-bits en microcontrolador PIC.*

En cambio, en los microcontroladores **AVR**, los FUSE-bits no se pueden modificar desde el código fuente. Se debe además tener en cuenta las siguientes observaciones para los mismos:

- Los FUSE-bits contienen un '0' cuando están programados y un '1' cuando no están programados.
- El estado de los FUSE-bits no se ve afectado por un borrado de chip (chip erase).
- Los FUSE-bits se bloquean si se programa Lock bit 1 (LB1). Por lo tanto, primero se deben programar los FUSE-bits y luego los bits de BLOQUEO en caso de requerirlo.

Por ejemplo, el *Atmega328p* tiene tres registros para sus FUSE-bits. Una descripción de los mismos se muestra a continuación. Para más detalles ver catálogo del microcontrolador.

I. EXTENDED FUSE-BYTE

Extended Fuse Byte	Bit No.	Description	Default Value
–	7	–	1
–	6	–	1
–	5	–	1
–	4	–	1
–	3	–	1
BODLEVEL2[1]	2	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL1[1]	1	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL0[1]	0	Brown-out Detector trigger level	1 (unprogrammed)

Figura 16. Extended FUSE-Byte en Atmega328p.

- **BODLEVEL2:0 →** nivel de tensión por debajo del cual el microcontrolador se resetea:

Table 32-8. BODLEVEL Fuse Coding[1][2]

BODLEVEL [2:0] Fuses	Min. V_{BOT}	Typ. V_{BOT}	Max V_{BOT}	Units
111	BOD Disabled			
110	1.7	1.8	2.0	V
101	2.5	2.7	2.9	
100	4.1	4.3	4.5	
011	Reserved			
010				
001				
000				

Figura 17. Detalle de Extended FUSE-Byte en Atmega328p.

II. HIGH FUSE-BYTE

High Fuse Byte	Bit No.	Description	Default Value
RSTDISBL[1]	7	External Reset Disable	1 (unprogrammed)
DWEN	6	debugWIRE Enable	1 (unprogrammed)
SPIEN[2]	5	Enable Serial Program and Data Downloading	0 (programmed, SPI programming enabled)
WDTON[3]	4	Watchdog Timer Always On	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed), EEPROM not reserved
BOOTSZ1	2	Select Boot Size (see Boot Loader Parameters)	0 (programmed)[4]
BOOTSZ0	1	Select Boot Size (see Boot Loader Parameters)	0 (programmed)[4]
BOOTRST	0	Select Reset Vector	1 (unprogrammed)

Figura 18. High FUSE-Byte en Atmega328p.

- **RSTDISBL** → deshabilitación del Reset en el pin 1/PC6.
  1: Reset habilitado (PC6 = RESET),
  0: Reset deshabilitado (PC6 = PICINT14). Cuidado! No permite otra grabación.
- **DWEN** → habilitación del DebugWire. Cuidado!
- **SPIEN** → habilitación de Programación Serie. Cuidado!
- **WDTON** → habilitación del Watch Dog Timer
- **EESAVE** → preservación de la memoria EEPROM ante un borrado de chip
- **BTSZ1:0** → tamaño de Boot Loader
- **BOOTRST** → selección del Reset Vector
  0: el dispositivo salta a la dirección del Boot Loader en el Reset,
  1: el dispositivo salta a la dirección 0x000 en el Reset.

III. LOW FUSE-BYTE

Table 31-7. Fuse Low Byte

Low Fuse Byte	Bit No.	Description	Default Value
CKDIV8(4)	7	Divide clock by 8	0 (programmed)
CKOUT(3)	6	Clock output	1 (unprogrammed)
SUT1	5	Select start-up time	1 (unprogrammed)(1)
SUT0	4	Select start-up time	0 (programmed)(1)
CKSEL3	3	Select Clock source	0 (programmed)(2)
CKSEL2	2	Select Clock source	0 (programmed)(2)
CKSEL1	1	Select Clock source	1 (unprogrammed)(2)
CKSEL0	0	Select Clock source	0 (programmed)(2)

Figura 19. Low FUSE-Byte en Atmega328p.

