Installing an Arduino Bootloader a learn.sparkfun.com tutorial

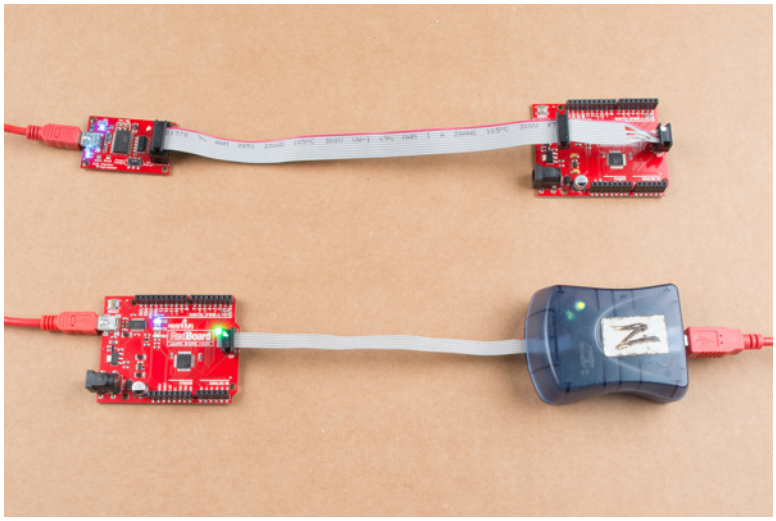
Available online at: http://sfe.io/t22

Contents

Introduction

Heads up! This tutorial was written for AVR microcontrollers with an Arduino bootloader using ICSP pins. If you are using an ARM microcontroller with SWD pins, you will need a dedicated programmer (i.e. Atmel JTAG ICE 3 or Atmel-ICE) to connect to the SWD port. For more information, check out our ARM programming tutorial.

Do you have a bricked Arduino that won't accept code anymore? Or, maybe you wrote your own firmware and would like to upload it to your Arduino? Or, maybe you just want to learn more about the inner-workings of Arduino, AVR, and microcontrollers in general. Well, you're in luck! This tutorial will teach you what a bootloader is, why you would need to install/reinstall it, and go over the process of doing so.



Suggested Reading

You may want to check out these tutorials before continuing down the bootloader path.

PCB Basics

What exactly IS a PCB? This tutorial will breakdown what makes up a PCB and some of the common terms used in the PCB world.

Serial Peripheral Interface (SPI)

SPI is commonly used to connect microcontrollers to peripherals such as sensors, shift registers, and SD cards.

What is an Arduino?

What is this 'Arduino' thing anyway? This tutorials dives into what an Arduino is and along with Arduino projects and widgets.

Installing Arduino IDE

A step-by-step guide to installing and testing the Arduino software on Windows, Mac, and Linux.

Integrated Circuits

An introduction to integrated circuits (ICs). Electronics' ubiquitous black chips. Includes a focus on the variety of IC packages.

Pocket AVR Programmer Hookup Guide

Skip the bootloader and load your program directly onto an AVR with the AVR Pocket Programmer.

What is a Bootloader?

Atmel AVRs are great little ICs, but they can be a bit tricky to program. You need a special programmer and some fancy **hex** files, and its not very beginner friendly. The Arduino has largely done away with these issues. They've put a **.hex** file on their AVR chips that allows you to program the board over the serial port, meaning all you need to program your Arduino is a USB cable.

The bootloader is basically a **.hex** file that runs when you turn on the board. It is very similar to the BIOS that runs on your PC. It does two things. First, it looks around to see if the computer is trying to program it. If it is, it grabs the program from the computer and uploads it into the ICs memory (in a specific location so as not to overwrite the bootloader). That is why when you try to upload code, the Arduino IDE resets the chip. This basically turns the IC off and back on again so the bootloader can start running again. If the computer isn't trying to upload code, it tells the chip to run the code that's already stored in memory. Once it locates and runs your program, the Arduino continuously loops through the program and does so as long as the board has power.

Why Install a Bootloader?

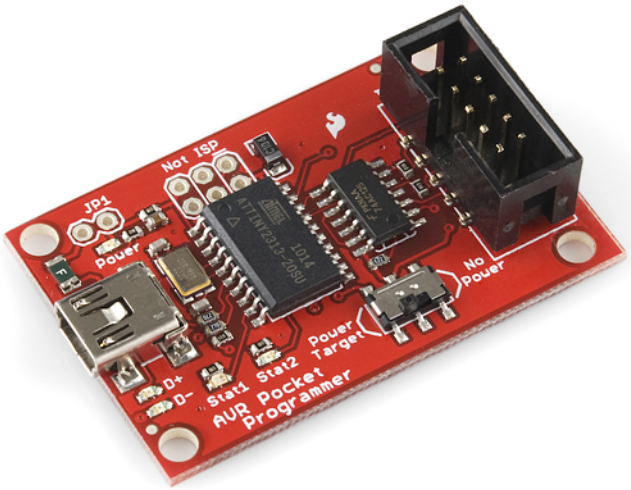
If you are building your own Arduino, or need to replace the IC, you will need to install the bootloader. You may also have a bad bootloader (although this is very rare) and need to reinstall the bootloader. There are also cases where you've put your board in a weird setting and reinstalling the bootloader and getting it back to factory settings is the easiest way to fix it. We've seen boards where people have turned off the serial port meaning that there is no way to upload code to the board, while there may be other ways to fix this, reinstalling the bootloader is probably the quickest and easiest. Like I said, having a bad bootloader is actually very very rare. If you have a new board that isn't accepting code, 99.9% of the time its not the bootloader. For the other 1% of the time it is, this guide will help you fix that problem.

