

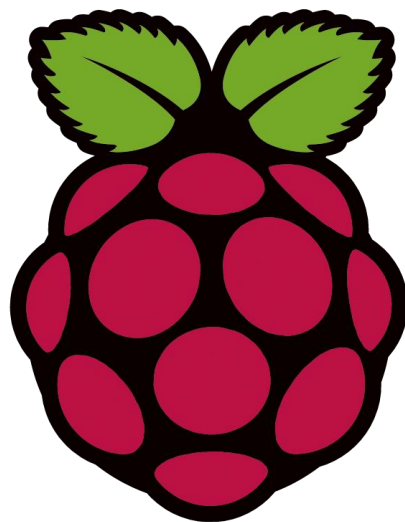
Paul KLAKE

François MONTEIL

# Compte rendu

—

## Raspberry



## Table des matières

I – Préparation de la micro carte SD .....	3
II – Installation de busybox.....	4
III – Installation librairie wiringPi.....	5
Compilation wiringPI .....	5
Compilation devLib .....	6
Compilation de gpio.....	6
Utilisation de gpio sur la RPI .....	7
Approfondissement : Le système de boot de la RPI .....	8

## I – Préparation de la micro carte SD

Dans un premier temps nous devons préparer la micro carte SD nous allons alors la partitionner en deux :

- Une partie en Fat32 dans laquelle nous retrouverons la configuration du RPI, le noyau...
  - Partitionner Fdisk /dev/sdb
  - Rendre cette partition bootable (t)
  - 128 Mo
- Une partie Ext4 dans laquelle nous retrouverons le système de fichier.

Pour la suite nous avons récupéré la partition boot d'un Raspbian, nous nous occuperont d'installer un noyau linux sur la partition root.

Il nous faut ensuite modifier le fichier cmdline.txt et remplacer le champ : PARTUUID par /dev/mmcdlk0p2

## II – Installation de busybox

Pour pouvoir installer busybox sur la RPI il faut utiliser un compilateur croisé car la Raspberry Pi possède un processeur ARM alors que les ordinateurs possèdent des processeurs x86.

Pour cela nous avons réalisé un script de compilation :

```
#!/bin/sh
set -e

CROSS_COMPILE="/home/francois/Desktop/RPI/tools-master/arm-bcm2708/gcc-
linaro-arm-linux-gnueabi-hf-raspbian-x64/bin/arm-linux-gnueabi-hf-"

cd busybox-1.29.3
make CROSS_COMPILE="$CROSS_COMPILE" CONFIG_PREFIX="/mnt/root" install
```

Une fois busybox installer, il faut regarder ses dépendances, pour cela nous devons utiliser la commande ldd, mais celle fournie par le compilateur croisé :

```
./arm-linux-gnueabi-hf-ldd --root /mnt/root/bin /mnt/root/bin/busybox
```

La commande nous renseigne alors sur les bibliothèques manquantes, bibliothèque que nous devons récupérer dans un sous-dossier du compilateur croisé et les copier dans le dossier lib à la racine de la partition root.

### III – Installation librairie wiringPi

La librairie wiringPi met à notre disposition un programme nommé gpio qui permet de contrôler et voir le statut des port GPIO de la Raspberry.

Pour l'installer dans notre busybox depuis notre environnement de travail sur ordinateur, il va falloir utilisé de nouveau le compilateur croisé.

Tout d'abord nous téléchargeons le dossier wiringPi qui est à notre disposition sur <http://wiringpi.com/>.

```
$ git clone git://git.drogon.net/wiringPi
```

Pour obtenir les librairies wiringPi (.so) ainsi que l'exécutable gpio, il va nous falloir compiler les éléments dans le bon ordre.

#### Compilation wiringPi

Tout d'abord nous compilons les librairies wiringPi, pour cela nous allons modifier le Makefile afin qu'il utilise notre compilateur croisé :

Tout d'abord on redirige l'installation dans un dossier à la base du bureau, on aurait très bien pu la rediriger directement sur la partition root de la raspberry :

```
DESTDIR?=/home/francois/Desktop  
PREFIX?=/wiringPi
```

Ensuite on lui fournit le chemin vers le compilateur croisé, pour enfin lui définir quel gcc utilisé :

```
PATH_CROISE = /home/francois/Desktop/RPI/tools-master/arm-bcm2708/gcc-linaro-  
arm-linux-gnueabihf-raspbian-x64/bin  
GCC_CROISE = $(PATH_CROISE)/arm-linux-gnueabihf-gcc  
CC          = $(GCC_CROISE)
```

Une fois ces modifications faites, on peut compiler :

```
$ make all  
$ make install
```

Une fois ces commandes effectuées, on retrouve dans le dossier wiringPi sur le bureau deux dossier lib et include dans lesquelles nous retrouverons nos librairies compilées avec le compilateur croisé.

