Raspberry Pi
Administration

Thierry Vaira <tvaira@free.fr>

v1.0 - 16 février 2019

Sommaire

1. Présentation
2. Distribution Raspbian
3. GPIO
4. Administration
5. Réseau
6. Bluetooth
7. Mode *Kiosk*

Sommaire

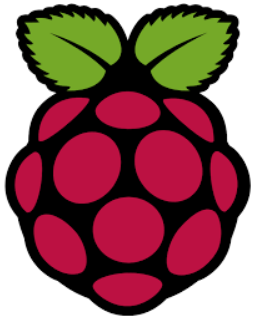
1 Présentation
2 Distribution Raspbian
3 GPIO
4 Administration
5 Réseau
6 Bluetooth
7 Mode *Kiosk*

1 Présentation

- Introduction
- Les différents modèles
- Caractéristiques
- Systèmes d'exploitation
- Distributions
- Systèmes alternatifs
- Liens

La (le) Raspberry Pi

- nano-ordinateur monocarte à processeur ARM conçu par David Braben (UK) 2011-2012
- destiné à encourager l'apprentissage de la programmation informatique
- plus de 12 millions d'exemplaires vendus
- disponible à moins de 40 € (entre 10 et 35 €)



✍ Il est généralement vendu "nu" (carte mère seule, sans boîtier, alimentation, clavier, souris ni écran) dans l'objectif de diminuer les coûts et de permettre l'utilisation de matériel de récupération.

Soc ARM

Plusieurs générations de Raspberry Pis ont été produites.

- Tous les modèles disposent d'un système SoC (*System on a Chip*) Broadcom avec une unité centrale de traitement (CPU) et une unité de traitement graphique (GPU) d'architecture ARM.
- Les architectures ARM sont des architectures de type RISC 32 bits (ARMv1 à ARMv7) et 64 bits (ARMv8) développées par ARM Ltd depuis 1990.
- Le SoC Broadcom BCM2835 utilisé dans la première génération de Raspberry Pi comprend un CPU ARM11 (ARMv6) de 700 MHz, un GPU VideoCore IV et de la RAM. Il possède un cache de niveau 1 (L1) de 16 Ko et un cache de niveau 2 (L2) de 128 Ko principalement utilisé par le GPU (*Graphics Processing Unit*).
- Le Raspberry Pi 3+ utilise un SoC Broadcom BCM2837B0 avec un processeur ARM Cortex-A53 (ARMv8-A) quatre coeurs 1,4 GHz à 64 bits, avec une mémoire cache L2 partagée de 512 Ko.

Versions

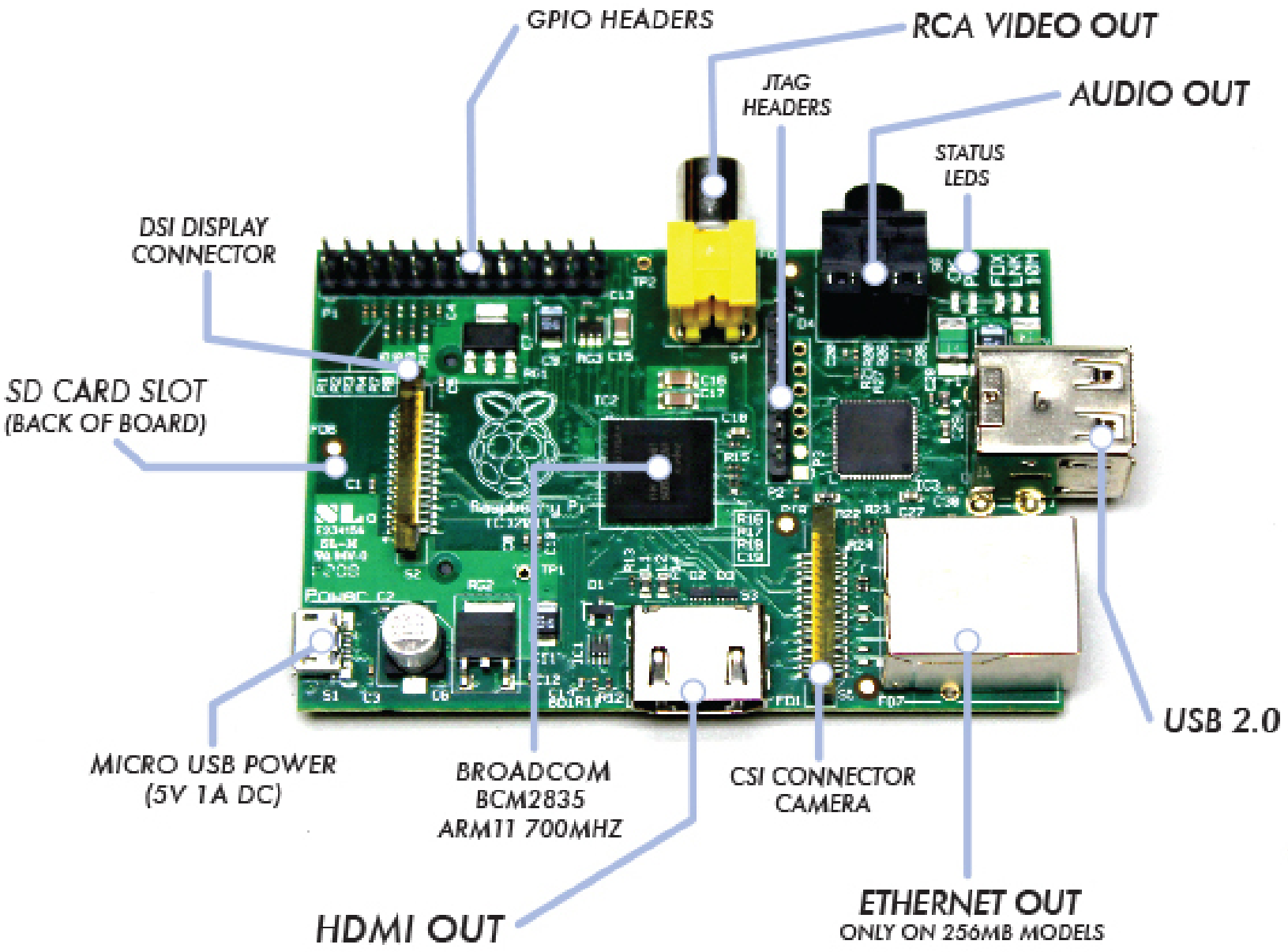
Depuis 2012, la Raspberry Pi s'est déclinée en plusieurs versions :

- **Modèle A** : ARMv6 à 700 MHz Broadcom 2835 - RAM 256 Mo ;
- **Modèle A+** : Lecteur de carte microSD - GPIO 40 broches ;
- **Modèle B** Rev1 : RAM 512 Mo - 2 ports USB 2.0 - 1 port réseau Fast Ethernet (10/100 Mbit/s) ; Rev2 : JTAG - I2C ;
  Modèle B+ : 4 ports USB 2.0 - micro SD ;
- **Modèle 2B** (Raspberry Pi 2) : quatre cœurs ARMv7 à 900 MHz Broadcom BCM2836 - RAM 1GO ;
- **Modèle 3B** (Raspberry Pi 3) : quatre cœurs ARM Cortex-A53 à 1,2 GHz Broadcom BCM2837 64 bits - Wifi 802.11n et Bluetooth 4.1 ;
- **Modèle 3B+** (Raspberry Pi 3) : quatre cœurs ARM Cortex-A53 cadencé à 1,4 GHz Broadcom BCM2837B0 64 bits - WiFi Dual-band 802.11ac et Bluetooth 4.2 ;
- **Modèle Zero W** : ARM11 à 1 GHz Broadcom BCM2835 - Mini-HDMI - Wifi 802.11n et Bluetooth 4.1 ;

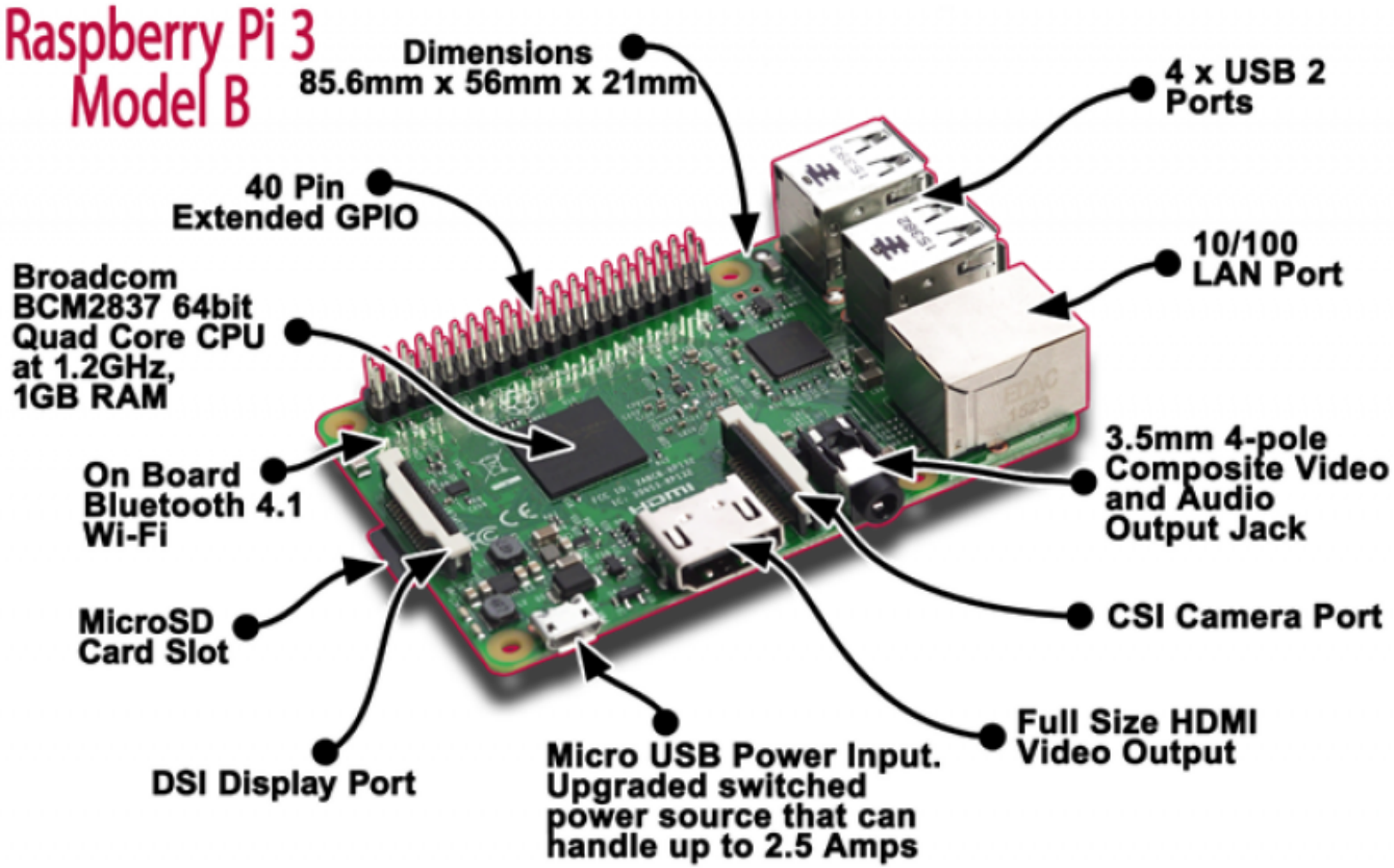
Tableau comparatif

Generation	Model A		Model B						Zero		
	1	1 +	1	1 +	2	2 ver 1.2	3	3+	PCB ver 1.2	PCB ver 1.3	W (wireless)
Release date	February 2013	November 2014	April–June 2012	July 2014	February 2015	October 2016	February 2016	14 March 2018	November 2015	May 2016	28 February 2017
Target price (USD)	$25	$20	$35	$25	$35	$35	$35	$35	$5	$5	$10
Instruction set	ARMv6Z (32-bit)				ARMv7-A (32-bit)	ARMv8-A (64/32-bit)			ARMv6Z (32-bit)		
SoC	Broadcom BCM2835				Broadcom BCM2836	Broadcom BCM2837		Broadcom BCM2837B0	Broadcom BCM2835		
FPU	VFPv2; NEON not supported				VFPv3 + NEON	VFPv4 + NEON			VFPv2; NEON not supported		
CPU	1× ARM1176JZF-S 700 MHz				4× Cortex-A7 900 MHz	4× Cortex-A53 900 MHz	4× Cortex-A53 1.2 GHz	4× Cortex-A53 1.4 GHz	1 GHz single-core ARM1176JZF-S		
GPU	Broadcom VideoCore IV @ 250 MHz (BCM2837: 3D part of GPU @ 300 MHz, video part of GPU @ 400 MHz) OpenGL ES 2.0 (BCM2835, BCM2836: 24 GFLOPS / BCM2837: 28.8 GFLOPS) MPEG-2 and VC-1 (with license), 1080p30 H.264/MPEG-4 AVC high-profile decoder and encoder (BCM2837: 1080p60)										
Memory (SDRAM)	256 MB (shared with GPU)	512 MB (shared with GPU) as of 4 May 2016. Older boards had 256 MB (shared with GPU)	1 GB (shared with GPU)						512 MB (shared with GPU)		

Raspberry Pi B

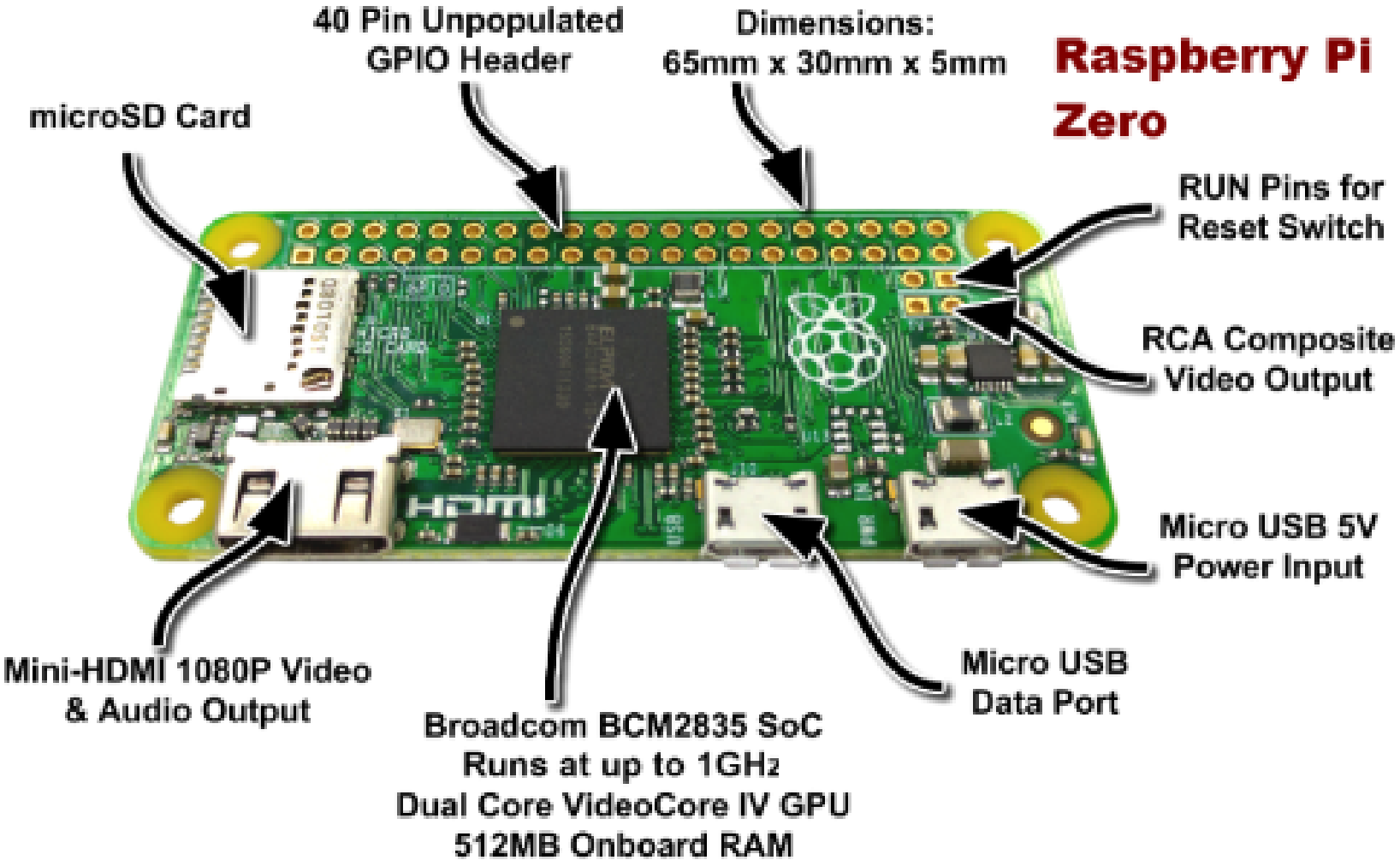


Raspberry Pi 3B



✍ Même si le processeur supporte le 64 bits, la Rapsberry Pi 3 fonctionne encore actuellement en mode 32 bits.