Selecting a Programmer

We are going to talk about two different types of programmers you can use to install or reinstall bootloaders.

Option 1: Dedicated Programmers

For a quick easy programmer we recommend looking into the AVR Pocket Programmer (Windows only).



Pocket AVR Programmer

PGM-09825
$19.95

Or, you can use the official Atmel-ICE programmer for ARM, SAM, and AVR.

Atmel-ICE Programmer and Debugger

PGM-14950
Retired
Note: The following programmers that have been retired also can be used.

AVR MKII *JTAG ICE3*

The AVR Pocket Programmer or most cheaper options will work just fine for most applications, but they may have problems with some boards, specifically ones with lots of memory like the ATMega2560 based boards.

Option 2: Using the Arduino as a Programmer

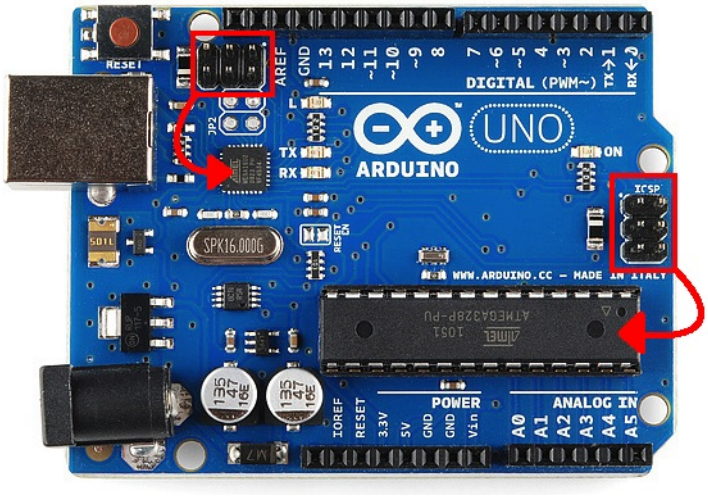
The other option is grabbing an Arduino Uno (or Duemilanove). If you go into the Arduino IDE you will see an example sketch called**Arduino as ISP.'** If you

upload this code to your Arduino, it will basically act as an AVR programmer. This isn't really recommended for production of boards, or boards with lots of memory, but, in a pinch, it works pretty well. Also as of this writing the code only works on ATmega328 boards. Maybe one day it will work on the Leonardo or Due, but not yet.

Hardware Hookup

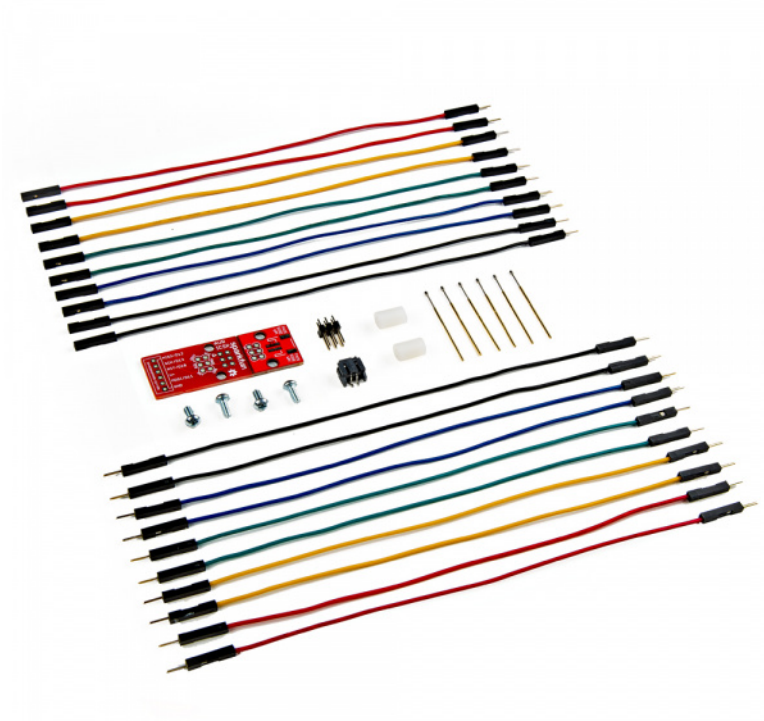
In-Circuit Serial Programming (ICSP)

It's very uncommon to program ICs before they are soldered onto a PCB. Instead, most microcontrollers have what's called an in-system programming (ISP) header. Particularly, some IC manufacturers, such as Atmel and Microchip, have a specialized ISP method for programming their ICs. This is referred to as in-circuit serial programming (ICSP). Most Arduino and Arduino compatible boards will have a 2x3 pin ICSP header on them. Some may even have more than one depending on how many ICs live on the PCB. It breaks out three of the SPI pins (MISO, MOSI, SCK), power, ground, and reset. These are the pins you'll need to connect your programmer to in order to reflash the firmware on your board.