## Compilation devLib

Nous pouvons passer à la compilation des devLib, cette fois si en plus de lui indiquer ou s'installer et quel compilateur choisir, il va falloir lui indiquer où aller chercher les includes :

```
DESTDIR?=/home/francois/Desktop  
PREFIX?=/wiringPi  
INCLUDE = -I$(DESTDIR)$(PREFIX)/include
```

Une fois chose fait on compile de la même manière que wiringPi et on obtient de nouveau fichier dans les dossier include et lib.

## Compilation de gpio

Pour compiler le programme gpio, il faut comme précédemment indiquer :

1. Le dossier où l'installer
2. Le chemin du compilateur croisé
3. Quel fichier include utiliser

Et il faut de plus lui indiquer le chemin vers les librairies :

```
DESTDIR?=/home/francois/Desktop  
PREFIX?=/wiringPi  
LDFLAGS = -L$(DESTDIR)$(PREFIX)/lib
```

On compile et dans notre dossier wiringPi nous retrouvons un dossier bin avec notre programme gpio.

On copie include, lib et gpio dans la partition root, on remet la micro SD dans la raspberry et on la démarre.

## Utilisation de gpio sur la RPI

Une fois la raspberry lancer on utilise une commande tel que gpio readall.

La raspberry nous renvoie un message d'erreur, on répertorie alors le dossier lib :

```
# ls -al
```

On remarque alors que libwiringPi.so et libwiringPiDev.so sont des liens symboliques qui pointe vers des fichiers se trouvant dans un répertoire de l'ordinateur de préparation.

On supprime alors ces fichiers et rééditons les liens :

```
# rm libwiringPi.so
# rm libwiringPiDev.so
# ln -s /lib/libwiringPi.so.2.46 libwinringPi.so
# ln -s /lib/libwiringPiDev.so.2.46 libwinringPiDev.so
```

On réutilise la commande et on obtient alors :

