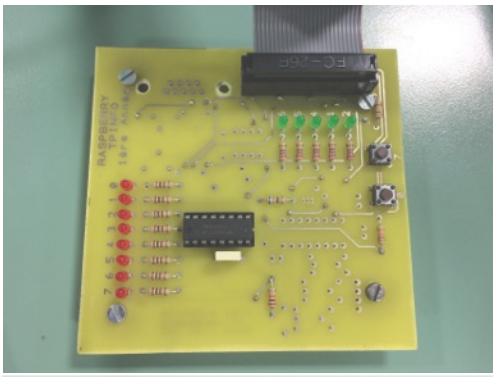
Compte rendu de projet:

Utilisation de Bootstrap sur un Raspberry Pi avec WiringPi:



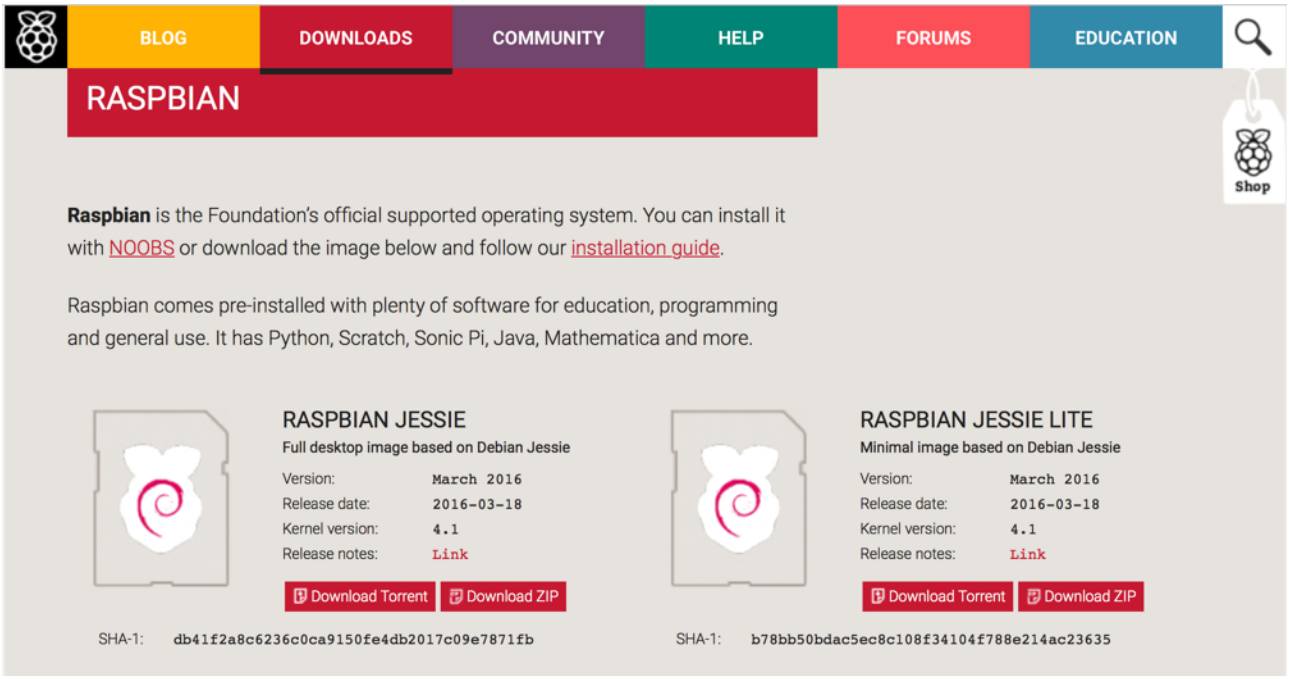
Introduction:

Le but du projet a été dans un premier temps de maitriser le raspberry et son interface en ligne de commande. Différents tests sur une carte WiringPi ont été fait afin d'interagir sur les LED de celle ci directement depuis la ligne de commande.
La suite du projet consiste à utiliser le Raspberry comme serveur web afin d'afficher une interface qui agira sur les LED de la carte connectée au Raspberry. La dernière partie consistera à utiliser notre système comme point d'accès wifi, afin de lui épargner une connexion filaire.