Here we have the Arduino Uno R3. It has two ICSP headers: one for the ATmega16U2 and one for the ATmega328. To reflash the bootloader on this board, you would use just the ICSP header for the ATmega328.

On some smaller boards you may not see this connector, but the pins should be broken out elsewhere. Whether you're using an SMD IC or a DIP IC, the ISP pins should be accessible in one form or another. Some boards might only have test points for the ISP header. If this is the case, you may want to consider getting an ISP Pogo Adapter. This kit allows you to temporarily make a good connection with test test points in order to reprogram your IC.



SparkFun ISP Pogo Adapter v2

KIT-23451
$15.50
Note: If you are having trouble finding the ICSP pins on your particular Arduino board, simply check out the Eagle board files or graphical datasheet for the development board. At SparkFun, we have generated a few graphical datasheets for select boards that we manufacture.

Once you have located the **six ICSP pins** on your board, it's time to hook up your programmer to the board. You can use a programming cable to connect the two, or, if you don't have a cable, you can just use some male-to-female jumper wires.

Jumper Wires Premium 6" M/F Pack of 10

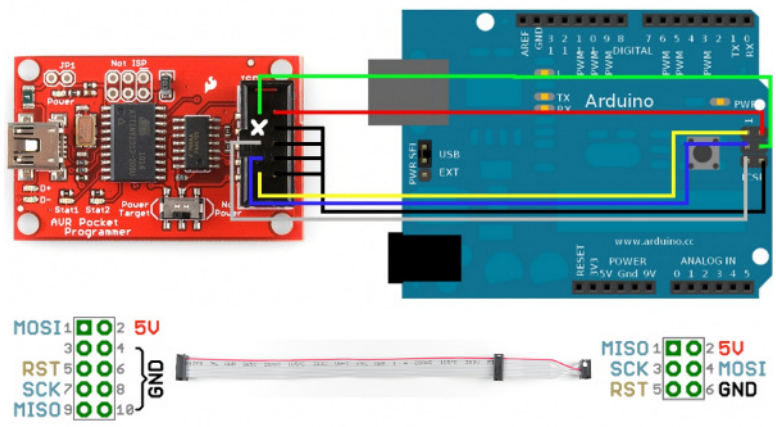
PRT-09140
$4.60

AVR Programming Cable

CAB-09215
$2.25

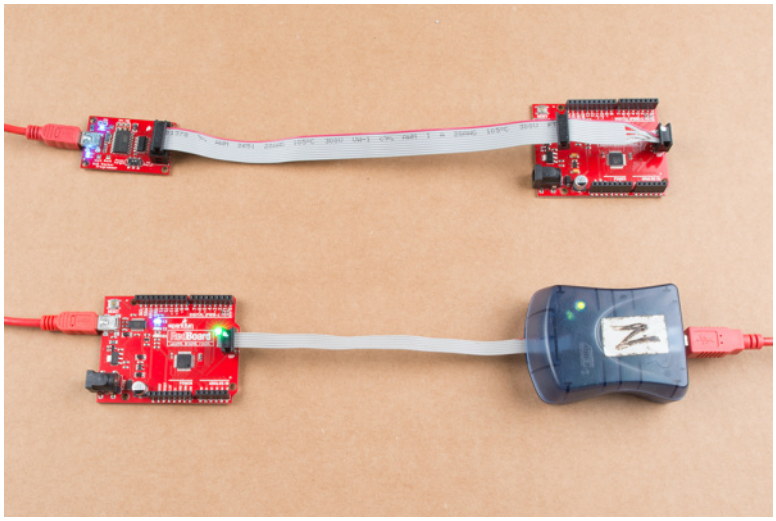
Connecting the Pocket AVR Programmer to Target

If you are using a programmer such as the pocket AVR programmer, your setup should look something like the connection below with the AVR programming cable's **arrow (◄)** connected to **MISO**. If you look really closely at the molding of the 2x3 connector, you should be able to notice the **arrow (◄)** pointing to pin 1 relative to the position of a standard ICSP header.



Click for larger image.

