MICROCONTROLADORES Y Electrónica de potencia

INSTRUCTIVO ATMELSTUDIO Y AVR

Tabla de contenido

1.	Pasos de instalación y configuración de AtmelStudio	3
2.	Parámetros para los Argumentos de External Tools	5
3.	Creación de proyecto en AtmelStudio	6
4.	Cómo grabar nuestro programa en el microcontrolador	8
5.	Cómo "quemar" el bootloader en un microcontrolador	9
6.	¿Qué son los FUSE-bits del microcontrolador?	11
7.	Cómo modificar los FUSE-bits de un micro AVR	14
8.	Cómo leer el programa de un microcontrolador	16

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_4m3NjGYNWl6aGlsdHEtZUk

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 \rightarrow External Tools \rightarrow 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" \rightarrow Apply \rightarrow OK

External Tools	_	-		_	? <mark>x</mark>
Menu contents:					
ArduinoUno					Add
				(Delete
				[Move Up
					Move Down
Title:	ArduinoUn	D			
Command:	C:\AVRDUD	ESS\avrd	ude.exe		
Arguments:	-c arduino ·	-p m328p	-P COM3 -ł	o 1152	00 -U flasl 🕨
Initial directory:					
🛙 Use Output window 🖣	(Pro	mpt for argu	iment	s
Treat output as Unicod	le	√ Clo	se on exit		
	0	K	Cance	I	Apply

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:

c:•. S	ímbolo del sistema	_		\times
C:\Us Jsage	sers\Emmanuel Jordán\Des e: avrdude [options]	ktop\Programas\AVRDUDESS>avrdude		^
optio	ons:			
-р	<partno></partno>	Required. Specify AVR device.		
-b	<baudrate></baudrate>	Override RS-232 baud rate.		
-B	<bitclock></bitclock>	Specify JTAG/STK500v2 bit clock period (us).		
-C	<config-file></config-file>	Specify location of configuration file.		
- C	<programmer></programmer>	Specify programmer type.		
-D		Disable auto erase for flash memory		
-i	<delay></delay>	ISP Clock Delay [in microseconds]		
-P	<port></port>	Specify connection port.		
- F		Override invalid signature check.		
-e		Perform a chip erase.		
-0		Perform RC oscillator calibration (see AVR05	3).	
-U	<memtype>:r w v:<filena< td=""><td>me>[:format]</td><td></td><td></td></filena<></memtype>	me>[:format]		
		Memory operation specification.		
		Multiple -U options are allowed, each request	t	
		is performed in the order specified.		
-n		Do not write anything to the device.		
-V		Do not verify.		
-u		Disable safemode, default when running from a	a script	
-5		Silent safemode operation, will not ask you	if	
		fuses should be changed back		
-+		Enter terminal mode		
-F	<pre>(exitspec)[<exitspec)]< pre=""></exitspec)]<></pre>	list programmer exit specifications		
	(extended naram)	Pass (extended naram) to programmer		
-1/	Concentraca_partains	Count # erase cycles in FEPROM		
y -∨		Initialize erase cycle # in FEPROM		
- 1		Verbose output $-y - y$ for more		
- 0		Quall prograss output		
1	logfilo	Use logile nother than stdern for diagnostic		
-T	TOBLITE	Display this wasse	.5.	
- ŗ		Display this usage.		
avrdı	ude version 6.1. URL: <h< td=""><td>ttp://savannah.nongnu.org/projects/avrdude/></td><td></td><td></td></h<>	ttp://savannah.nongnu.org/projects/avrdude/>		
	,			

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 \rightarrow New \rightarrow Project

2) GCC C Executable Project \rightarrow OK

ŏ	Start Page - AtmelStudio (Administr Standa	rd Mo	de 【	Quick Launch (Ctrl+Q)	_ م	□ ×
File	Edit View VAssistX ASF Project	Deb	ug	Tools Window Help		
	New	•	殾	Project	Ctrl+Shift+N	E ÷
	Open	•	ٹ*	File	Ctrl+N	- 82
	Close Close Solution		As As	Atmel Start Project Atmel Start Example Project		 ×
	Import	•	₫	Example Project	Ctrl+Shift+E	

Figura 5. Creación de proyecto en AtmelStudio.

