

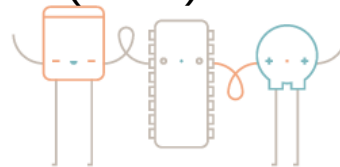
# Starting with the Arduino IDE using a Linux platform



## Simplified approach to programming AVR and STM32 devices (Draft)

AN OPEN PROJECT INITIATIVE, DEVELOPED  
AND SUPPORTED BY ARDUINO.CC AND  
THE ARDUINO COMMUNITY WORLDWIDE

LEARN MORE ABOUT THE CONTRIBUTORS  
OF **ARDUINO.CC** on [arduino.cc/credits](http://arduino.cc/credits)



**Author:** Nick KASPEROVIC – VK3TY

**Date:** 10<sup>th</sup> October 2016

**Last Updated:** Monday, 24 October 2016

Document has been drafted on a **Linux Mint 17.3** platform and using **LibreOffice Writer**.

As a committed Linux user, I use my works to highlight the fact that commercial quality material can be undertaken using Linux and ancillary programs without financial burden to developers.

I trust you will find this document of benefit.

This document has been prepared based on personal experience and gleaned from public domain sources. I make no claims to originality of the material.

You are free to use the material contained within. The only limitations may be constraints of your country's legal guidelines.

All due care has been taken in preparing this material and, if there is an error, I will correct it.

Any damages incurred as a consequence of your actions using these references is and remains your responsibility and I absolve myself of all and any claimable damages.

## Table of Contents

Introduction.....	1
Notational Guidance.....	1
Basic steps to installing an Arduino IDE.....	2
Arduino AVR Library Installation.....	3
Loading Ancillary Modules.....	3
STM32 32-bit ARM Cortex MCUs.....	5
Background.....	5
Making use of newer technology.....	5
Getting Arduino IDE to co-operate with ARM STM32F103 boards.....	5
Downloading supplementary files.....	5
The All Important Download and Installation.....	7



# Arduino Installation - Linux devices

## Introduction

This document provides a one-stop, easy to follow series of instructions helping a user to set up an Arduino Integrated Development Environment (IDE) and includes the set-up for using the STM321F003C boards.

From the outset, a special thank you to Roger Clark for his extensive knowledge on both the AVR and STM32 devices used. In my mind, Roger is quite the “*font of knowledge*” on these matters. I’m merely the author.

From a personal background, I have been involved with many and varied aspects of computing since the early 1970’s. Yes, a geriatric in body but mentally – the tenacity of a terrier.

As a professional, I have seen a huge change from those halcyon days driven my main-frame then mini-computers that, compared with the current trend of computers are so archaic as to belong with the dinosaurs.

## Notational Guidance


Where there is a requirement for you to do something, they will highlighted in yellow eg:-

<code>sudo ./install.sh</code>	Terminal commands that are specific to issuing commands. These use a Courier 10 Pitch (typical terminal display font).
<code>left-click</code>	Mouse or keyboard initiated commands.
<a href="http://www.arduino.cc">www.arduino.cc</a>	Uniform Resource Locator (URL) that points to web-sites.
<code>arduino</code>	Reference to specific Linux directories/folders.
<code>&lt;user&gt;</code>	Requirement to enter a value without the “<” or “>”. May also denote a computer returned value particular to your configuration

## Basic steps to installing an Arduino IDE

There are two ways to undertake the installation of Arduino into your computer. I will use the Graphic User Interface (commonly referred to as GUI). This will make it easier and less prone to errors (hopefully).

Using the following steps, I'm sure you will be successful. So let's start - step by step:-

1. Download the required Linux version of Arduino from [www.arduino.cc](http://www.arduino.cc) - in my case, I selected the 64-bit version. First check whether your computer has a 32-bit or 64-bit operating system.
2. Open the File Manager (at the bottom of the screen you will see a  representing a folder. **Left-click** and you will see your computer's file structure.
3. From within **Downloads** find the file **arduino-1.#.##-linux64.tar.xz** or **arduino-1.#.##-linux32.tar.xz** depending on your operating system (32-bit or 64-bit). (the current version is 1.6.10 (as of September 2016) – but this will change over time – get the latest stable version).
  4. **Right-click** and select Open with Archive Manager to extract.
  5. The files will be extracted to the directory **arduino-1.#.##/**.
  6. My preference is to use **arduino** as the “directory of choice”. So, whilst still in the File Manager, **Right-click** **arduino-1.#.##**. About ten lines down, you will see “Rename . . .”. **Left-click** on “Rename . . .” and type **arduino** and then press **Enter**.
  7. Now you can start developing your own Arduino sketches/programs.  
... and everyone said it was so-o-o hard to do.
  8. Go to the desktop (main screen), you should see a blue circle with an infinity symbol



Double **left-click** and, after a square “splash screen”, you will have your first Arduino IDE basic script before your eyes.

9. One more useful step I would recommend;-  
**Left-click** on "File" and then select "Preferences". Under the "Settings" tab. The first field will be "Sketchbook location:". This is where all your personal sketches/programs will be stored.