Raspberry Pi Zero



Caractéristiques

- Processeur ARM (voir modèles)
- Sorties vidéo composite et HDMI
- USB, Ethernet, Wifi et Bluetooth (suivant les modèles)
- GPIO, UART, SPI, I2C
- Audio
- DSI/CSI (*Display/Camera Serial Interface*)

Affichage

Il existe trois options de connexion pour l'affichage :

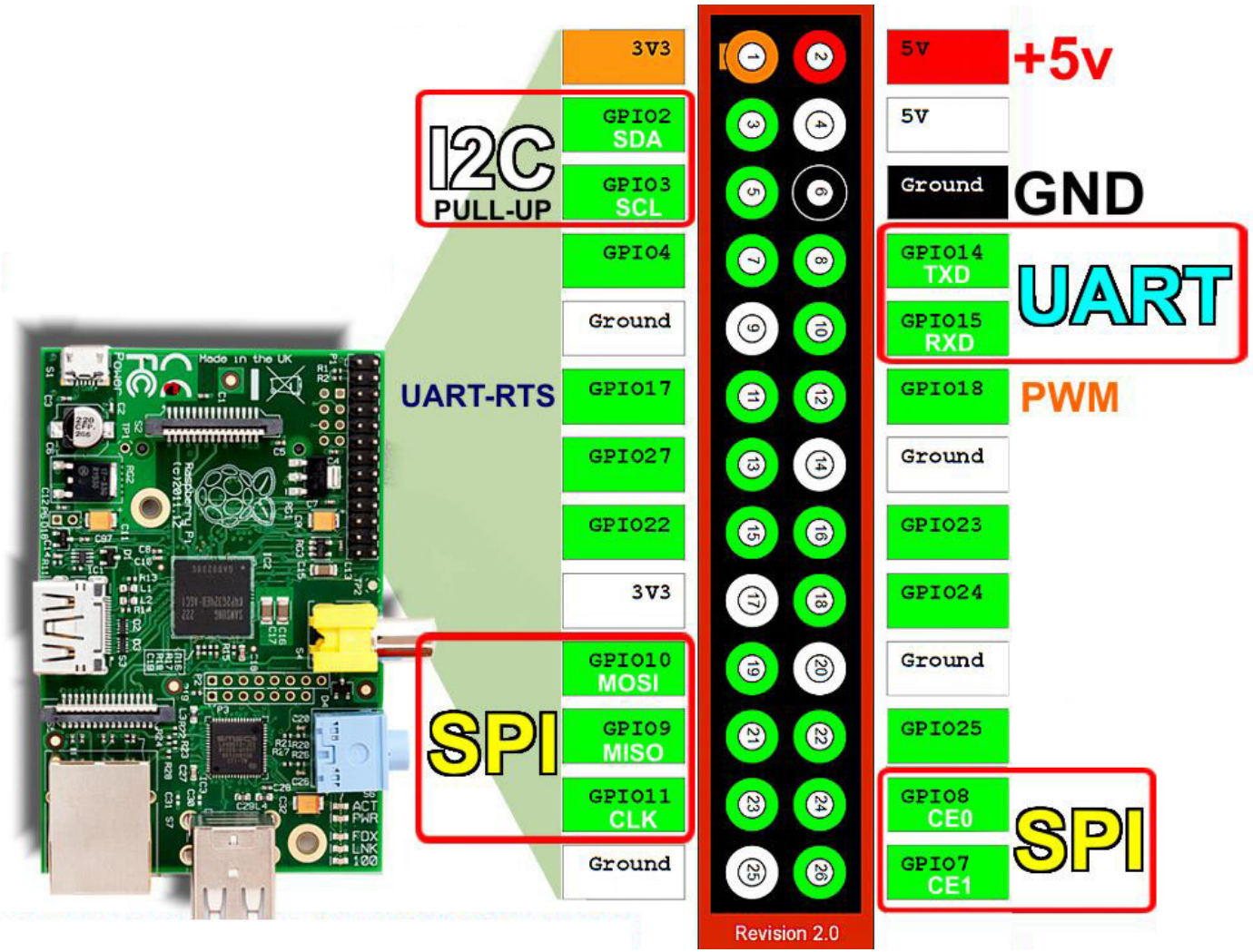
- HDMI (haute définition) : Les téléviseurs HD et des moniteurs LCD peuvent être connectés à l'aide d'un câble HDMI "mâle" standard ou d'un adaptateur DVI ou VGA. Les versions HDMI 1.3 et 1.4 sont prises en charge. L'audio et la vidéo sont transmis via HMDI, mais les entrées HDMI ne sont pas supportées. cf. elinux.org/RPiconfig#Video

- Composite (définition standard) : Les téléviseurs plus anciens peuvent être connectés à l'aide de la vidéo composite (câble RCA jaune) ou via la prise péritel (avec un adaptateur vidéo composite vers péritel). Les téléviseurs aux formats PAL et NTSC sont pris en charge. L'audio est disponible à partir de la prise jack 3,5 mm et peut être utilisé avec un téléviseur, des écouteurs ou un amplificateur.

- DSI (*Display Serial Interface*) : https://www.raspberrypi.org/products/raspberry-pi-touch-display/

✍ Il n'y a pas de sortie analogique VGA disponible.

GPIO



http://pinout.xyz/

Systèmes d'exploitation

➤ La Raspberry Pi n'ayant pas de mémoire de masse interne ni de système d'exploitation intégré, on aura besoin d'une carte SD préchargée avec un système d'exploitation.

➤ Depuis sa commercialisation, la Raspberry Pi exécute des variantes du système d'exploitation libre GNU/Linux-Debian. Suivant les modèles, on pourra aussi exécuter :

- plusieurs variantes du système d'exploitation libre GNU/Linux
- Windows 10 IoT Core (`developer.microsoft.com/en-us/windows/iot/downloads`)
- Android Pi (`androidpi.wikia.com/wiki/Android_Pi_Wiki`)

Distributions

- GNU/Linux : Debian « stretch » (Raspbian), Fedora, Arch Linux, Gentoo, Slackware, Ubuntu et Suse
- Autres : Firefox OS, RISC OS, NetBSD, FreeBSD, ...
- Distributions spécialisées : médias center (LibreELEC/OpenElec, OSMC/Rasbmc, ...), audio, retrogaming, réseaux (Kali), serveurs (OwnCloud, ...), ...

✍ **NOOBS** est un programme d'installation simple de système d'exploitation contenant Raspbian et LibreELEC. Il fournit également une sélection de systèmes d'exploitation alternatifs qui sont ensuite téléchargés sur Internet et installés. Voir aussi : **BerryBoot**.

Alternatives

Le succès de la Raspberry Pi a créé un développement de solutions alternatives :

- BeagleBoard : dès 2008, le projet BeagleBoard proposait une carte mère ARM *low cost*
- Orange Pi : propose probablement les prix les plus bas
- Banana Pi : un des premiers clones du Raspberry Pi
- Odroid : le haut de gamme
- Olimex : cartes mères ARM embarquées en Open Hardware
- ...

Liens

www.raspberrypi.org

www.raspberrypi.org/downloads/

elinux.org/RPi_Hub

raspbian-france.fr

www.raspberrypi-france.fr

www.framboise314.fr

raspberry-pi.developpez.com/cours-tutoriels/

Sommaire

1 Présentation

2 Distribution Raspbian

3 GPIO

4 Administration

5 Réseau

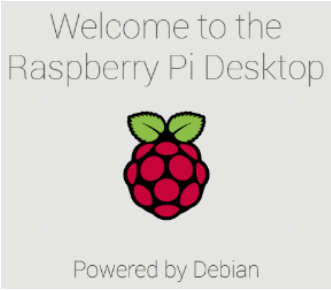
6 Bluetooth

7 Mode *Kiosk*

2 Distribution Raspbian
- Présentation
- Installation
- Post-Installation

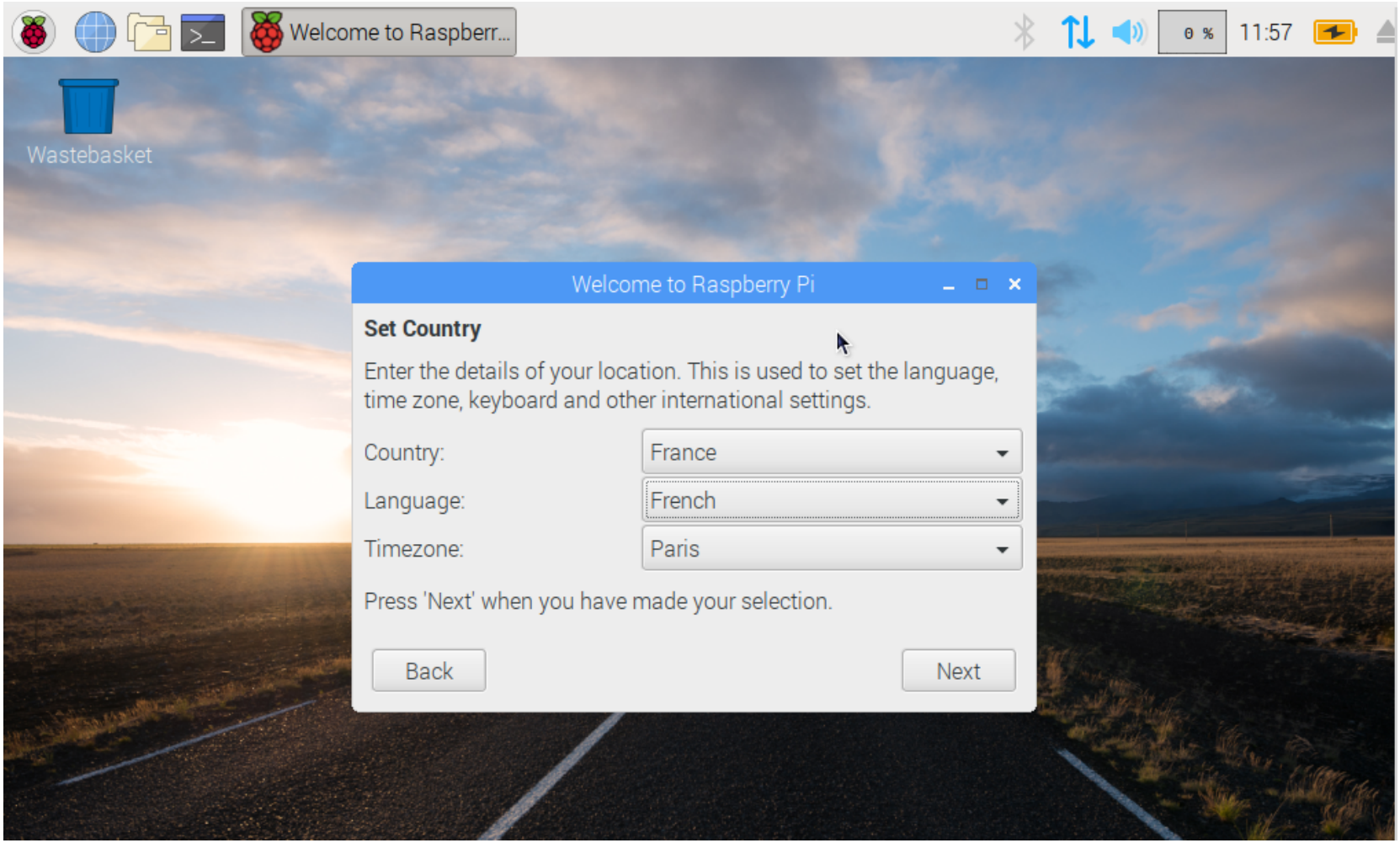
Raspbian

- portage de la distribution Debian pour la Rapsberry Pi (ARM)
- considérée comme la distribution par défaut
- distribution classique et non pas embarquée ni « temps réel »
- livré pré-installé avec de nombreux logiciels pour l'éducation, la programmation et l'utilisation générale
- `www.raspberrypi.org/downloads/raspbian/` ➟ deux versions DESKTOP et LITE
- existe en version PC et MAC (Raspberry Pi Desktop) : `https://www.raspberrypi.org/downloads/raspberry-pi-desktop/`



✍ La plupart des utilisateurs développent en Python !

Raspberry Pi Desktop



Mode graphique

Par défaut, on aura :

- Serveur d'affichage (*Display Server*) : **Xorg** (version officielle de X11) ➟ `/etc/X11/`
- Environment de bureau : **RPD** (*Raspberry Pi Desktop*) un **LXDE** (*Lightweight X11 Desktop Environment*) modifié ➟ `/etc/xdg/`
- Gestionnaire de fenêtres : **Openbox** ➟ `/etc/X11/openbox/` et `/etc/xdg/openbox/`
- Gestionnaire de *login* : **LightDM** ➟ `/etc/lightdm/lightdm.conf`

```
$ dpkg -get-selections | grep -E "lightdm|xorg|lxde|openbox"
```

✍ www.raspberrypi.org/forums/viewtopic.php?t=133691

Installation

La Raspberry Pi n'ayant pas de mémoire de masse interne ni de système d'exploitation intégré, on utilisera une carte SD sur laquelle un système d'exploitation sera installé.

Pour cela, il faudra :

- formater (ou effacer) une carte SD (facultatif)
- installer une image sur la carte SD

Formater la carte

- Windows et Mac : L'association **SanDisk** (SD) fournit un utilitaire pour le formatage de la carte SD ➠ https://www.sdcard.org/downloads/formatter_4/index.html
- GNU/Linux : l'outil de partitionnement **GParted** (ou `parted`) permet le formatage en FAT32. Sinon on utilisera `fdisk` ou `sfdisk` et `mkfs.vfat -F 32`
- Effacer la carte SD :

```
$ dd if=/dev/zero of=$DRIVE bs=1024
```

`$DRIVE` contient le chemin vers le périphérique (`/dev/sdg` par exemple). Si on ajoute un numéro, on effacera seulement la partition (`/dev/sdg1` par exemple).

Le formatage détruit toutes les données actuellement présentes sur la carte.

