Notes cours embarqué et projet

Bastion Lab Cogitux

Fichier des clés acceptées sur une machine Linux

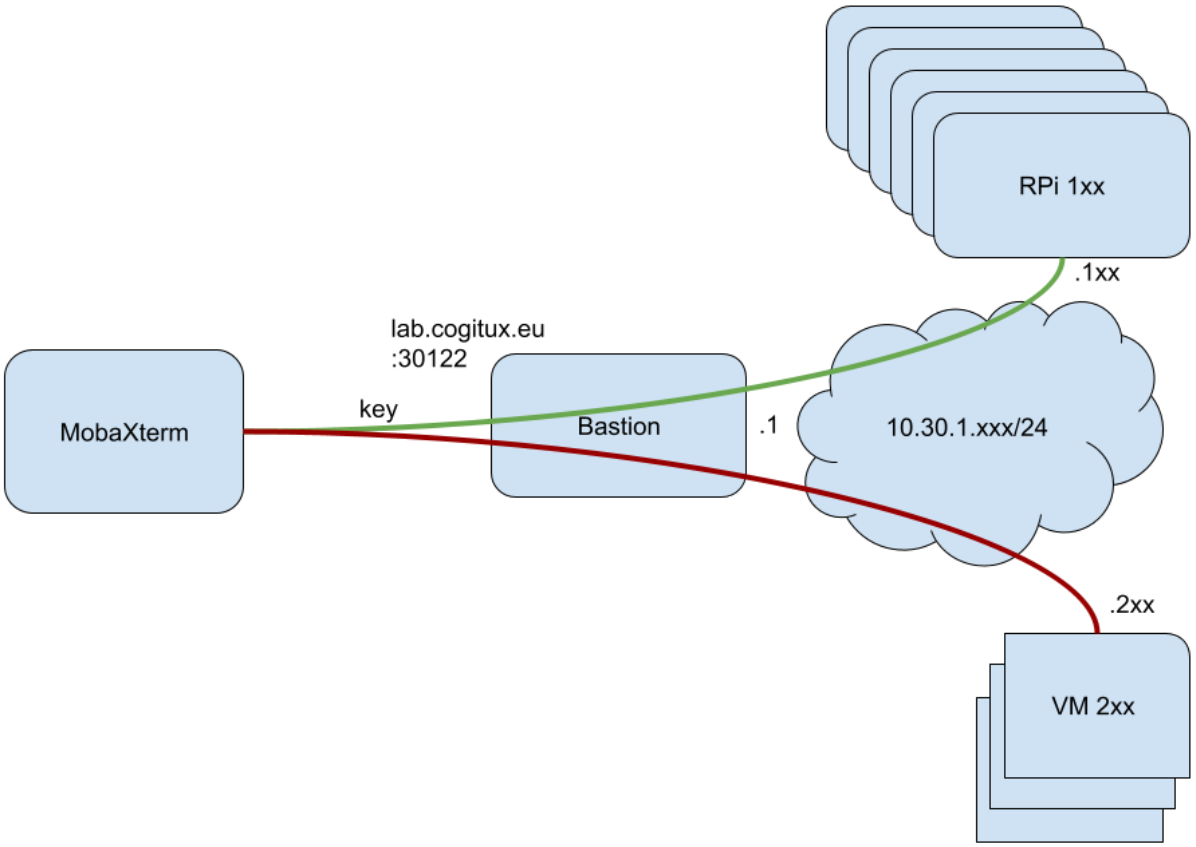
- ~/.ssh/authorized_keys

Joindre la Raspberry Pi #07 via SSH et un rebond sur le bastion

- ssh -J guest@lab.cogitux.eu:30122 **pi**@10.30.1.107

Joindre la VM Debian #01 (dev) via SSH et un rebond sur le bastion

- Adresse IP termine par 230 + N° de VM
- ssh -J guest@lab.cogitux.eu:30122 **dev**@10.30.1.231



VM Debian 11 de développement embarqué

- Utilisateur dev (passwd: dev) peut accéder en SSH, sudoer
- Super-utilisateur root via commande sudo uniquement
- Personnaliser le hostname :
  - `sudo hostnamectl set-hostname debianVM-Julien`
  - Modifier le `/etc/hosts`

