Arduino Uno as ICSP Programmer for Bare Bones ATmega328

By: Bob Glicksman; v1.0; 7/20/24

Introduction.

This document provides complete instructions for programming a “bare bones” ATmega328 (or ATmega328P) chip using “In Circuit Serial Programming” (ICSP). Bare-Bones means a factory-fresh chip that you will be using:

- With the internal 8 MHz clock (no crystal/capacitors for the external 16 MHz clock used on Uno and other Arduino boards).
- The chip does not come with a bootloader already flashed onto it.

The Arduino IDE (version 1.8.7) is used for both software development and for uploading compiled code to the chip. An Arduino Uno (R3) is used as the hardware ICSP programmer.

The information herein is based upon the official Arduino documentation that can be found at: https://docs.arduino.cc/built-in-examples/arduino-isp/ArduinoToBreadboard/

The Arduino documentation focuses on the use of ICSP to upload the Arduino bootloader to the bare bones chip. Thereafter, user-created programs are uploaded to the chip using the bootloader. This document describes a procedure that uses ICSP exclusively and overwrites the bootloader. There are several reasons for our wanting to do this:

- The bootloader takes up 512 bytes of the 32 Kbytes of flash memory on the ATmega328 chip. This is small and usually unimportant.
- When the bootloader is used, resetting the ATmega328 executes the bootloader first. When no new code is available for uploading (particularly when no programmer is attached to the chip’s serial I/O port), the bootloader times out before transferring control to the existing user’s program. The timeout delay is short, however this delay can be important for applications that are battery powered and use RESET to wake up from deep sleep, perform a small function, and then go back to deep sleep. Adding bootloader execution time to the regular wakeup execution time can significantly add to battery drain.
- The bootloader uses the (single) hardware serial port on the ATmega328. If the user’s application also uses the serial port, there can be a conflict if the chip is to be

programmed in-circuit. Extra components will be necessary in the circuit to allow both an external serial programmer and the internal (to the project) circuitry to use the one hardware serial port without conflict.

This document describes the procedure to upload code to both the ATmega328 and the ATmega328P chips. The -P version of the chip can be powered down to a much lower power draw then the non-P version (< ⅓ microAmp as opposed to about 25 microAmps). The -P version also has a few more machine language instructions than does the non-P version, but this does not seem to make any difference when programming in the Arduino programming language ("Wiring") using the Arduino IDE.

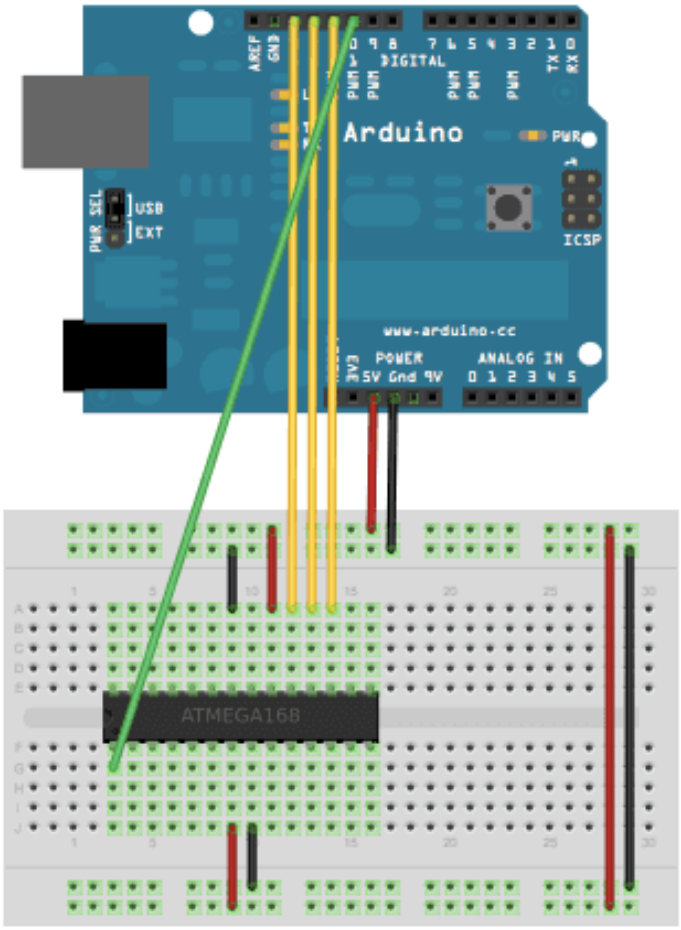
Hardware Requirements.

The following hardware is needed for this project:

- 1 ea. Arduino Uno (R3), or clone.
- 1 ea. Solderless breadboard.
- 2 ea. 0.1 uF decoupling capacitors.
- Male-male jumper wires
- ATmega328-PU or ATmega329P-PU microcontroller chips (without bootloader):
    - non-P version: Digikey P/N ATMEGA328-PU-ND
    - -P version: Digikey P/N ATMEGA328P-PU-ND
- (optional) 28 pin, 0.3" ZIF socket: Adafruit P/N 382
- (optional) 3 ea LED and 3 ea resistor (220 or 330 ohm) – wire per the instructions in the comments in the ArduinoISP program if you want these status indicators.

Hardware Setup for Using Uno as ISP Using ICSP Protocol.

The following figure shows how to wire an Arduino Uno to the breadboard in order to use it as an ISP. This figure is taken directly from the official Arduino documentation:

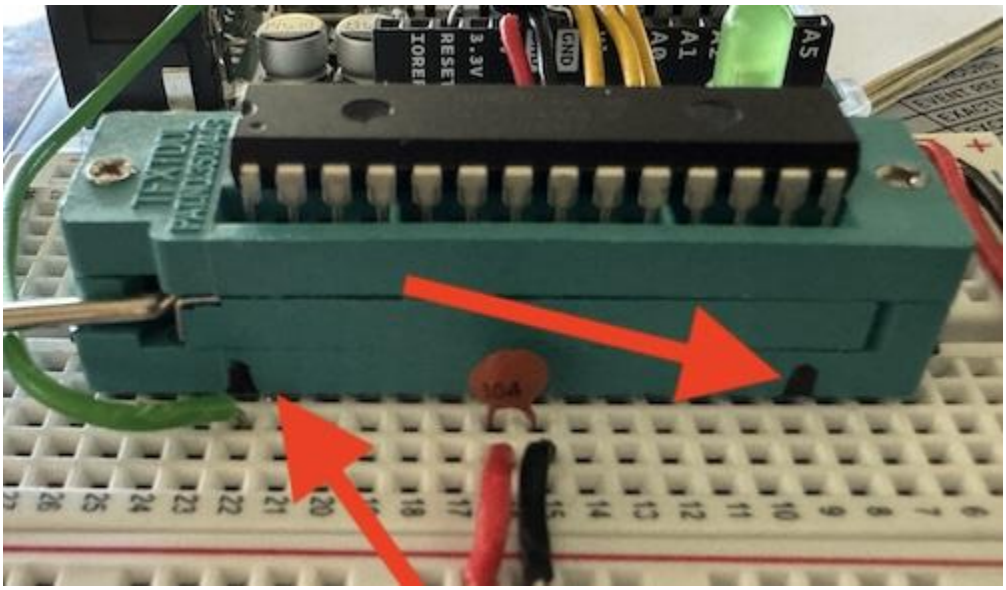


Here are the instructions:

1. Wire the breadboard power rails: Wire the + power bus on one side of the breadboard to the + power bus on the opposite side of the breadboard.
2. Wire the breadboard ground rails: Wire the - (gnd) power bus on one side of the breadboard to the - (gnd) power bus on the opposite side of the breadboard.
3. Wire the Uno GND pin to the closest - (gnd) power bus on the breadboard
4. Wire the Uno +5 volt power pin to the closest + power bus on the breadboard.
5. Place the target ATmega328 or ATmega328P chip on the breadboard, as shown in the diagram. OPTIONALLY, place a 28 pin ZIF socket where the ATmega is shown. Note that in the diagram above, ATmega pin 1 (notch at the top of the chip) is to the left.
6. Wire IC pin 7 to the closest + power rail.
7. Wire IC pin 8 to the closest - (gnd) power rail.