Recent	Sort by:	Default 👻 📰 📘		Search Installed Templates (Ctrl+E)	P
Installed		GCC C ASF Board Project	C/C++	Type: C/C++	
C/C++ Assembler AtmelStudio Solution		GCC C Executable Project	C/C++ 🗲	Creates an AVR 8-bit or AVR/ARM 32- bit C project	
	ecc	GCC C Static Library Project	C/C++		L
	occ	GCC C++ Executable Project	C/C++	include cavr/io.h>	l
	ecc	GCC C++ Static Library Project	C/C++	printf("unid)	l
	\odot	Create project from Arduino sketch	C/C++	· me110	l
				GCC	J
Name: C	iccApplica	tion1			
ocation:	:\Users\ta	ller2\Desktop\EMMA\Proyectos Atmel\		Browse	
olution name: 0	SccApplica	tion1		Create directory for solution	

Figura 6. Elección de tipo de proyecto en AtmelStudio.

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

Device Selection	-					×
Device Family:	All				328p	×
Name	App./Boot Memory (Kbytes)Data Memory (bytes)	EEPROM (bytes)	Device Info:		
ATmega328P	32	2048	1024	Device Name	ATmega328P	
ATmega328PB	32	2048	1024	Speed:	N/A	
					ОКС	ancel

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 \rightarrow 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* \rightarrow *Build* (*nombre proyecto*) y lo grabamos desde *Tools* \rightarrow *nombre de la placa*.

• A través de AVRDUDESS:

Una vez realizado el programa en *AtmelStudio*, lo compilamos desde *Build* \rightarrow *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*.

AVRDUDESS 2.6 (avrdude version 6.3)	- 🗆 X
Programmer (-c) Arduino Port (-P) Baud rate (-b) Bit clock (-B)	MCU (p) ATmega328P ~ Flash: 32 KB
COM4 V (115200 Flash C:\Users\Emmanuel Jordán\Desktop\MyEP 2018\MyEP\- EJERCICIOS\2- UAF	EEPROM: 1 KB Detect Presets Arduino Uno (ATmega328P)
Write Read Verify Go Format Auto (writing only)	Manager Fuses lock bits L 0x00 Read Write H 0x00 ✓ Set fuses E 0x00 Fuse settings LB 0x3F Read Write
Program! Stop Options ? -c arduino -p m328p -P COM4 -b 115200 -U flash:w:"C:\Users\Emmanuel Jor avrdude.exe: load data etuse data from input file 0x00: avrdude.exe: input file 0x00 contains 1 bytes avrdude.exe: reading on-chip efuse data: Reading ####################################	Additional command line args
Ready	

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.

<u>Nota</u>: 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)

\odot	ArduinoISP Arduin	o 1.6.13		—		\times		
Arch	ivo Editar Progran	na Herramientas	Ayu	da				
	Nuevo	Ctrl+N				Q		
	Abrir	Ctrl+O						
	Abrir Reciente	>			Δ			
	Proyecto	>		Ejemplos	Constr	uidos		
	Ejemplos	>		01.Basics			>	
	Cerrar	Ctrl+W		02.Digital			>	
	Salvar	Ctrl+S		03.Analog)		>	
	Guardar Como	Ctrl+Mavús+S		04.Comm	unicati	on	>	
		,		05.Contro	bl		>	
	Configurar Página	Ctrl+Mayús+P		06.Sensor	s		>	
	Imprimir	Ctrl+P		07.Display	/		>	
	Preferencias	Ctrl+Coma		08.Strings	;		>	
				09.USB			>	
	Salir	Ctrl+Q		10.Starter	Kit_Bas	icKit	>	
				11.Arduin	oISP		>	Arduino

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*.

lerramientas Ayuda				
Auto Formato	Ctrl+T			
Archivo de programa.				
Reparar codificación & Recargar				
Monitor Serie	Ctrl+Mayús+M			
Serial Plotter	Ctrl+Mayús+L			
WiFi101 Firmware Updater				
Placa: 'Arduino/Genuino Uno"	2	•		
Puerto	2	•		
Obtén información de la placa		_		
Programador: "ArduinoISP"	;		AVR ISP	
Quemar Bootloader			AVRISP mkll	
			USBtinyISP	
		•	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*).

🞯 ArduinoISP Arduino 1.6.13
Archivo Editar Programa Herramientas Ayuda
ArduinoISP §
<pre>// The standard pin configuration. #ifndef ARDUINO_HOODLOADER2</pre>
<pre>#define RESET 10 // Use pin 10 to reset the target rather than SS #define LED_HB 9 #define LED_ERR 8 #define LED_PMODE 7</pre>
<pre>// Uncomment following line to use the old Uno style wiring // (using pin 11, 12 and 13 instead of the SPI header) on Leonardo, Due</pre>
<pre>#define USE_OLD_STYLE_WIRING</pre>
<pre>#ifdef USE_OLD_STYLE_WIRING</pre>
<pre>#define PIN_MOSI 11 #define PIN_MISO 12 #define PIN_SCK 13</pre>
<pre>#endif Figura 12. Quemando bootloader en microcontrolador, paso 3.</pre>