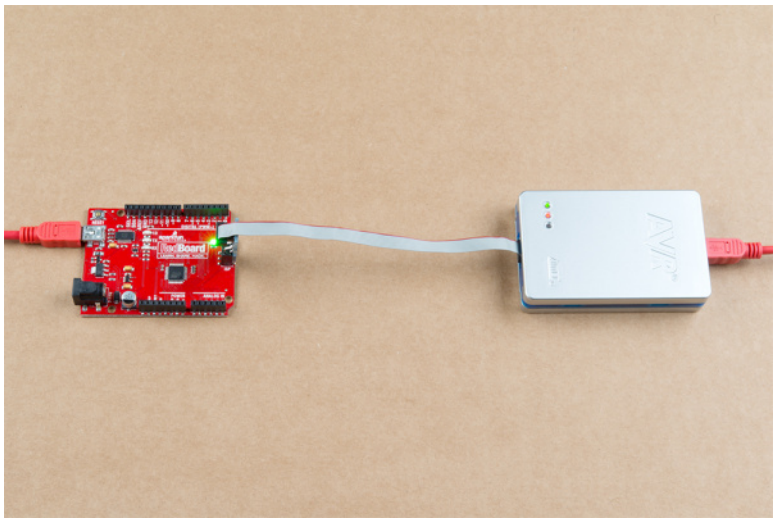
Heads up! There is a subtle difference in the orientation of the 2x5 to 2x3 AVR programming cable compared to other 2x3 cables that are attached to official Atmel programmers. As shown in the image below with MISO highlighted by a white dot, the pocket AVR programmer's cable has MISO connected closer toward the inside of the cable. The AVR MKII's cable has MISO connected closer toward the outside of the cable.



Click for larger image.

You also might notice that there is **not a USB cable connected to one of the RedBoards** Since the pocket AVR programmer can provide 5V power to the target AVR with the switch flipped to the *Power Target* position, a USB cable is not needed for the RedBoard. However, the official Atmel AVR MKII is not able to provide power to the target board. Thus, a cable is required to connect to the target AVR.

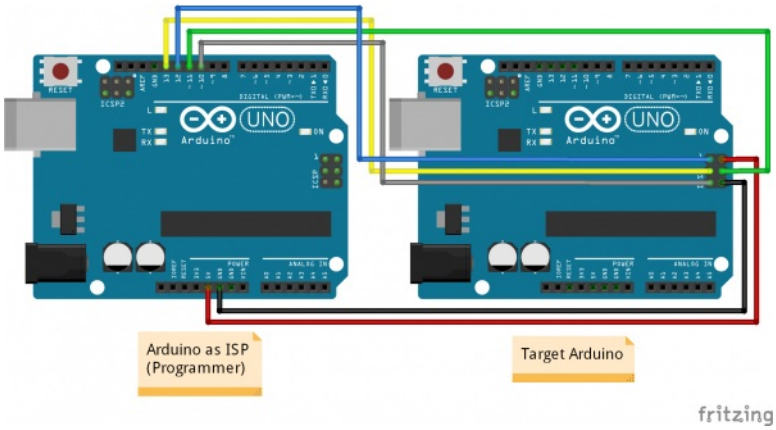
The same goes for newer Atmel programmers. The image below shows the Atmel JTAG ICE3 connected to a RedBoard. As you can see, the position of the AVR cable is connected to the RedBoard similar to the image with the AVR MKII. Since the programmer is not able to provide power to the target, you would need an additional cable connected to the target AVR.



Click for larger image.

Connecting the Arduino as ISP to Target

Or, if you're using the Arduino as your programmer, it should look the image below. Make sure to power the Arduino as ISP by connecting it to your computer.



Click for larger image.

Tip Looking for more examples besides connnecting two Arduino Unos together? Try checking out Arduino.cc's ArduinoISP page for more information.

Arduino.cc - ArduinoISP

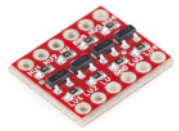
Hookup Table

⚡ **Power Your Target Arduino!** Make sure to power your target Arduino. Otherwise, you will not be able to flash the**hex** file to your target.

Here's a table to help clarify which connections go where. Depending on the Arduino, you may have access to the ICSP pins only on the 2x3 ICSP header. Make sure to refer to the board layout for more information on the Arduino's SPI connections.

AVR Programmer	Arduino as ISP	2x3 ICSP Header	ATmega328	ATmega2560	ATmega32U4
5V	Vcc/5V	Pin 2	Vcc	Vcc	Vcc
GND	GND	Pin 6	GND	GND	GND
MOSI	MOSI/D11	Pin 4	D11	D51	D16
MISO	MISO/D12	Pin 1	D12	D50	D14
SCK	SCK/D13	Pin 3	D13	D52	D15
Reset	D10	Pin 5	Reset	Reset	Reset

Warning! The connection is for a 5V Arduino. If you are connecting to a Arduino that is using a lower voltage, make sure to safely convert the logic levels using a converter between the programmer and target Arduino.



SparkFun Logic Level Converter - Bi-Directional

BOB-12009
$3.95

SparkFun Voltage-Level Translator Breakout - TXB0104

BOB-11771
$5.95

SparkFun Level Translator Breakout - PCA9306

BOB-15439
$5.25
For more information, check out our tutorial about logic levels.