My preference is to store programs/sketches away from the Arduino parent folder (**arduino**) and save them in a library of my own - (**ArduinoDev**). It's up to you how you do it. I do it this way to allow better management and tracking of the stuff I develop.

Replace the default “Sketchbook location:” with **home/<User ID>/ArduinoDev** and then **left-click** the “OK” at the bottom right.

That's the end of the basic installation of an Arduino IDE.

## Arduino AVR Library Installation

Goups (Adafruit, GitHub and others provide a large module repositories that cover numerous functionary areas.

As is the want of most programmers/developers, there's always a need for more. As new technology is made available, rather than trying to develop you own routines to access this new technology, there are many around the world that are privy to pre-release samples. Rather than idividually writing your own code, various libraries are made available (eventually).

These libraries generally come in a **.zip** form. Arduino provides a simple way to load ancillary modules. Expanded, they are automatically loaded into the `~/arduino/libraries` folder. These modules are accessed by using the statement `#include <name.h>` directive in your coding.

Some of the more commonly used libraries;-

- **LiquidCrystal\_I2C.h**
  - The above requires the (mandatory) inclusion `#include <Wire.h>` to use the above.
  - A current model Liquid Crystal Display (LCD) may well have only a four wire connection (+ve, -ve and two signalling pins). This is a far cry from having to connect 12 separate wires.
- Real Time Clock (RTC)

As stated previously, Adafruit, GitHub and others provide a serious selection of different libraries.

Before giving up in despair, I would strongly suggest Googling these (and other) sites. Quite often you will come across sample code to assist you.

### ***Loading Ancillary Modules.***

Mentioned previously that it can be done, when you are ready, navigate to Sketch → Include Library → Add .ZIP Library...  
It's that simple

As example, I have a RTC module that I would like to use with my development board. A quick search on eBay (Australia) - <http://www.ebay.com.au/itm/like/262122358249?lpid=107&chn=ps> I find that there is a module I think will do the job. But, like all things on eBay, documentation is sparse.

Knowing that the item number is DS3231 – why not simply type in Arduino DS3231 and see what comes up.

Several options pop up. The one that interests me the most has the title;- “Time library – Arduino Playground” with a URL of <http://playground.arduino.cc/Code/Time>

Not only does it provide a library but it is an Arduino supported library with detailed instructions – with included sketch. Couldn't be better.

Quoting the Arduino site, it also informs that the original library has been superceded and you might be better served elsewhere;-

**Update:** newer versions of [Time](#), [TimeAlarms](#), and [DS1307RTC](#) are available, featuring more updates for newer version of Arduino and compatibility with Arduino Due.

The code is derived from the earlier Playground DateTime library but is updated to provide an API that is more flexible and easier to use.

[This old download](#) does not work on Arduino 1.6.1. Use the links above. Time includes example sketches illustrating how similar sketch code can be used with: a Real Time Clock, Internet NTP time service, GPS time data, [DCF77 radio signal](#), and Serial time messages from a computer. To use all of the features in the library, you'll need the UDPbitwise library, found [here](#).

From this we now can navigate via our preferred web-client to:-

[http://www.pjrc.com/teensy/td\\_libs\\_DS1307RTC.html](http://www.pjrc.com/teensy/td_libs_DS1307RTC.html)

Here you find another supplier that goes into further details and, not surprisingly, shows where you will find the library you will need;=

Download:    Included with the Teensyduino Installer  
                 Latest Developments on Github

<https://github.com/PaulStoffregen/DS1307RTC>

Okay, not as straight forward but it does show that there are many ways to get a result.

Once downloaded to your **Home/Downloads** directory you will be able to navigate to Sketch → Include Library → Add .ZIP Library... and load it.



# Arduino ARM (STM32F103C) Library - optional

## STM32 32-bit ARM Cortex MCUs

### Background

The STM32 family of 32-bit Flash micro-controllers based on the ARM® Cortex®-M processor is designed to offer new degrees of freedom to MCU users. It offers a 32-bit product range that combines very high performance, real-time capabilities, digital signal processing, and low-power, low-voltage operation, while maintaining full integration and ease of development.

The unparalleled and large range of STM32 devices, based on an industry-standard core and accompanied by a vast choice of tools and software, makes this family of products the ideal choice, both for small projects and for entire platform decisions.

### *Making use of newer technology*

The ARM processor boards are a little more complex both functionally and in setting up compared to the (original Arduino) AVR boards. Do not be daunted. The Arduino IDE has been altered/kludged to accommodate the new ARM chip-set.

From the outset, those on the “bleeding edge” of the change have been challenged to adjust to the differences. The new concept on the mini development boards was to have a USB-micro port and a separate four pin J-Link connectivity port.

Roger Clark summarised it pretty well in his **Arduino STM32 -USB Serial and DFU** (8<sup>th</sup> June 2015) article - *“There is a common misconception that the STM32duino-bootloader and the older Maple boot-loader contain both DPU (upload) and USB support and also (Maple Serial) functionality. However, this is not the case.”*. Whereas the AVR boards can be programmed via the USB-B port and the port used for serial communications, the STM32 development boards are different.