8. Place a 0.1 uF decoupling capacitor between IC pins 7 and 8, close to the power wiring to the chip.
9. Wire IC pin 20 to the closest + power rail.
10. Wire IC pin 22 to the closest - (gnd) power rail.
11. Place a 0.1 uF decoupling capacitor between IC pins 20 and 22, close to the power wiring to the chip.
12. Wire Uno digital pin 10 to pin 1 of the IC on the breadboard.
13. Wire Uno digital pin 11 to pin 17 of the IC on the breadboard.
14. Wire Uno digital pin 12 to pin 18 of the IC on the breadboard.
15. Wire Uno digital pin 13 to pin 19 of the IC on the breadboard.

TIP: If using the ZIF socket, then with a Sharpie marker, put four marks on the ZIF case at the pin position for each of the ends of the ZIF socket. This makes it easier to map ATmega chip pins to the solderless breadboard wire holes.



Overview of Operating Instructions.

Detailed operating instructions are presented in the next section of this document. Here is an overview of the entire process:

1. Install a “hardware/breadboard” folder into your Arduino Uno IDE sketchbook. This folder is necessary to tell the IDE about the chip that you are going to program. Optionally edit the file “boards.txt” to change from the default -P chip to the non-P version of the chip.

2. Open the Arduino IDE and upload the example sketch “ArduinoISP” to your Uno board. This makes your Uno into an ICSP programmer for your bare bones chips, using the wiring described above.

3. Flash the Arduino bootloader to the ATmega328(P).

4. Configure the Arduino IDE for programming the bare bones chip.

5. Compile and verify your program on the Arduino IDE.

6. Upload your program to the bare bones chip using ICSP.

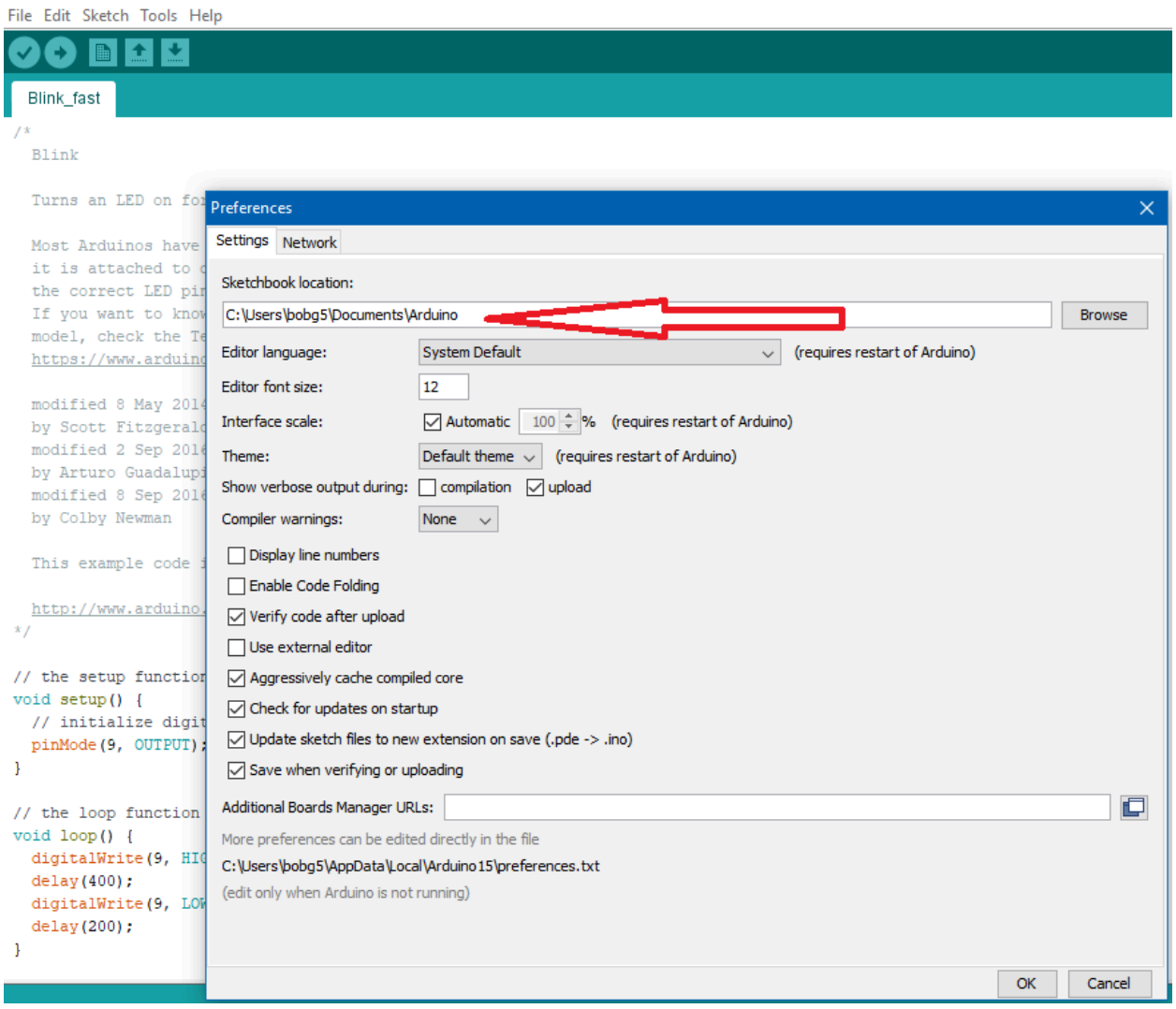
NOTE: This procedure will flash the Arduino bootloader to the bare bones chip but will subsequently overwrite it. You need to flash the bootloader because this process also sets necessary fuses and locks on the bare bones chip. After executing this procedure in full the first time, you can skip steps 1-3 thereafter, as the fuses and locks will remain properly set. The bootloader program itself will be overwritten, but it is not needed when using this procedure. You can, however, always restore the bootloader to the bare bones chip by following steps 1-3 of this procedure.

Detailed Operating Instructions.

This section provides step by step detailed instructions for each of the steps listed above.