Télécharger l'image

Il faut télécharger l'image du système (portant généralement l'extension `.img` ou `.zip`) :

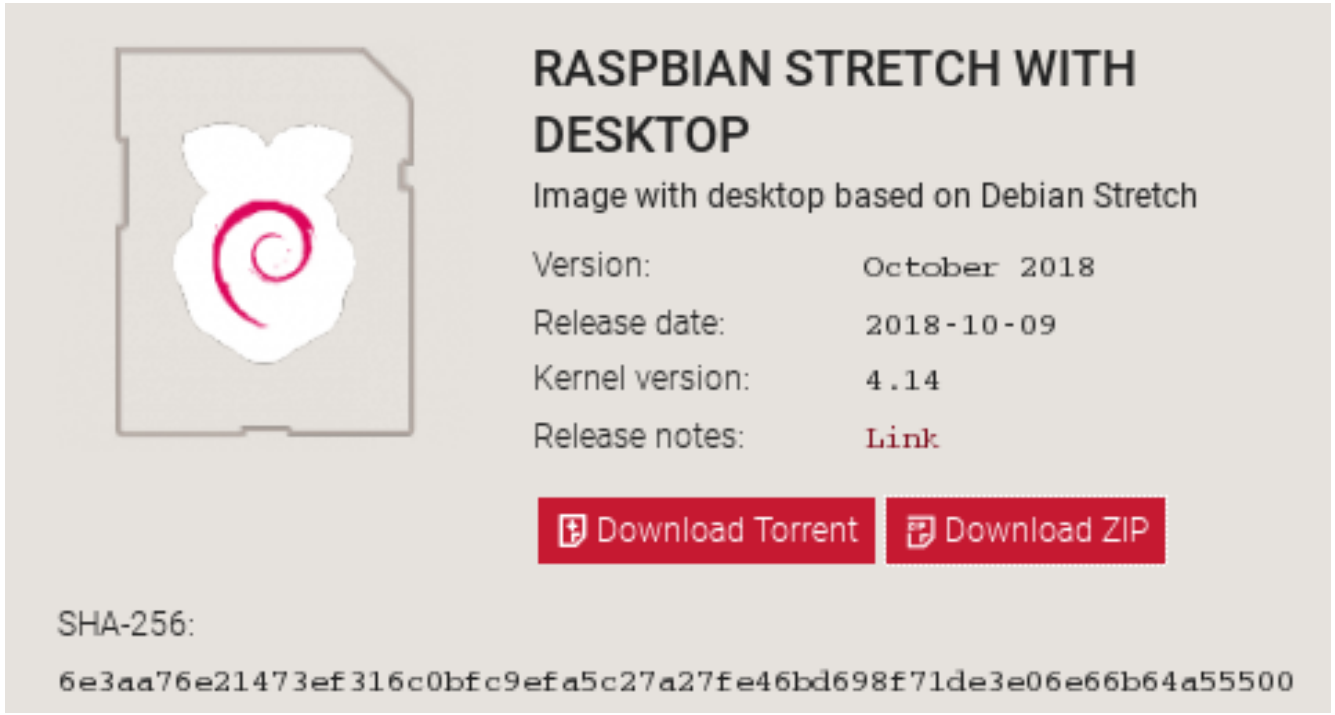
- `downloads.raspberrypi.org` pour toutes les versions proposées
- `www.raspberrypi.org/downloads/raspbian/` pour les deux dernières versions DESKTOP et LITE
- La commande `wget` permet de télécharger en mode CLI sous GNU/Linux :

```
$ wget http://raspbian-france.fr/download/raspbian_latest.zip
```

✍ L'image Raspbian est au format **ZIP64**. Les outils supportant ce format : **7-Zip** sous Windows (`www.7-zip.org`), The **Unarchiver** sous Mac (`unarchiver.c3.cx/unarchiver`) ou **Unzip** sous GNU/Linux

Vérifier l'image

Vous pouvez vérifier l'intégrité de l'archive téléchargée avant de l'écrire sur la carte SD :




```
$ sha256sum 2018-10-09-raspbian-stretch.zip
6e3aa76e21473ef316c0bfc9efa5c27a27fe46bd698f71de3e06e66b64a55500 2018-10-09-
    raspbian-stretch.zip
```

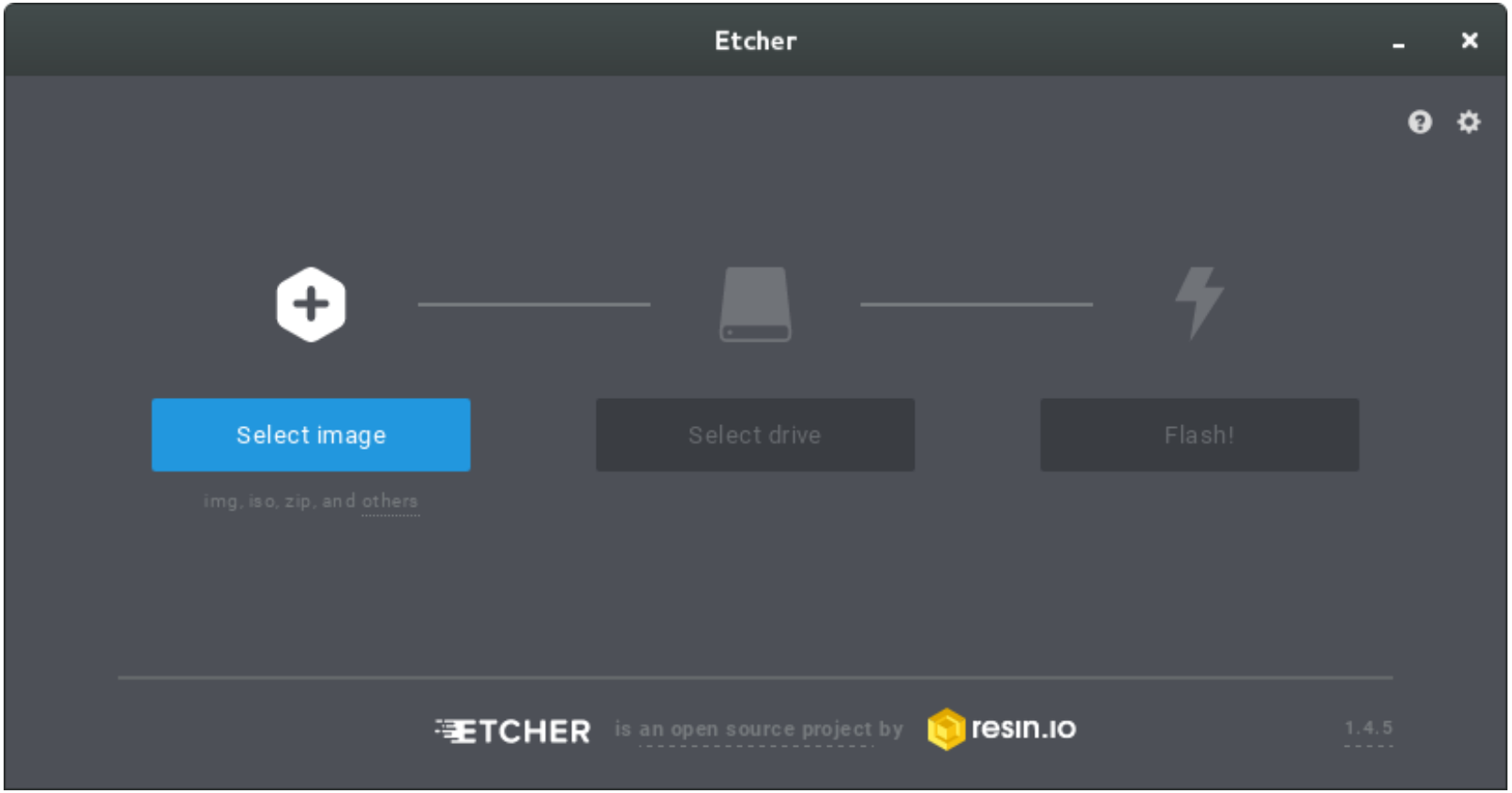
Installer une image sur la carte SD

➳ Utiliser **Etcher** (`etcher.io`) qui est un outil libre de gravure d'images (ZIP, img, iso) sur différents supports clé USB, Carte SD, pour GNU/Linux, Windows, MacOS. Ou :

- Windows : avec l'utilitaire **Win32 Disk Imager** ➟ `https://sourceforge.net/projects/win32diskimager/files/latest/download`.
- Mac : avec **PiFiller** (`ivanx.com/raspberrypi/files/PiFiller.zip`) ou encore **PiWriter** (`https://sourceforge.net/projects/piwriter/`). On peut aussi utiliser la commande dd comme sous Linux.
- GNU/Linux : avec la commande `dd`.

✍ Une image est une copie conforme (à l'identique) d'une carte SD.

Etcher


Etcher
Select image
img, iso, zip, and others
Select drive
Flash!
ETCHER
is an open source project by
resin.io
1.4.5

Installer une image sur la carte SD (CLI sous Linux) I

Dans un terminal :

- Exécuter la commande **`lsblk`**. La colonne de gauche donnera le nom de périphérique de votre carte SD et le nom des éventuelles partitions sur celle-ci. La colonne de droite indiquera où les partitions ont été montées (si elles ne l'ont pas été, ce sera vide).
- Démonter toutes les partitions avec `umount`. Par exemple, **`umount /dev/sdX1`** (remplacez `sdX1` par le nom de périphérique de votre carte SD et le numéro de la partition).
- Télécharger l'image au format `.img` ou `.zip` avec la commande **`wget`**.
- Écrire l'image (format `.img`) sur la carte avec la commande :

```
$ dd bs=4M if=raspbian.img of=/dev/sdX conv=fsync &
```

Installer une image sur la carte SD (CLI sous Linux) II

- Ou écrire l'image (format `.zip`) sur la carte avec la commande :

```
$ unzip -p raspbian.zip | sudo dd of=/dev/sdX bs=4M conv=fsync &
```

- Suivre l'avancement :

```
$ kill -s SIGUSR1 $(pidof dd)
```

- Finalisez les écritures avec la commande **`sync`**.

Structure de la carte SD

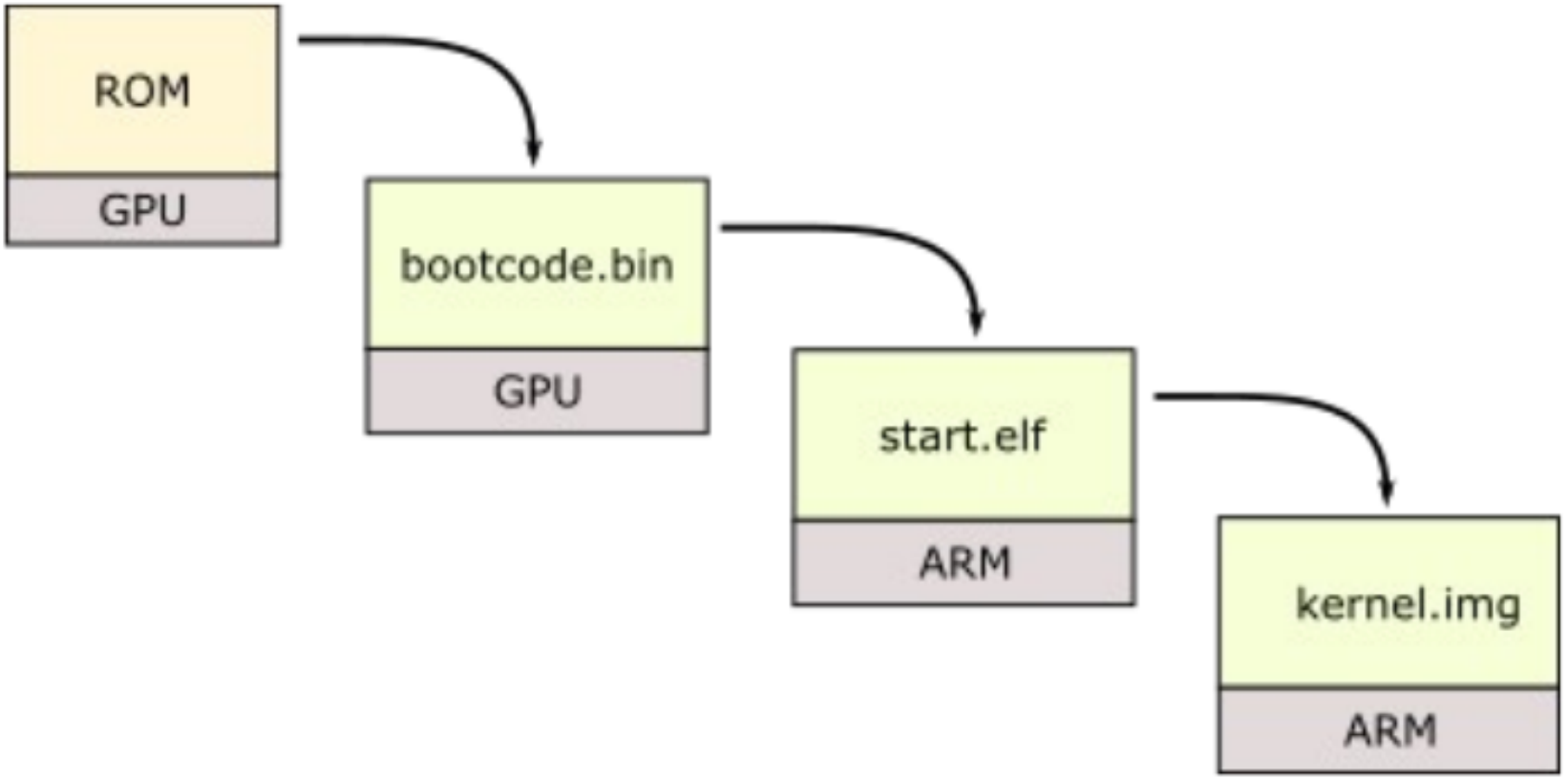
La carte SD contient au minimum 2 partitions :

- Une partition **VFAT** de démarrage (*boot*)
- Une partition **EXT4** contenant l'arborescence de la distribution (*rootfs*)

La partition VFAT contient les fichiers de démarrage, de configuration et les images noyaux disponibles :

- `bootcode.bin` : initialisation du GPU (*bootloader* phase 1)
- `start.elf` (+ `fixup.dat`) : initialisation (*binary firmware image*) du GPU (*bootloader* phase 2)
- `kernel.img` : noyau par défaut
- `cmdline.txt` : configuration du noyau Linux
- `config.txt` : configuration (matérielle) de la Raspberry Pi

Séquence de *boot*



Fichier cmdline.txt

Le noyau Linux accepte des paramètres lors du démarrage (*boot*). Ces paramètres sont définis dans le fichier **cmdline.txt** de la partition de démarrage.

- console série : **console=serial0,115200 console=tty1**
- système de fichiers principal : **root=/dev/mmcblk0p2 rootfstype=ext4**
- ...

✍ **www.raspberrypi.org/documentation/configuration/cmdline-txt.md** et **elinux.org/RPi_cmdline.txt**

Fichier config.txt

La Raspberry Pi utilise le fichier de configuration **config.txt** de la partition de démarrage à la place du BIOS d'un PC conventionnel. Le fichier est lu par le GPU avant que le processeur ARM et Linux ne soient initialisés. Toutes les modifications ne prendront effet qu'après le redémarrage de la Raspberry Pi.

- affichage des paramètres avec la commande :

```
$ vcgencmd get_config <config>
```

- options pour configurer la mémoire, la vidéo, l'audio, la camera, les gpio, l'*overclocking*, ...

✍ www.raspberrypi.org/documentation/configuration/config-txt/README.md et elinux.org/RPiconfig

Installation *headless* (Raspberry Pi Zero W) I

Un système *headless* est un système qui fonctionne sans moniteur (écran), clavier et souris. Ce type de système est généralement contrôlé via une connexion réseau ou liaison série. Cela peut être le cas d'une **Raspberry Pi Zero W qui dispose d'une interface sans-fil**.