So it is not surprising to find that ancillary preparation is needed before we can use the Arduino IDE to program and upload code. To that end, Roger Clark wrote in his forums, a basic three step process to overcome the short-comings the current (as of version 1.5.## of the Arduino IDE ).

There are three components that are required;-

1. Arduino IDE (covered previously).
2. Arduino STM32 supplementary modules.
3. Segger J-Link Software.

If you follow the steps outlined from here onward, you should be able to use the Arduino IDE for both the AVR and ARM chips without further change.

## Getting Arduino IDE to co-operate with ARM STM32F103 boards

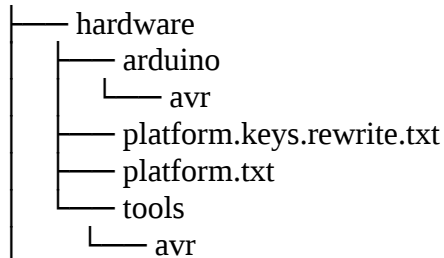
### *Downloading supplementary files*

Steps to follow;-

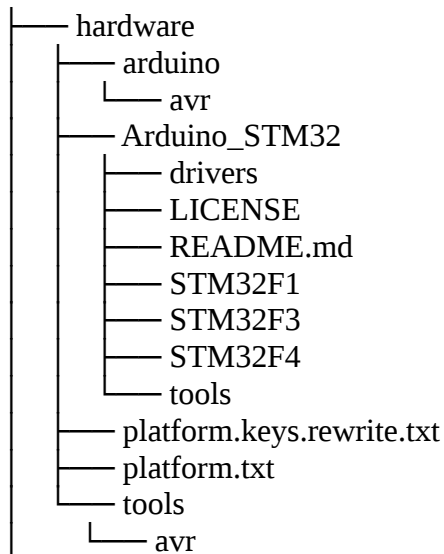
First we need the STM2 files;-

1. Go to GitHub to find and download the library latest **Arduino\_STM32** file. As of mid-October 2016, the version is 0.1.2. Simply, go to this link;- [https://github.com/rogerclarkmelbourne/Arduino\\_STM32](https://github.com/rogerclarkmelbourne/Arduino_STM32) and click on "Clone or download". This will be (at least it should be) found in your Downloads directory when loading has finished. Change the name from **Arduino\_STM-master** to just **Arduino\_STM**.
2. In the Arduino program folder/directory, ensure that a folder titled "hardware" exists. If it doesn't - create it. Note, with Linux, folders and files are case sensitive. I have found the folder must to be in lower case.
3. Using the GUI within Linux, **right click** the downloaded **Arduino\_STM32** and using the Archive Manager have the zip file extracted into the **Home/arduino/hardware** folder.

#### Pre STM2 Installation – Tree



#### Post STM2 Installation – Tree



Note that the original name **Arduino\_STM32-master** has not been retained. I prefer to drop the "-master" from all my Arduino library .zip importations. We then end up with a more simplistic file name of **Arduino\_STM32**.


Thanks to Roger Clark (Melbourne, Australia) who re-developed a Linux specific J-Link script (**jlink\_upload**) which you downloaded as part of the Arduino\_STM32. That's all there is to it.

## The All Important Download and Installation

The last stage is to download some additional software to “drive” the J-Link.

Go to the Segger site and download the necessary zipfile;-

<https://www.segger.com/downloads/jlink>

... navigate down the page until you come to “**J-Link Software and Documentation Pack**”. Next, at the bottom of that frame you will see “**Click for downloads**”. You will be shown a list of files titled “**J-Link Software and documentation pack for Linux, DEB Installer, ##-bit**”. The ## will be either 32 or 64 (depending on your operating system. Take note that you pick the entry with a little Linus emblem . **Left-click** on “**Download**”. Read the disclaimer and tick the term acceptance and then **left-click** the “**Download Software**” button.

Have a look in your “Downloads” folder. You should find an installation file;- **Jlink\_Linux\_V610g\_i386.deb**. The rest is simple. **Right-click** on the file and it will invite you to “**Open Gdebi Package Installer**”. This will expand the file and again, invite you to “Install Package”. Just **left-click** to continue. An oddity is that it doesn't exit instead. re-invites you to “Install Package” - Argh-h-h-h. Just cancel it.

That's it.

As a final check, get into terminal mode and enter JLinkExe. If everything is fine, you will presented with;-

```
$ JLinkExe
SEGGER J-Link Commander V6.10g (Compiled Oct 19 2016 16:40:35)
DLL version V6.10g, compiled Oct 19 2016 16:40:27

Connecting to J-Link via USB...FAILED: Can not connect to J-
Link via USB.
J-Link>exit
$
```

If you now start Arduino, when you go into Tools>Board: as you scroll down, you will see a list of STM32-type boards. We've made fantastic headway.

**That's it. The rest is up to you to do the programming.**

**Stop reading and start enjoying using both Arduino AVR with STM32!!!**

----- **END OF DOCUMENT** -----