1. Install the “hardware/breadboard” folder into your Arduino sketchbook.

You can find the location of your Arduino sketchbook using the Arduino IDE. Open the IDE and select “File : Preferences”. You will see the location of your sketchbook here:



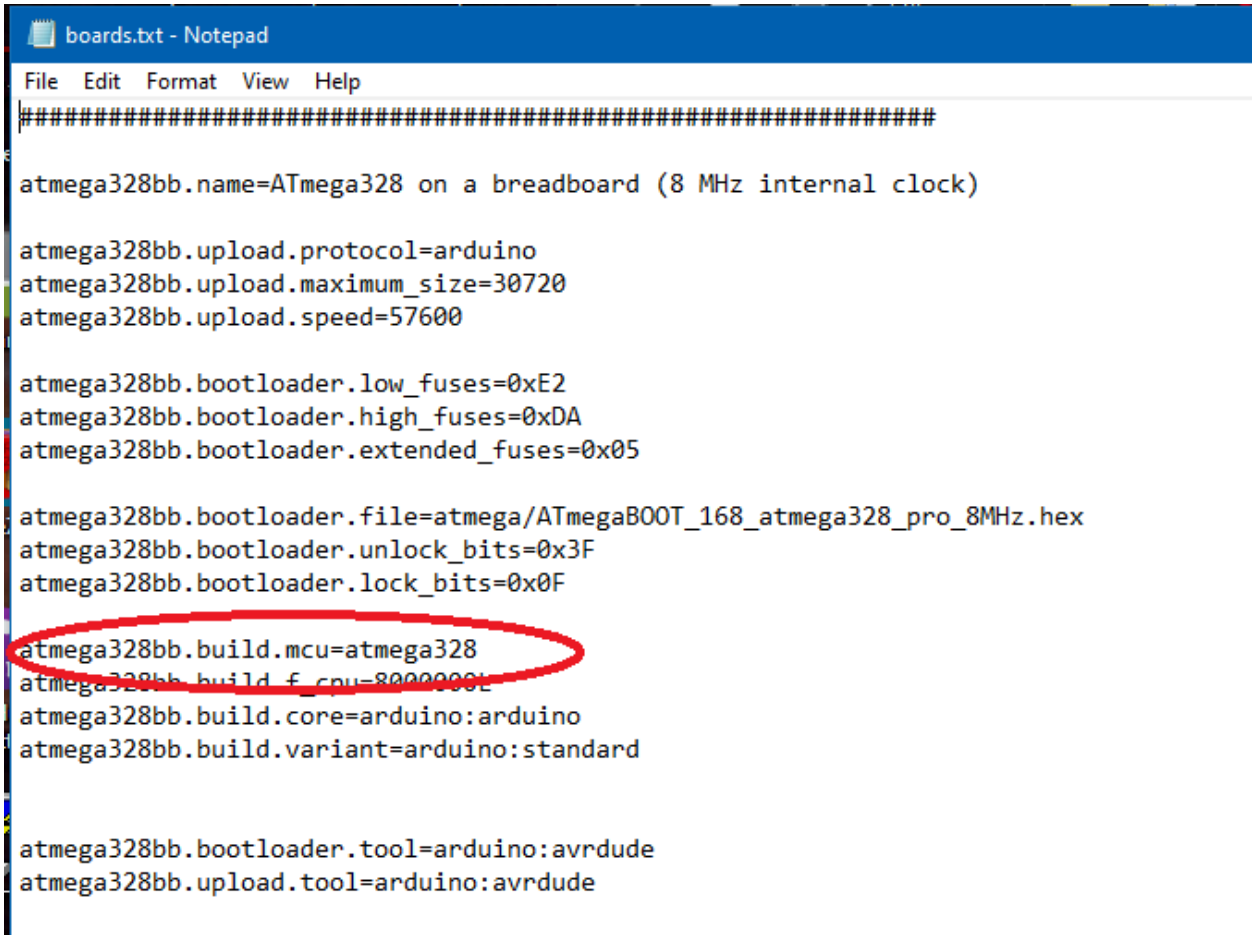
Open the sketchbook location on your computer using your computer's file explorer. See if there is already a folder called "hardware" here. If not, create an empty folder called "hardware".

Download the following zip file from the Arduino website:
https://www.arduino.cc/en/uploads/Tutorial/breadboard-1-6-x.zip

Open this zip file and extract the contents. The contents will be a folder called "breadboard" with some subfolders and other files inside of it. Drag and drop the "breadboard" folder inside of the "hardware" folder in your Arduino sketchbook.

If you plan to use only the -P chips, then you don't need to do anything else in this step. If you plan to use non-P chips, follow these additional steps:

- Open the “breadboard” folder that you just installed into the “hardware” folder in your Arduino sketchbook location.
- Open the “avr” folder.
- Open the file “boards.txt” in any text editor (e.g. Notepad on a Windows PC). It will look like the following figure:

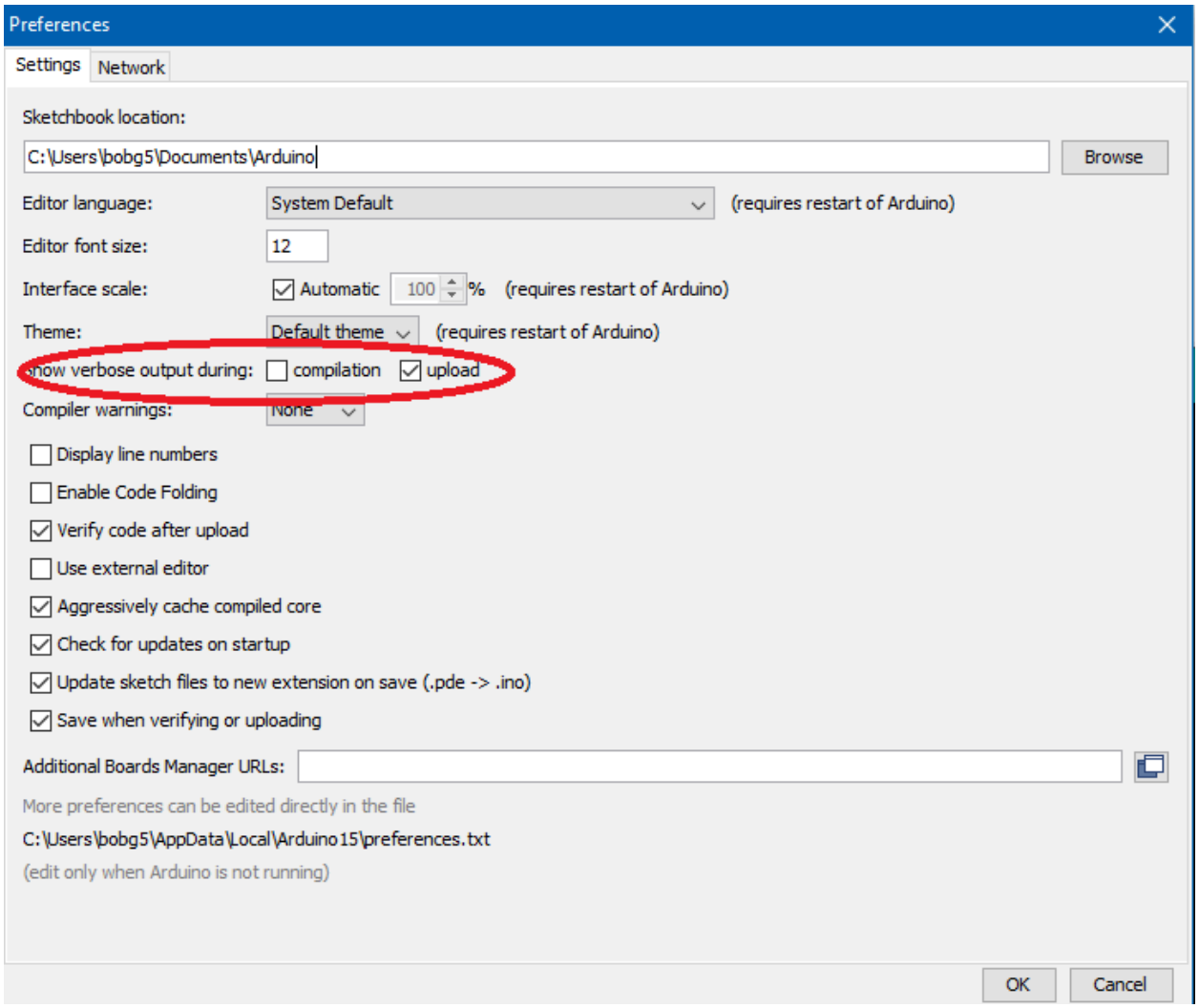


Note the circled line in the figure. The downloaded file will say “atmega328bb.build.mcu=atmega328p”. Note the “p” at the end of this line. This tells the Arduino IDE that the bare bones chip will be an ATmega328P chip. Remove the “p” at the end of this line (as shown in the figure) if you want to upload code to non-P chips. SAVE the edited file and exit your text editor.