- installer l'image sur la carte SD (une version LITE suffira pour la RPI Zero W)
- monter la carte SD puis exécuter la commande `lsblk` pour déterminer les points de montage
- activer le serveur SSH : il suffit de créer un fichier `ssh` dans la partition *boot* ➠ `$ sudo touch /media/tv/boot/ssh`

Installation *headless* (Raspberry Pi Zero W) II

- configurer le WiFi : il suffit de créer un fichier `wpa_supplicant.conf` dans la partition *boot* ou `/etc/wpa_supplicant/wpa_supplicant.conf` dans la partition *rootfs* ➠ `$ sudo vim /media/tv/boot/wpa_supplicant.conf`

wpa_supplicant.conf

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=FR

network={
  ssid="XXXX"
  scan_ssid=1
  psk="very secret passphrase"
  key_mgmt=WPA-PSK
}
```

Installation *headless* (Raspberry Pi Zero W) III

- configurer une adresse IP statique (facultatif) :
  `$ sudo vim /media/tv/rootfs/etc/dhcpcd.conf`

/etc/dhcpcd.conf

```
# l'interface WiFi
interface wlan0

static ip_address=192.168.52.254/24
static routers=192.168.52.1
static domain_name_servers=192.168.52.1
```

Mode de démarrage (*boot*) sans carte SD

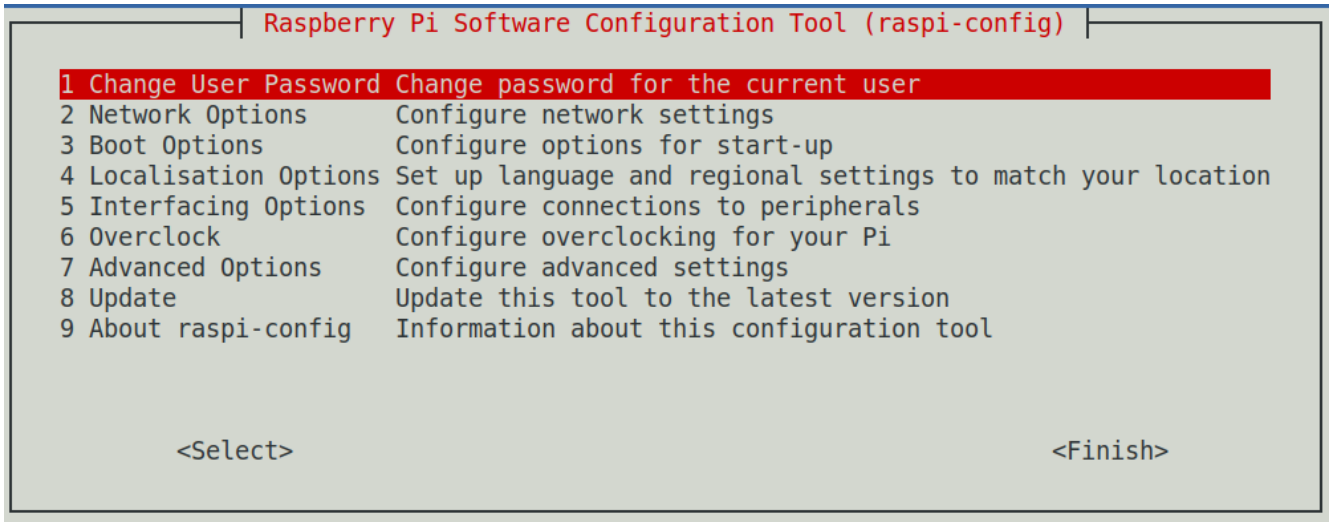
Il existe deux modes de démarrage sans carte SD :

- à partir d'un périphérique de stockage de masse USB (lecteur flash ou un disque dur USB) ➟ www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/msd.md
- à partir du réseau (interface sans-fil non prise en charge) ➟ www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/net.md et www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/net_tutorial.md

github.com/raspberrypi/documentation/blob/master/hardware/raspberrypi/bootmodes/bootflow.md

Configuration post-installation I

- Lorsque vous allumez votre Raspberry Pi pour la première fois avec sa nouvelle image Raspbian sur la carte SD, vous voudrez probablement configurer certains paramètres du système. Il faudra utiliser l'outil **raspi-config**.



✍ www.raspberrypi.org/documentation/configuration/

Configuration post-installation II

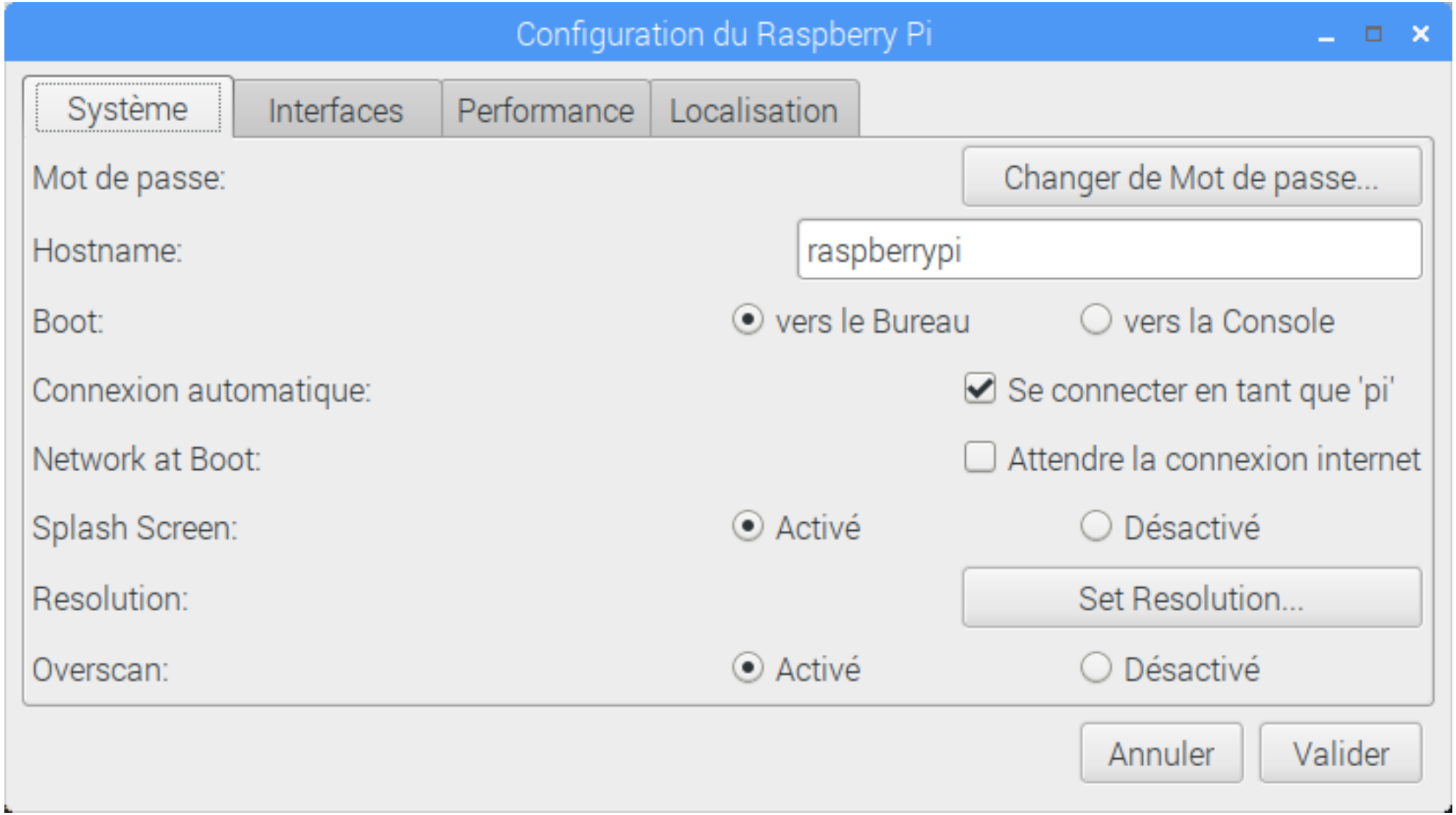
- Il existe aussi un outil graphique accessible à travers le menu Menu → « Preferences » (Préférences) → « Raspberry Pi Configuration » (Configuration du Raspberry Pi).
- Le système de carte SD peut démarrer directement en mode GUI (*Graphical User Interface*). Si vous avez démarré en mode CLI (*Command Line Interface*), vous pouvez probablement lancer l'interface graphique en exécutant : `$ startx`
- Vous allez certainement installer des logiciels. Il faudra tout d'abord resynchroniser la liste des paquets (*packages*) disponibles avec la commande : `$ sudo apt-get update`. Ensuite, vous pouvez mettez à jour tous vos paquets déjà installés vers leurs dernières versions avec la commande : `$ sudo apt-get dist-upgrade`

Configuration post-installation III

- Les paquets téléchargés (fichiers `.deb`) sont conservés dans `/var/cache/apt/archives`. Vous pouvez les supprimer afin de libérer de l'espace avec la commande : `$ sudo apt-get clean`
- Il est conseillé de changer les mots de passe des comptes existants avec la commande `passwd` ou avec `raspi-config`.

✍ Dans les options avancés (*Advanced Options*), vous pouvez régler notamment : l'extension de la partition principale à la taille maximale de la carte SD (*Expand Filesystem*) et définir la quantité de mémoire vive dédiée au GPU (*Memory Split*).

raspi-config



Informations I

- Sur le système :

```
$ uname -a
$ cat /proc/version
$ cat /etc/issue
$ cat /etc/debian_version
$ cat /etc/os-release
$ vcgencmd version # firmware
```

- Sur le processeur :

```
$ vcgencmd get_config int
$ cat /proc/cpuinfo
# cf. www.raspberrypi.org/documentation/hardware/raspberrypi/revision-codes/
```

Informations II

- Sur les processus :

```
$ ps aeux
$ top

$ lxtask # gui
```

- Sur la mémoire :

```
$ free -h
$ cat /proc/meminfo

$ swapon -s
$ cat /proc/swaps
$ vmstat
```

Informations III

- Sur le(s) système(s) de fichiers :

```
$ lsblk
$ sudo fdisk -l
$ df -hT
$ cat /proc/partitions
$ cat /proc/mounts
$ mount
$ sudo du -sh /
```

Informations IV

- Divers :

```
$ ls -l /boot/
$ cat /boot/config.txt
$ cat /boot/cmdline.txt
$ lsusb
$ hostname
$ hostname -I
$ ip address
$ ifconfig
$ iwconfig
$ cat /etc/network/interfaces
$ hciconfig
```

Informations V

- Sur les GPIO :

```
$ ls /sys/class/gpio/
$ ls -l /dev/gpio*
$ dpkg -l | grep -i gpio
$ pinout
$ raspi-gpio get
$ gpio readall # WiringPi
```

Sauvegarde de la carte SD (*backup*)

Il sera peut-être nécessaire de créer une image de la carte SD une fois installée, configurée et personnalisée.

- Windows : **Raw HDD Copy Tool** (hddguru.com/software/HDD-Raw-Copy-Tool/)
- Mac : **PiCloner** (sourceforge.net/projects/picloner/)
- GNU/Linux et Mac : la commande **dd**