- ***CKDIV8*** → División del clock por 8
- ***CKPUT*** → Salida del clock por pin PB0
- ***SUT1:0*** → Tiempo de "start-up". Junto con el bit CKSEL0, y dependiendo de la fuente de oscilación presente, determina el tiempo de retardo al encendido.
- ***CKSEL3:0*** → Fuente de oscilación

Table 13-1. Device Clocking Options Select

Device Clocking Option	CKSEL[3:0]
Low Power Crystal Oscillator	1111 - 1000
Full Swing Crystal Oscillator	0111 - 0110
Low Frequency Crystal Oscillator	0101 - 0100
Internal 128kHz RC Oscillator	0011
Calibrated Internal RC Oscillator	0010
External Clock	0000
Reserved	0001

Note: For all fuses, '1' means unprogrammed while '0' means programmed.

Figura 20. Selección de fuente de oscilación.

Ejemplo de configuración de FUSE bits de Atmega328p para trabajar en placa Arduino Uno o Nano:

$$\boldsymbol{EFuse = 0xFD = 0b11111101}$$

- Reseteo a 2.7Vcc.

$$\boldsymbol{LFuse = 0xF7 = 0b11110111}$$

- Sin división del clock por 8
- Sin salida del clock por pin PB0
- Máximo tiempo de "start –up"
- Full Swing Crystal Oscillator (más estable pero mayor consumo que Low Swing Crystal)

$$\boldsymbol{HFuse = 0xDE = 0b11011110}$$

- Reset Habilitado
- Debug Wire No Habilitado
- SPI Habilitado
- Watch Dog No habilitado
- EEPROM no preservada ante borrado de chip
- Mínimo tamaño de Boot Loader
- Dirección de Boot Loader en el Reset

7. Cómo modificar los FUSE-bits de un micro AVR

- ***A través de una placa Arduino y con AVRDUDESS***

1) Seguimos los mismos pasos para quemar el bootloader en el microcontrolador, hasta el conexionado de la placa grabadora con el microcontrolador a grabar.

2) Abrimos el programa *AVRDUDESS*, seleccionamos grabador *Arduino*, el puerto *COM* correspondiente y la velocidad adecuada de baudrate de grabación (ejemplo 19200 para *Arduino UNO*).

3) Con la placa grabadora conectada por cable USB y el microcontrolador a grabar conectado según los pasos anteriores, hacemos click en *Detect*. Inmediatamente detectará el microcontrolador a grabar (por ejemplo *Atmega328p*).

4) Escribimos los FUSE-bits ya sea desde "*bit selector*" o con la ayuda de "*Fuse settings*", y hacemos click en *Write*.