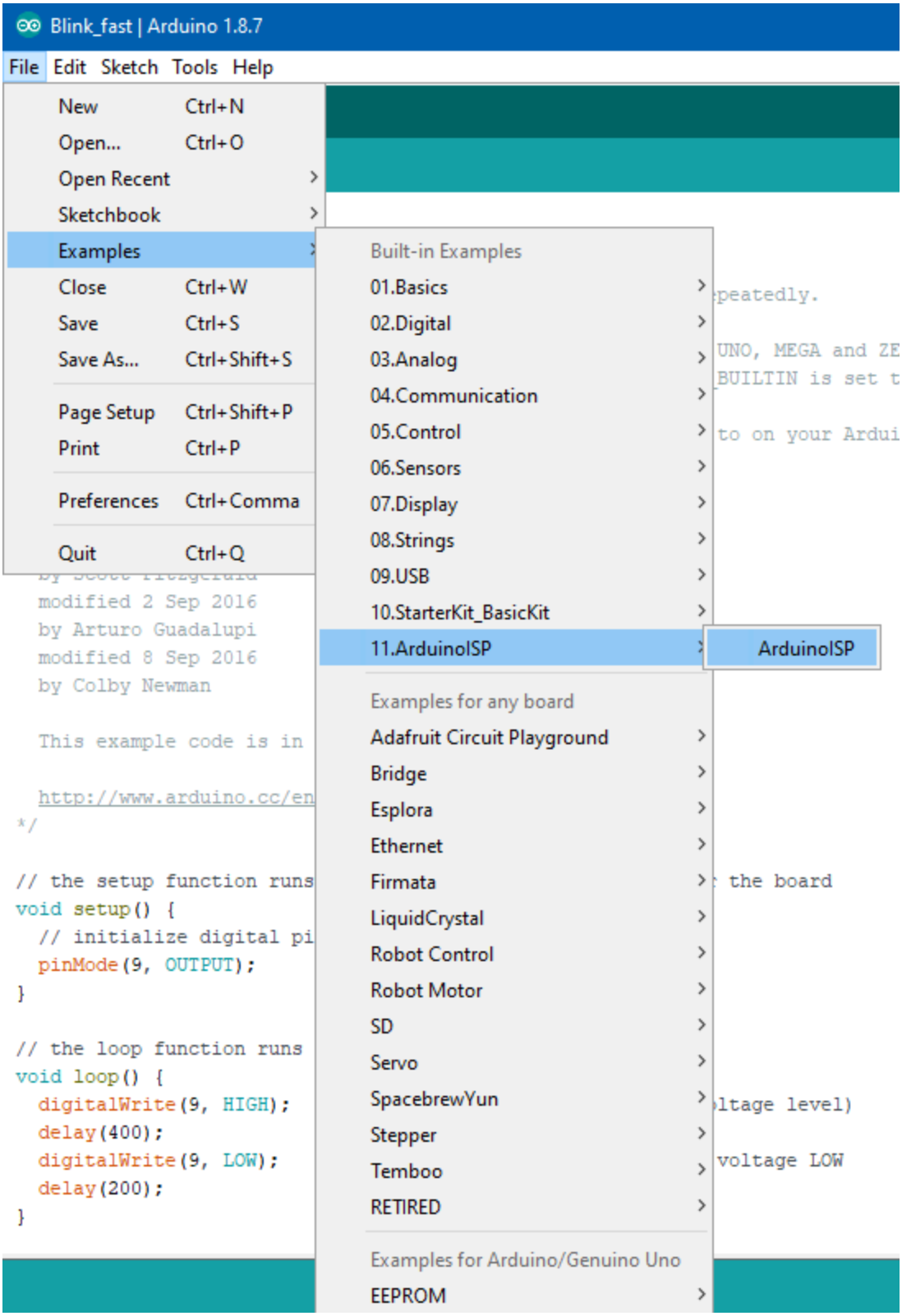
2. Upload “ArduinoISP” Code to Your Uno to Make It Into An In-System Programmer (ISP).

Open your Arduino IDE. If it is already open, you must close it and open it again.

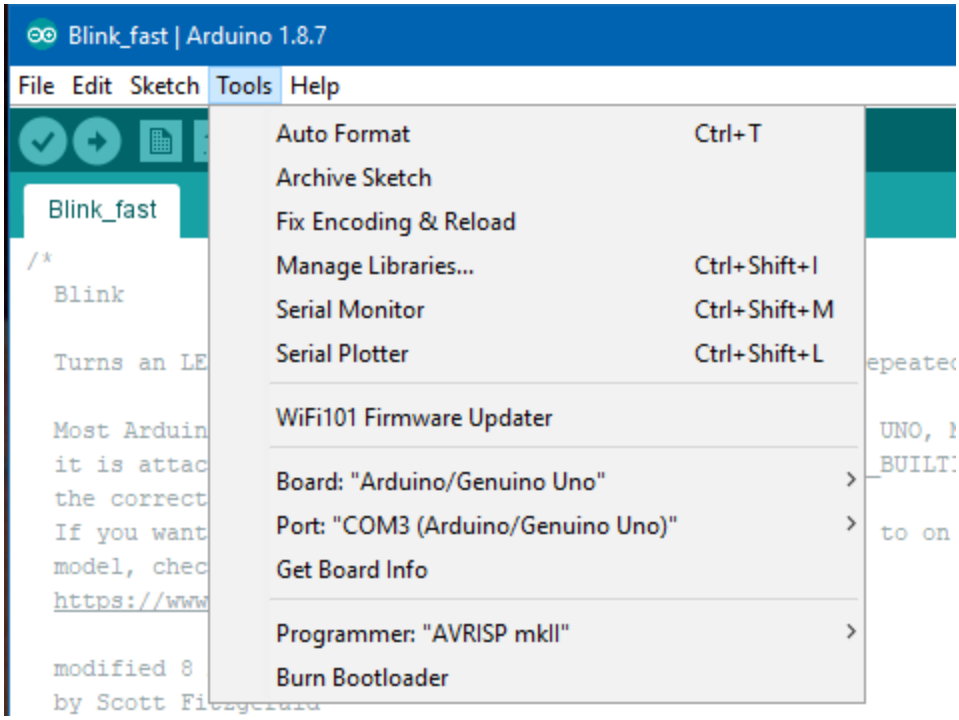
As an *optional* first step, we recommend that you enable “verbose” status on the IDE when uploading code. Open the Arduino IDE preferences and check the “upload” check box, as shown in the figure below. Then click “OK” to exit the preferences.



