

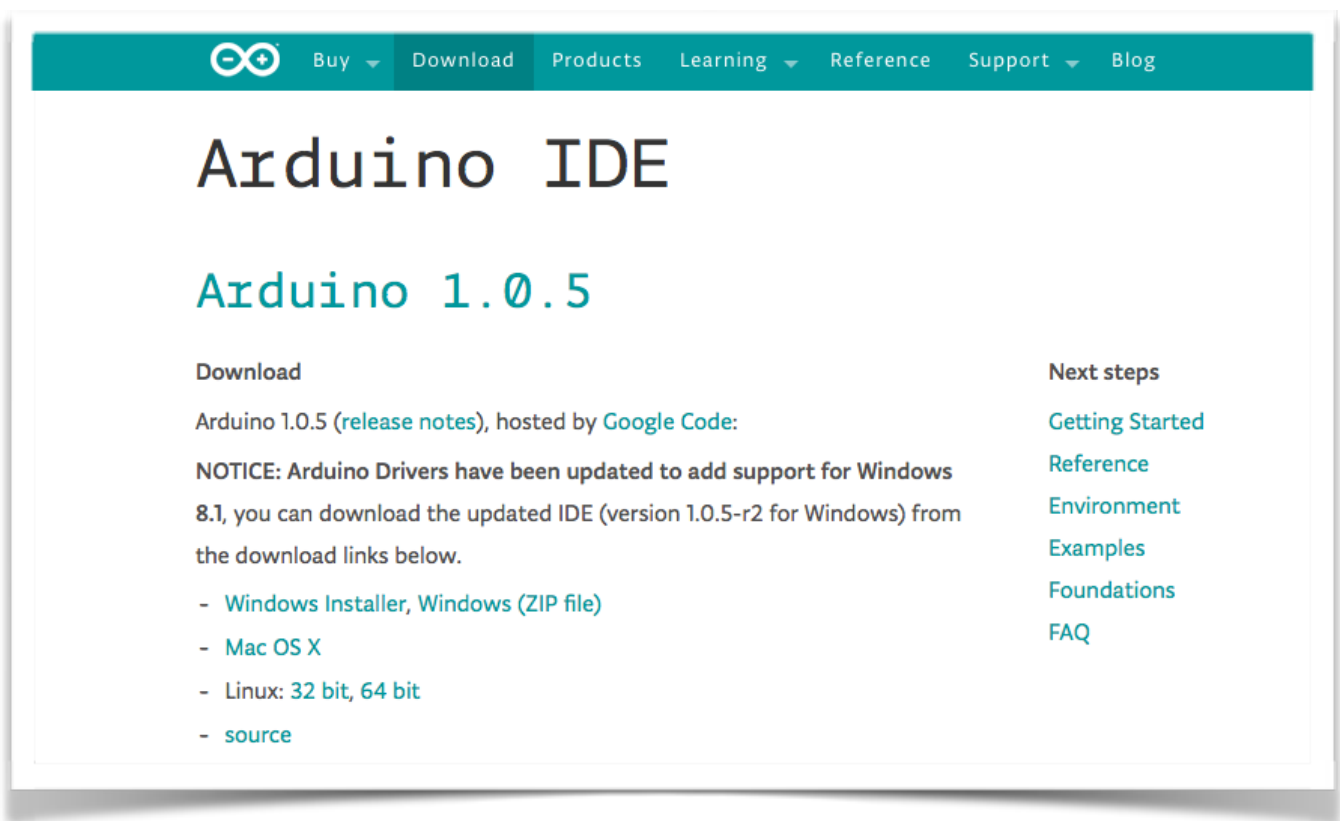
# MICRODUINO STARTUP GUIDE

# 1. Microduino Configuration

The Arduino IDE software does not support Microduino boards out of the box, but it is very easy to configure Microduino-Core/Core+ in Arduino IDE environment. Only in 5 minutes, you will be able to upload Arduino compatible sketches directly to Microduino-Core/Core+ boards.