Sommaire:

1) Récupération de l'image Raspberry:

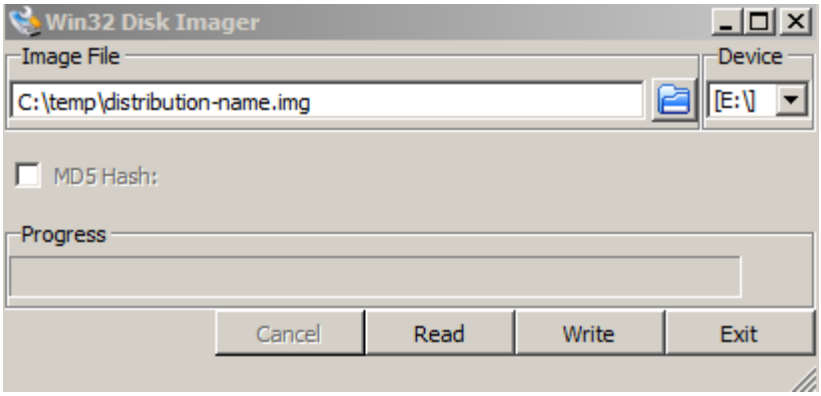
Pour télécharger l'image Raspberry : https://www.raspberrypi.org/downloads/raspbian/



Pour installer l'image sur le Raspberry, il y a 2 solutions:

Solution 1 :

Télécharger Win32DiskImager http://sourceforge.net/projects/win32diskimager/ puis récupérer la dernière image RAW de raspbian sur http://www.raspberrypi.org/downloads (actuellement la 2013-09-25-wheezy-raspbian.zip) et la décompresser. Puis ouvrir windiskimager et charger l'image de votre OS (xxx.img) puis copier sur la SDcard (Cf figure ci-dessous) Attention, le nom de l'image et le nom du lecteur ne sont pas forcément les bons !



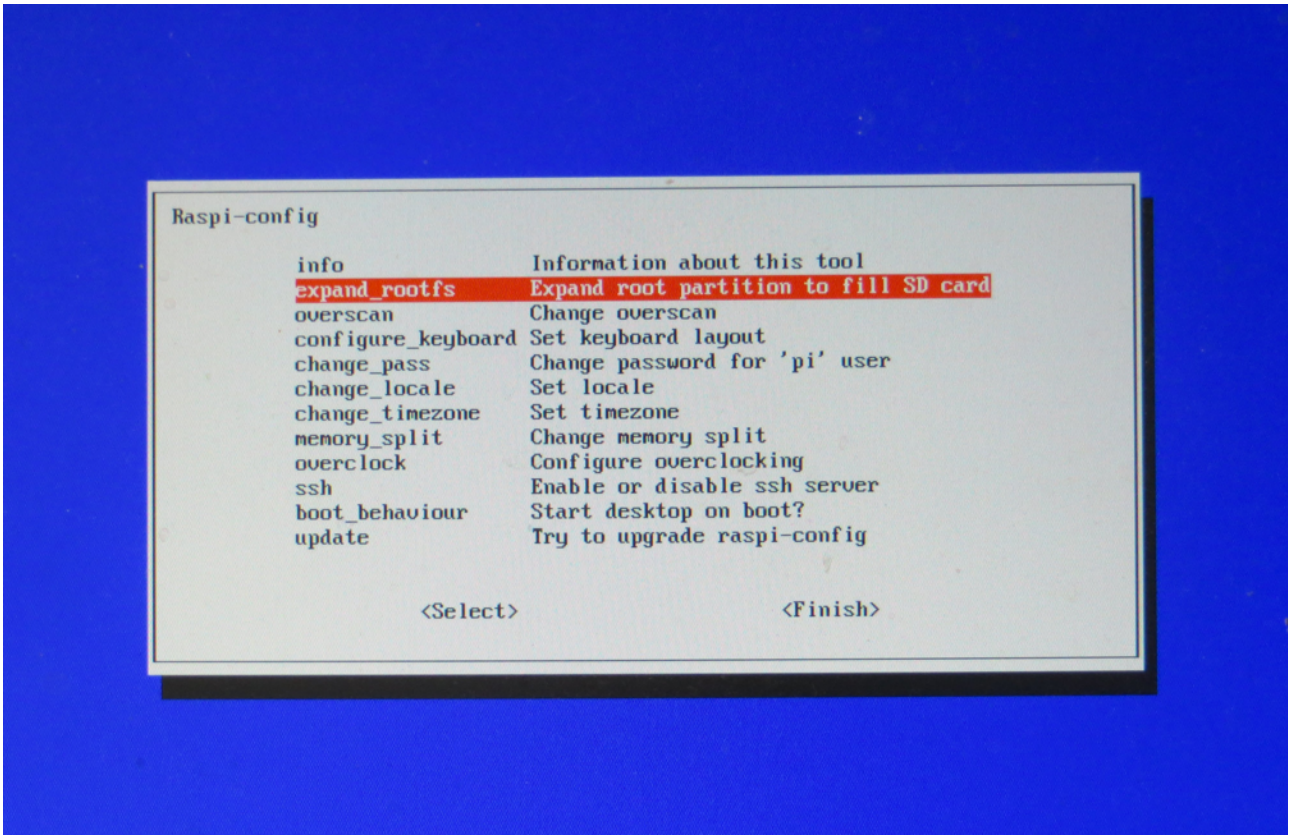
Solution 2 :