Next, you need to load the file “ArduinoISP” into the IDE, see the figure below. Select “File: Examples:11. ArduinoISP: ArduinoISP”. The IDE code window will show the source code for the Example file called “ArduinoISP”.



Before compiling and uploading “ArduinoISP” to your Uno board, make sure that your Tools settings are correct, see figure below:



Note the following in this figure:

- The “Board:” must be “Arduino/Genuino Uno”
- The “Port” will be the COM or tty port that your Uno is connected to (may not be COM 3, as in the above figure).
- The “Programmer” must be set to “AVRISP mkII”.

After double checking these Tools settings, click on the Compile and Upload button: the Right facing arrow at the top left of the IDE window. This will flash the ArduinoISP code to your Uno. NOTE: You only need to do this once - thereafter, the ArduinoISP code will be running when you power up your Uno (unless you change it, of course).

3. Flash the Arduino Bootloader to the Bare Bones Chip.

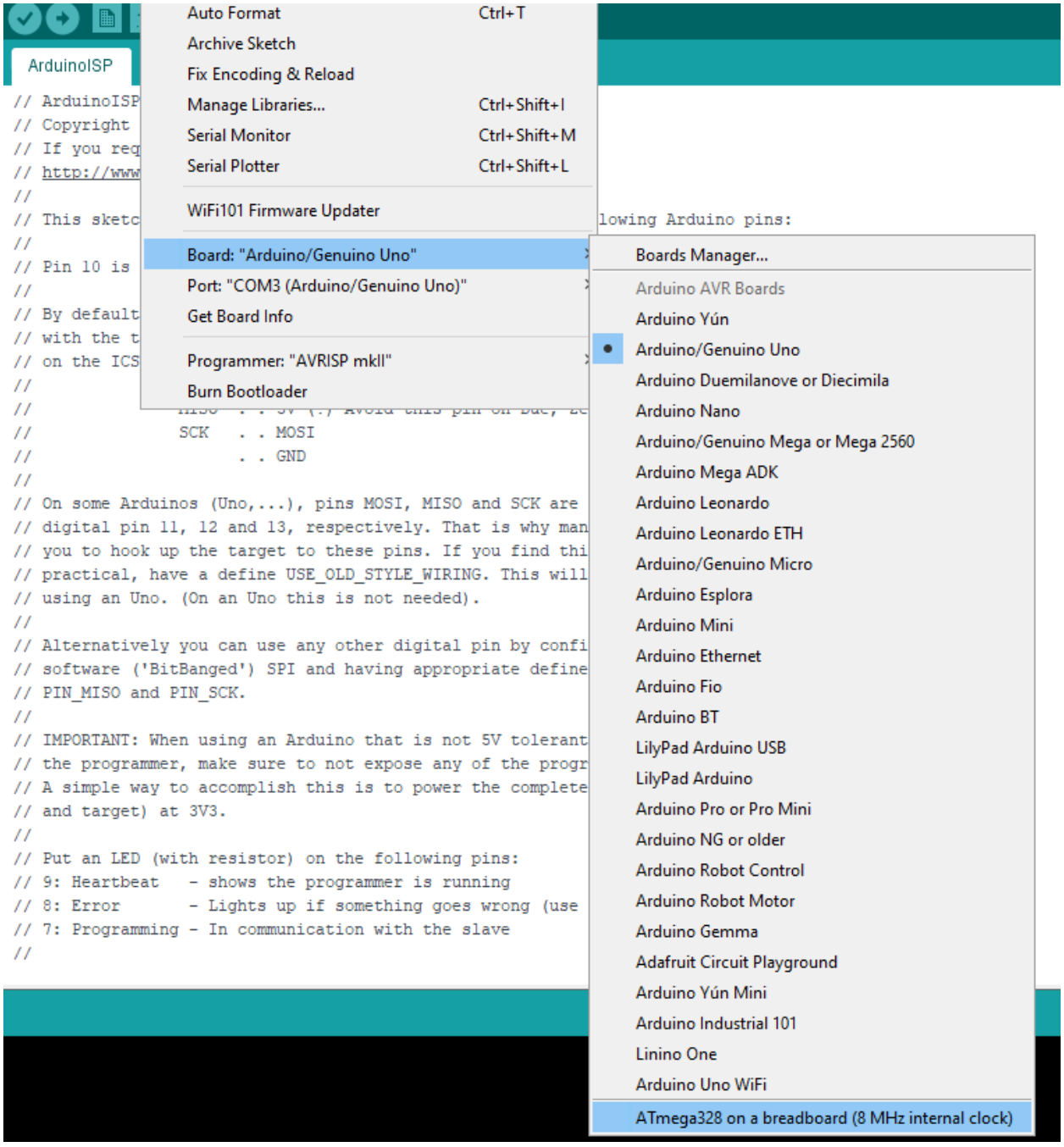
You must next flash the Arduino bootloader to the bare bones chip. This step is necessary because it sets fuses and locks on the bare bones chip that are needed for the chip to operate properly. This step will also flash the bootloader program to your bare bones chip, but the bootloader will be overwritten in the subsequent steps of this procedure.

MAKE SURE THAT YOU HAVE YOUR BARE BONES CHIP PROPERLY INSERTED INTO YOUR BREADBOARD OR ZIF SOCKET. MAKE SURE THAT THE WIRING FROM THE UNO TO YOUR BREADBOARD IS CORRECT! MAKE SURE THAT YOU HAVE SET YOUR

BOARD.TXT FILE TO THE CORRECT CHIP (-P or non-P) PER THE FIRST STEP IN THIS DETAILED PROCEDURE.

