**BURNING AVR MICROCONTROLLER USING ARDUINO: migrating from Arduino to a professional AVR.**

1. **Upload AVR ISP program to Arduino**

In tools menu, check your board type and ports – make sure they are correct.

Now, File -> Examples -> ArduinoISP

You should see ArduinoISP code in your IDE.

Then, Tools -> Programmer -> AVRISP mkII (just as usual)

Now, Upload this code to your Arduino by pressing upload button. **This makes your Arduino an ISP** (In system programmer).

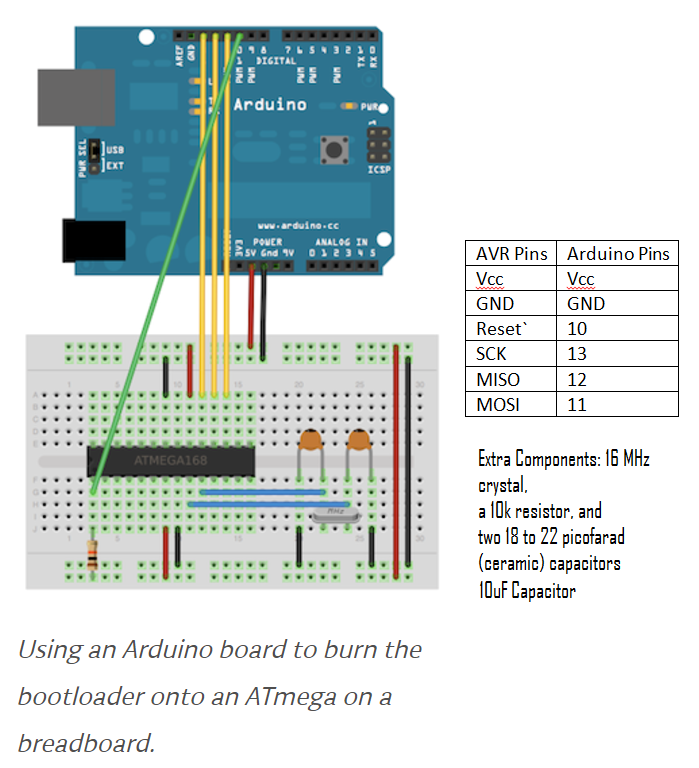
1. **Burn Bootloader**

Bootloader is a small program that lives in every Arduino. It is much like BIOS in computer. This, basically, tells the micro-controller how to communicate with the computer and paves the path to receive or send programs. Every micro-controller contains fuses bits. These fuses save basic parameters about micro-controller: such as clock frequency and how the chip resets itself. When we burn the Arduino bootloader to the chip, these fuses will be programmed correctly.

**Therefore, if you are using brand new chip, you should burn the bootloader else it not required**; if your vendor provided you the bootloaded chips, again, you don’t need to burn the bootloader. Also, in the case of chips that come along with Arduino, we don’t need to burn bootloader because it is already done for you. If you are not sure whether your chip is bootloaded or not, then you can do burn bootloader – this won’t damage your IC.

The good news is, bootloader is burnt only once to your chip.

**Here is how** you can burn the bootloader to your Atmel chip. Make the following connections:



Source: <https://www.arduino.cc/en/Tutorial/ArduinoToBreadboard>

If you are using some other Atmel micro-controller, the pin connections as in table should work for you. Also, note that you can burn bootloader without XTAL circuit, but this method is better for many reasons.

After connecting these pins, open a new Arduino IDE sketch.

NOTE: Connect 10µF polar capacitor to reset pin of Arduino in order to preempt auto-reset. Negative pole of capacitor should face Gnd.

Now, Tools -> Programmer -> Arduino as ISP

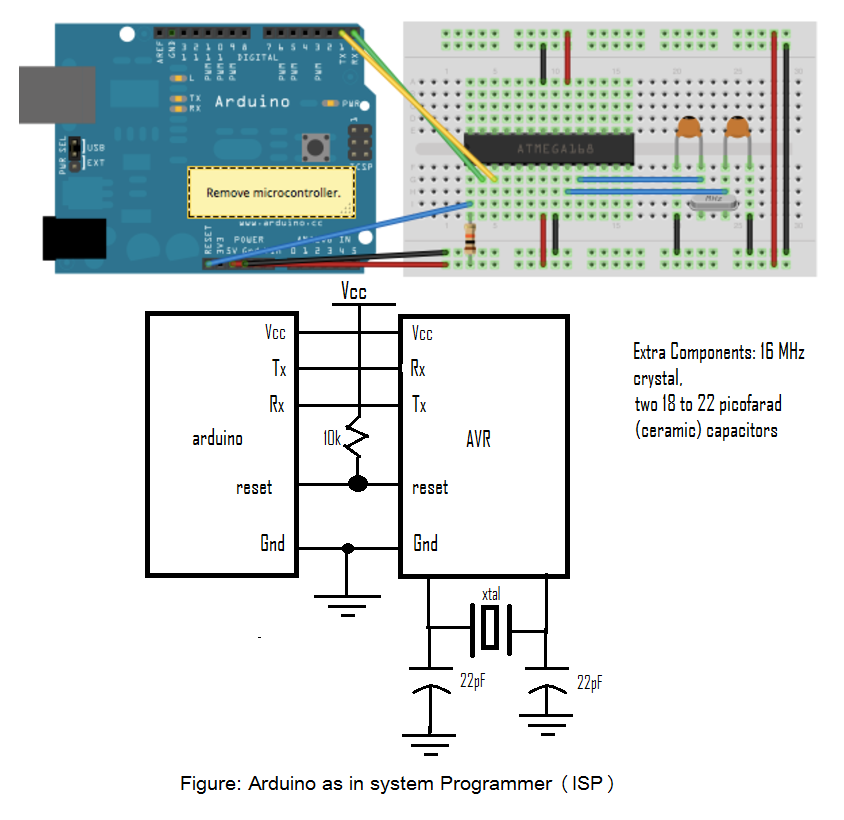
Note: You should disconnect all the external circuits connected to I/O pins. Do manually; or you can use tri-stated buffer: operate in high impedance mode.