Raspberry-Pi

- Utilisateur pi (passwd: **raspi**), sudoer

TP LFS

Crosstool-NG

- `export CT_PREFIX=/home/dev/usr`

Linux kernel

- DTB/DTS
  - `cat linux-5.15.46/arch/arm/boot/dts/versatile-pb.dts`
  - `cat linux-5.15.46/arch/arm/boot/dts/versatile-ab.dts`
    `# included by PB`

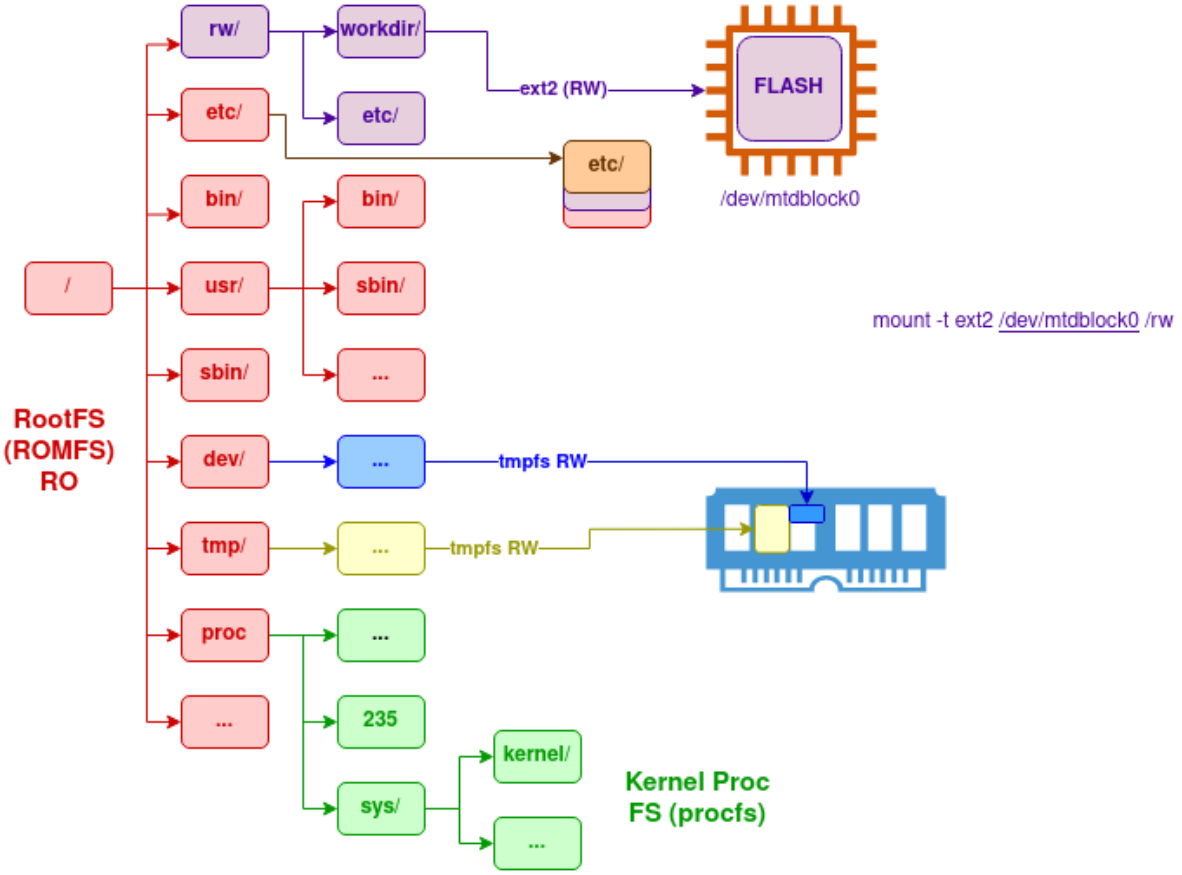
Rootfs

- Montage
  - `mount -t proc none /proc`
  - `cat /proc/filesystems`
  - `mount -t sysfs none /sys`
- Automatisation
  - `mkdir -p rootfs1/etc/init.d/`
  - `nano rootfs1/etc/init.d/rcS`
    - Shebang : `#!/bin/sh`
  - `chmod +x rootfs1/etc/init.d/rcS`
    - ***puis :*** `genromfs -f romfs.img -V "RMFS Busybox RootFS" -d rootfs1`
  - `Alternative : créer un /etc/fstab et ajouter “mount -a” dans son script rcS`
    - /etc/fstab :