## Instructions

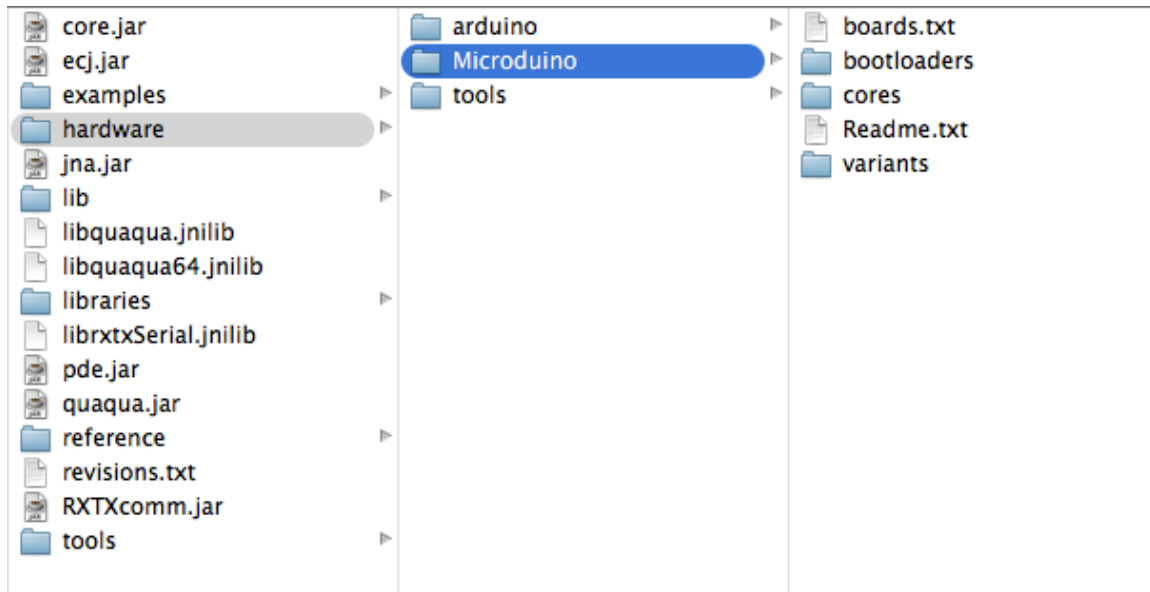
- **1. Install Arduino IDE Software** (v.1.0.5), downloading the program from [Arduino IDE official web](#).



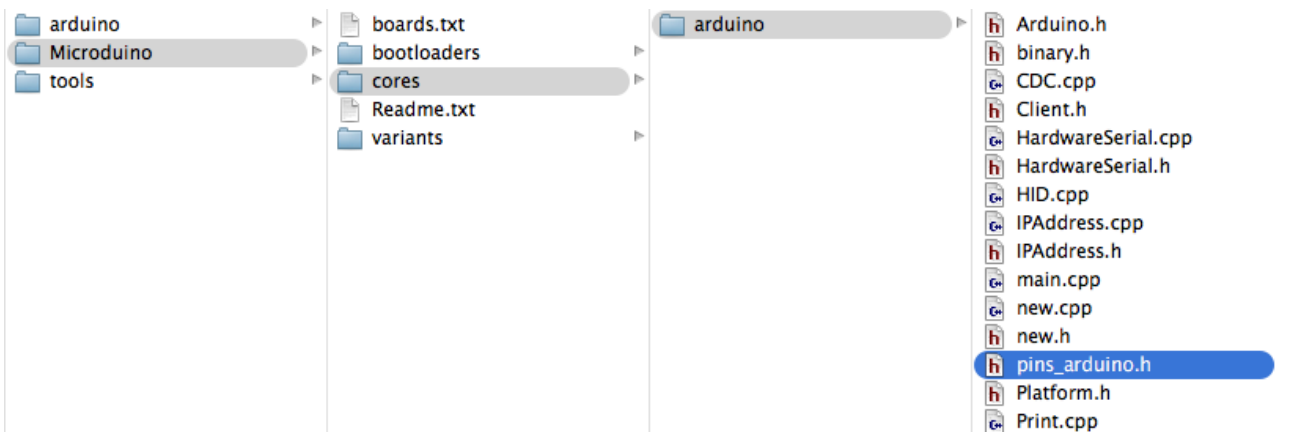
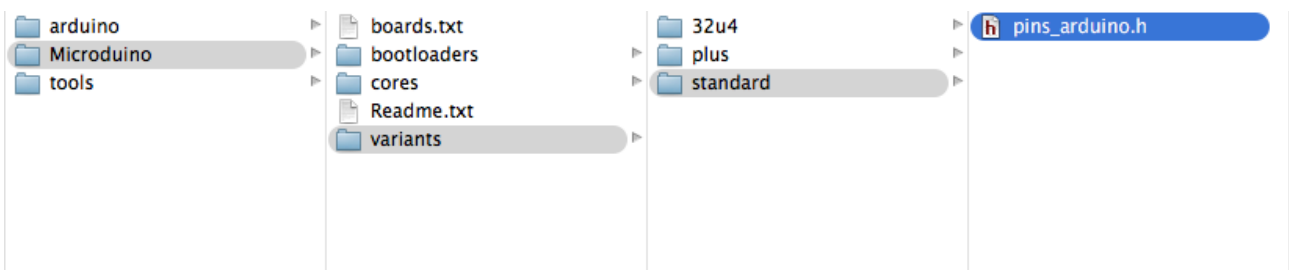
The screenshot shows the Arduino IDE 1.0.5 download page. The page has a teal header with navigation links: Buy, Download, Products, Learning, Reference, Support, and Blog. The main heading is "Arduino IDE" followed by "Arduino 1.0.5". Under the "Download" section, it states "Arduino 1.0.5 (release notes), hosted by Google Code:" and includes a "NOTICE" about updated Windows drivers. Below this, there are four download links: "Windows Installer, Windows (ZIP file)", "Mac OS X", "Linux: 32 bit, 64 bit", and "source". On the right side, under "Next steps", there are links for "Getting Started", "Reference", "Environment", "Examples", "Foundations", and "FAQ".