```
$ dd bs=4M if=/dev/sdX of=my-image-sd.img
```

✍ Réduire l'image à une taille optimale ➠ softwarebakery.com//shrinking-images-on-linux

Sommaire

1 Présentation

2 Distribution Raspbian

3 GPIO

4 Administration

5 Réseau

6 Bluetooth

7 Mode *Kiosk*

3 GPIO

- Présentation
- Brochage et usage
- Tests (sysfs, gpiod, WiringPi)
- HAT

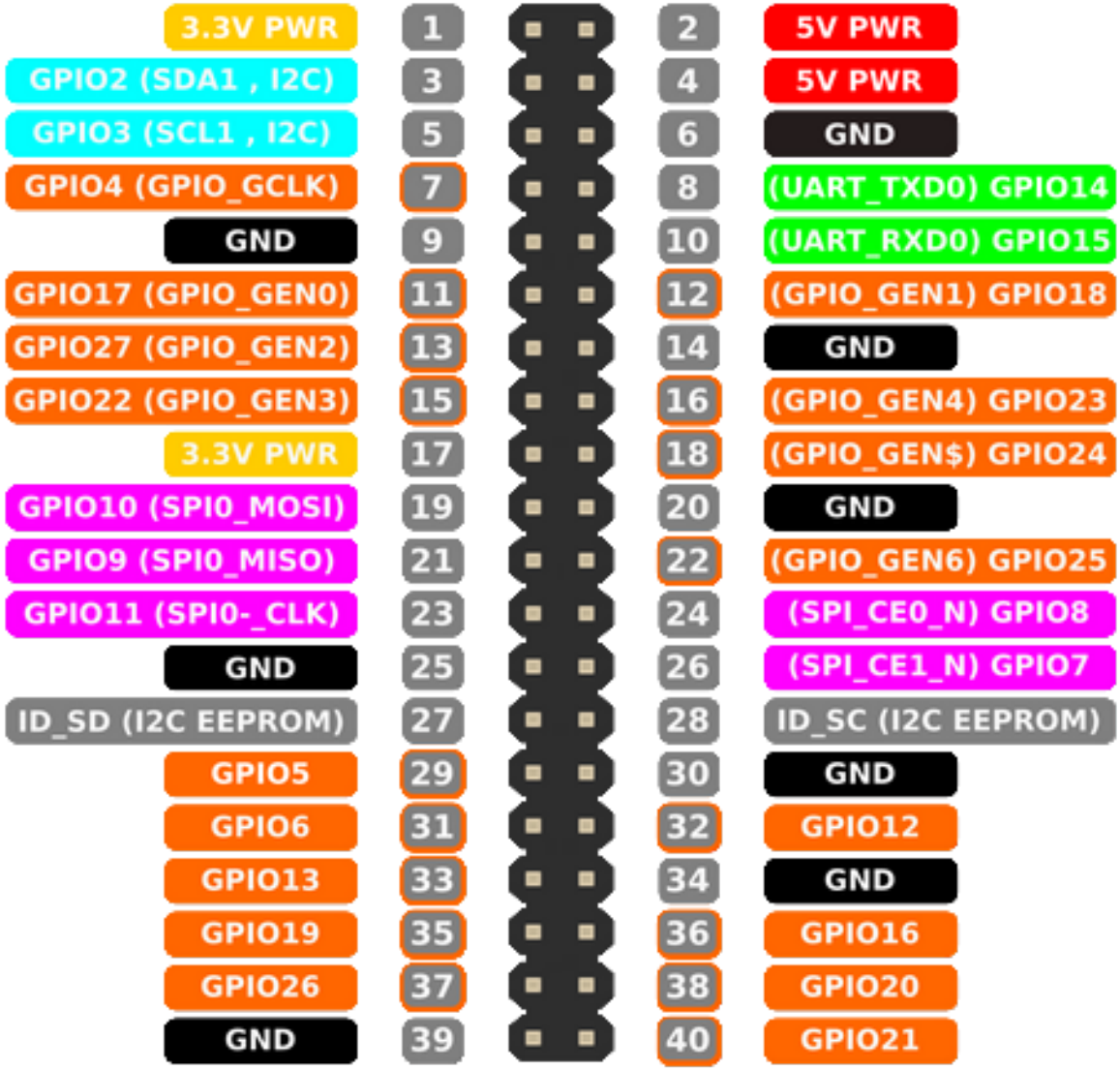
GPIO

GPIO (*General Purpose Input/Output*) est un port d'entrées-sorties très utilisés dans le monde des microcontrôleurs et de l'électronique embarquée.

- Les GPIO sont gérés par les pilotes du noyau du système d'exploitation. **Il n'y a pas d'entrée/sortie analogique.**
- Linux reconnaît nativement les ports GPIO, une documentation complète est même disponible : `www.kernel.org/doc/Documentation/gpio/gpio.txt`
- Les GPIO sont accessibles depuis leur *export* dans `/sys/class/gpio/` via `sysfs` (obsolète)
- Depuis la version 4.8 du noyau Linux, les GPIO sont accesibles par le pilote de périphérique ABI (*Application Binary Interface*) *chardev GPIO* via `/dev/gpiochipN` ou `/sys/bus/gpio`

✍ `www.raspberrypi.org/documentation/usage/gpio/`

Brochage



http://pinout.xyz/

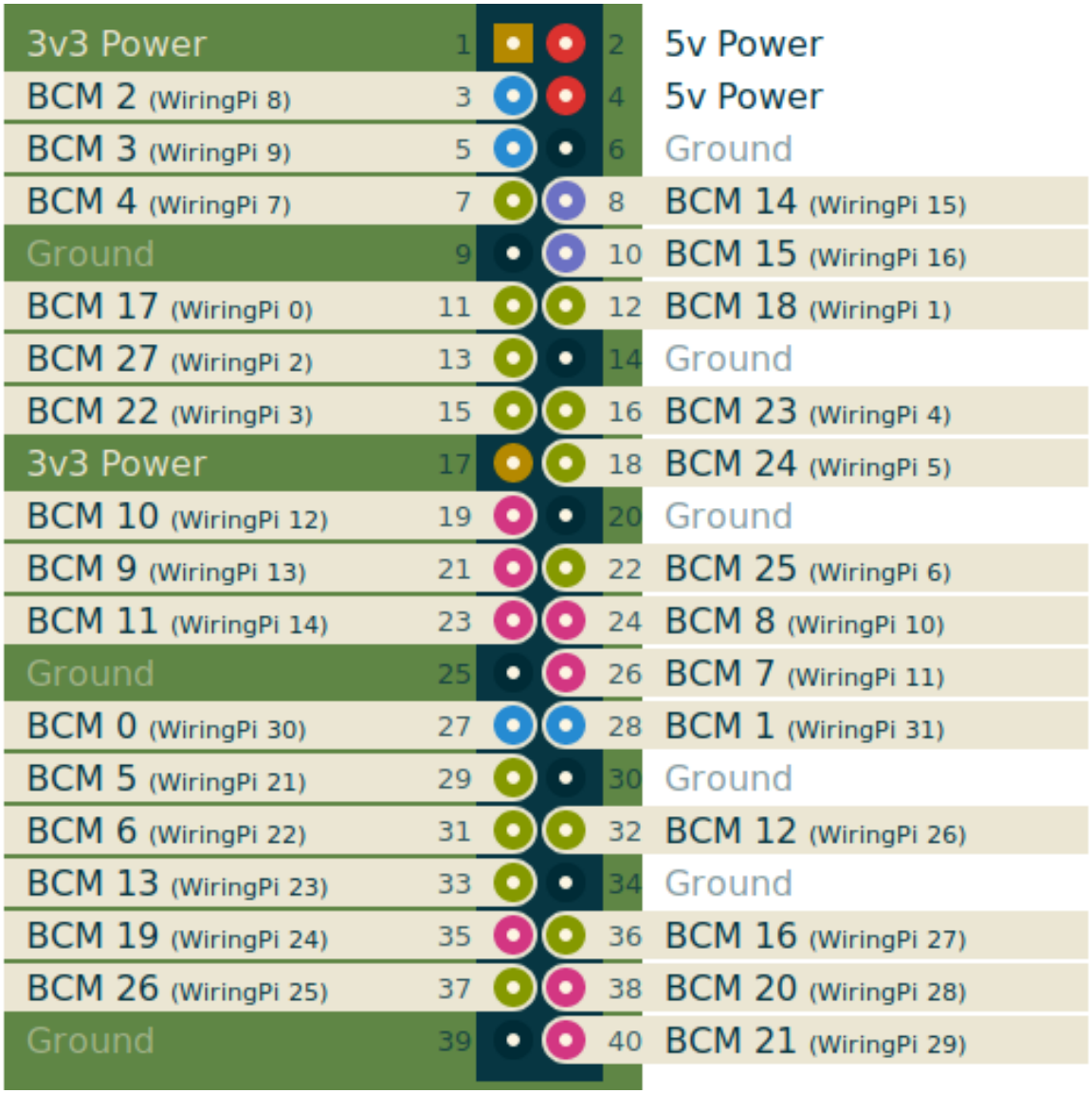
Usage

- Une sortie peut être fixée sur un niveau haut (3V3) ou bas (0V).
- Une entrée peut être lue comme un niveau haut (3V3) ou bas (0V). Il est possible d'utiliser de résistances internes *pull-up* ou *pull-down*. Les GPIO2 et GPIO3 ont des résistances de *pull-up* fixes, mais pour les autres broches, elles peuvent être configurées logiciellement.
- *Software* PWM (*Pulse-Width Modulation*) disponible sur toutes les broches et *Hardware* PWM seulement sur GPIO12, GPIO13, GPIO18, GPIO19
- I2C : SDA (GPIO2) SCL (GPIO3) et EEPROM *Data* (GPIO0) EEPROM *Clock* (GPIO1)
- Port série (UART) : TX (GPIO14) RX (GPIO15)
- SPI :
  - SPI0 : MOSI (GPIO10) MISO (GPIO9) SCLK (GPIO11) CE0 (GPIO8) CE1 (GPIO7)
  - SPI1 : MOSI (GPIO20) MISO (GPIO19) SCLK (GPIO21) CE0 (GPIO18) CE1 (GPIO17) CE2 (GPIO16)

Bibliothèques GPIO

- En plus de la bibliothèque standard du noyau, il existe des bibliothèques spécialisées de plus haut niveau, facilitant la programmation des GPIO.
  - En C/C++/*shell* : `libgpiod` qui fournit une bibliothèque C et des outils pour interagir avec le périphérique de caractères GPIO.
  - En C/C++/*shell* : **WiringPi** ➟ `wiringpi.com` comprend un utilitaire CLI `gpio`, qui peut être utilisé pour programmer, configurer les broches GPIO et l'utiliser dans des scripts *shell*.
  - En Python : **RPi.GPIO** ➟ `pypi.org/project/RPi.GPIO/`

Numérotation



http://pinout.xyz/

/sys/class/gpio/ via sysfs

➠ www.kernel.org/doc/Documentation/gpio/sysfs.txt

Lire une entrée

```
$ echo 23 > /sys/class/gpio/export
$ echo in > /sys/class/gpio/gpio23/direction
$ cat /sys/class/gpio/gpio23/value # puis la relier à la broche 1 (3V3)
```

Commander une sortie

```
$ echo 17 > /sys/class/gpio/export
$ echo out > /sys/class/gpio/gpio17/direction
$ echo 1 > /sys/class/gpio/gpio17/value # puis la relier à la broche 11 (GPIO
    23)
$ echo 17 > /sys/class/gpio/unexport
$ echo 23 > /sys/class/gpio/unexport
```

L'accès aux GPIO via l'interface sysfs est considéré comme obsolète depuis la version 4.8 du noyau Linux.

chardev GPIO I

Le paquet **gpiod** fournit six outils permettant de manipuler les GPIO :

- **gpiodetect** : liste tous les *gpiochips* présents sur le système, leurs noms, étiquettes et nombre de lignes GPIO
- **gpioinfo** : liste toutes les lignes des *gpiochips* spécifiés, leurs noms, direction, état actif ...
- **gpioget** : lit les valeurs des lignes GPIO spécifiées
- **gpioset** : définit les valeurs des lignes GPIO spécifiées
- **gpiofind** : recherche une ligne GPIO
- **gpiomon** : attend les événements sur les lignes GPIO

chardev GPIO II

Tests gpiod

```
$ sudo apt-get install gpiod
$ dpkg -L gpiod

$ gpiodetect

$ gpioinfo gpiochip0

$ gpioset --help
$ gpioset -m time -s 5 -b gpiochip0 17=1
$ gpioget gpiochip0 23
```

➠ www.blaess.fr/christophe/2018/10/15/pilotage-de-gpio-avec-lapi-libgpiod-partie-1/

WiringPi I

➟ wiringpi.com/download-and-install/

Installation

```
$ sudo apt-get install git-core
$ cd
$ git clone git://git.drogon.net/wiringPi
$ cd ~/wiringPi
$ git pull origin
$ cd ~/wiringPi
$ ./build

# Tests
$ gpio -v
$ gpio readall
```

WiringPi II

Commander une sortie

```
# WiringPi pin 0
$ gpio mode 0 out
$ gpio write 0 1
# BCM_GPIO pin 17
$ gpio -g mode 17 out
$ gpio -g write 17 1
# Physical P1 pin 11
$ gpio -1 mode 11 out
$ gpio -1 write 11 1
# Lecture
$ gpio -g mode 23 in
$ gpio -g read 23
$ gpio readall
```

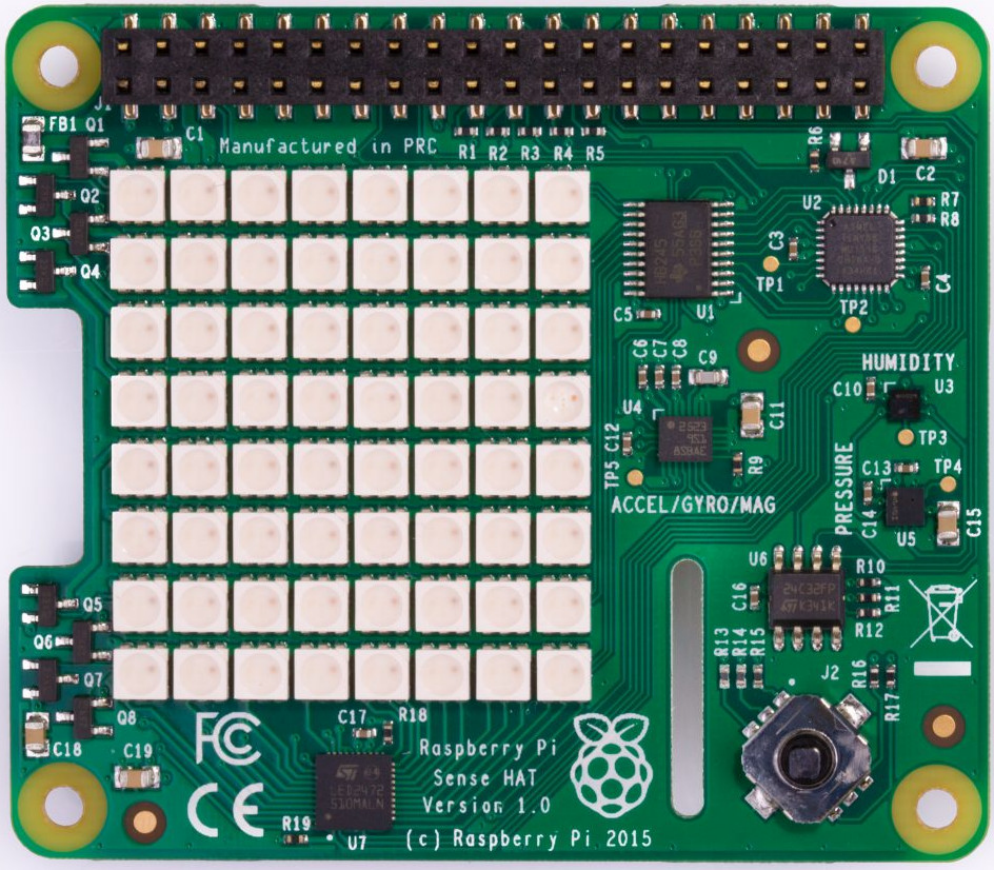
HAT I

Depuis la Raspberry Pi B+, de nombreuses cartes additionnelles appelées HAT (*Hardware Attached on Top*) ont été concues.

- Un HAT est une carte complémentaire qui se conforme à un ensemble de règles spécifiques qui permettront d'identifier le HAT connecté et de configurer automatiquement les GPIO et les pilotes de la carte (une sorte de *plug-and-play*).
- Cela sera réalisé par une mémoire EEPROM I2C sur le HAT, permettant de renseigner le nom du fabricant, le type de carte, un numéro de série unique et de préconfigurer les broches GPIO.
- Spécifications d'un HAT : `github.com/raspberrypi/hats`
- Liste de *Raspberry Pi HATs, pHATs & Add-ons* : `pinout.xyz/boards`

HAT II

Exemple : le *Sense HAT* est équipé d'une matrice LED 8x8, accéléromètre, gyroscope, magnétomètre, capteurs de pression d'air, de température, d'humidité et d'un petit joystick.



Sommaire

1 Présentation

2 Distribution Raspbian

3 GPIO

4 Administration

5 Réseau

6 Bluetooth

7 Mode *Kiosk*

4 Administration
- Noyau Linux
- Gestion des logiciels
- Journalisation
- Démarrage du système
- Utilitaires d'audit
- *Swap*
- *ramfs* et *tmpfs*
- Divers

Le noyau (*kernel*) Linux

- Le noyau (*kernel*) Linux est un noyau de système d'exploitation de type UNIX. Le noyau Linux est un logiciel libre développé essentiellement en langage C par des milliers de bénévoles et salariés via Internet.
- Le noyau est le cœur du système, c'est lui qui s'occupe de **fournir aux logiciels une interface pour utiliser le matériel**. Le noyau Linux a été créé en 1991 par **Linus Torvalds** pour les compatibles PC construits sur l'architecture processeur x86 (INTEL PC). Depuis, il a été porté sur nombre d'architectures dont m68k, PowerPC, ARM, ...
- Ses caractéristiques principales sont d'être multitâche et multi-utilisateur tout en respectant les normes **POSIX** (UNIX).
- Au départ, le noyau a été conçu pour être monolithique. Depuis sa version 2.0, le noyau est modulaire, c'est-à-dire que certaines fonctionnalités peuvent être ajoutées ou enlevées du noyau à la volée.

Configuration du noyau I

- Le code source du noyau Linux est disponible sur le site `kernel.org` et `github.com/raspberrypi/linux` pour la Raspberry Pi, mais les distributions GNU/Linux fournissent également des sources empaquetées sur leurs dépôts.
- L'étape la plus importante de la compilation d'un noyau personnalisée est sa configuration. Les options de configuration sont déclarées dans le fichier `.config`, chacun correspond à une fonctionnalité du noyau, qu'on décide d'utiliser ou non. Trois choix sont généralement possibles :
  - Y : la fonctionnalité est compilée et implantée dans l'image du noyau
  - M : la fonctionnalité est compilée comme module
  - N : la fonctionnalité est ignorée

Configuration du noyau II

- Il existe plusieurs cibles (`config`, `menuconfig`, `gconfig` ou `xconfig` pour un mode GUI) pour régler la configuration avec la commande **`make`** : `$ make config` par exemple.
- La compilation du noyau et des modules se fait par la commande **`make`**. Cette opération peut être assez longue. L'installation est automatisée, les commandes **`make install`** et **`make modules_install`** permettent respectivement d'installer l'image du noyau et ses modules.
- Sur les systèmes Linux, `vmlinux` (ou `vmlinuz` pour sa forme compressé) est le nom du fichier exécutable qui contient le noyau Linux dans l'un des formats de fichiers objet supporté par Linux. Pour la Raspberry Pi, le fichier se nomme **`kernel.img`** (format `zImage`).

Configuration du noyau III

- Le SoC de la Raspberry Pi 3 est un Broadcom BCM2837 pouvant fonctionner en 64 bits. Néanmoins les distributions actuelles assurent une compatibilité avec la Raspberry Pi 2 en fonctionnant en 32 bits.
- Pour les Raspberry Pi 2 et 3, le *bootloader* charge par défaut le noyau `kernel7.img` alors qu'il charge le noyau `kernel.img` au format ARMv6 pour les modèles précédents. C'est l'option `kernel` du fichier `config.txt` qui permettra de choisir le noyau a chargé au démarrage.

Les modules chargeables

- La modularité du noyau Linux est assurée par des modules chargeables ou LKM (*Loadable Kernel Module*).
- Un module chargeable du noyau est un fichier objet qui contient le code qui permet d'étendre les fonctionnalités du noyau d'un système d'exploitation en cours d'exécution. Ces fichiers portent maintenant l'extension `.ko` (*kernel objet*). Ils sont stockés dans `/lib/modules/`.
- Ils sont généralement utilisés pour ajouter le support d'un nouveau matériel (pilote de périphérique ou *driver*), la prise en charge d'autres systèmes de fichiers (FAT, NTFS, ...), ou pour ajouter tout simplement des fonctionnalités systèmes (`ip6`, *firewall*, ...).

Les commandes dédiées aux modules

Il existe des commandes dédiées à la gestion des modules chargeables :

- `lsmod` affiche l'état des modules chargés actuellement dans le noyau
- `insmod` charge un module dans le noyau
- `rmmod` décharge un module dans le noyau
- `modprobe` charge et décharge un module dans le noyau
- `modinfo` affiche des informations sur un module chargeable

✍ La commande `dmesg` est aussi utile pour afficher le tampon des messages du noyau.

Device Tree I

Les derniers noyaux de Raspberry Pi utilisent désormais *Device Tree*, soit une Arborscence Matérielle (*DT* = *Device Tree*), pour gérer les attributions de ressources et le chargement de modules par défaut.

- Le *Device Tree* est une description du matériel d'un système. Il représente donc la configuration matérielle sous la forme d'une hiérarchie de nœuds (arborescence).
- Par défaut, aucun matériel n'est activé. Pour utiliser des interfaces externes (I2C, SPI, ...), il faudra ajouter de nouveaux paramètres au fichier `config.txt`.
- Le *Device Tree* est décrit par des fichiers texte (syntaxe de type C), appelés DTS (*Device Tree Source*) et stockées dans des fichiers d'extension `.dts`.

Device Tree II

- Le format binaire compilé de ces fichiers est appelé DTB (*Device Tree Blob*) avec l'extension `.dtb`.
- Les *Device Tree* de base sont stockés dans la partition FAT (/boot sous Linux) à côté du chargeur `start.elf` et sont nommés `bcm2708-rpi-*.dtb` (où l'étoile est remplacée par le modèle `2-b`, `3-b`, ...), `bcm2709-rpi-*.dtb` et `bcm2710-rpi-*.dtb`. La sélection est automatique et permet à la même image de carte SD d'être utilisée sur divers modèles.
- Avec le *Device Tree*, le noyau recherchera et chargera automatiquement les modules prenant en charge les périphériques à activer.
- Il est possible de décrire des composants optionnels en utilisant des « overlays » qui s'ajouteront au *Device Tree* de base.

Device Tree III

- On pourra aussi paramétrer de petits changements à la configuration d'un périphérique avec `dtparam` et `dtoverlay` dans le fichier `config.txt`.

✍ `www.raspberrypi.org/documentation/configuration/device-tree.md` et sa version traduite `www.framboise314.fr/un-point-sur-le-device-tree/`

Les paquetages (packages)

- Une majorité de logiciels sont fournis sous forme de **paquetages (*packages*)** dans l'environnement GNU/Linux.
- Un paquet (*package*) contient :
  - des fichiers décrivant le *package* (description, version, signature, dépendances, ...)
  - les fichiers à installer
  - des scripts qui s'exécutent avant ou après l'installation ou la suppression

✍ `.deb` pour les paquets DEBIAN

Les gestionnaires de paquets DEBIAN

- **dpkg** est un outil pour l'installation, la création, la suppression et la gestion des paquets Debian.
- **aptitude** est la principale interface CLI à **dpkg**.
- **APT** est un système de gestion de paquets logiciels.
- **synaptic** est la principale interface GUI à **APT**.

✍ Les sources pour les paquets sont énumérées dans le fichier `/etc/apt/sources.list`.

Informations sur les paquets