Logic Levels

Learn the difference between 3.3V and 5V devices and logic levels.

Uploading Code - Easy Way

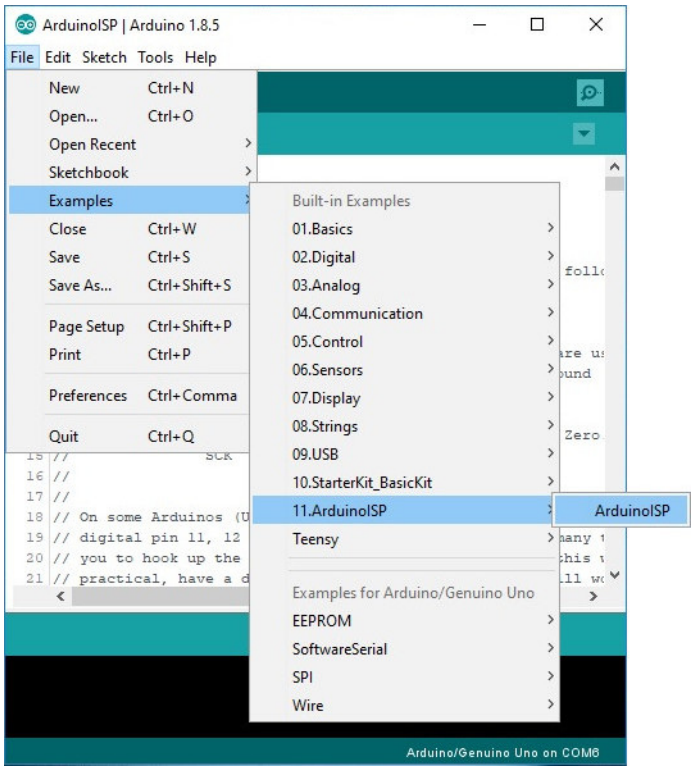
Note: This example assumes you are using the latest version of the Arduino IDE on your desktop. If this is your first time using Arduino, please review our tutorial on installing the Arduino IDE.

The easy way to upload the bootloader involves using the Arduino IDE.

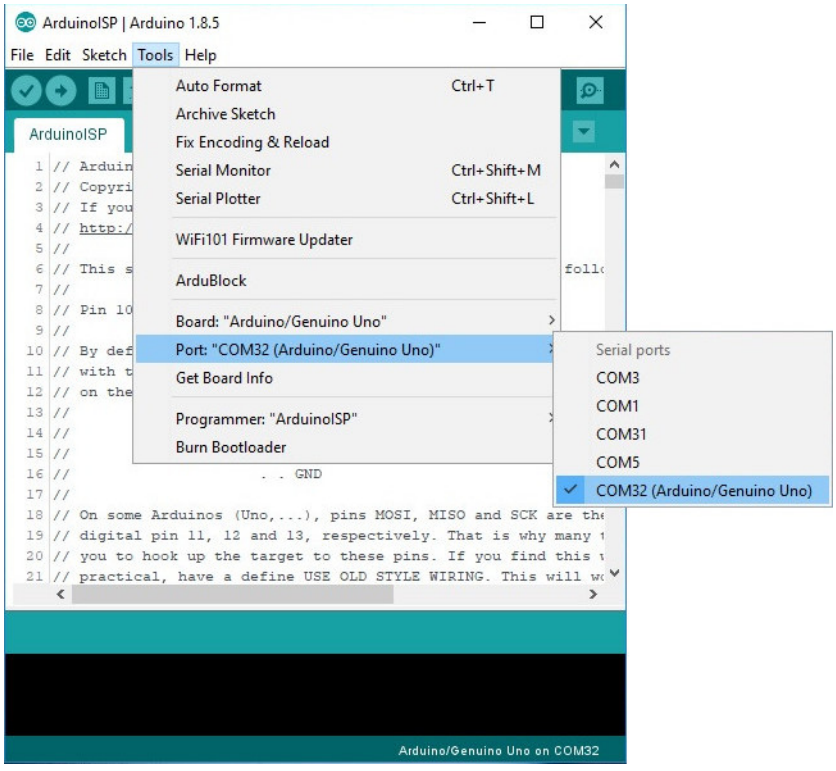
Upload Arduino as ISP

Warning! You will need to upload the *ArduinoISP.ino* to your known good Arduino (i.e. your programming board) before being able to use it as a programmer.