- **Microduino-Core** use Arduino IDE and add some small modifications to support Microduino boards.

- 2. Install Microduino-Core IDE package folder:
- Copy the Microduino folder into the "Arduino/hardware" directory.



- Open the "...Microduino/variants/standard" directory. Copy the "pins\_arduino.h" file into "...Microduino/cores/arduino" directory



- Restart (or start) your Arduino IDE software.
- From the tools → Board menu, select the "Microduino-Core" board. You will now be able to use your Microduino boards just as using a normal Arduino board. Programs can be uploaded now!
- Bootloader can be burned -- as Arduino board does.
- Just deleting "Microduino/" directory, it's removed Microduino IDE environment.

- **3. Install FTDI USB Serial Driver:**

- Download the appropriate VCP Driver file from "http://www.ftdichip.com/Drivers/VCP.htm"
- Install the driver.
- Reboot the system.

The screenshot shows the FTDI website header with the logo and navigation menu. The main content area is titled "Virtual COM Port Drivers" and contains the following text:

**Virtual COM Port Drivers**  
 This page contains the VCP drivers currently available for FTDI devices.  
 For D2XX Direct drivers, please click [here](#).  
 Installation guides are available from the [Installation Guides](#) page of the [Documents](#) section of this site for select

**VCP Drivers**  
 Virtual COM port (VCP) drivers cause the USB device to appear as an additional COM port available to the PC.

*This software is provided by Future Technology Devices International Limited "as is" and any express or implied liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised FTDI drivers may be used only in conjunction with products based on FTDI parts.  
 FTDI drivers may be distributed in any form as long as license information is not modified.  
 If a custom vendor ID and/or product ID or description string are used, it is the responsibility of the product manufacturer.*

At the bottom left of the screenshot, there is a "Google Site Search" box and a magnifying glass icon.