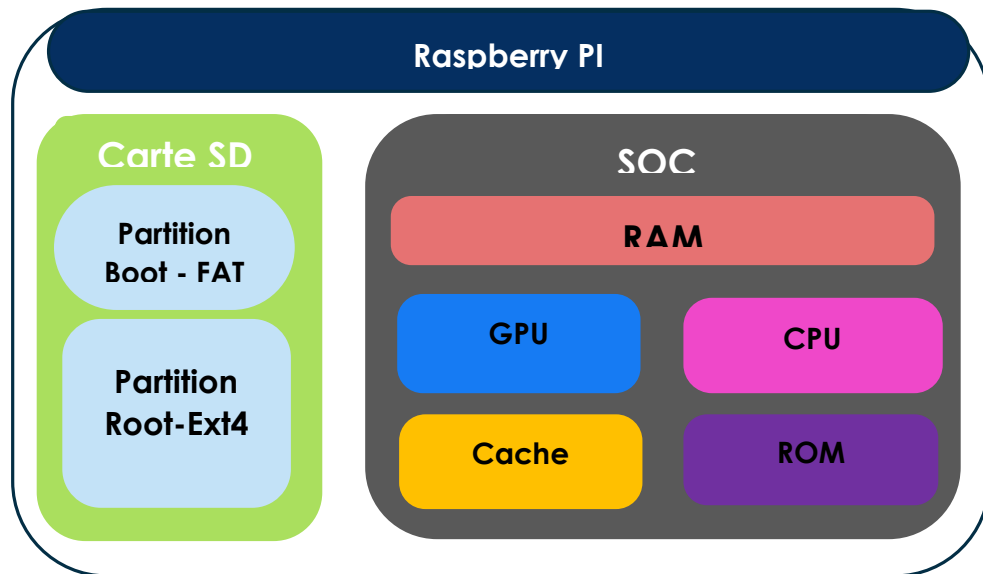
```
/ # gpio readall
```

Pi 3											
BCM	wPi	Name	Mode	U	Physical	U	Mode	Name	wPi	BCM	
		3.3v			1	2		5v			
2	8	SDA.1	IN	1	3	4		5v			
3	9	SCL.1	IN	1	5	6		0v			
4	7	GPIO. 7	IN	1	7	8	0	IN			
		0v			9	10	1	IN			
17	0	GPIO. 0	IN	0	11	12	0	IN			
27	2	GPIO. 2	IN	0	13	14		0v			
22	3	GPIO. 3	IN	0	15	16	0	IN			
		3.3v			17	18	0	IN			
10	12	MOSI	IN	0	19	20		0v			
9	13	MISO	IN	0	21	22	0	IN			
11	14	SCLK	IN	0	23	24	1	IN			
		0v			25	26	1	IN			
0	30	SDA.0	IN	1	27	28	1	IN			
5	21	GPIO.21	IN	1	29	30		0v			
6	22	GPIO.22	IN	1	31	32	0	IN			
13	23	GPIO.23	IN	0	33	34		0v			
19	24	GPIO.24	IN	0	35	36	0	IN			
26	25	GPIO.25	IN	0	37	38	0	IN			
		0v			39	40	0	IN			

```
/ #
```

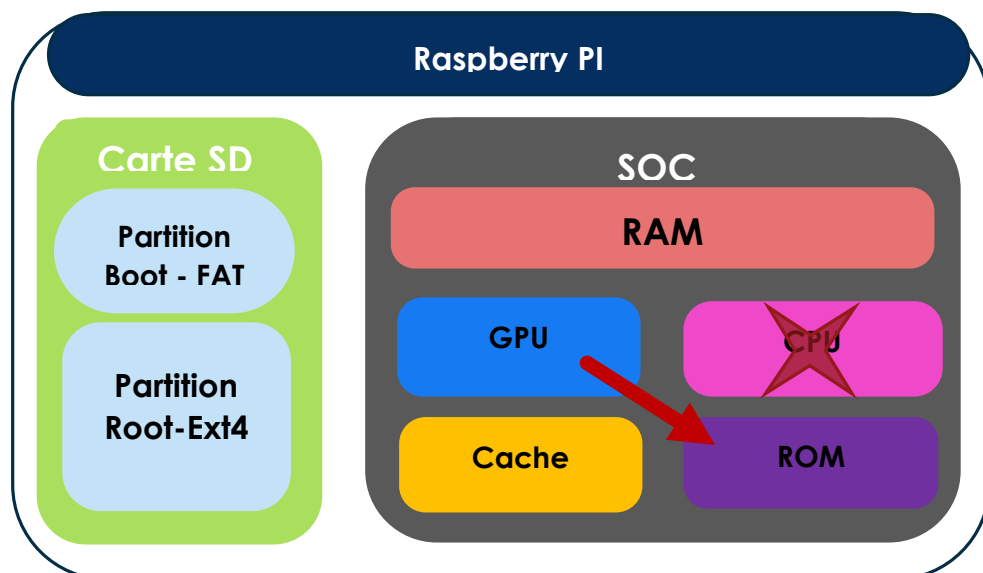
Le programme gpio fonctionne.

## Approfondissement : Le système de boot de la RPI



Au démarrage de la Raspberry Pi, un signal RUN est établi, et le SOC démarre (System On a Chip), sur celui-ci nous retrouvons votre processeur ARM, un GPU responsable de l'affichage sur écran, la mémoire RAM, un cache et les port GPIO (entre autre).

Au démarrage du SOC, seul le GPU s'active, et la première chose qu'il fait c'est exécuter le programme se trouvant dans la ROM (Read Only Memory) dans laquelle le constructeur à programmer le premier étage du bootloader.



L'objectif de ce programme est d'accéder à la partition FAT (mmcblk0p1) se trouvant sur la carte SD.

