Arduino / AVR Development Environment Setup

# Introduction

I develop complex applications and libraries for Atmel AVR microcontroller (on Arduino or directly on chip). I work exclusively on Linux, I use Fedora (as of September 2019, I use Fedora 29).

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++ 17 and heavily depend on C++ templating facilities.

For this reason and a few other constraints, I always try to use the most recent Atmel AVR toolchain (based on GNU GCC) so that I have the best possible C++ support and the least bugs. Fedora repositories include AVR GCC packages, that are often updated, I decided to use exclusively those packages.

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 have used both).

Note, however, that I will not describe how to setup a VM. In the past, for my VM, I have used both VirtualBox (free) and VMWare Workstation (commercial) and both worked fine.

The Linux setup I describe here is based on Fedora 29 but should be easily adaptable to other distributions. One main advantage of Fedora is its repositories include all AVR build tools.

# Linux setup

For FastArduino-based development, you will first need to get the following packages (on Fedora; if you use other distributions, you will have to find equivalent packages on your own):

* avr-g++ 9.2
* avr-gcc 9.2
* avr-libc 2.0
* avr-binutils 2.32
* avrdude 6.3

All required executables will be installed at default locations (typically /usr/bin) and automatically available from executable path.

Installing avrdude is needed 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).

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/avrdude/

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

In addition, you may want to 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 <user>  
> 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:

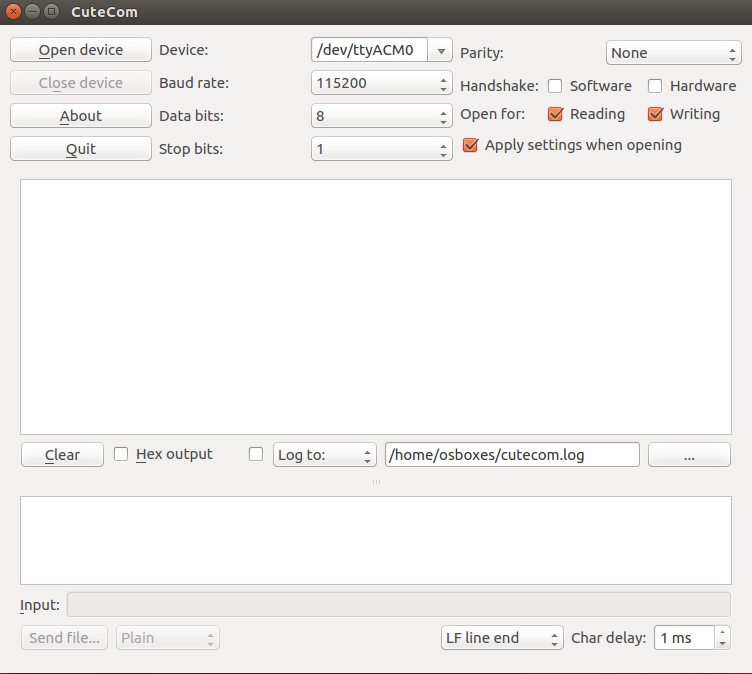
> stty -F /dev/ttyACM0 cs8 115200 ignbrk -brkint -icrnl -imaxbel -opost -onlcr -isig -icanon -iexten -echo -echoe -echok -echoctl -echoke noflsh -ixon –crtscts

**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 system, 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