      ```
      none /proc   proc    defaults 0 0
      none /sys sysfs   defaults 0 0
      ```
- Peupler /dev automatiquement
  - `mount -t tmpfs -o size=64k none /dev`
  - `mdev -s`

OverlayFS

- But : rendre /etc virtuellement RW



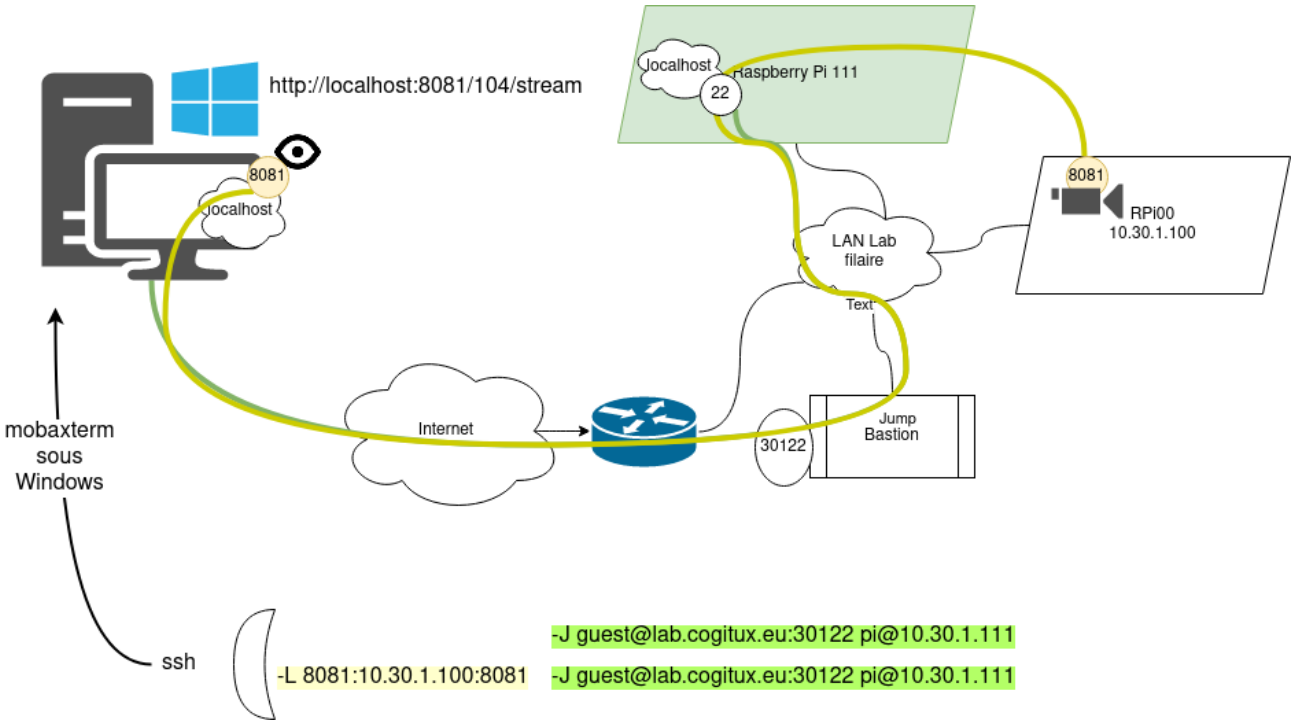
- Pour monter /rw en synchrone (pas de cache) et sans les “Access Time’ fichiers et répertoires :
  `mount -t ext2 -o sync,noatime,nodiratime /dev/mtdblock0 /rw/`