- 4) Verificamos y grabamos el programa.
- 5) Conectamos los pines de GND, 5V, SCK, MOSI y MISO de la <u>placa grabadora</u> con los correspondientes del <u>microcontrolador</u> 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 \rightarrow 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*" \rightarrow "*Config Bits*" (PIC C Compiler) y aparece una ventana como la siguiente:

PIC16F 628 A ~						
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	<pre>//Sin programación con bajo voltaje B3(PIC16) o B5(PIC18)</pre>
#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:

BODLEVEL [2:0] Fuses	Min. V _{BOT}	Тур. 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				

Table 32-8. BODLEVEL Fuse Coding⁽¹⁾⁽²⁾

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.

o **RSTDISBL** → deshabilitación del Reset en el pin 1/PC6.
 l: Reset habilitado (PC6 = RESET),

0: Reset deshabilitado (PC6 = PICINTI4). Cuidado! No permite otra grabación.

- \circ **DWEN** → habilitación del DebugWire. Cuidado!
- SPIEN → habilitación de Programación Serie. Cuidado!
- O WDTON → habilitación del Watch Dog Timer
- **EESAVE** → preservación de la memoria EEPROM ante un borrado de chip
- **BTSZ1:0** → tamaño de Boot Loader
- \circ BOOTRST \rightarrow 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⁾
SUTO	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* \rightarrow División del clock por 8
- $\circ \quad CKPUT \qquad \rightarrow Salida \ del \ clock \ por \ pin \ PBO$
- *SUTI:0* → Tiempo de "start-up". Junto con el bit CKSELO, y dependiendo de la fuente de oscilación presente, determina el tiempo de retardo al encendido.
- \circ *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:

EFuse = 0xFD = 0b11111101

• Reseteo a 2.7Vcc.

LFuse = 0xF7 = 0b11110111

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

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
- o 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*.

AVRDUDESS 2.6 (avrdude version 6.3)		- 🗆 X	
Programmer (-c)	MCU (-p)		
Arduino	~	Almega328P V	
Port (-P) Baud rate (-b) Bit	clock (-B)	Flash: 32 KB	
COM4 ~ 19200		EEPROM: 1 KB Detect	
Flash		Presets	
C:\Users\Emmanuel Jordán\Desktop\MyEP 2018\MyEP\- EJE	RCICIOS\2- UAF	Arduino Uno (ATmega328P) 🛛 🗸	
● Write ○ Read ○ Verify Go Format A	uto (writing only) \sim	Manager	
EEPROM		Fuses lock bits	
		L 0xFF Read Write	
		H OxDE Set fuses	
Write O Read O Venty Go Format A	uto (writing only)		
Options			
Force (-F) Erase flash and EEPROM (-e	LB 0x3F Read Write		
Disable verify (-V) Do not write (-n)	Set lock		
□ Disable flash erase (-D) Verbosity 0 ∨		Bit selector	
Program! Stop	Options ?	Additional command line args	
-c arduino -p m328p -P COM4 -b 19200 -U flash:w:"C	:\Users\Emmanuel Jord		
avrdude.exe: input file 0xFD contains 1 bytes			
avrdude.exe: verifying avrdude.exe: 1 bytes of efuse verified			
avrdude.exe done. Thank you.			
		\vee	
Ready		.::	

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.

AVRDUDESS 2.6 (avrdude version 6.3)	- 🗆 X
Programmer (-c)	MCU (-p)
Arduino 🗸	ATmega16 ~
Port (-P) Baud rate (-b) Bit clock (-B)	Flash: 16 KB
COM4 ~ 19200	EEPROM: 512 B Detect
	Prosto
	Arduino Uno (ATmega328P) V
○ Write ● Read ○ Verify Go Format Hexadecimal (reading (∨	Manager
EEPROM	Fuses lock bits
	L 0xFF Read Write
	H 0xDE Set fuses
	F Duos Fuse settings
Options	
Force (-F) Erase flash and EEPROM (-e)	LB 0x3F Read Write
Disable verify (-V) Do not write (n)	Set lock
Disable flash erase (-D) Verbosity 0 ~	Bit selector
Program! Stop Options ?	Additional command line args
-c arduino -p m16 -P COM4 -b 19200 -U flash:r:C:/miArchivo.hex:i	-U flash:r:C:/miArchivo.hex:i
~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~	<u>^</u>
avrdude.exe: AVR device initialized and ready to accept instructions	
avrdude.exe: Device signature = 0x1e950f (probably m328p) avrdude.exe: reading flash memory:	
Acauring ***********************************	×
Ready	

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.