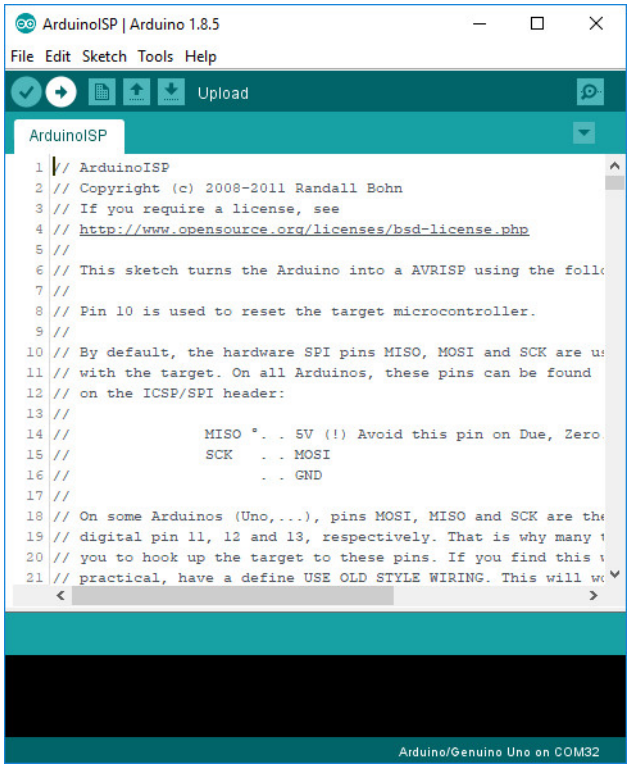
Grab a known good RedBoard or Arduino Uno. Open your Arduino IDE. In your menu, select **File > Examples > 11.ArduinoISP > ArduinoISP** to open up the Arduino as ISP sketch



Select the COM port for your Arduino as ISP. The COM port may be different depending on how it enumerated on your computer.

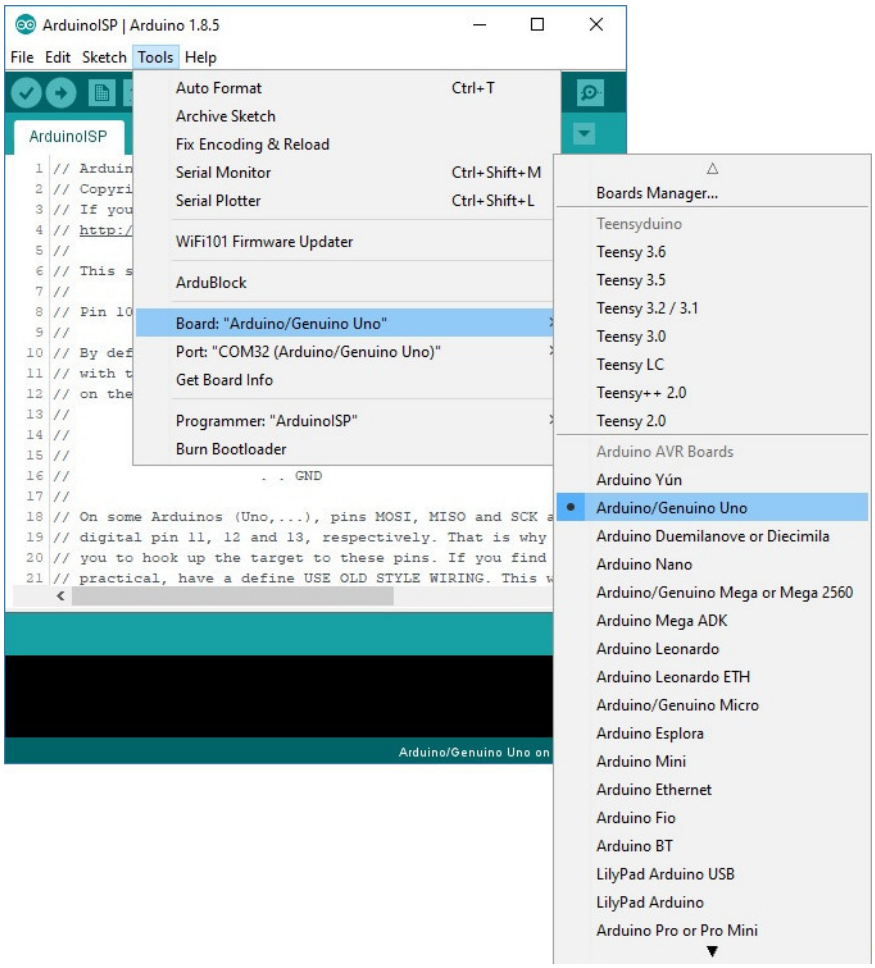


Upload the code to your Arduino to turn it into a AVRISP.