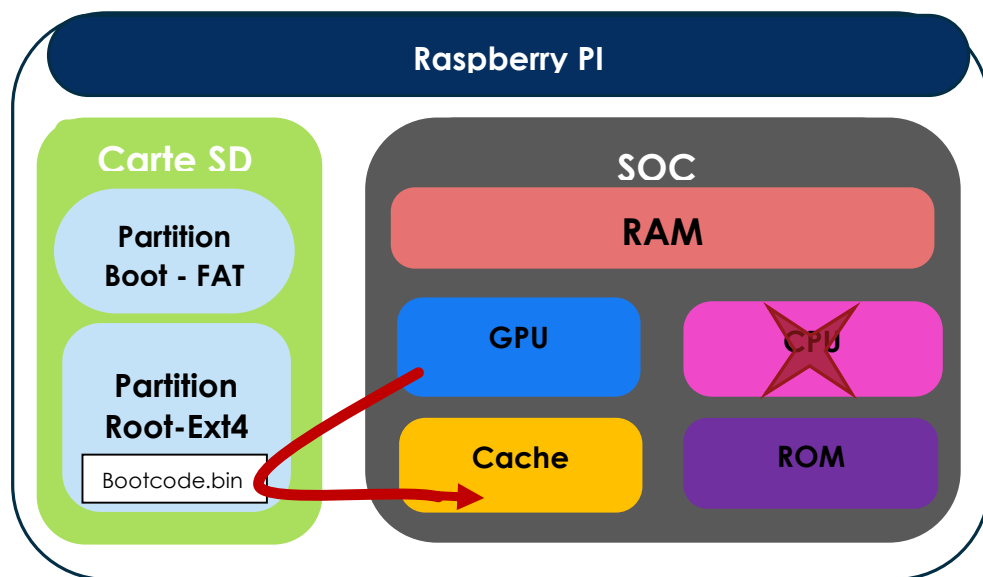


Dans cette partition boot nous retrouvons les éléments suivants :

```
bcm2708-rpi-0-w.dtb      cmdline.txt      kernel.img
bcm2708-rpi-b.dtb       config.txt       LICENCE.broadcom
bcm2708-rpi-b-plus.dtb  COPYING.linux   LICENSE.oracle
bcm2708-rpi-cm.dtb      fixup_cd.dat    overlays
bcm2709-rpi-2-b.dtb     fixup.dat       start_cd.elf
bcm2710-rpi-3-b.dtb     fixup_db.dat    start_db.elf
bcm2710-rpi-3-b-plus.dtb fixup_x.dat      start.elf
bcm2710-rpi-cm3.dtb     issue.txt       start_x.elf
bootcode.bin            kernel7.img     'System Volume Information'
```

- bootcode.bin : second étage du bootloader
- start.elf: firmware binaire du GPU
- kernel.img: Le noyau de l'OS à exécuter par le processeur ARM : ici Linux
- cmdline.txt: paramètres passés au noyau lors du boot.

Une fois que le programme de trouvant dans la ROM a trouvé la partition Boot de notre carte, le GPU charge le second étage du bootloader qu'est bootcode.bin dans le cache puis l'exécute.



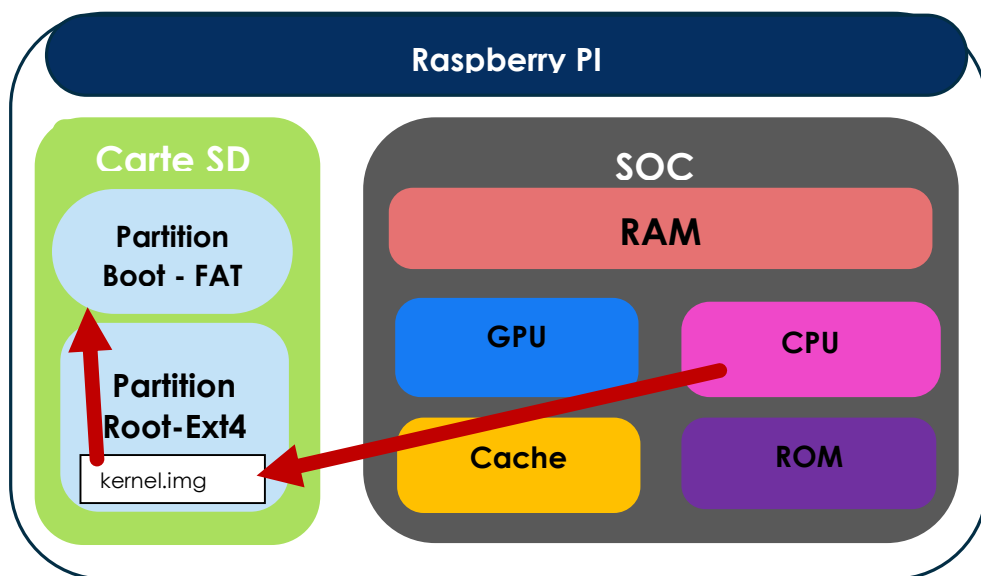
L'objectif de bootcode.bin est de mettre en service la RAM et de charger start.elf.

Une fois start.elf chargé, il est exécuté par le GPU, ce programme lit config.txt et cmdline.txt qui permettent de paramétrer des éléments du boot (tel que l'emplacement de la partition Root mmcblk0p2).

Ensuite start.elf charge l'image du noyau linux : kernel.img.

C'est au tour du CPU de travailler, son rôle est alors d'exécuter l'image du noyau.

Le noyau charge le système de fichier qui se trouve sur la partition Root et linux démarre « normalement ».



Source : <https://www.framboise314.fr/booter-le-raspberry-pi-sur-un-disque-dur-usb/>