```
$ dpkg -l # liste les paquets
$ dpkg --get-selections # liste les paquets installés
$ dpkg -L <paquet> # liste les fichiers d'un paquet installé
$ dpkg -S <fichier> # recherche le <fichier> dans les paquets installés
$ dpkg -p <paquet> # affiche les informations sur un paquet
$ apt-cache show <paquet> # affiche les informations sur un paquet
$ dpkg-query -s <nom> # affiche les informations sur un paquet installé
$ apt-cache search <mot> # recherche <mot> (RE) dans les noms et descriptions

$ apt-file search|list <pattern> # utilitaire pour rechercher dans les paquets
```

Gestion des paquets

Quelques commandes :

```
# resynchroniser l'index répertoriant les paquets disponibles et sa source
$ sudo apt-get update

# installer les versions les plus récentes de tous les paquets
$ sudo apt-get upgrade|dist-upgrade

# installer un paquet
$ sudo dpkg -i <paquet>
$ sudo apt-get install <paquet>

# réconfigurer un paquet
$ sudo dpkg-reconfigure <paquet>

# réinstaller un paquet
$ sudo apt-get --reinstall <paquet>

# désinstaller un paquet
$ sudo apt-get remove|purge <paquet>
$ sudo apt-get remove --purge <paquet> # avec les fichiers de configuration
```

syslog/rsyslog

Syslog a été la solution de journalisation standard sur les systèmes Unix et Linux. Les journaux sont stockés dans `/var/log/`.

- `syslog` est à la fois un **protocole** définissant un service de journalisation et un **logiciel** responsable de la prise en charge des fichiers de journalisation du système (`/var/log/syslog`). Sa configuration est réalisée dans le fichier `/etc/syslog.conf`.
- `rsyslog` est un logiciel libre utilisé sur des systèmes d'exploitation Unix et Linux pour remplacer `syslog`. Sa configuration est réalisée dans le fichier `/etc/rsyslog.conf`.
- La journalisation dans `syslog/rsyslog` (local ou distant) se fait via la commande `logger`. Par défaut, les messages sont enregistrés dans le fichier `/var/log/messages`.

```
$ logger -p syslog.notice -t "test" -i "mon message"
```

journalctl

systemd possède son propre mécanisme de journalisation, **syslog** n'est plus requis par défaut. La commande **journalctl** permet d'accéder au *log* (journal).

- Tout le log : **journalctl** (l'option `-f` pour un affichage continu comme pour `tail`)
- Par service : **journalctl -u ssh**
- Par PID : **journalctl _PID=1**
- Par exécutable : **journalctl /usr/sbin/ntpd**
- Par jour : **journalctl -since="today"**
- Par niveau : **journalctl -p err**
- Gestion de l'espace : **journalctl -disk-usage**
- Suppression : **sudo journalctl -vacuum-size=10M**

✍ La configuration du journal de **systemd** est réalisée avec le fichier **/etc/systemd/journald.conf**.

systemd

- À l'origine les systèmes Unix/Linux utilisaient le processus **init** (**sysvinit**) pour assurer le démarrage de l'OS. Certains systèmes ont utilisé ensuite **Upstart**. La majorité utilise maintenant **systemd**.

```
$ ps -p1 | grep systemd && echo systemd || echo upstart
```

- **systemd** est un système d'initialisation du système qui active et maintient les **services** pour le noyau Linux.
- **systemd** introduit la notion d'**unité**. Une unité représente un fichier de configuration. Une unité peut être un service (`*.service`), un *target* (`*.target`), un montage (`*.mount`), un *socket* (`*.socket`), ...
- L'outil de gestion des services (et des autres unités d'ailleurs) dans **systemd** s'appelle **systemctl**.

systemctl I

```
# liste les unités
$ sudo systemctl list-units

# liste les services
$ sudo systemctl list-units --type=service
$ sudo systemctl --state=running

# contrôle d'un service
$ sudo systemctl start|stop|relaod|status|enable|disable <service>

# visualise le contenu d'un service
$ sudo systemctl cat <service>

# une GUI pour systemd
$ sudo apt-get install systemd-ui
$ sudo systemadm
```

runlevel

- Dans **systemd**, la notion de *runlevel* est remplacé par le concept de *target*.

Runlevel	*Target*
0	poweroff.target
1	rescue.target
2, 3, 4	multi-user.target
5	graphical.target
6	reboot.target

- Pour afficher le *runlevel* : `$ runlevel`
- Pour changer de *runlevel* : `$ sudo systemctl isolate multi-user.target`
- Pour activer un *runlevel* : `$ sudo systemctl enable multi-user.target`
- Pour mettre un *runlevel* par défaut :

  `$ sudo systemctl set-default multi-user.target`

Création et gestion d'un service systemd I

Les services gérés par **systemd** sont stockés dans `/etc/systemd/system/` (configuration locale) et `/lib/systemd/system/` (unités installées).

Un simple service

```
$ sudo vim /etc/systemd/system/hello.service
[Unit]
Description=bla bla ...
After=multi-user.target

[Service]
Type=simple
ExecStart=/bin/sh -ec "exec echo \"Bonjour Maître\" > /tmp/bonjour"

[Install]
WantedBy=default.target
```

Création et gestion d'un service systemd II

```
# Recharger systemctl
$ sudo systemctl daemon-reload

# Démarrer le service
$ sudo systemctl start hello.service

# Consulter l'état
$ sudo systemctl status hello.service

# Consulter le journal (log)
$ sudo journalctl -u hello.service

# Visualiser les propriétés assoiées à un service
$ sudo systemctl show hello.service

# Arrêter le service
$ sudo systemctl stop hello.service

# Activer le service (puis redémarrer)
$ sudo systemctl enable hello.service
```

Paramètres systèmes

- La commande **vcgencmd** permet d'accéder aux paramètres du *firmware*. Usage : **elinux.org/RPI_vcgencmd_usage**
- Les paramètres du *firmware* peuvent être définis au démarrage dans le fichier **confif.txt**.
- La commande **sysctl** permet d'accéder aux paramètres du système d'exploitation Linux. Usage : **man sysctl**
- Les paramètres du système d'exploitation Linux peuvent être définis au démarrage dans le fichier **/etc/sysctl.conf**.

Les fichiers d'initialisation I

- Un fichier d'initialisation contient une série de commandes qui sont exécutées lorsque le *shell* démarre. Il permet de personnaliser l'environnement de travail.
- Il existe deux sortes de fichiers d'initialisation : système et utilisateur.
- Les fichiers d'initialisation système sont gérés par l'administrateur (*root*), se trouvent dans le répertoire `/etc` et sont communs à tous les utilisateurs.
- Les fichiers d'initialisation utilisateur se trouvent à la racine du répertoire personnel (`$HOME`) de chaque utilisateur.
- On distingue deux types de fichiers d'initialisation : les fichiers exécutés à la connexion (*login*) seulement et les fichiers exécutés à chaque lancement de *shell*.

Les fichiers d'initialisation II

File	Description
/etc/profile	The system-wide initialization file; executed when you log in.
/etc/bashrc	Another system-wide initialization file; may be executed by a user's *.bashrc* for each **bash** shell launched.
~/.bash_profile	If this file exists, it is executed automatically after */etc/profile* when you log in.
~/.bash_login	If *.bash_profile* doesn't exist, this file is executed automatically when you log in.
~/.profile	If neither *.bash_profile* nor *.bash_login* exists, this file is executed automatically when you log in.
~/.bashrc	This file is executed automatically when **bash** starts.
~/.bash_logout	This file is executed automatically when you log out.
~/.inputrc	This file contains optional key bindings and variables that affect how **bash** responds to your keystrokes.

- Le répertoire `/etc/skel` contient des modèles de fichiers d'initialisation pour les utilisateurs. Le contenu de ce répertoire est copié dans le répertoire de l'utilisateur lors de sa création à l'aide de la commande **useradd**, si l'option `-m` est utilisée.

Des outils d'audit

- L'utilitaire **`who`** peut fournir des informations sur les utilisateurs connectés et la commande **`w`** affiche les utilisateurs connectés et ce qu'ils font.
- La commande **`last`** peut être utilisée pour déterminer toutes les connexions/déconnexions d'un utilisateur.
- La commande **`uptime`** nous renseigne depuis quand le système fonctionne et la charge de celui- ci.
- La commande **`vmstat`** permet de suivre en temps réel l'activité de la mémoire virtuelle, des zones de swap, des processus, ...
- La commande **`top`** permet de suivre en temps réel l'activité des processus.
- La commande **`netstat`** affiche les informations du sous-système réseau.
- La commande **`fuser`** identifie les processus qui utilisent des fichiers et la commande **`lsof`** liste les fichiers ouverts.
- Voir aussi : `ps`, `pstree`, `pgrep`, `free`, `top`, `df`, `du`, `slabtop`, ...

Zone de swap

La zone de *swap* est utilisée lorsque la mémoire physique (RAM) est remplie. Si le système a besoin de plus de ressources mémoires et que la mémoire physique est remplie, les pages de mémoire (bloc de taille fixe) inactives (non utilisées depuis un certain temps) sont déplacées dans la zone de *swap*.
Quelques règles de base pour la zone de *swap* :

- Elle peut être stockée dans une partition dédiée ou un fichier
- Le paramètre `swappiness` (une valeur entre 0 et 100) contrôle la tendance du noyau à déplacer les processus de la mémoire physique vers le *swap*. La valeur par défaut est 60. Une valeur faible fait que le noyau évitera de permuter (0 désactivant le *swap*). Son réglage se fait dans le fichier `/etc/sysctl.conf` : `vm.swappiness = 10`
- Avec une Raspbian, un fichier de *swap* `/var/swap` de 100M est utilisé par défaut.

Configuration

- Les zones de *swap* en cours d'utilisation par le système peuvent être listées avec la commande **`swapon -s`** (partition ou fichier)
- La création d'une partition sur un disque existant ou un nouveau disque peut se faire avec les outils **`parted`**, **`fdisk`** ou autre. Pour un fichier, on utilisera la commande **`dd`**.
- Il faut ensuite créer les structures de la zone de *swap* avec la commande **`mkswap`** et l'activer avec **`swapon`**
- L'arrêt de l'utilisation de la zone de *swap* se fera avec la commande **`swapoff`** (partition ou fichier).
- Pour les partitions de *swap* (seulement), il faut renseigner leur montage dans le fichier **`/etc/fstab`**.

Fichier de swap

- Les fichiers de *swap* sont contrôlés par la commande `dphys-swapfile setup|install|swapon|swapoff|uninstall` ou `swapon/swapoff` (partition ou fichier)
- La configuration des fichiers de *swap* est réalisée dans le fichier `/etc/dphys-swapfile` :
```
$ cat /etc/dphys-swapfile | sed '/^$/d;/^#/d'
CONF_SWAPSIZE=100
```
- Contrôle du service de fichiers de *swap* :
```
$ sudo systemctl start|stop|status|enable|disable
dphys-swapfile.service
```
- Statistiques et informations : `free`, `vmstat`, `top`, ...

✍ Suivant l'usage de son système, on peut envisager de désactiver ou déplacer sur un disque USB externe le *swap* sur une Raspberry Pi.

Systèmes de fichiers en mémoire : *ramfs* et *tmpfs*

Les systèmes de fichiers en mémoire sont parfois nommés pseudo systèmes de fichiers. Ils ne résident pas sur un disque physique mais en mémoire. Il en existe deux sortes :

- ceux qui permettent de créer un disque en mémoire : **ramfs**.
- ceux qui permettent d'accéder à des informations du système par l'intermédiaire de fichiers : **tmpfs** (il utilise la mémoire partagée du système).

✍ Les répertoires `/tmp`, `/var/log`, `/var/tmp` et `/var/run` sont de « bons candidats » pour `tmpfs`. C'est déjà le cas pour `/var/run` (cf. `/run`). Il suffit d'ajouter une ligne dans `/etc/fstab` :

