Arduino / AVR Development Environment Setup

# Introduction

I develop complex applications and libraries for Atmel AVR microcontroller (on Arduino or directly on chip). I essentially work from Windows 10, although I am also familiar with Linux.

Currently, I don’t consider Arduino IDE suitable for professional environment.

I have decided to use a real IDE for my AVR developments; there are several good IDE supporting C and C++ on the market; some are more or less easy to configure for Arduino development.

Originally in 2016 I had chosen Netbeans 8 for various reasons, but I finally decided to go with Microsoft Visual Studio Code for its broad range of supported languages and its easy extensibility.

My Arduino projects use C++ 11 and heavily depend on C++ templating facilities.

For this reason and a few other constraints, I always try to use the latest Atmel AVR toolchain (based on GNU GCC) so that I have the best possible C++ support and the least bugs. My main constraint is that, with Arduino 1.6.x integrated toolchain, I always have had issues at link time with the provided GCC.

After my first attempts using the latest AVR Toolchain (3.5.3) on Windows, I found an issue with the archiver tool I was using (this one is not integrated in the toolchain, you have to get GNU binutils for Windows, i.e. Cygwin or MinGW).

Due to this latest issue, I decided to finally build all my projects on Linux either in native mode on my laptop or within a VM on my desktop.

In this document, I will first describe how to setup a Linux environment suitable for FastArduino development, either on a native Linux box or on a VM that can run on Windows (I use both).

Note, however, that I will not describe how to setup a VM. Until now, for my VM, I have used both VirtualBox (free) and VMWare Workstation (commercial) and feel unable to provide relevant recommendations here.

The Linux setup I describe here is based on Fedora 26 but should be easily adaptable to other distributions.

# Linux setup

First off, you’ll need to setup your box as you need and like (displays, background, audio, date & time…); I will not describe this here.

For FastArduino-based development, you will first need to get the AVR GNU Toolchain from Atmel: <http://www.atmel.com/tools/atmelavrtoolchainforlinux.aspx>



You will be requested for some personal information before you can get a download link by email. Just copy this link and paste it in your Linux terminal:

> wget <pasted link>  
> gunzip avr8-gnu-toolchain-3.5.3.1700-linux.any.x86\_64.tar.gz  
> tar xvf avr8-gnu-toolchain-3.5.3.1700-linux.any.x86\_64.tar

This shall have created a directory named “avr8-gnu-toolchain-linux\_x86\_64” on your Linux home directory.

One additional step consists in updating the path in Linux so that it includes ~/avr8-gnu-toolchain-linux\_x86\_64/bin which contains all GNU-based executables for AVR (all the binaries are prefixed with “avr-“).

Actually the easiest way to do so on most Linux distibutions is to add a link named “bin” on your home directory and make it point to ~/avr8-gnu-toolchain-linux\_x86\_64/bin:

> ln –s avr8-gnu-toolchain-linux\_x86\_64/bin bin

Last, you have to install avrdude so that your builds will be able to upload AVR programs to the targets (Arduino through USB or an AVR MCU directly through an ISP programmer):

> sudo dnf install avrdude

This will install avrdude 6.3 on your VM at /usr/bin.

You’ll also have to install a configuration file for avrdude; I use the one provided with Arduino 1.8.2 (can be found at hardware\tools\avr\etc\avrdude.conf).

Personally, I set it as default configuration file so that I don’t have to specify it in command line, for that it must be copied to /etc:

> sudo cp avrdude.conf /etc

> sudo cp avrdude.conf /etc/avrdude/

> sudo chmod a+r /etc/avrdude.conf

In addition, you should install doxygen and graphviz as this is the tool used for generating FastArduino documentation:

> sudo dnf install doxygen  
> sudo dnf install graphviz

In order to have access to devices (Arduino or Programmers), you need to do:

> sudo usermod –aG dialout osboxes  
> sudo chmod a+rw /dev/ttyACM0

Then you are able to use the device, e.g. in a serial terminal or avrdude.

The command line below shows stty settings that work well:

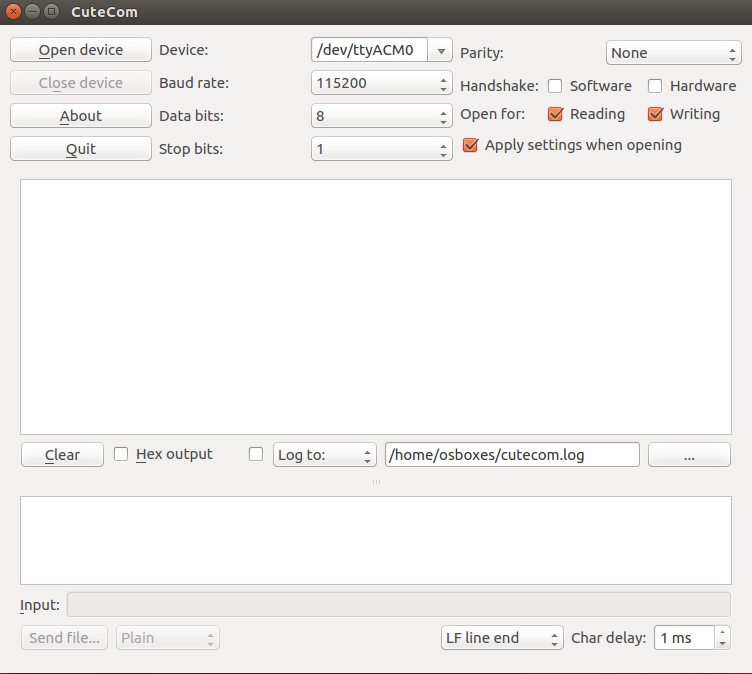
> stty -F /dev/ttyACM0 cs8 115200 ignbrk -brkint -icrnl -imaxbel -opost -onlcr -isig -icanon -iexten -echo -echoe -echok -echoctl -echoke noflsh -ixon –crtscts

**Important!** If you intend to use an ISP programmer that does not get viewed directly as a tty device (e.g. ArduinoISP), then you will need to ensure that avrdude is run as suid so that it is executed in SSH as root, this can be done with the simple command line below:

> sudo chmod +s /usr/bin/avrdude

References:  
<https://learn.adafruit.com/usbtinyisp/avrdude>

## Tip for serial communication

If you need to communicate with your AVR device through a serial port, then you need a serial terminal; I use Cutecom which is a simple terminal which is easy to configure:  


To install it on your VM, just type the command line:

> sudo dnf install cutecom

Then you can invoke it directly from your shell:

> cutecom &

That’s it!

# Visual Studio Code setup

For all my Arduino/AVR projects, I now use Visual Studio Code as a good replacement for a specialized Integrated Development Environment. My projects repositories always include VSCode configuration files.

## First installation

> sudo rpm --import https://packages.microsoft.com/keys/microsoft.asc  
> sudo sh -c 'echo -e "[code]\nname=Visual Studio Code\nbaseurl=https://packages.microsoft.com/yumrepos/vscode\nenabled=1\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" > /etc/yum.repos.d/vscode.repo'  
> dnf check-update  
> sudo dnf install code  
> sudo dnf install clang

You can now launch VS Code:  
> code

When in VS Code, add the following extensions:

* C/C++
* VSCode FastArduino
* Include Autocomplete
* TODO-Highlight
* Auto Comment Blocks
* Eclipse Keymap (if you are used to Eclipse IDE shortcuts)

## Setup for FastArduino library contribution

TODO

FastArduino clone

Update json files

## Project setup

TODO

FastArduino download/clone

Update json files

FastArduino project template download

Update json files