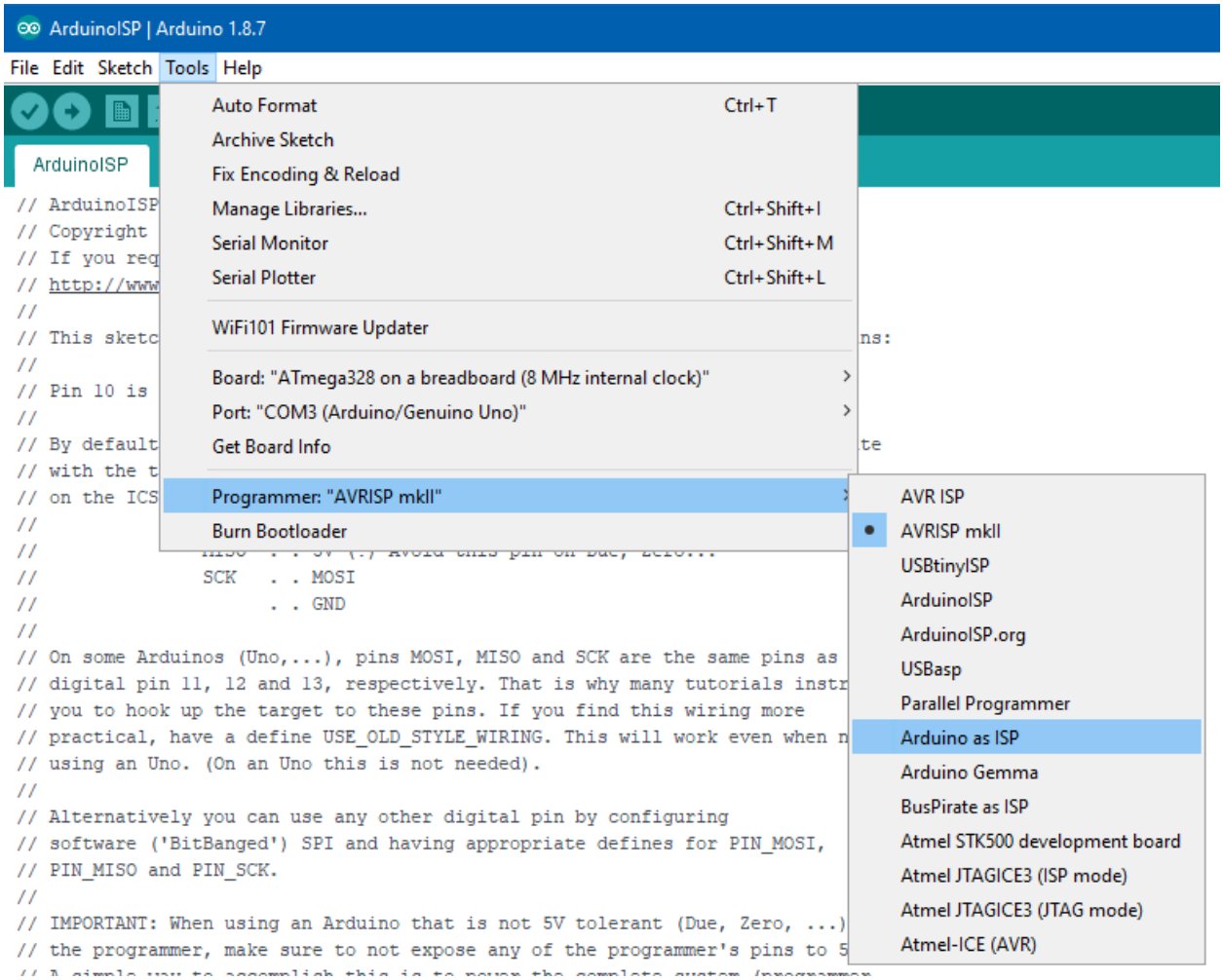
Open the “Tools” menu on the Arduino IDE and go to “Board:” Look down the drop-down list for “ATmega328 on a breadboard (8 MHz internal clock)”, as shown in the figure below. Select “ATmega328 on a breadboard (8 MHz internal clock)” as the board to be programmed.

NOTE: if you don’t see this option, then go back to the initial step of this procedure where you installed the “hardware/breadboard” folder into your sketch folder using the downloaded zip archive. Repeat the steps, close the Arduino IDE and then reopen it. You should now see this Board option in your “Tools:Board” list.

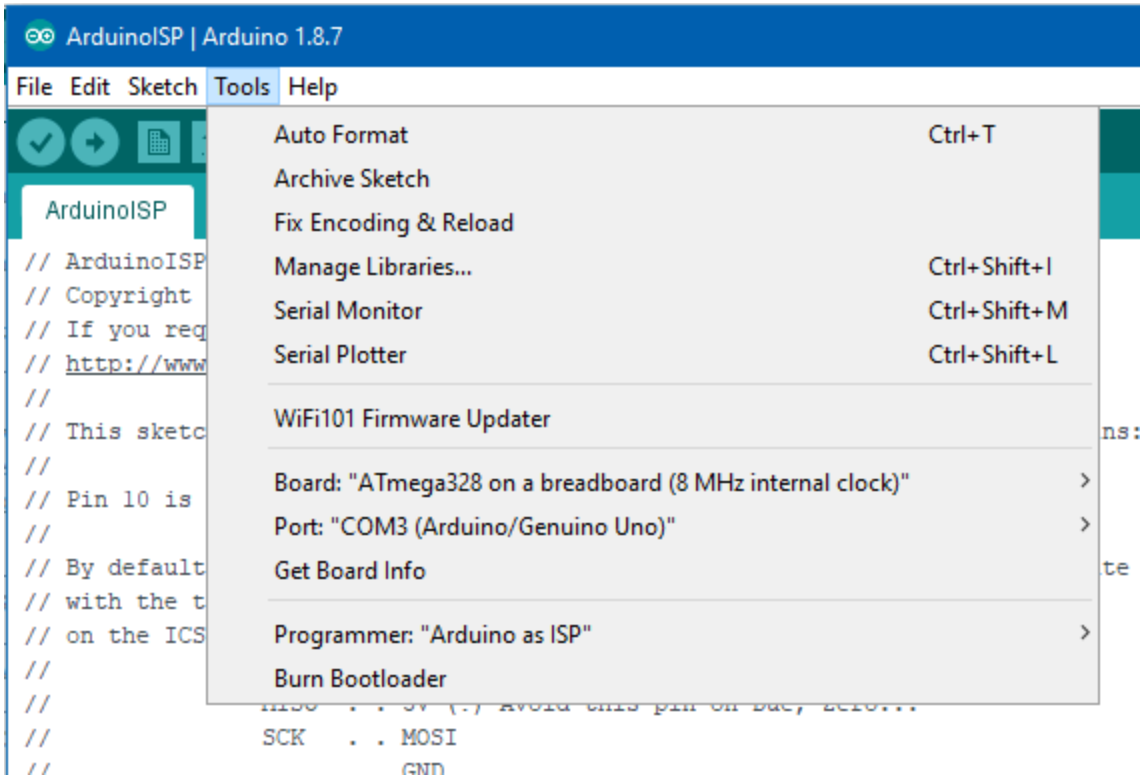


Next, go to “Tools” again and change the “Programmer” setting to “Arduino as ISP”

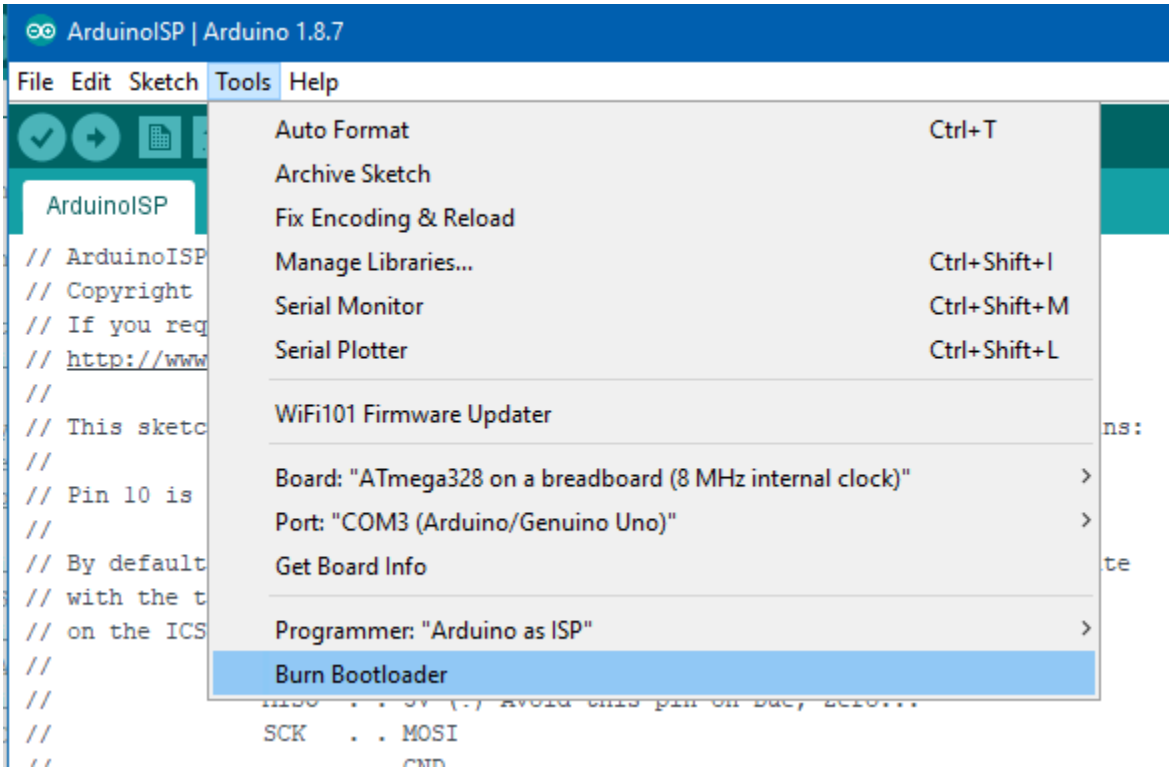
NOTE: do NOT confuse this with “ArduinoISP” which is also shown on the dropdown list. “ArduinoISP” is the program that you uploaded to your Uno in the previous step. In this step, you are selecting “Arduino as ISP” which tells the IDE to use your Uno as a programmer, rather than flashing new code to it.