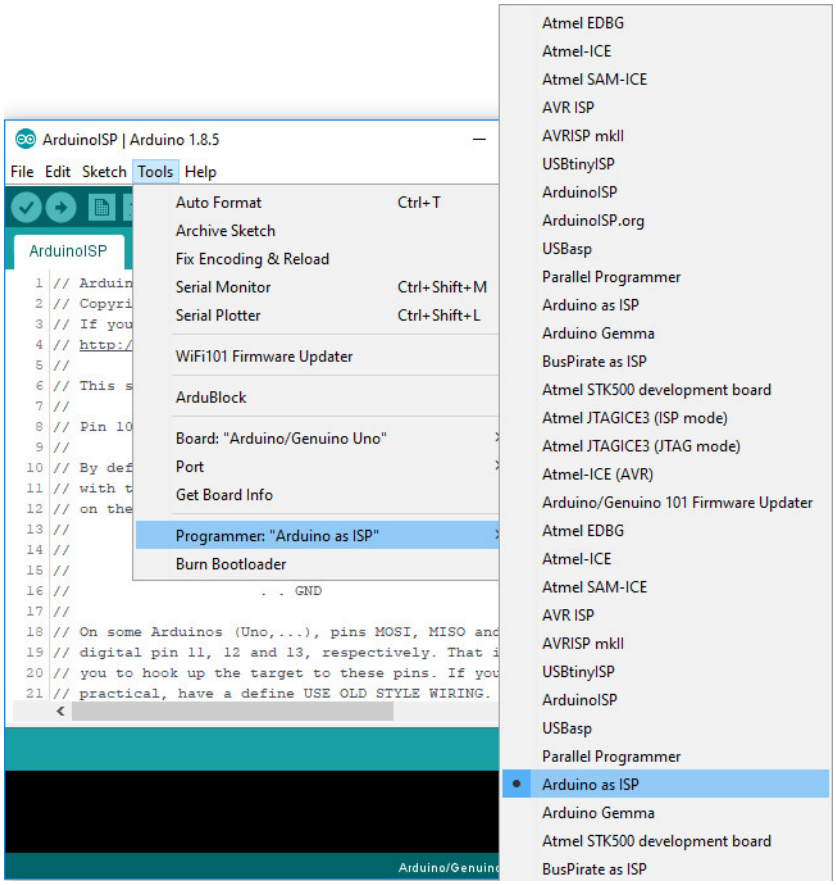


Burning a Bootloader to Your Target

Leave the Arduino as ISP (i.e. your programmer) connected to your computer. If you have not already, connect your target Arduino. Then select the board definition for your target Arduino under **Tools > Board**.

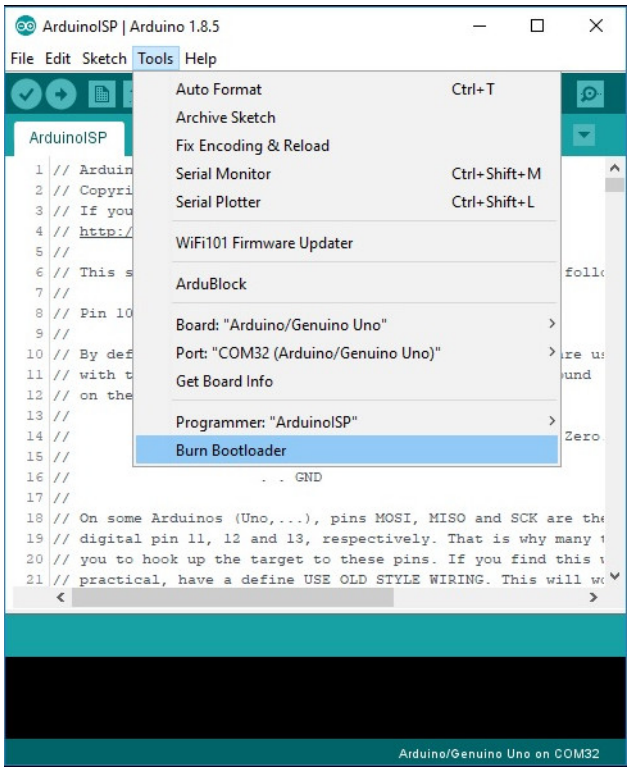


Select the programmer that you are using under**Tools > Programmer**. In this case, our programmer is an Arduino as ISP so select**Arduino as ISP**. You will also need to select the COM port that the Arduino as ISP is connected to if you have not selected the COM port already.



Heads up! You may encounter more than one selection for "Arduino as ISP" in the list of programmers. Either one will work. However, "ArduinoISP" will not work if it is selected.

Finally, select **Burn Bootloader**. This will take the board you selected in the Arduino IDE and look up the associated bootloader in the *boards.txt* file. Then, it will find the bootloader in the Arduino IDE's program folder (specifically "**...\Arduino\hardware\arduino\avr\bootloaders**") and install it. This only works if the board is installed correctly in the IDE and you have the correct bootloader.



If for some reason you want to use a bootloader that isn't installed in the Arduino IDE, visit the next section. However, it's probably easier to just install the bootloader from the Arduino IDE. For those who are curious about settings such as fuse bits, have no fear. Arduino takes care of all the messy details for you when you burn bootloaders through it.

Uploading Code - Hard Way

The hard way is for those people who want to use thecommand line. This method may be more preferable if you are modifying and recompiling and don't want to have to keep updating the IDE, but otherwise it's pretty unnecessary. Again, you will need to get the programmer and hook everything up. In this example, we are using avrdude on Windows.