RPiOS

Préliminaires

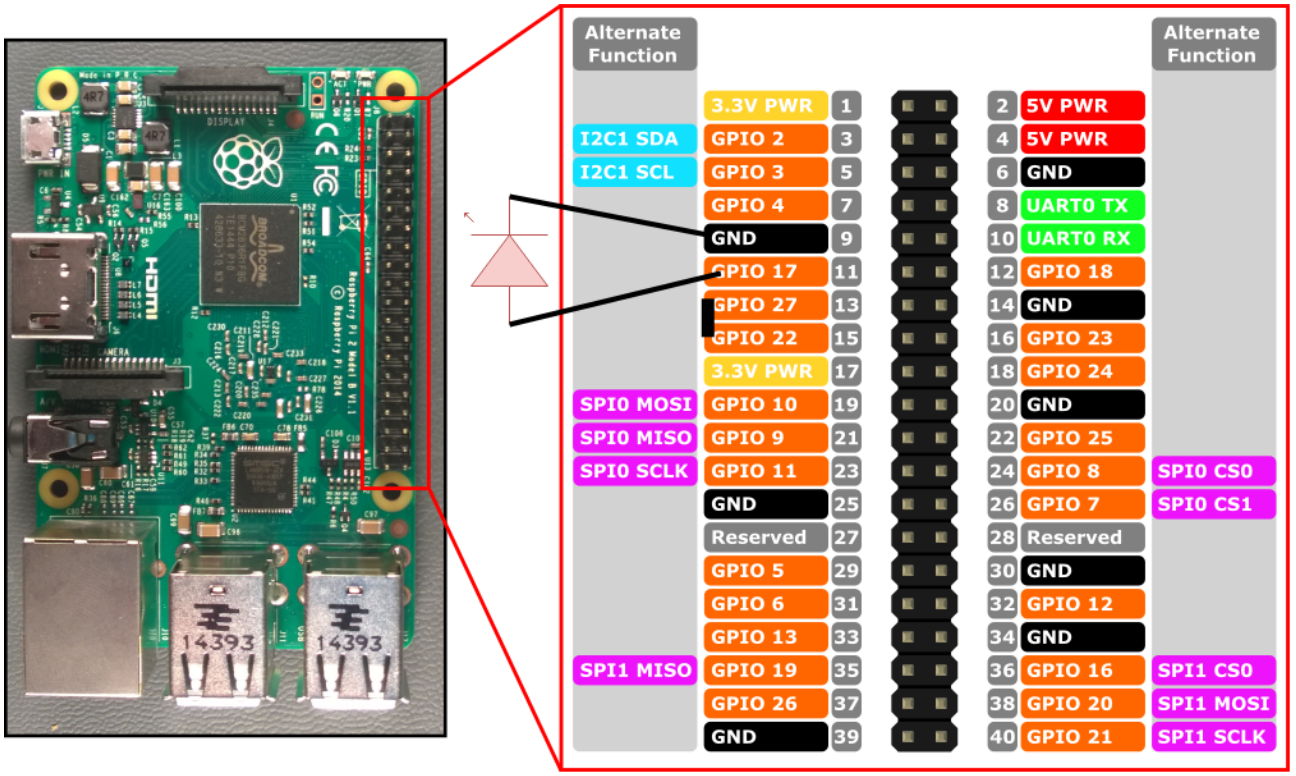
- Se détacher du tmux LFS (CTRL-b d)
- Renommer le tmux LFS si c’était le n°1
  `tmux rename-session -t 1 LFS`
- Créer un nouveau tmux pour le TP RPiOS
  `tmux new -s RPiOS`
- Générez votre paire de clé SSH sur la VM et transférez là sur votre RPi
  `ssh-keygen`
  `ssh-copy-id pi@10.30.1.10x` # Avec la bonne IP pour votre RPi

Accès à la caméra du Lab



- Dans un terminal <u>local</u> MobaXterm (nouvel onglet avec [+])
  ssh -L 8081:10.30.1.100:8081 -J guest@lab.cogitux.eu:30122 pi@10.30.1.1XX
- ou, si vous préférez utiliser la VM Debian comme passerelle :
  ssh -L 8081:10.30.1.100:8081 -J guest@lab.cogitux.eu:30122 **dev**@10.30.1.**23X**
- Après ça, la caméra est dispo dans votre navigateur Windows à l'adresse
  http://localhost:8081/104/stream