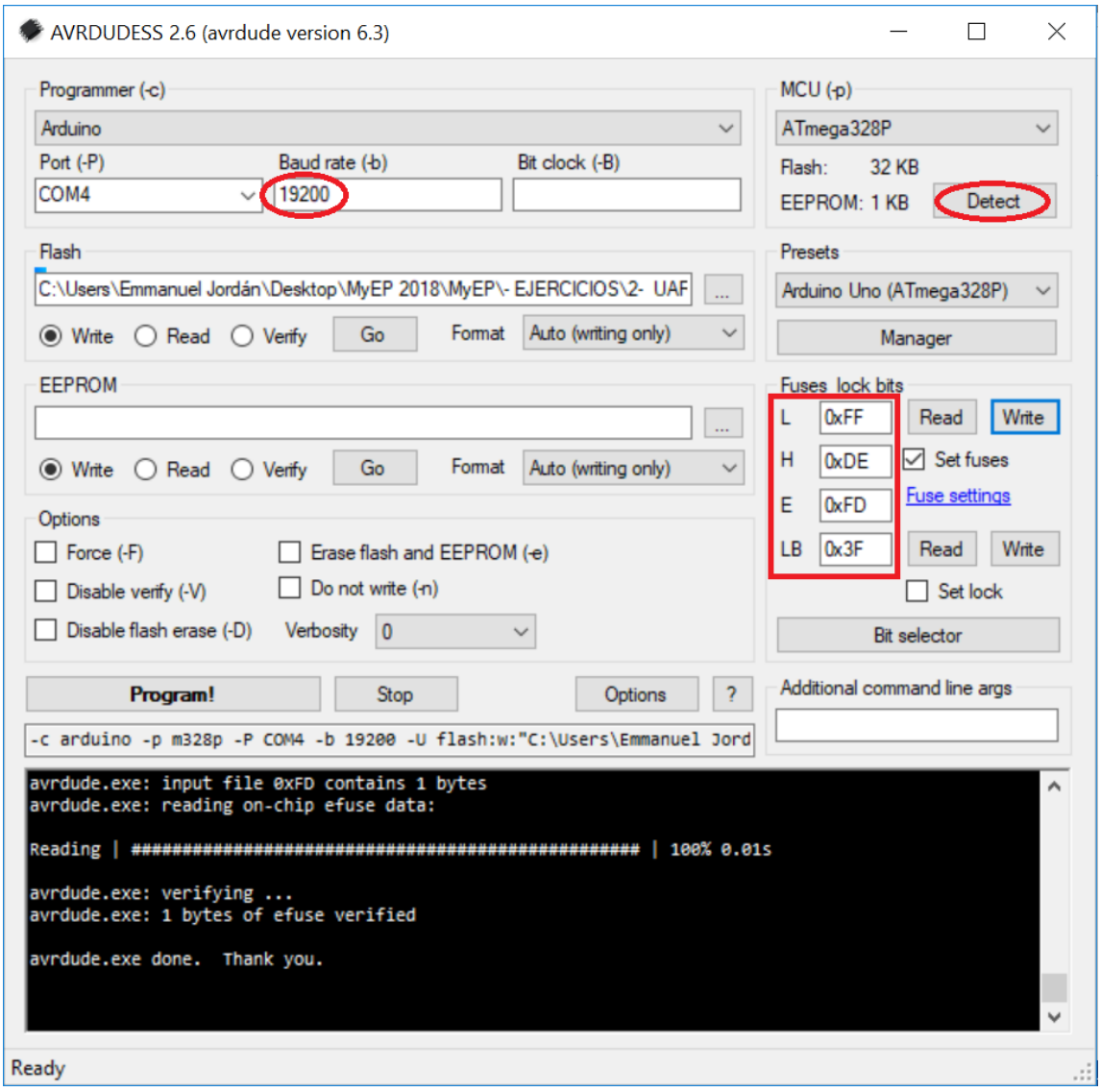


Figura 21. *Modificación de FUSE-bits desde AVRDUDESS.*

8. Cómo leer el programa de un microcontrolador

- ***A través de una placa Arduino y con AVRDUDESS***

1) Seguimos los mismos pasos para quemar el bootloader en el microcontrolador, hasta el conexionado de la placa grabadora con el microcontrolador a grabar.

2) Abrimos el programa *AVRDUDESS*, seleccionamos grabador *Arduino*, el puerto *COM* correspondiente y la velocidad adecuada de baudrate de grabación (ejemplo 19200 para *Arduino UNO*).

3) Con la placa grabadora conectada por cable USB y el microcontrolador a grabar conectado según los pasos anteriores, hacemos click en *Detect*. Inmediatamente detectará el microcontrolador a grabar (por ejemplo *Atmega328p*).

4) Luego elegimos la opción *Read* en memoria Flash. Escribimos la línea *Additional command line* como se muestra en la figura siguiente, especificando lugar de destino y nombre del archivo. Por último hacemos click en *Go*.