**Currently Supported VCP Drivers:**

Operating System	Release Date	x86 (32-bit)	x64 (64-bit)
Windows 8.1	2013-10-21	<a href="#">2.08.30 8.1</a>	<a href="#">2.08.30 8.1</a>
Windows*	2013-08-01	<a href="#">2.08.30</a>	<a href="#">2.08.30</a>
Linux	2009-05-14	<a href="#">1.5.0</a>	<a href="#">1.5.0</a>
Mac OS X	2012-08-10	<a href="#">2.2.18</a>	<a href="#">2.2.18</a>
Windows CE 4.2-5.2**	2012-01-06	<a href="#">1.1.0.10</a>	-
Windows CE 6.0	2012-01-06	<a href="#">1.1.0.10</a>	-

## 2. Burn Bootloader to Microduino-Core with an Arduino

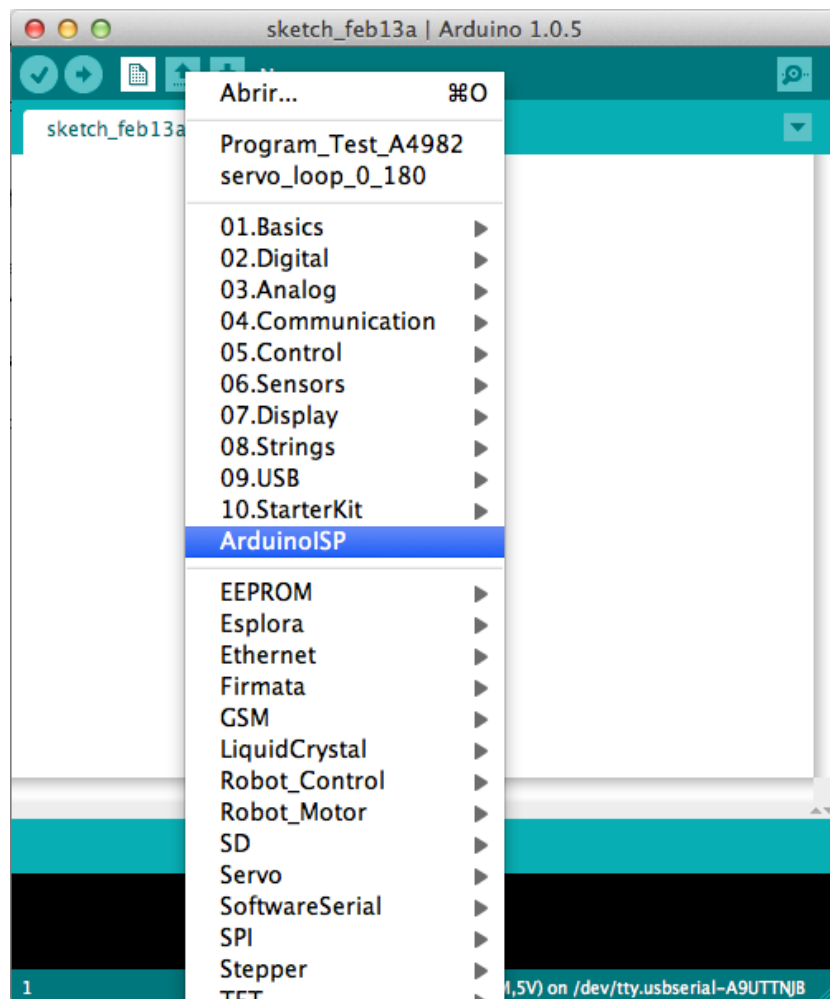
This tutorial explains how to use an Arduino or Microduino-Core/Core+ board as an AVR ISP (in-system programmer). This allows you to use the board to burn the bootloader onto an AVR (e.g. the Microduino-Core or Core+ used in Arduino).