GPIO



- LED sur la GPIO 17 (câblée entre pins 11 et 9)
- Sortie sur la GPIO 27 (câblée entre pins 13 et 15)
- Entrée sur la GPIO 22 pour poll() (câblée entre pins 13 et 15)

Automatisation au démarrage

- On peut utiliser un script /etc/rc.local (attention droits +x) qui sera exécuté par SystemD au démarrage (via service rc-local)
- Le script est exécuté par root au démarrage donc pas besoin de sudo ou autre élévation de droits
- Au démarrage, on configure les LEDs, on prépare les GPIO utiles

Cross-débogage

- Téléchargez toolchain de Raspberry-Pi (même si plus supportée elle a le cross-débogueur alors que le paquet Debian ne l'a pas), téléchargez le Zip (git) depuis Bouton Code
- Sur l'host, on utilisera tools-master/arm-bcm2708/arm-linux-gnueabihf/bin/arm-linux-gnueabihf-gdb pour déboguer à distance
- Sur la cible il faut installer le paquet gdbserver

Buildroot

Préliminaires

- Se détacher du tmux LFS ou RPiOS (CTRL-b d)
- Créer un nouveau tmux pour le TP Buildroot

```
tmux new -s buildroot
```

- Aller dans le dossier buildroot que j'ai créé sur vos VMs
- Extraire (tar xf …) l'archive buildroot
- Suivre le PDF TP/buildroot (AJC Classroom) sauf pour :
  - Au §2.2, choisir **make raspberrypi3_defconfig**
  - **Ne pas tenir compte du §2.3**
  - Au §2.4, pour lancer le make, ajoutez une attente de "1H x votre ID" (ex: Jérém = 7)

```
sleep $(bc <<< 3600*7) && time make
```

Première installation

- ~~Reprendre après les préliminaires au §2.5~~
- ~~On passe §2.6, §2.7 et §2.8 pour l'instant~~
- ~~On reprend au §3 sans Q5~~
- ~~À la Q7, changer le port en 8080 pour le serveur HTTP de Busybox~~
- ~~C'est moi qui vous flashe la SDcard en Q8~~

Fix compilation Buildroot

- ~~Pour erreur PHP, dans output/build/php-8.0.19/ext/standard/crc32.c~~
  - ~~ajouter " && defined(HAVE_SYS_AUXV_H)" aux 2 lignes qui ont juste "#if HAVE_AARCH64_CRC32"~~
- ~~Pour forcer le recompilation de GDB suite à l'ajoute de gdbserver et WCHAR et thread debug de la uclibc :~~
  - ~~rm output/build/uclibc-1.0.40/.stamp*~~
  - ~~rm package/gdb -rf~~
    - ~~Si vous avez fait ça, il faut restaurer gdb :~~
      ~~cd .. && tar xf buildroot-2022.02.2.tar.xz buildroot-2022.02.2/package/gdb/~~
  - ~~rm -rf output/build/gdb-10.2~~

- ~~make menuconfig # avec tous les trucs nécessaires~~
- ~~make clean~~
- ~~make pour relancer la compilation de l'image~~
- J'ai ajouté le fichier package/php/0006-fix-sys_auxv_include.patch sur vos machines pour permettre de patcher automatiquement PHP lors de la compilation
- Normalement la compilation a été relancée avec un clean (et les options WCHAR et thread debug comme indiqué dans la nouvelle version du TP FR_Buildroot_no_graphic §2.3)
- Pour anticiper les besoins, augmenter la taille du rootfs à 256M (Filesystem images > (256M) exact size)
- Pour serveur web Busybox
    - make busybox-menuconfig
        networking utilities > httpd et port 8080