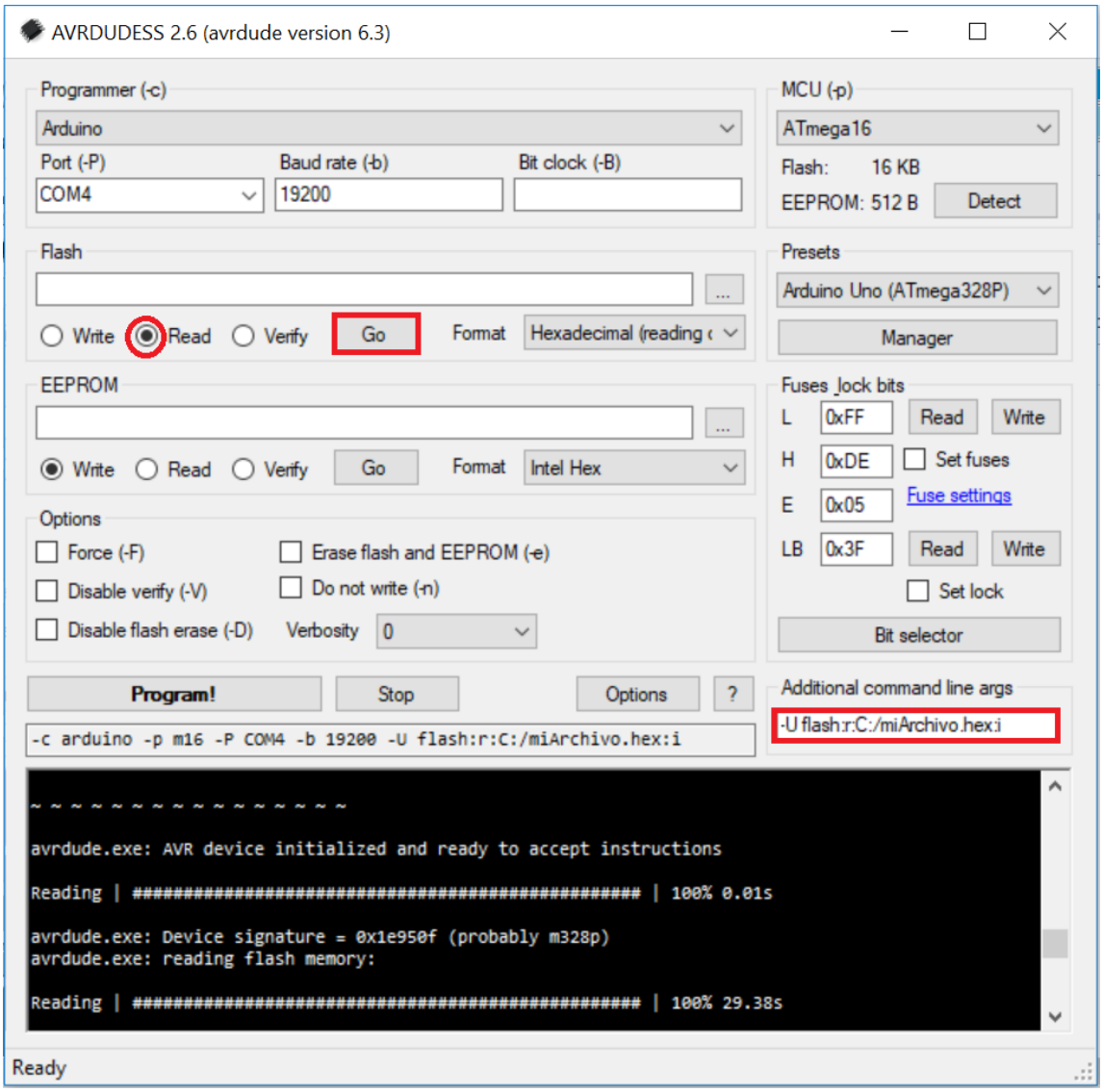


Figura 22. *Extracción de archivo .hex de un microcontrolador desde AVRDUDESS.*

Importante: la velocidad de **baudrate** para el microcontrolador en **MODO GRABADOR** (para leer/escribir el programa o los FUSE-bits de otro microcontrolador) no es la misma que la velocidad de baudrate para grabar directamente al microcontrolador por puerto serie. Por ejemplo, para el *Atmega328p* la velocidad para su grabación directa por USB es de 115200, mientras que la velocidad para el mismo en modo grabador es de 19200. Notar que al cargar el programa de *ArduinoISP* en el micro grabador, la velocidad seteada en el código fuente de *Arduino IDE* fue 19200 y no 115200, y esta velocidad es la seleccionada en *AVRDUDESS* para que actúe como tal.