```
tmpfs /tmp tmpfs defaults,noatime,nosuid,size=100m 0 0
```

ramdisk

- Un *ramdisk* est une zone de mémoire utilisée comme partition.
- Une fois montée, cette partition est utilisable comme n'importe quelle partie du système de fichiers.
- Pour créer un *ramdisk*, rien de plus simple : il suffit de le monter avec la commande `mount` et l'option `-t ramfs` :

```
$ sudo mkdir /ramdisque
$ sudo mount -t ramfs -o maxsize=10M none /ramdisque
```

- On peut aussi faire en sorte qu'il soit créé au démarrage en rajoutant une ligne au fichier `/etc/fstab`.

La température interne

```
$ vcgencmd measure_temp
temp=44.0'C

$ cat /sys/class/thermal/thermal_zone0/temp
44008

$ let TEMP=$(cat /sys/class/thermal/thermal_zone0/temp)/1000
$ echo $TEMP
44

# installe une calculatrice
$ sudo apt-get install bc

$ echo "$(cat /sys/class/thermal/thermal_zone0/temp)/1000" | bc -lq
44.008

$ echo "scale=2;$(cat /sys/class/thermal/thermal_zone0/temp)/1000" | bc -lq
44.00
```

UART I

Un **UART** (*Universal Asynchronous Receiver Transmitter*) est un émetteur-récepteur asynchrone universel. C'est le composant de base pour obtenir une **liaison série (port série)**.

- Une **trame UART** est constituée des bits suivants :
  - 1 bit de start (toujours à 0) servant à la synchronisation du récepteur
  - *n* bits de données : *n* compris entre 5 et 8 bits (bits envoyés du LSB (bit de poids faible) au MSB (bit de poids fort))
  - 0 ou 1 bit de parité : aucune ou paire (nombre de bits à 1 égal à un nombre pair) ou impaire (idem pour un nombre impair)
  - 1 bit de stop (toujours à 1) : durée variable entre 1, 1.5 et 2 *bit-time*
- Le débit de transmission des données est exprimé encore en **bauds**, où l'unité baud correspondant à un **bit par seconde**.

UART II

- Les paramètres sont parfois décrits sous forme condensée : « `115200 8N1` » ce qui correspond à 115200 bauds, 8 bits de données, pas de parité (N), 1 bit de stop.
- Les niveaux de tension sont de type TTL soit 0V pour le niveau logique bas et +5V (ou **3.3V pour la Raspberry Pi**) pour le niveau logique haut.
- Il est possible d'ajouter un circuit type MAX232 pour mettre en forme des signaux conformes au standard RS-232.
- Pour le relier à un PC, il faut utiliser un adaptateur USB/RS-232 car les PC ne disposent plus de port série RS-232.
- La commande `screen` (Ctrl-a k pour sortir) ou les utilitaires `picocom` (Ctrl-a Ctrl-x pour sortir), `minicom` ou `cutecom` permettent de transmettre et de recevoir des données via la liaison série.

UART de la Raspebrry Pi I

Le SoC du Raspberry Pi est le BCM2835 qui comporte deux UARTs : le premier est un « vrai » UART (le PL011) et le second un « mini » UART.

- Sur les premières générations de Raspberry Pi (model 1 B, B+ et 2), l'UART0 PL011 est connecté aux broches 8 (TX) et 10 (RX) du GPIO. Les messages de démarrage du système sont envoyés par défaut sur ce port. Une exécution de `getty` (ou `agetty`) sur ce port permet de se connecter via la liaison série. Le « mini » UART n'est pas utilisé.
- Sur la Raspberry Pi 3, l'UART0 PL011 est maintenant utilisé pour le Bluetooth et le « mini » UART est connecté aux broches 8 (TX) et 10 (RX) du GPIO. Attention, le « mini » UART ne garantit pas un débit constant car celui-ci est généré à partir de la fréquence d'horloge du processeur.

UART de la Raspebrry Pi II

- Il est possible de modifier le comportement actuel de la gestion des UARTs de la Raspberry Pi 3 à partir du fichier `config.txt`.
  Lire cet article : www.framboise314.fr/le-port-serie-du-raspberry-pi-3-pas-simple/.

Sommaire

1 Présentation

2 Distribution Raspbian

3 GPIO

4 Administration

5 Réseau

6 Bluetooth

7 Mode *Kiosk*

5 Réseau
- Protocoles
- Interfaces réseaux
- Contrôle à distance
- Services réseau

Protocoles utilisés

La plupart des systèmes d'exploitation utilisent actuellement le protocoles "**TCP/IP**" basés sur des réseaux de type **Ethernet** (du moins pour les réseaux locaux). Un protocole est un **ensemble de règles** gérant l'échange de données entre deux entités. Les protocoles interviennent à plusieurs niveaux dans une communication (cf. modèle à couches).

- Protocoles de niveau **réseau** : IP (*Internet Protocole*), ARP, RARP (*Adresse Resolution Protocol*, *Reverser ARP*), ICMP (*Internet Control Message Protocol*), ...
- Protocoles de niveau **transport** : TCP (*Transmission Control Protocol*) et UDP (*User Datagram Protocol*)
- Protocoles de niveau **application** : HTTP (*HyperText Transfert Protocol*), FTP, TFTP (*File Transfert Protocol*, *Trivial FTP*), SMTP (*Simple Mail Transfert Protocol*), NFS (Network File System), ...

Les interfaces réseaux

Sous Linux, les périphériques de type interfaces réseau sont accessibles par l'intermédiaire d'un fichier de périphérique nommé :

- `/dev/ethX` où `X` est le numéro de l'interface réseau Ethernet, attribué par le système en fonction de l'ordre de détection de la carte.
- `/dev/wlanX` où `X` est le numéro de l'interfac réseau sans-fil, attribué par le système en fonction de l'ordre de détection de la carte.

Quelques commandes utiles :

- la commande `dmesg` qui affiche les messages systèmes depuis le démarrage du système
- la commande `ifconfig` qui configure et affiche les interfaces réseaux
- la commande `iwconfig` idem pour les interfaces réseaux sans-fil
- contrôler le service réseau avec **systemd** :
  `systemctl (start|stop|restart) networking`

Notions de bases

Les paramètres réseaux les plus souvent modifiés sont :

- Les **adresses IP et masques de sous réseau** des cartes
- L'**adresse de la passerelle par défaut**
- Le **nom d'hôte** de la machine
- La **résolution des noms** de machines

Les commandes de configuration

- La commande **ifconfig** permet de stopper ou de démarrer le fonctionnement d'une interface réseau, de voir et de changer sa configuration. Voir aussi : la commande **ip**.
- La commande **route** permet de modifier la table de routage et donc de configurer l'adresse de la passerelle par défaut (la route par défaut pour tous les paquets non destinés au réseau local).
- La commande **hostname** permet de manipuler le nom d'hôte.

Les fichiers de configuration

Les paramètres de configuration du système se trouvent (dans le cas de **Debian/Ubuntu** Linux) dans le répertoire `/etc/network`.

- La configuration des interfaces réseau dans `/etc/network/interfaces`.
- Le nom d'hôte (et le nom de domaine DNS si besoin) par défaut se trouve dans le fichier `/etc/hostname`.
- La résolution des noms de machines peut utiliser deux bases de données pour convertir des adresses IP en nom de machine ou inversement : la base locale (fichier `/etc/hosts`) et la base distribuée DNS (`/etc/resolv.conf`).

Exemple de configuration

Obsolète ?

```
$ cat /etc/network/interfaces
auto lo # activation de l'interface au démarrage
iface lo inet loopback # l'interface de loopback lo

auto eth0 # activation de l'interface au dmarrage
iface eth0 inet static # adressage statique IPv4
    address 192.168.1.1
    netmask 255.255.255.0
    broadcast 192.168.1.255
    gateway 192.168.1.254
    #dns-nameservers 192.168.1.3 10.0.0.1
    #dns-search intra.net

auto eth1 # activation de l'interface au démarrage
iface eth1 inet dhcp # adressage dynamique par un serveur DHCP

allow-hotplug wlan0
iface wlan0 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Configuration WiFi

- *Wi-Fi Protected Access* (WPA et WPA2) est un mécanisme pour sécuriser les réseaux sans-fil de type Wi-Fi.

Pour des réseaux WPA-Personal (PSK) et WPA-Enterprise avec EAP-TLS

```
$ cat /etc/wpa_supplicant/wpa_supplicant.conf
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=FR

network={
  ssid="XXXX"
  scan_ssid=1
  psk="very secret passphrase"
  key_mgmt=WPA-PSK
}
```

✍ Voir aussi : `man 5 wpa_supplicant.conf`

Configuration adresse IP statique

- **dhcpcd** est un client DHCP et DHCPv6. Il est activé par défaut dans une Raspbian : **systemctl status dhcpcd.service**
- La configuration est réalisée dans **/etc/dhcpcd.conf**.

/etc/dhcpcd.conf

```
# exemple pour l'interface WiFi
interface wlan0

static ip_address=192.168.52.254/24
static routers=192.168.52.1
static domain_name_servers=192.168.52.1
```

✍ **man 5 dhcpcd.conf**

Accès à distance

L'accès à distance est une méthode qui permet, depuis un ordinateur éloigné et sans limite théorique de distance, de prendre le contrôle d'un autre ordinateur en affichant l'écran de celui-ci et en manipulant les fonctions d'un périphérique d'entrée comme un clavier.

- Deux modes d'affichage du poste distant sont actuellement possible :
  - le **mode texte** qui affiche l'interface distante en ligne de commande (CLI), souvent appelé par abus de langage « terminal distant ». Exemple : SSH.
  - le **mode graphique** (GUI) qui est appelé bureau à distance. Exemple : VNC.
- L'accès à distance permet : la télémaintenance, l'enseignement à distance, l'aide aux utilisateurs, ... et surtout l'**administration de serveurs**.

SSH

- SSH (*Secure Shell*) est à la fois un programme informatique et un protocole de communication sécurisé.
- Le protocole SSH a été conçu pour remplacer les différents protocoles non chiffrés : `rlogin` et `telnet` pour **`ssh`**, `rcp` pour **`scp`**, `ftp` pour **`sftp`**.
- Habituellement, le protocole SSH utilise le port TCP 22. Il est particulièrement utilisé pour ouvrir un *shell* sur un ordinateur distant, généralement pour administrer un serveur.
- **OpenSSH** est une suite d'outils libres utilisée sur les systèmes BSD, GNU/Linux et donc la Raspberry Pi. Vérifier son installation :

```
$ dpkg -l | grep openssh
```

✍ **SSHFS** vous permet de monter les fichiers d'une Raspberry Pi via SSH.

Serveur SSH

- Avant votre accès distant initial, il est recommandé de générer des clés publiques / privées d'hôte uniques avec la commande suivante :

```
$ sudo rm /etc/ssh/ssh_host_* && sudo dpkg-reconfigure
openssh-server
```

- Vérifier que le serveur ssh est démarré :

```
$ systemctl status ssh.service
```

- Contrôle du serveur ssh :

```
$ systemctl [start|stop|enable|disable] ssh.service
```

✍ Si au démarrage du système, vous obtenez le message "`SSH is enabled and the default password for the 'pi' user has not been changed.`", il vous faut soit modifier le mot de passe du compte **pi** ou modifier les fichiers qui produisent ce message : `/etc/profile.d/sshpwd.sh` (pour le mode CLI) et `/etc/xdg/lxsession/LXDE-pi/sshpwd.sh` (pour le mode GUI).

Connexion SSH

L'utilitaire `ssh-keyscan` enregistrera la clé publique de l'hôte dans un fichier nommé **`known_hosts`**. Le fichier **`known_hosts`** conserve l'ensemble des clés publiques associées aux hôtes (serveurs SSH) qui ont été approuvées. On peut supprimer les clés d'un hôte en particulier :
`ssh-keygen -f "/home/tv/.ssh/known_hosts" -R "192.168.1.2"`

```
$ ssh -X pi@192.168.1.2
...
$ echo $DISPLAY
localhost:10.0

# Lancement sur le poste local
$ lxterminal
# Lancement sur la Raspberry Pi
$ DISPLAY=:0.0 lxterminal
```

✍ L'option **`-X`** ou **`-Y`** (*X11 forwarding*) permet de transmettre l'interface graphique X11 et donc de pouvoir exécuter des programmes GUI.

VNC

- VNC (*Virtual Network Computing*) est un système de visualisation et de contrôle de l'environnement de bureau d'un ordinateur distant.
- Il fonctionne en mode client/serveur en utilisant le protocole RFB (*Remote Frame Buffer*) pour les communications.
- Il permet d'assurer un support technique à distance, l'administration et la maintenance de systèmes ou logiciels ne permettant que des contrôles graphiques et demandant l'utilisation de la souris.
- La Raspberry Pi embarque le serveur RealVNC (`https://www.realvnc.com/fr/`) par défaut. Il faut néanmoins l'activer.

✍ Le protocole VNC n'est pas sécurisé (les données transitent en clair et la connexion ne requiert pas d'authentification). Il est cependant possible de chiffrer la transmission de données en SSL, et d'imposer la saisie d'identifiant et mot de passe. VNC peut également être utilisé à travers un tunnel chiffré via une connexion SSH ou VPN.

Serveur VNC (Raspberry Pi)

- La Raspberry Pi embarque le serveur **RealVNC** (www.realvnc.com/fr/) par défaut. Sinon exécutez la commande :

```
$ sudo apt-get install realvnc-vnc-server
```

- Il faut néanmoins l'activer. En mode CLI, exécutez `raspi-config` puis dans « *Interfacing Options* » sélectionnez « VNC » et l'activer. En mode GUI, dans Menu → Preferences → Raspberry Pi Configuration → Interfaces sélectionnez « VNC » et l'activer.
- Vous pouvez contrôler le serveur VNC avec `systemctl` :

```
$ sudo systemctl [start|stop|status|enable|disable]
vncserver-x11-serviced.service
```

- Il faut ensuite installer un client VNC sur la machine Windows, Mac ou Linux : **RealVNC Viewer** www.realvnc.com/fr/connect/download/viewer/.

✍ Il est aussi possible d'installer un autre serveur VNC sur la Raspberry Pi, comme **TightVNC** : `$ sudo apt-get install tightvncserver`. Puis utiliser un client vnc (`vncviewer` par exemple sous Linux).

Connexion vncviewer

Sommaire

1 Présentation

2 Distribution Raspbian

3 GPIO

4 Administration

5 Réseau

6 Bluetooth

7 Mode *Kiosk*

5 Réseau

- Protocoles
- Interfaces réseaux
- Contrôle à distance
- Services réseau
  - DHCP
  - DNS
  - NFS
  - Samba
  - Autres : FTP, Web

DHCP

- DHCP (*Dynamic Host Configuration Protocol*) permet d'automatiser la configuration TCP/IP des machines du réseau, quel que soit leur système d'exploitation.
- L'utilisation de DHCP simplifie l'administration système en regroupant en un seul point la configuration de tout un réseau (adresses dynamiques, semi-statiques, masque de sous-réseaux, DNS, passerelle par défaut, ...).
- Le service DHCP peut aussi servir à renvoyer la configuration des démarrages par réseau (serveur de *boot*, fichiers de démarrage réseau, ...).
- Il est nécessaire de configurer le serveur DHCP avec une adresse IP statique.

isc-dhcp-server I

- Il est nécessaire d'installer le paquet **isc-dhcp-server** car seuls les paquets clients DHCP sont installés par défaut.
- Il faut éditer le fichier de configuration **/etc/dhcp/dhcpd.conf**.

