Setting up your Raspberry Pi for programming

Mattscheibe, January 2021

If you are going for more than one project, set up one Rpi, then clone its SD-Card for all successive RPis.

1. Download latest RaspberrypiOS from raspberrypi.org and install it to an (empty) SD-Card (there are deatiled instructions on that page).

2. When starting your RPi for the first time, change **password** of standard user pi. Only then connect your RPi to your network.

3. Do some setup stuff (*locales*, *keyboard*, *time zone*) (might re-do this later, via "`sudo raspi-config`" of from preferences menu).

4. Update your software:

   ```
   sudo apt-get update ; sudo apt-get upgrade
   sudo apt-get install synaptic joe avrdude gcc-avr arduino arduino-mk screen gedit git
   sudo apt-get install gedit-plugins atool tcl-tclreadline xfig xfig-doc mercurial apt-file
   sudo apt-get install locate evince git checkinstall gdebi raspi-gpio libx11-dev imagemagick
   ```

   From now on, always type "`sudo apt-get update;sudo apt-get upgrade`" if you want to update your system, or use synaptic for that very purpose.

5. You RPi now has some software for programming **Arduino-Boards**. These can drive WS2812B LED strands autonomously.

   It also now has all the prerequisites for installing libraries to drive WS2812B LEDs by itself. You can use the arduino-IDE or gedit for typing in your programs.[1]

6. If you need **more software**, use synaptic (has about 60000 programs to be click-activated or de-activated). If some certain command is missing for you, you might search for it via
   `apt-file find` *`name_of_program_that's_reported_as_missing`*
   for example this way: `apt-file find arduino` . (Before doing a search like this, run `sudo apt-file update` to be up-to-date with your search-database.)

   Additional *externally offered* software can be downloaded to your RPi (e.g. via cloning github repositories), can be compiled und installed.[2] Instead of the standard "`sudo make install`" with self-compiled software, you might also say "`sudo checkinstall`". This would allow you to a. use synaptic (or any other package manger) to uninstall this software again later, and to b. get hold of a *.deb installation file, that you can find in your compilation directory and that you can save somewhere in order to distribute it to other computers as well.

   If you need to **find** a file or a directory on your SD-Card,
   `locate` *`name_of_what_you_search_for`*
   might help.

   locate command's database can be updated by running command `sudo updatedb`. You should run this command every now and then (and before calling locate for the first time).

   `find -name` *`name_of_what_you_search_for`*
   will also do a search, but starting only from the current directory (which you are in whilst evoking this command).

   To look for **string patterns** in all files of the current directory, type
   `fgrep` *`string_your_are_searching_for`* `*`
   If you type a file name instead of a star "`*`", the search will be in that file only. If you add an option "`-R`" to the line above, searching will take place in all sub-diretories, too.

   A good **document viewer**, e.g. for PDF files, is `evince`. A good **browser** is `chromium-browser`.

   (Network and other) **printers** can be set up by navigating your browser to http://localhost:631 after having installed software package cups.

   In a terminal program ("console"), `cd` *`name`* will change to a different directory, `mkdir` *`name`* will create a new directory, `ls` will list all files and subdirectories of the current directory (more verbose with `ls -l` or with `ls -al` including hidden files).
   Program files are made executable by typing `chmod +x` *`name_of_program_script`* and executed by typing *`./name_of_program_script`* . Any editing, compiling etc. of your own programs can be controlled by a proper Makefile that lists all dependencies and all necessary steps and can be activated using the `make` command.

---

[1] It might be a good idea to update your Arduino-IDE directly from its homepage. RaspberryPi OS's version is very, very old and cannot handle directory names with dashes. Newer versions are also much more versatile when adding external libraries to your software collection.

[2] One hint that you might find helpful as a beginner: If in a tutorial a line starts with a dash "`#`" as a prompt (instead of the standard "`$`"), you need root user's privileges to execute it. Put a "`sudo`" in front of such a command. And don't type the prompt itself. You might want to install ready-made *.deb-files with "`sudo gdebi` *`file.deb`*". This would trigger a download of everything that's necessary for such a file to be run.