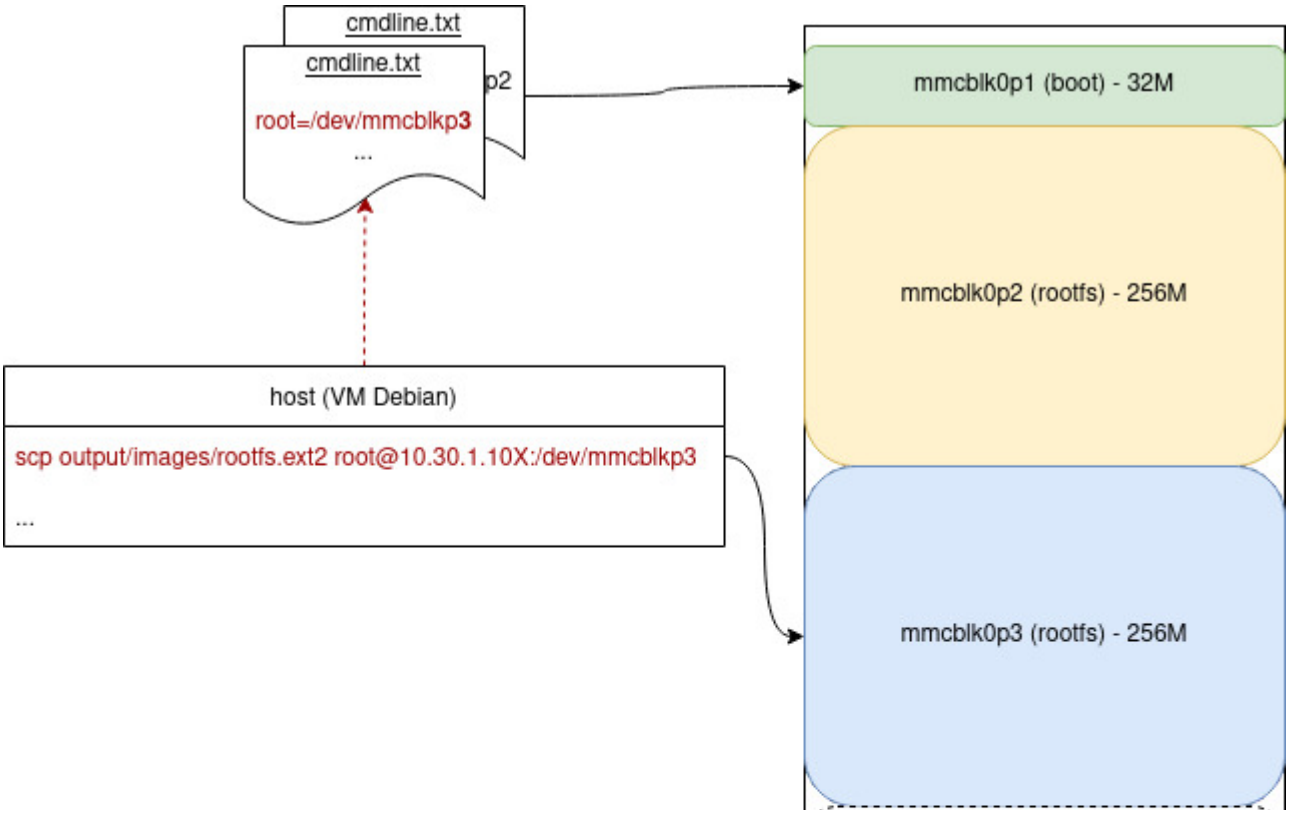
SSH et Dropbear

Rootfs et partition de boot

- Montage en RW / RO
    mount -o remount,rw /
    mount -o remount,ro /
- Monter la partition de boot pour modifier config.txt par exemple
    mount /dev/mmcblk0p1 /mnt
    vi /mnt/config.txt

Mise à jour du rootfs

- Préparatifs
    - Ajouter la partition mmcblkp3 (une seule fois)
        - n (new)
        - p (primaire)
        - 3 (n°3)
        - start par défaut (entrée)
        - size +256M
    - Il faudra rebooter pour voir la nouvelle partition
- Pour chaque mise à jour
    - Transférer la nouvelle image du rootfs (output/images/rootfs.ext2) directement dans la partition destination
      `scp output/images/rootfs.ext2 root@10.30.1.10X:/dev/mmcblk0p3`
    - Monter la partition de boot dans /mnt
      `mount /dev/mmcblk0p1 /mnt`
    - Editer le fichier /mnt/cmdline.txt pour changer mmcblk0p2 en mmcblk0p3
        - Attention de ne rien changer d'autre


cmdline.txt
cmdline.txt
p2
root=/dev/mmcblkp3
...
mmcblk0p1 (boot) - 32M
mmcblk0p2 (rootfs) - 256M
mmcblk0p3 (rootfs) - 256M
host (VM Debian)
scp output/images/rootfs.ext2 root@10.30.1.10X:/dev/mmcblkp3
...