Adresses dynamiques

```
subnet 10.0.0.0 netmask 255.0.0.0
{
  option broadcast-address 10.255.255.255; # adresse de
      diffusion
  range 10.0.0.100 10.0.0.250; # plage d'adresses
      dynamiques
}
```

isc-dhcp-server II

Adresses semi-statiques

```
deny unknown-clients; # rejete les clients inconnus
subnet 10.0.0.0 netmask 255.0.0.0
{
  option broadcast-address 10.255.255.255; # adresse de
      diffusion
  host machine1
  {
    hardware ethernet 01:01:02:ae:34:c4; # adresse MAC
    fixed-address 10.0.0.2; # adresse IP fixe
  }
}
```

Options DHCP

L'utilisation de DHCP permet de fournir une configuration complète de tout un réseau (adresses dynamiques, semi-statiques, masque de sous-réseaux, DNS, passerelle par défaut, ...). Quelques options :

- `option routers 10.0.0.1;` pour indiquer la passerelle par défaut (elle sera ajoutée à la table de routage)
- `option domain-name-servers 10.0.0.1, 10.0.0.6;` pour fournir les serveurs DNS (ils seront ajoutés dans `/etc/resolv.conf`)
- `option domain-name "intra.net";` pour fournir les domaines de recherche (ils seront ajoutés dans `/etc/resolv.conf`)

dnsmasq

- **Dnsmasq** est un serveur léger qui fournit les services DNS, **DHCP**, Bootstrap Protocol et TFTP. Il faut installer le paquet **dnsmasq**.
- Gérer le service **dnsmasq** :

```
$ sudo service dnsmasq {start|stop|restart|status}
$ sudo /etc/init.d/dnsmasq {start|stop|restart|status}
$ systemctl {start|stop|restart|status} dnsmasq.service
```

- Il faut éditer le fichier de configuration **/etc/dnsmasq.conf**.

Adresses dynamiques (DHCP)

```
interface=eth0
dhcp-range=10.0.0.100,10.0.0.250,255.0.0.0,12h
dhcp-option=option:router,10.0.0.1
```

✍ wiki.debian.org/HowTo/dnsmasq

DNS

- Le service DNS (*Domain Name System*) est utilisé pour associer les adresses IP aux noms complets (**FQDN**, *Fully Qualified Domain Name*) des machines (et inversement).
- Le DNS est une base de données distribuée, chaque domaine et sous domaine (appelés **zones**) étant gérés par un serveur DNS différent (un serveur peut gérer plusieurs zones). De plus, les serveurs sont organisés entre eux de façon hiérarchique.
- Une **zone** est gérée par un et un seul serveur DNS principal, et peut être répliquée sur un ou plusieurs serveurs secondaires.
- Chaque serveur DNS contient, pour chaque zone qu'il gère, les fichiers de la base de données permettant de convertir un nom de machine en adresse IP et inversement, ainsi que les noms et adresses des autres serveurs DNS de la zone et des serveurs de mail.

Configuration DNS

- Le serveur de nom fourni habituel sous Unix/Linux est **BIND** (*Berkeley Internet Name Domain*).
- **BIND** est composé, entre autre, du démon `/usr/sbin/named` et de la commande `/usr/sbin/rndc`.
- Il lit sa configuration dans le fichier `/etc/bind/named.conf`, et stocke ses informations dans le répertoire `/var/cache/bind/`.

```
zone "xxx.mondomaine" {
    type master;
    file "/etc/bind/db.mondomaine.xxx";
};

zone "0.168.192.in-addr-arpa" {
    type master;
    file "/etc/bind/db.mondomaine.xxx.rev";
};
```

Fichiers de zone I

- Les fichiers de zones contiennent les enregistrements constituant les différents noms de machines et serveurs du domaine.
- Les différents types d'enregistrements les plus courants sont :
  - **A** : *Address*, un nom de machine associé à une adresse IP.
  - **CNAME** : *Canonical NAME*, un alias sur un nom de machine.
  - **MX** : *Mail eXchange*, noms des serveurs de mails du domaine.
  - **NS** : *Name Server*, noms des serveurs DNS.
  - **SOA** : *Start Of Authority*, informations à propos de cette zone.
  - et pour la résolution inverse **PTR** : *PoinTeR*, une adresse IP associée à un nom de machine.

Fichiers de zone II

```
$TTL 86400
@   IN  SOA ns.xxx.mondomaine. root.xxx.mondomaine. (
            12345 ; Version
            21600 ; Refresh secondaires
            3600 ; Attente après demande erronee
            604800 ; TTL max dans les caches des DNS secondaires
            86400 ; TTL min dans les caches
        )
    IN  NS  ns.xxx.mondomaine.
    IN  MX  10 mail.xxx.mondomaine.

@   IN  A   192.168.0.2
ns  IN  A   192.168.0.2
mail    IN  A   192.168.0.2
server IN  A   192.168.0.2
client IN  A   192.168.0.3
www   IN CNAME server
```

NFS

- **NFS** (*Network File System*) est un système de partage de fichiers qui utilise les protocoles TCP/IP, RPC et XDR.
- Les termes utilisés dans NFS sont :
  - **Serveur** NFS : Désigne le système qui possède physiquement les ressources (fichiers, répertoires) et les partages sur le réseau avec d'autres systèmes.
  - **Client** NFS : Désigne un système qui monte les ressources partagées sur le réseau (option `-t nfs` de la commande `mount`). Une fois montées, les ressources apparaissent comme si elles étaient locales.

✍ Malgré ses défauts d'origine, NFS reste le standard pour les partages de fichiers en réseaux hétérogènes. En effet, les autres systèmes de fichiers réseaux sont soit trop liés à un type de système d'exploitation (SMB, CIFS), soit propriétaires (NCP), soit trop lourd à mettre en oeuvre pour la plupart des réseau locaux de petites tailles (Coda).

Configuration NFS

- La configuration du service NFS, côté serveur, se limite à lister les ressources partagées et les droits de montage.
- La liste des ressources partagées peut être obtenue à l'aide de la commande **showmount**.
- Le fichier **/etc/exports** contient la liste des ressources partagées, une ligne par ressource.
- L'option **root_squash** : L'utilisateur *root* local des clients (UID 0, GID 0) est considéré comme un utilisateur anonyme par le serveur.

```
/export/home 192.168.0.0/16(rw,root_squash) admin.intra.net(rw,no_root_squash)
/usr/local machin.intra.net(ro) 192.168.0.10(rw)
```

Samba

- Le logiciel **Samba** est utilisé pour le partage de fichiers et d'imprimantes à l'aide des protocoles **SMB** et **CIFS**.
- Ces protocoles étant ceux utilisés pour les systèmes d'exploitation *Microsoft*, l'installation de Samba sur une machine équipée de Linux permet :
  - d'intégrer celle-ci dans le "réseau Microsoft" de l'entreprise
  - de prendre la place d'un serveur *Microsoft Windows*
- On configure le service Samba à l'aide du fichier `/etc/samba/smb.conf`.

✍ www.raspberrypi.org/documentation/remote-access/samba.md

Configuration et commandes

- Le fichier **/etc/samba/smb.conf** est composé de deux parties :
  - Un partie globale, qui permet de configurer le fonctionnement du service.
  - Une partie partages, où sont listés les partages de répertoires et d'imprimantes et leurs paramètres.
- Quelques commandes : **smbclient**, **smbmount**, **smbstatus**, ...
- Ajouter manuellement des utilisateurs Samba : **smbpasswd**
- Vérifier la configuration de Samba en utilisant la commande : **testparm -s**

Autres : FTP, Web

- Serveur Web HTTP (Hypertext Transfer Protocol) :
  - Apache ➟ www.raspberrypi.org/documentation/remote-access/web-server/apache.md, raspberry-pi.developpez.com/cours-tutoriels/serveur-web/ et tvaira.free.fr/esimed/admin/AdminLinux-Apache.pdf
  - NGINX ➟ www.raspberrypi.org/documentation/remote-access/web-server/nginx.md
- Serveur FTP (*File Transfer Protocol*) :
  - proftpd ➟ raspbian-france.fr/installer-serveur-ftp-raspberry-pi/
  - Pure-FTPd ➟ www.raspberrypi.org/documentation/remote-access/ftp.md
  - vsFTPd ➟ tvaira.free.fr/esimed/admin/AdminLinux-Ftp.pdf

Sommaire

1 Présentation

2 Distribution Raspbian

3 GPIO

4 Administration

5 Réseau

6 Bluetooth

7 Mode *Kiosk*

6 Bluetooth
- Présentation
- Prise en charge
- Les commandes de base
- Normes

Bluetooth

Bluetooth est une norme de communications permettant l'échange bidirectionnel de données à très courte distance en utilisant des ondes radio UHF sur une bande de fréquence de 2,4 GHz.

La liaison Bluetooth présente les caractéristiques suivantes :

- très faible consommation d'énergie (entre 1 mW et 100 mW selon les classes)
- très faible portée (entre 1 m et 100 m selon les classes)
- faible débit (qq Mbits/s)
- très bon marché et peu encombrant.

Pour des raisons de sécurité, les périphériques Bluetooth ne peuvent pas simplement être "utilisés" sans une autorisation explicite appelée *pairing* (appairage ou couplage).

✍ Avec l'arrivée de la Raspberry Pi 3, le support Bluetooth a été nativement ajouté. Il est aussi intégré avec la Raspberry Pi Zero W.

Prise en charge

```
# Paquets installés pour le Bluetooth
$ dpkg --get-selections | grep -i "blue"
bluealsa        install
bluez           install
bluez-firmware      install
libbluetooth3:armhf     install
lxplug-bluetooth    install
pi-bluetooth        install

# Prise en charge au démarrage
$ dmesg | grep -i bluetooth

# Les modules du noyau pour le support du Bluetooth
$ lsmod | grep -i -E "bluetooth|bnep|rfcomm"

# Etat du service Bluetooth
$ sudo systemctl status bluetooth

# Le périphérique Bluetooth
$ hciconfig
hci0: BD Address: B8:27:EB:01:98:7E ACL MTU: 1021:8 SCO MTU: 64:1
```

Les commandes de base

- **bluetoothctl** est un outil de contrôle du Bluetooth (il remplace les anciennes commandes `bluez-xxx-xxx`).
- **hciconfig** est utilisé pour configurer les périphériques Bluetooth (`hciX` est le nom d'un périphérique Bluetooth installé dans le système).
- **hcitool** est utilisé pour configurer les connexions Bluetooth et envoyer une commande spéciale aux périphériques Bluetooth.
- **l2ping** envoie une requête d'écho L2CAP à l'adresse MAC Bluetooth `BD_ADDR` en hexadécimale.
- **rfcomm** est utilisé pour configurer, maintenir et inspecter la configuration RFCOMM du sous-système Bluetooth dans le noyau Linux.
- **sdptool** permet d'effectuer des requêtes SDP (*Service Discovery Protocol*) sur les périphériques Bluetooth.
- **hcidump** lit et affiche les données HCI brutes d'une communication Bluetooth.

Normes I

Depuis la version 4.0, on distingue deux normes principales : .

- Bluetooth (*Classic*)
- Bluetooth LE (*Low Energy*) ou *smart*

✍ **BlueZ** est un logiciel qui met en œuvre la technologie sans fil Bluetooth sur le système d'exploitation GNU/Linux. Les fichiers de configuration sont stockés dans `/etc/bluetooth/`.

Normes II

```
# Exemple d'appairage
$ bluetoothctl
[NEW] Controller B8:27:EB:01:98:7E raspberrypi [default]
[bluetooth]# scan on
Discovery started
[NEW] Device 20:15:11:16:76:04 HC-05
[bluetooth]# agent on
Agent registered
[bluetooth]# pair 20:15:11:16:76:04
Attempting to pair with 20:15:11:16:76:04
Request PIN code
[agent] Enter PIN code: 1234
Pairing successful
[bluetooth]# trust 20:15:11:16:76:04
Changing 20:15:11:16:76:04 trust succeeded
[bluetooth]# quit

# Recherche de périphériques à proximité
$ hcitool scan # Classic
$ sudo hcitool lescan # BLE
```

Sommaire

1 Présentation

2 Distribution Raspbian

3 GPIO

4 Administration

5 Réseau

6 Bluetooth

7 Mode *Kiosk*

7 Mode *Kiosk*

- Démarrage automatique
- Configuration *kiosk*

Mode *Kiosk*

Le mode « Kiosque » (*kiosk*) a été conçu à l'origine pour les bornes internet (*Internet kiosks*). Ces bornes nécessitent la mise en place d'un environnement logiciel restreint aux fonctions essentielles afin d'empêcher les utilisateurs de faire quoi que ce soit d'indésirable ou de dangereux. Cela comprend aussi :

- de dédier l'affichage graphique exclusivement à une application en plein écran
- d'enlever les composants graphiques habituels (curseur souris, barre de titre, tableau de bord, icônes, etc ...)
- de désactiver les modes économiseur d'écran (notamment l'écran de veille)

✍ Ce mode est particulièrement utile pour les afficheurs autonomes, comme des bornes accessibles au public. Parmi les exemples les plus couramment rencontrés : les distributeurs de billets, les affichages d'horaires dans les gares, ...

Démarrage automatique

- La configuration du démarrage automatique du bureau GUI peut se réaliser avec **raspi-config** dans "Boot Options Configure options for start-up" puis dans "B1 Desktop / CLI" choisir "B4 Desktop Autologin Desktop GUI, automatically logged in as 'pi' user". Cela crée un lien symbolique /etc/systemd/system/default.target vers /lib/systemd/system/multi-user.target.
- Le démarrage automatique des applications est configuré pour tous les utilisateurs dans le fichier /etc/xdg/lxsession/LXDE-pi/autostart (anciennement /etc/xdg/lxsession/LXDE/autostart)
- Le démarrage automatique des applications est configuré pour chaque utilisateur dans le fichier $HOME/.config/lxsession/LXDE-pi/autostart (anciennement $HOME/.config/lxsession/LXDE/autostart)

Autologin

La configuration de l'autologin de l'utilisateur `pi` peut se réaliser avec **`raspi-config`** dans "`Boot Options Configure options for start-up`" puis dans "`B1 Desktop / CLI`" choisir "`B4 Desktop Autologin Desktop GUI, automatically logged in as 'pi' user`" :

```
# configurer le service lightdm pour l'autologin de l'utilisateur pi
$ vim /etc/lightdm/lightdm.conf
autologin-user=pi
```

Démarrage automatique d'une application

Il y a plusieurs possibilités :

- directement dans un des fichiers `autostart` : `@MonApplication`
- en ajoutant une entrée `.desktop` dans `/etc/xdg/autostart/` ou dans `$HOME/.config/autostart/` :

```
[Desktop Entry]
Type=Application
Name=MonApplication
Comment=MonApplication
NoDisplay=false
Exec=/usr/bin/MonApplication
```

- en utilisant un service `systemd` (cf. Création d'un service)

Configuration du mode *kiosk*

```
# configurer le démarrage de LXDE (environnement de bureau X11)
$ vim /home/pi/.config/lxsession/LXDE-pi/autostart
#@lxpanel --profile LXDE # LXPanel la barre des tâches standard de LXDE
#@pcmanfm --desktop --profile LXDE-pi # PCMan File Manager le gestionnaire de
    fichiers par défaut
#@xscreensaver -no-splash # XScreenSaver l'écran de veille ou économiseur d'
    écran

# désactiver le gestionnaire d'économie d'énergie DPMS et l'écran de veille
$ vim /home/pi/.config/lxsession/LXDE-pi/autostart
@xset s off
@xset -dpms
@xset s noblank

$ vim /etc/lightdm/lightdm.conf
# don't sleep the screen
xserver-command=X -s 0 dpms

# Masquer le curseur de la souris
$ sudo apt-get install unclutter
```