### Instructions

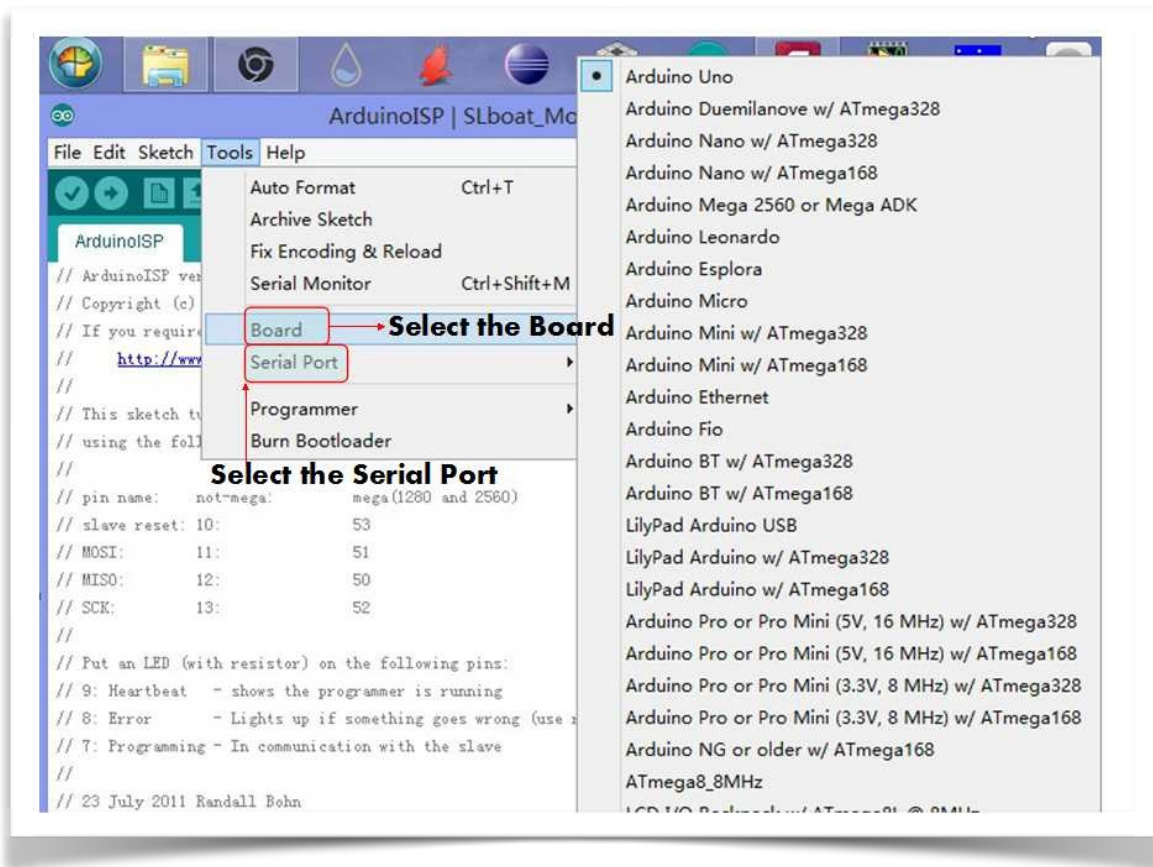
To use your Arduino or Microduino-Core/Core+ board to burn a bootloader onto an AVR, you need to follow a few simple steps.

#### Prepare your master Arduino or Microduino-Core/Core+ boards:

1. Open the Arduino ISP firmware (in Examples) to your Arduino board.



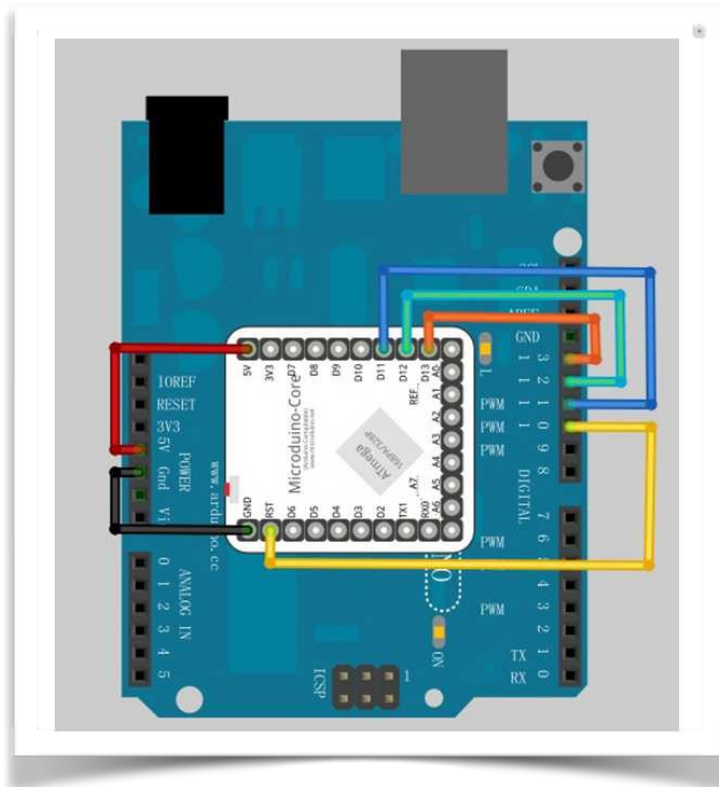
- Note for Arduino 1.0: you need to make one small change to the Arduino ISP code. Find the line in the heartbeat() function that says "delay(40);" and change it to "delay(20);".
- Select the items in the Tools > Board and Serial Port menus that correspond to the board you are using as the programmer (not the board being programmed).



