# Raspberry Pi: Booting from USB + SSh Server + AP

Fernando G. Tinetti*

Technical Report TR-RT-01-2020
III-LIDI, Facultad de Informática, UNLP
*Also with CIC, Prov. de Buenos Aires
Argentina
Contact e-mail: fernando@info.unlp.edu.ar
May 2020

**Abstract.** In many IoT (Internet of Thing) and Real-Time projects (as well as other related application areas) we have found the usefulness of using a hardware/software development board or a SBC (Single Board Computer) providing WiFi for system control. Recent Raspberry Pi models have included WiFi facilities and its Linux-based operating system makes it a good candidate for project development and experimentation in the above mentioned areas. In this Technical Report a general configuration is explained for installing a setting a Raspberry Pi providing several features which are documented but not included in default operating system installations. Besides, we focus our settings in future stand-alone and battery-powered Raspberry applications so that we do not include a graphic interface full installation, but a Raspbian lite installation as the starting point. As an extra, the USB boot is set because of the large number of SD reader failure reports and experiences.

## 1.- Introduction

The focus of this report is to have a single installation process for Raspberry Pi 3 models (and similar ones), all with integrated WiFi network hardware. Our intended usage is rather biased from the "general" SBC one, because instead of considering the Raspberry Pi as a low-cost computer we will use them as an enhanced and powerful IoT/Real-Time development board. Besides, we regularly use Raspberry Pi boards in several courses at the Facultad de Informática of the Universidad Nacional de La Plata (UNLP), in engineering projects such as some of those described at [9] so the "regular" Raspberry Pi operation environment is not necessarily that of a SBC. Several of those projects usually involve a relatively heavy usage of GPIO (General-Purpose Input/Output) pins as well as different OS configurations which, in turn, leads to physical handling of the Raspberry Pi.

The physical handling of raspberry Pi mostly affects GPIO pins and the micro-SD reader. Broken GPIO pins are relatively easy to *replace*:

- By physically replacement
- By using other GPIO pins at the project configuration phase.

A broken micro-SD card reader, however, basically makes a Raspberry Pi useless because the micro-SD reader is needed at the Raspberry Pi startup process.

Booting from USB is not necessarily a critical feature, but it becomes *necessary* as the Raspberry Pi are being used (and sometimes *shared*) in different projects by different (student) teams and the micro-SD reader is prone to have failures or is broken. Fig. 1 shows an example of a physically broken micro-SD reader.

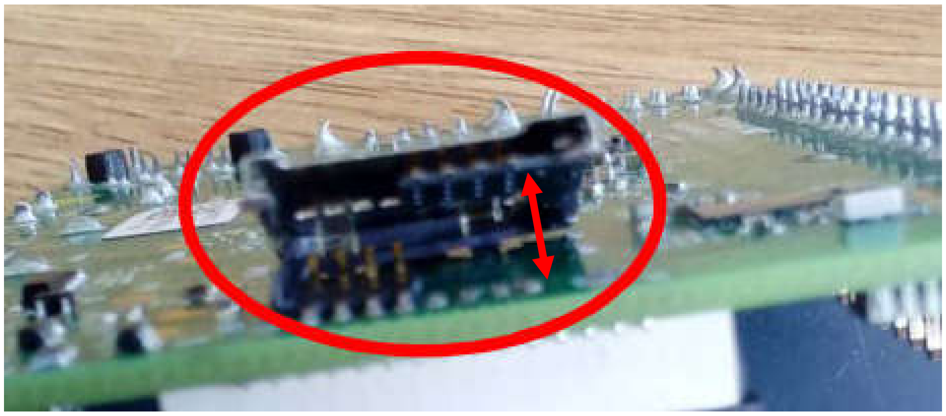

Figure 1: A Raspberry Pi 3B Broken Micro-SD Reader.

The Raspberry Pi wireless configuration as Access Point and providing ssh access (i.e. running the SSh server from startup) makes the Raspberry Pi a versatile hardware/software environment because:

- It is possible to pre-install the necessary software for project developments. Besides, if more software and/or libraries are needed, it is always possible to install it via another computer connected to the internet.
- It maintains the Raspberry Pi relatively isolated from the Internet, which reduces the likelihood of security issues along the project development process.
- It does not depend of other network software for accessing the Raspberry Pi, as well as maintains the Raspberry Pi in a state similar to that of the "production stage" for the projects we define and develop.
- Reduces the requirement of hardware for interaction with the Raspberry Pi: no keyboard, mouse, monitor, etc. are needed. This is also called a "headless" Raspberry Pi in the Raspberry Pi documentation [5].