WiFi

Lab

- Premier réseau
  - SSID : LabCogitux
  - Passphrase : LABO-LINUX-301
- Second réseau
  - SSID : LabCogituxSpare
  - Passphrase : LABO-LINUX-SPARE-301

Configuration via wpa_cli

Nom de l'interface à configurer

```
(for dir in /sys/class/net/*/wireless; do if [ -d "$dir" ]; then
basename "$(dirname "$dir")" ; fi; done) | tail -1
```

Configuration de l'interface

```
wpa_cli -i wlan0 status
wpa_cli -i wlan0 remove_network 0
wpa_cli -i wlan0 add_network
wpa_cli -i wlan0 set_network 0 ssid \"LabCogitux\"
wpa_cli -i wlan0 set_network 0 psk \"LABO-LINUX-301\"
wpa_cli -i wlan0 enable_network 0
wpa_cli -i wlan0 status
wpa_cli -i wlan0 list_networks
wpa_cli -i wlan0 save_config
wpa_cli -i wlan0 reconfigure
```

Export json du statut de configuration

```
echo -e "{\n$(wpa_cli -i wlan0 status | sed -e 's/^/  \"/;s/=/\":
\"/;s/$/\",/')\n  \"date\": \"$(date)\"\n}"
```

```
{
  "bssid": "c2:fb:e4:da:4e:36",
  "freq": "2412",
  "ssid": "LabCogitux",
  "id": "0",
  "mode": "station",
  "pairwise_cipher": "CCMP",
  "group_cipher": "CCMP",
  "key_mgmt": "WPA2-PSK",
  "wpa_state": "COMPLETED",
  "ip_address": "10.30.1.22",
  "p2p_device_address": "ba:27:eb:c4:11:6f",
  "address": "b8:27:eb:c4:11:6f",
  "uuid": "1b0b3733-9ff9-57b9-a9df-814e6203db94",
  "date": "Thu 22 Jul 10:51:59 CEST 2021"
}
```

Tests

```
pi@rpi-11:~ $ ip -s link show dev wlan0
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DORMANT group default qlen 10
00
    link/ether b8:27:eb:c4:11:6f brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    113760     683      0       0       0       461
    TX: bytes  packets  errors  dropped carrier collsns
    40005      337      0       0       0       0
pi@rpi-11:~ $ ping -I wlan0 1.1.1.1
PING 1.1.1.1 (1.1.1.1) from 10.30.1.22 wlan0: 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=59 time=21.6 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=59 time=20.4 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=59 time=20.7 ms
64 bytes from 1.1.1.1: icmp_seq=4 ttl=59 time=27.8 ms
^C
--- 1.1.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 7ms
rtt min/avg/max/mdev = 20.390/22.629/27.769/3.000 ms
pi@rpi-11:~ $ ip -s link show dev wlan0
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DORMANT group default qlen 10
00
    link/ether b8:27:eb:c4:11:6f brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    114096     687      0       0       0       461
    TX: bytes  packets  errors  dropped carrier collsns
    40493      341      0       0       0       0
```

TODO

1. Accès SSH
    a. alias ssh-anonyme='ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no'
2. Test des LEDs
3. Script de setup des LEDs au démarrage (via overlay)
4. Mettre en place le GIT et un dépôt centralisé (Github, Gitlab…) que vous me communiquerez
5. Setup de l'i2c et du capteur BMP180 via IIO
    a. Il faut modifier le fichier config.txt comme précédemment
    b. Il faut charger les modules i2c-bcm2835 et bmp280-i2c manuellement au boot
6. Setup d'une page Web lighttpd et test de httpd de busybox
7. Setup de PHP
8. Setup du WiFi
9. Setup d'un partition 4 RW
10. Protocole d'update