Then: Tools -> Burn Bootloader.

This may take some minutes. You are done with bootloading if you see “done burning bootloader.”

1. **Uploading your code to your AVR**

3.1 Make the ensuing connections.



Source: <https://www.arduino.cc/en/Tutorial/ArduinoToBreadboard>

The configuration shown in the schematic diagram will work for any Atmel Micro-controller.

Open your sketch or create one right now. For example, you can choose Blink code for now.

Now you can upload programs to breadboard (You use the USB-to-serial convertor (FTDI chip) on an Arduino board).

* 1. Remove the microcontroller from the Arduino board so the FTDI chip can communicate to the microcontroller on the breadboard.

Note: Here also, you should disconnect all the external circuits connected to I/O pins. Do it manually; or you can use tri-stated buffer: operate in high impedance mode. Doing it manually should not be messy – for me it was handy.

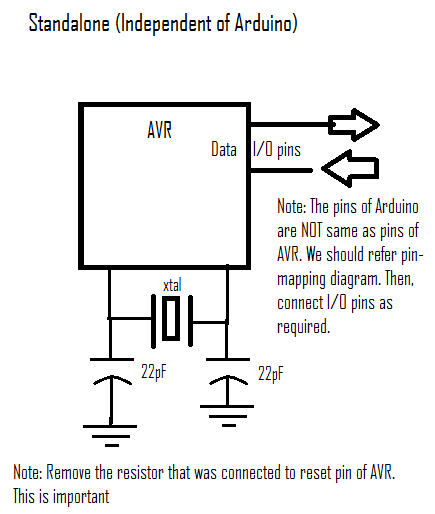
* 1. To program Atmega 328 or Atmega 168 microcontroller, select “Arduino Duemilanove” or “Nano w/ ATmega328” from Tools -> Board menu. Then upload as usual (i. e. you do NOT have to press shift to upload. Also, choose tools->programmer-> AVRISP mkll).

To program other Atmel chips: In the latest version of Arduino IDE, go to Tools -> Board -> Boards Manager and it shouldn’t be hard to get your Board type. If it still doesn’t work, or you are working from older IDE, just Google it – you will find plethora of resources on this. After that, upload as usual.

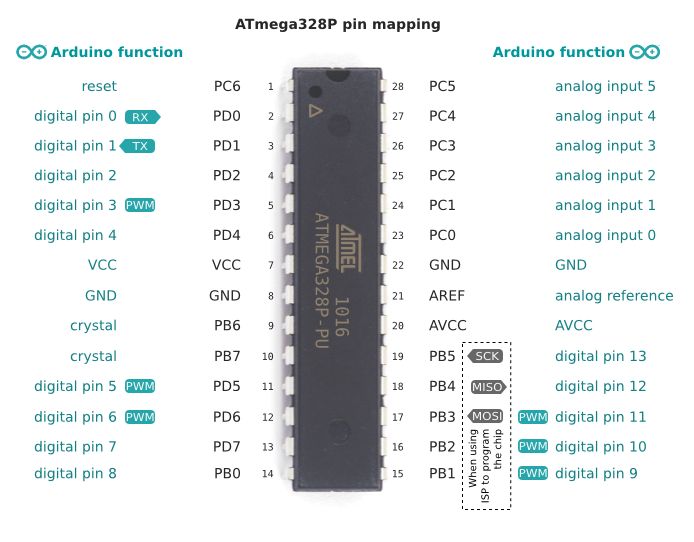
1. **Working independent of Arduino: Standalone System**

Get an external power source and remove all connections from Arduino.

Connect your desired circuit to this configuration. For example, for the Blink example code, you can connect your LED to pin 19.



1. **Getting Access to all pins after getting independent of Arduino**

****

**Image Source:** [**http://www.eevblog.com/forum/microcontrollers**](http://www.eevblog.com/forum/microcontrollers)

**For other Atmel chips, you will find similar pin-mapping if you Google it.**

**Cheers. Have a good one.**

**Author: Bigyan Chapagain**