## 2.- Basic Setup

Initially, we download the Raspbian lite version which, at the time of this writing is
2019-09-26-raspbian-buster-lite.zip
from [2]. The lite version is smaller and does not include the Raspbian graphical interface, which we will not use. However, it is not a requirement, the process is identical regardless the image, i.e. it is possible to install the so called "Raspbian Buster with desktop and recommended software" or the "Raspbian Buster with desktop" image versions.

You may set up the micro-SD card/Install the operating system image with the software you are most familiar with. In [3] the balenaEtcher [1] graphical SD card writing tool is suggested. In the Windows environment, we found the Win32DiskImager [10] easier/simpler to use. The objective, though, is to have the Raspbian operating system image in a micro-SD ready to boot a Raspberry Pi, independently of the tool used for copying the image.

Once the Raspbian image is copied in the micro-SD card, it is not needed to follow the "standard" recommended process of updating and upgrading the operating system (OS). The micro-SD card will be used only to set the Raspberry Pi for startup with USB, as explained in the next section.

## 3.- Enabling Boot from USB

The procedure for setting the Raspberry Pi so that it checks if a USB can be used at the startup process (i.e. with an operating system in a USB storage) is explained in [8]. In order to simplify the process, it is suggested that the same OS is copied in a USB storage device, with the same tool used to copy the image in the micro-SD card.

The Raspberry Pi is set to enable USB host boot mode by writing the so-called One Time Programmable Memory (OTPM). And the Raspbian writes the OTPM by adding a line in the config.txt file. Writing the config.txt file in Linux is shown in [8] as:
echo program_usb_boot_mode=1 | sudo tee -a /boot/config.txt
The specific line to add at the end of the config.txt file is
program_usb_boot_mode=1
Take into account that using the "echo" command as documented in the Raspberry site is assuming that the Raspberry Pi has been booted with the newly copied Raspbian image. However, the required text line in the config.txt file can be added before startup, thus avoiding two boots: one for adding the line in the config.txt file and one more for writing the OTPM. In Windows, the micro-SD is seen as shown in Fig. 2. The assigned drive identification is "E:" in this case, but it will depend on the specific Windows configuration/settings. The name, however, is contained in the micro-SD, i.e. it will always be shown as "boot". Depending on the Windows version, the micro-SD is reported as "having problems", but it is read anyway. Do not change anything in Windows other than the config.txt file in the micro-SD card root directory (i.e. do not use any windows tool to fix the reported errors/micro-SD card drive problem/s).

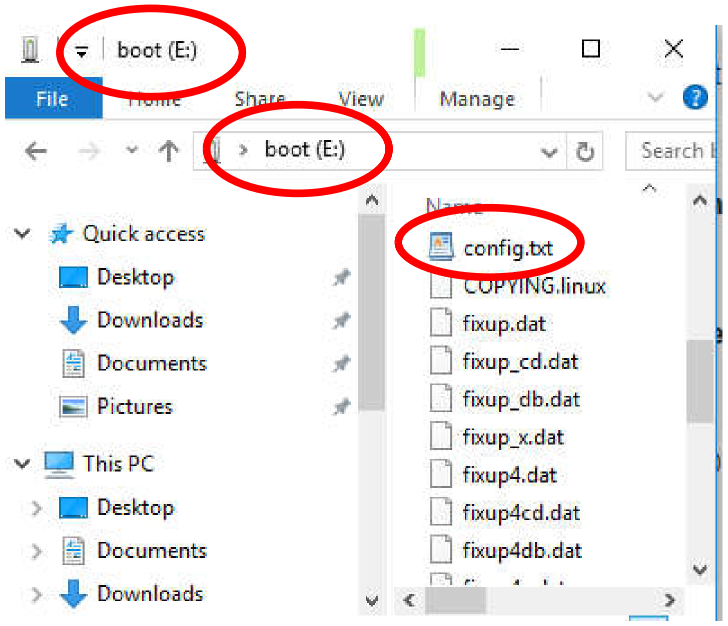

Figure 2: Raspbian Micro-SD Card Seen in a Windows Environment.

In the windows environment, we edited the config.txt file with a plain text editor such as WordPad and added the corresponding line at the end of the file. The actual last lines of the config.txt file used for enabling host boot mode are shown in Fig. 3. Do not change/edit any other line in the config.txt file, it contains all the Raspbian configurations options/setting for several Raspberry Pi hardware models (3B, 3B+, Zero W, 4, etc.). Summarizing, there is only one change to be made to a Raspbian image copied in a micro-SD card: add a specific text line in the config.txt file at the root directory.

```
[all]
#dtoverlay=vc4-fkms-v3d

program_usb_boot_mode=1
```

Figure 3: Last Lines of the config.txt File.