Télécharger l'outil de formatage de la carte sd (https://www.sdcard.org/downloads/formatter_4/eula_windows/) . Insérer la carte sd dans un lecteur, puis la formater. Ensuite télécharger http://downloads.raspberrypi.org/noobs , le décompresser et copier les fichiers décompressés à la racine de la carte. La carte SD est alors prête à être insérée dans la raspberry. Au démarrage, veiller à avoir une connection internet filaire (rj45), puis un menu guide l'installation. On choisira d'installer Raspbian.

2) Configurations dans le bios:

Le clavier sera configuré en qwerty à l'installation, rien de bien embêtant mais il reste plus pratique de configurer le clavier en azerty.
Pour cela, lancer le bios du raspberry en tapant sudo raspi-config dans la ligne de commande.
La configuration du clavier se fait dans « configure_keyboard », celle de la langue dans « change_locale ».
Il faut veiller à autoriser le ssh. Pour cela, aller dans « ssh » et sélectionner « Enable ».

Par défaut, les login et password donc respectivement « pi » et « raspberry ».



3) Installation de vnc server:

L'installation de vnc server sur le Raspberry est très simple, il suffit de taper:
sudo apt-get install vncserver dans la ligne de commande
VNC facilitera l'utilisation du raspberry depuis une machine distante, le nombre de ports USB étant relativement limité.

4) WiringPi:

Sources : http://wiringpi.com/download-and-install/ https://projects.drogon.net/raspberry-pi/wiringpi/the-gpio-utility/

WiringPi is maintained under GIT for ease of change tracking, however there is a *Plan B* if you're unable to use GIT for whatever reasons (usually your firewall will be blocking you, so do check that first!)

If you do not have GIT installed, then under any of the Debian releases (e.g. Raspbian), you can install it with:

```
sudo apt-get install git-core
```

If you get any errors here, make sure your Pi is up to date with the latest versions of Raspbian:

```
sudo apt-get update
sudo apt-get upgrade
```

To obtain WiringPi using GIT:

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

If you have already used the clone operation for the first time, then

```
cd wiringPi
git pull origin
```

Will fetch an updated version then you can re-run the build script below.

To build/install there is a new simplified script:

```
cd wiringPi
./build
```

The new build script will compile and install it all for you – it does use the sudo command at one point, so you may wish to inspect the script before running it.

Plan B

Click on this URL: (it should open in a new page)

https://git.drogon.net/?p=wiringPi;a=summary

Then look for the link marked **snapshot** at the right-hand side. You want to click on the top one. This will download a tar.gz file with a name *like* wiringPi-98bcb20.tar.gz. Note that the numbers and letters after **wiringPi** (98bcb20 in this case) will probably be different – they're a unique identifier for each release.

You then need to do this to install:

```
tar xfz wiringPi-98bcb20.tar.gz
cd wiringPi-98bcb20
./build
```

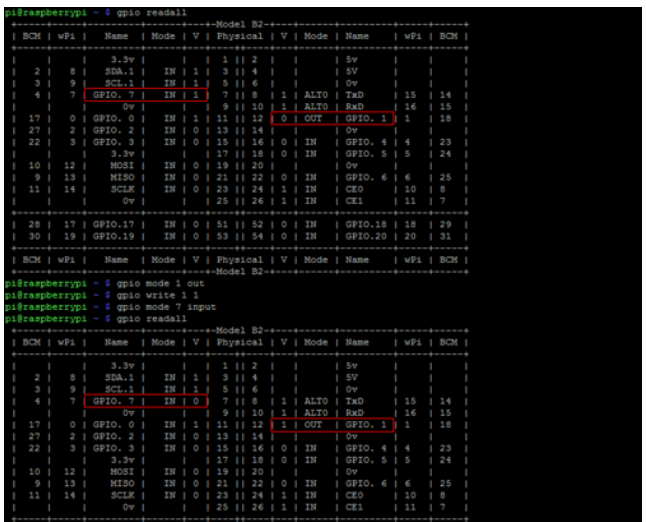
Note that the actual filename will be different – you will have to check the name and adjust accordingly.