Before proceeding, verify all of your IDE settings, as in the figure below:



Now, click on “Burn Bootloader” at the bottom of the Tools menu:

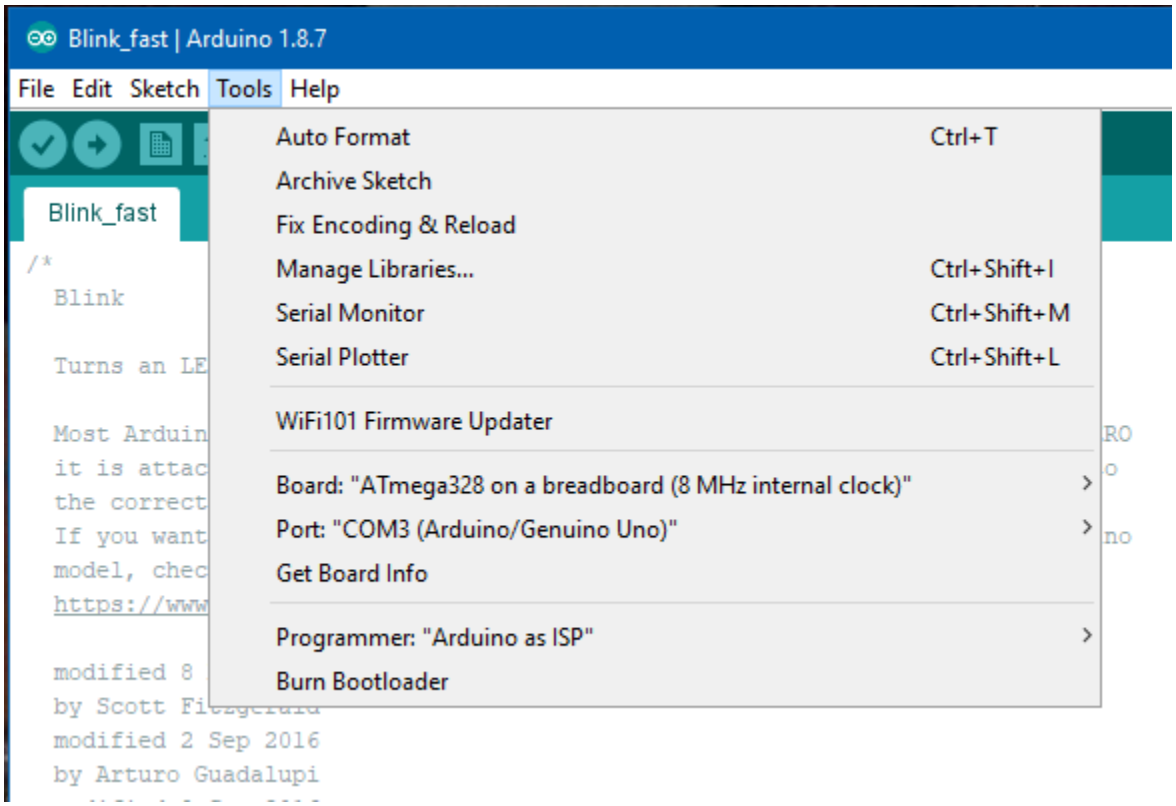


You will see the lights on the Uno flash for a bit while the fuses, locks and bootloader code are uploaded to your bare bones chip.

NOTE: if you get a signature error in the IDE status window, then you are probably configured for the wrong version of the ATmega328 chip (-P or not -P). Go back to the first step in this procedure and check the indicated line in the “boards.txt” file. Make the correction, save the file, close and open the IDE, and repeat this step.

4. Configure the Arduino IDE for ICSP Programming of a Bare Bones Chip.

You should have the proper settings in the Tools menu, but you should verify this, as in the figure below:

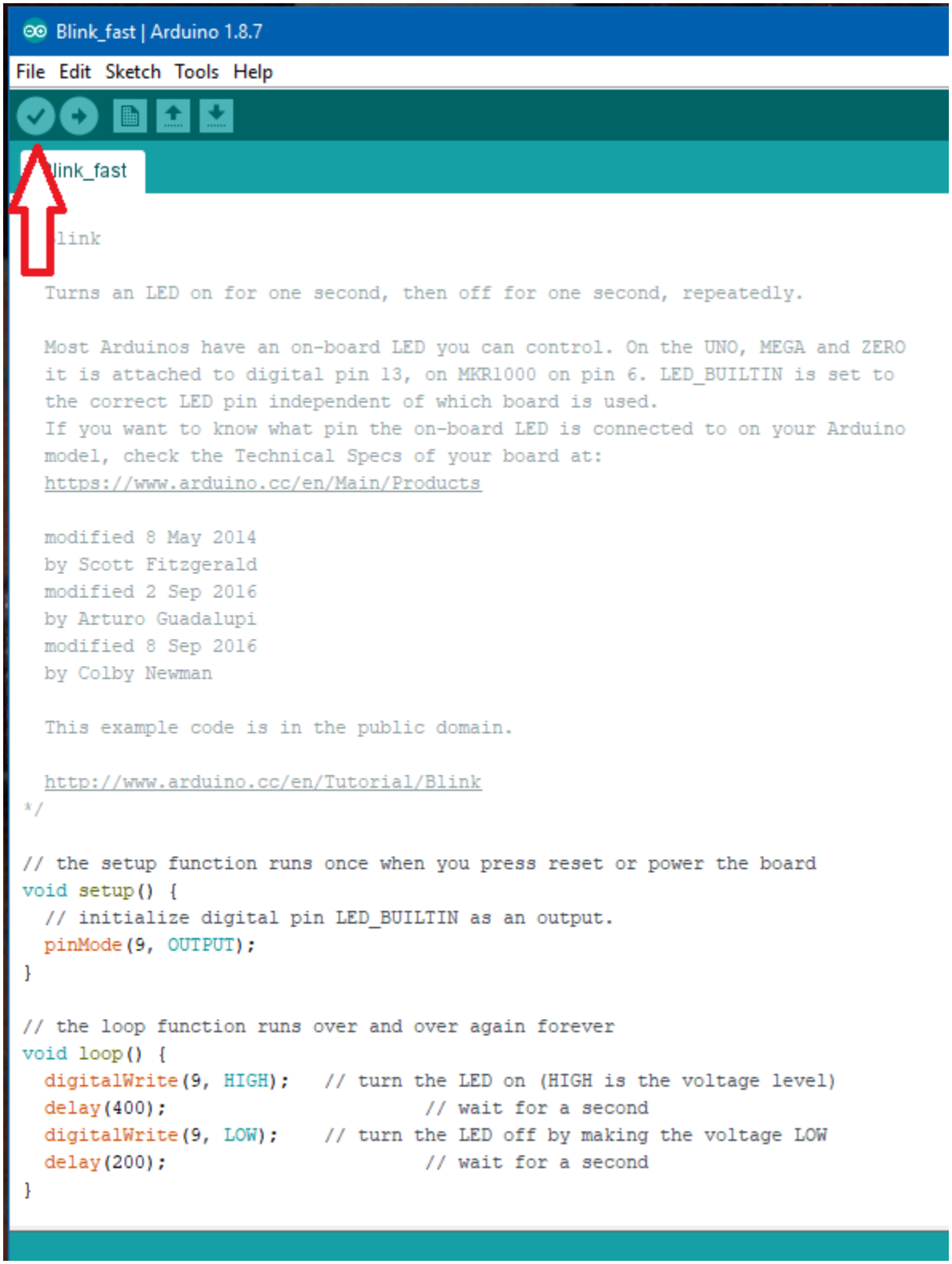


Double-check the following:

- “Board: ATmega328 on a breadboard (8 MHz internal clock)”
- “Port:” The COM or tty port that your Uno is connected to.
- “Programmer:” “Arduino as ISP”.

5. Compile and Verify Your Program on the Arduino IDE.

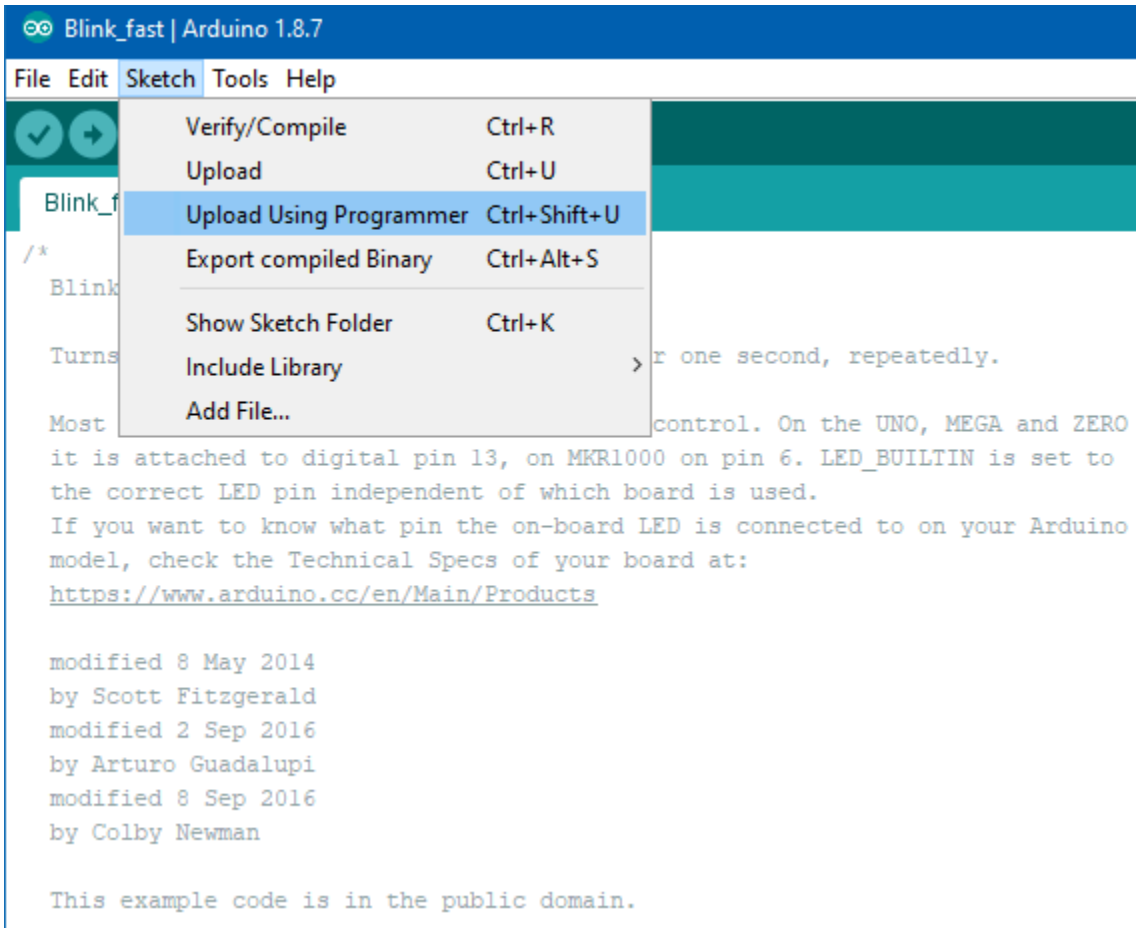
You are now ready to open or create a new program of your own that you want to upload to the bare bones chip. Proceed as you would for any program that you import, select, or create on the Arduino IDE. When your program is ready, you can compile and verify (but NOT upload) your code. You do this by clicking on the CHECK icon at the top left of the IDE (NOT the right arrow icon!), as shown in the figure below:


Blink_fast | Arduino 1.8.7
File Edit Sketch Tools Help
link_fast
link
Turns an LED on for one second, then off for one second, repeatedly.
Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
the correct LED pin independent of which board is used.
If you want to know what pin the on-board LED is connected to on your Arduino
model, check the Technical Specs of your board at:
https://www.arduino.cc/en/Main/Products
modified 8 May 2014
by Scott Fitzgerald
modified 2 Sep 2016
by Arturo Guadalupi
modified 8 Sep 2016
by Colby Newman
This example code is in the public domain.
http://www.arduino.cc/en/Tutorial/Blink
*/
// the setup function runs once when you press reset or power the board
void setup() {
// initialize digital pin LED_BUILTIN as an output.
pinMode(9, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
digitalWrite(9, HIGH); // turn the LED on (HIGH is the voltage level)
delay(400); // wait for a second
digitalWrite(9, LOW); // turn the LED off by making the voltage LOW
delay(200); // wait for a second
}

Check the status window on the IDE to make sure that your code has compiled and verified correctly!

6. Upload Your program to the Bare Bones Chip.

In order to upload your program to the bare bones chip using ICSP, you must open the “Sketch” menu on the Arduino IDE and select “Upload Using Programmer”:



After a short delay, you will see lights flashing on the Uno indicating that it is working as an ICSP programmer and uploading your sketch to the bare bones chip.

At this point the ATmega will be executing code while in the programmer. One tip is to attach and LED and appropriate resistor to one digital pin and have your setup() code blink this once. That way you can verify in the programmer that your download has worked.

Subsequent Uploads to the Bare Bones Chip.

Once you have successfully completed all of these steps, you need not repeat steps 1-3 in order to modify the code and upload it to this same chip. You only need to perform steps 4 - 6, unless you (a) change the code flashed to your Uno, or (b) wish to restore the bootloader to your bare bones chip or (c) wish to upload code to another factory-fresh chip.