Once the Raspberry Pi is booted with this config.txt file, the Raspberry Pi will be enabled to boot from a USB storage in all subsequent boots (the OTPM is already written). Thus, the micro-SD card and its Raspbian image is useful only to write the OTPM, the rest of the Raspbian update, upgrade, and configuration will be made in the corresponding USB Raspbian image, as explained in the next section.

## 4.- Setting Up Raspbian, with ssh Server

Once the startup process is completed with the modified config.txt in the micro-SD card, it is possible to boot from the USB storage containing the Raspbian copied as suggested above. Actually, the Raspbian in the USB storage and the micro-SD card are identical except for the config.txt file: the micro-SD card one contains one more text line. Before booting the Raspberry Pi with the “brand new” Raspbian in the USB storage, we suggest to enable the Raspbian ssh server at startup so that we will be able to remotely access the Raspberry Pi and complete the installation and configuration process.

The Raspberry Pi can be configured for starting the ssh server at system startup [6], and this configuration is sometimes part of a “headless” Raspberry Pi setup [5]. There are several alternative ways of enabling the ssh server at startup, one of the simple ones only requires to create a file named “ssh” (without any extension and without any contents) in the root directory (*folder*) of a USB copied with the Raspbian image in Windows. Thus, in the Windows environment, after copying the Raspbian image in a USB storage and before disconnecting the USB storage we create the ssh file in the USB drive root directory.

For the startup process after writing the OTPM with the micro-SD set up described above, we arranged the Raspbian setup as follows:

- ⇨ Remove the micro-SD card
- ⇨ Connect the Raspberry Pi Ethernet interface to an internet router
- ⇨ Insert the USB storage in a Raspberry Pi USB slot
- ⇨ Turn on the Raspberry Pi.

Once the Raspberry Pi boot process is completed from the USB storage, we were able to login via ssh, using the default sudoer Raspberry Pi user, which is
pi
and its default password is
raspberry
At this point, we followed the suggestion of updating and upgrading the Raspbian OS, as documented in [7], i.e. with

```
$ sudo apt update; sudo apt full-upgrade
```

Now, it is possible to install and configure the Raspberry Pi for the specific task/s it will be used in. Our next step will be explained in the next section: set the Raspberry Pi wireless interface as a WiFi Access Point.

## 5.- Raspberry Pi as a WiFi Access Point

The documented steps for setting the Raspberry Pi wireless interface as an access point (AP) are documented in [4]. We copy here the sequence of steps without further explanations, which are given in the Raspberry Pi documentation.

Install and stop dnsmasw and hostapd services:

```
$ sudo apt install dnsmasq hostapd
$ sudo systemctl stop dnsmasq
$ sudo systemctl stop hostapd
```

Edit the dhcpcd,conf file for setting the wireless interface fixed IP. This IP must be used to access the Raspberry Pi once another wireless device is connected to the LAN provided by the Raspberry Pi, i.e. the Raspberry Pi will "provide" a wireless LAN access in which the same Raspberry Pi will be accessible. The three text lines between the lines of "====" are added at the end of the file dhcpcd.conf file:

```
$ sudo nano /etc/dhcpcd.conf
=============
interface wlan0
   static ip_address=192.168.4.1/24
   nohook wpa_supplicant
=============
```

Restart the service dhcpcd:

```
$ sudo service dhcpcd restart
```

Create a new dnsmasq.conf file. In this case, after renaming the original one, the new one will only have the two text lines shown between the lines of "====":

```
$ sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
$ sudo nano /etc/dnsmasq.conf
=============
interface=wlan0      # Use the require wireless interface - usually wlan0
dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h
=============
```

Restart the dsnmasq service:

```
$ sudo systemctl start dnsmasq
```

Create the file /etc/hostapd/hostapd.conf with the necessary contexts for setting the Raspberry Pi AP. The most important settings are

- ⇨ ssid and wpa_passphrase, which should be set according to your needs, remember: the passphrase should be between 8 and 64 characters in length.