There are three steps to this process:

- Set Fuse Bits (i.e. Low, High, and Extended)
- Flash **.hex** File
- Set Lock Bits

The first step involves setting the fusebits. Fusebits are the part of the AVR chip that determine things like whether you are using an external crystal or whether you want brown out detection. The commands listed below are specifically for the Arduino Uno using an ATMega328, they will probably work on some other similar boards such as the Duemilanove, but make sure you know what you are doing before playing with fusebits. All the required fuse bits are listed in the *boards.txt* file for different boards. Again, if you have a*boards.txt* file installed then just use the Easy Way. The second step is actually uploading the program. The final step is setting the lock bits.

Note: These fusebits will **not** work on a 3.3V/8MHz board. If you are using a different microcontroller, you will also need adjust thepartno parameter.

Pocket AVR Programmer

Fuse Bits

If you are using the AVR Pocket Programmer to program your target Arduino Uno, type the following commands in the command line to set the fuse bits.

language:bash
avrdude -b 19200 -c usbtiny -p m328p -v -e -U efuse:w:0x05:m -U hfuse:w:0xD6:m -U lfuse:w:0xFF:m

Hex File and Lock Bits

Once the fuse bits are set, we can flash a compiled**.hex** file to the target board and set the lock bits. Enter the following in a command line. Make sure that you are in same directory as your **.hex** file and adjust the ...hexfilename.hex that you are using to flash for your target. To flash the Arduino Uno Bootloader, head over to the Arduino program folder. On a Windows OS, it will look similar to this path **...\arduino-1.8.5\hardware\arduino\avr\bootloaders\optiboot**. There area few files in the folder but the one we are interested in is the **optiboot_atmega328.hex** file.

language:bash
avrdude -b 19200 -c usbtiny -p m328p -v -e -U flash:w:hexfilename.hex -U lock:w:0x0F:m

---

Arduino as ISP

Fuse Bits

If you are using the Arduino as ISP to program your target Arduino Uno, type the following commands in the command line to set the fuse bits.

language:bash
avrdude -P comport -b 19200 -c avrisp -p m328p -v -e -U efuse:w:0x05:m -U hfuse:w:0xD6:m -U lfuse:w:0xFF:m

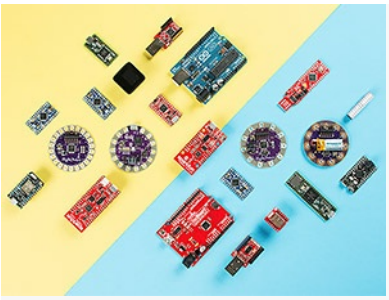
Hex File and Lock Bits

Once the fuse bits are set, we can flash a compiled**.hex** file to the target board and set the lock bits. Enter the following in a command line. Make sure that you are in same directory as your **.hex** file and adjust the ...hexfilename.hex that you are using to flash for your target. To flash the Arduino Uno Bootloader, head over to the Arduino program folder. On a Windows OS, it will look similar to this path **...\arduino-1.8.5\hardware\arduino\avr\bootloaders\optiboot**. There area few files in the folder but the one we are interested in is the **optiboot_atmega328.hex** file.

language:bash
avrdude -P comport -b 19200 -c avrisp -p m328p -v -e -U flash:w:hexfilename.hex -U lock:w:0x0F:m

One last bit of info. As we stated earlier, a bootloader is essintially a**.hex** file. Thus, you can use this method to upload and code you wish to your ICs.

Resources and Going Further



For more info on AVRs, bootloaders, and flashing firmware to other boards, check out these other great tutorials.

- Arduino.cc
  - Bootloader Info from Arduino
  - Arduino as ISP
- AVRDUDE
- Pinouts for Various ICs and Development Boards
- SparkFun Graphical Datasheets
- Getting Started with the Pocket Programmer

You can also check out these related tutorials to learn more about uploading code to AVR chips or modifying firmware.

Tiny AVR Programmer Hookup Guide

A how-to on the Tiny AVR Programmer. How to install drivers, hook it up, and program your favorite Tiny AVRs using AVRDUDE!

Installing a Bootloader on the MicroView

Fix your bootloader-less MicroView! This tutorial covers how to: disassemble the MicroView, wire it up to an assortment of programmers, program the bootloader, and test it out.

Servo Trigger Programming Guide

Looking under the hood of the Servo Trigger -- using the development environment and some finer details of the firmware.

How to Install an ATtiny Bootloader With Virtual USB

With this, you will be able to upload Arduino sketches directly to the ATtiny84 over USB without needing to use a programming device (such as another Arduino or FTDI chip).

Pi AVR Programmer HAT Hookup Guide

In this tutorial, we will use a Raspberry Pi 3 and the Pi AVR Programmer HAT to program an ATMega328P target. We are going to first program the Arduino bootloader over SPI, and then upload an Arduino sketch over a USB serial COM port.

Or check out our ARM programming tutorial for ARM chips.

ARM Programming

How to program SAMD21 or SAMD51 boards (or other ARM processors).