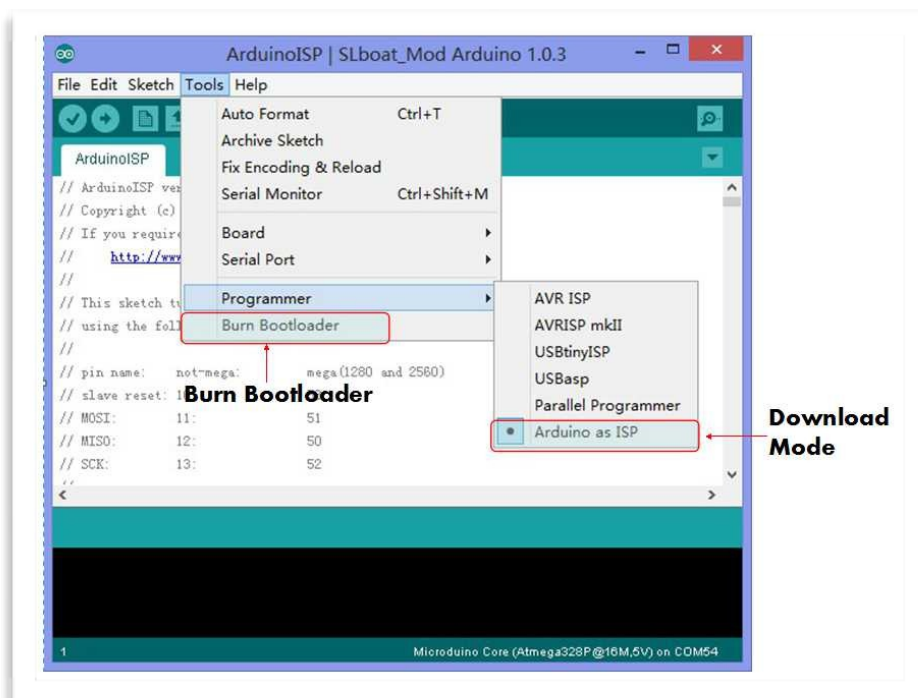
- Upload the Arduino ISP sketch.

#### Burn Bootloader into target Microduino-Core/Core+ boards:

- Wire your Microduino-Core/Core+ board to the target as shown in the diagram below. (Note for the Arduino Uno: you'll need to add a 10 uF capacitor between reset and ground.)



2. Select the item in the Tools > Board menu that corresponds to the board on which you want to burn the bootloader (not the board that you're using as the programmer). See the board descriptions on the environment page for details.
3. Use the Burn Bootloader > Arduino as ISP command.



## 3. CodeBlocks Arduino IDE

CodeBlocks Arduino IDE is a customized distribution of the open-source **Code::Blocks** IDE enhanced for Arduino development. It provides more demanding software developers with everything a modern IDE should have including code folding, code completion, code navigation, compiling as well as uploading for Arduino. With a dedicated project wizard, it's easy create a ready-to-go Arduino project. The distribution integrates latest Arduino core files, standard Arduino libraries, AVR toolchain, **Arduino Builder**, a serial terminal and most interesting, an API-level Arduino simulator (under development).

### Characteristics

1. Dedicated project wizard for Arduino development.
2. Integrated Arduino core files and libraries.
3. Compiled core files cached for faster compiling speed (comparing to original Arduino IDE).
4. Integrated pre-configured AVR compiler toolchain.
5. Popular Arduino boards supported as build targets.
6. Uploading HEX to Arduino boards (Leonardo supported) by running the built target.
7. Arduino API-level simulator (very early stage) integrated (as a build target).