Take into account that a single error in a single line may generate a rather meaningless error report and the AP will not be set up correctly. As in the previous example the text lines between "====" (not including these lines) should be in the hostapd.conf file:

```
$ sudo nano /etc/hostapd/hostapd.conf
=============
interface=wlan0
driver=nl80211
```

```
ssid=NameOfNetwork
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=AardvarkBadgerHedgehog
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
==============
```

Change a single text line in the /etc/default/hostapd file, find the line with
#DAEMON_CONF …
and replace it so that it "refers to" the hostapd.conf file recently created:

```
$ sudo nano /etc/default/hostapd
==============
DAEMON_CONF="/etc/hostapd/hostapd.conf"
==============
```

Config and start services:

```
$ sudo systemctl unmask hostapd
$ sudo systemctl enable hostapd
$ sudo systemctl start hostapd
```

Check status:

```
$ sudo systemctl status hostapd
$ sudo systemctl status dnsmasq
```

If restarted at this point, the Raspberry Pi will be configured as an access point. Since the ssh server it is already enabled (explained in Section 4.- Setting Up Raspbian, with ssh Server), the Raspberry Pi is fully installed and configured so that

- Raspbian is updated and upgraded
- The Raspberry Pi provides a WiFi AP, connected devices will be assigned IP numbers in the LAN 192.168.4.xx
- The WiFi IP of the Raspberry Pi is 192.168.4.1
- It is possible to login via ssh to the Raspberry Pi

And this configuration is suggested, so that:

- Every software tool/library can be installed if the Raspberry Pi is connected using the Ethernet interface. We suggest to connect the Ethernet interface only for this task, actually, for avoiding security issues and clear identification and isolation of specific projects software dependencies.
- Project development via ssh access and project experimentation via the Raspberry Pi WiFi AP. Eventually, the same WiFi AP will be used in the production stage.

Two interesting configurations are available as "collateral effects" of setting the Raspberry Pi WiFi interface as access point:

1) The IP address of the WiFi can be pre-determined/set in advance. This provides a well-known IP for possible servers, such as ssh, http, etc.

2) The DHCP can be configured so that specific WiFi stations receive specific IP addresses (by MAC Address configuration, for example).

## 6.- Identifying One-Time and Regular Tasks

If a single micro-SD card and USB storage can be left for installation and configuration of a "Raspberry Pi: Booting from USB + SSh Server + AP", it is suggested to follow the sequence:

1. Copy the Raspbian OS in micro-SD and USB storage. This may be in the Windows or Linux environment, using the balenaEtcher, Win32DiskImager (only available in Windows), or any similar tool.
2. Set the micro-SD Raspbian for writing the OTPM, for enabling the Raspberry Pi to boot from a USB storage, as explained in section "3.- Enabling Boot from USB".
3. Set the USB storage Raspbian for enabling the ssh server at system startup, as explained at the beginning of section "4.- Setting Up Raspbian, with ssh Server".

This sequence of steps is suggested to be carried out without any Raspberry Pi, i.e. use a regular computer (either desktop or laptop) with the OS you prefer. The micro-SD can be used in as many Raspberry Pi as needed, no other/specific tasks are required to set every Raspberry Pi. The next steps, now involving the Raspberry Pi are

4. Boot the Raspberry Pi with the micro-SD for OTPM writing.
5. Shutdown the Raspberry Pi and remove the micro-SD.
6. Insert the USB storage.
7. Connect the Raspberry Pi through Ethernet to a router with Internet access.
8. Boot the Raspberry Pi.
9. Follow the steps in section "5.- Raspberry Pi as a WiFi Access Point".

At this point, the USB storage can be used for starting and setting the Raspberry Pi with all the requirements (Raspbian software tool/s and configuration/s) specific to each project. Actually, the USB filesystem at this point can be directly copied to any USB and used in any Raspberry Pi.

## References

[1] Balena, balenaEtcher - Home
https://www.balena.io/etcher/

[2] Raspberry Pi Foundation, Download Raspbian for Raspberry Pi
https://www.raspberrypi.org/downloads/raspbian/

[3] Raspberry Pi Foundation, Installing operating system images - Raspberry Pi Documentation
https://www.raspberrypi.org/documentation/installation/installing-images/README.md

[4] Raspberry Pi Foundation, Setting up a Raspberry Pi as a Wireless Access Point
https://www.raspberrypi.org/documentation/configuration/wireless/access-point.md

[5] Raspberry Pi Foundation, Setting up a Raspberry Pi headless
https://www.raspberrypi.org/documentation/configuration/wireless/headless.md

[6] Raspberry Pi Foundation, SSH (Secure Shell)
https://www.raspberrypi.org/documentation/remote-access/ssh/

[7] Raspberry Pi Foundation, Updating and upgrading Raspbian

https://www.raspberrypi.org/documentation/raspbian/updating.md

[8] Raspberry Pi Foundation, USB mass storage boot
https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/msd.md

[9] Fernando G. Tinetti, "Some work on Real-Time & Embedded Systems"
http://fernando.thats.im/links/embed-rt/index.html

[10] Win32 Disk Imager
https://sourceforge.net/projects/win32diskimager/