Test wiringPi's installation

run the gpio command to check the installation:

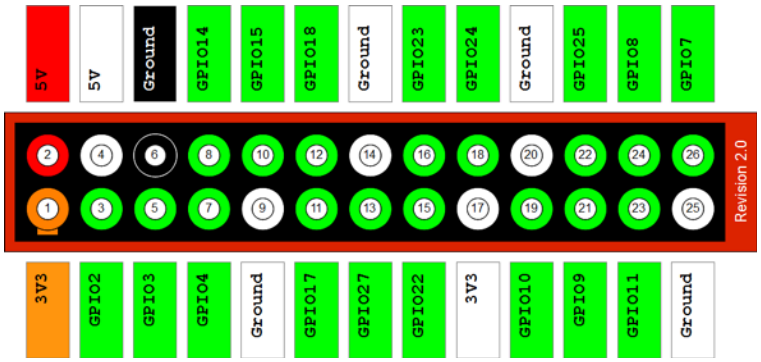
```
gpio -v
gpio readall
```

That should give you some confidence that it's working OK.



Examples

```
gpio mode 0 out
gpio write 0 1
```



This uses the ***wiringPi*** pin numbers to set pin 0 as an output and then sets the pin to a logic 1.
`gpio -g mode 0 in`
`gpio -g read 0`
This uses the BCM_GPIO pin numbering scheme and reads pin 0 (SDA0 on a Rev. 1 Raspberry Pi)

5) Installation du serveur web NPM:

Sources: https://github.com/sidwarkd/pimonitor http://revryl.com/2014/01/04/nodejs-raspberry-pi/

How to Use It
You can watch the *µ*Cast episode to see the program developed but if you just want to grab the source code and start messing around you can clone this repository right on the Raspberry Pi. Once the repo is present you need to ensure you have NodeJS installed. You can test this by running:
`node —version`

If it is not installed I've found the easiest way to get it is by following the instructions found at http://revryl.com/2014/01/04/nodejs-raspberry-pi/.
With NodeJS and NPM installed you need to install the dependencies by running the following command from within the **pimonitor** folder:
`npm install`

This can take several minutes on the Pi depending on connection speed. Once finished you can start the PiMonitor server running on port 3000 with the command:
`node bin/www`

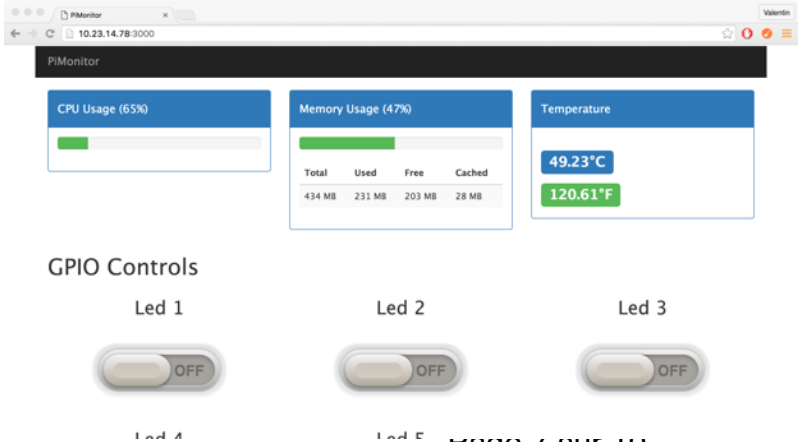
Following Along
If you wish to follow along in the video and create the project from scratch you will need one additional dependency and that's the ExpressJS generator which can be installed from the command line (after NodeJS and NPM are installed) with:
`sudo npm install -g express-generator`

```
pi@raspberrypi ~/pimonitor-master/views $
```

6) L'interface web:

La racine des fichiers pour l'accessibilité au serveur web se trouve:

Nous avons réalisé une interface qui agit directement sur les diodes du GPIO et qui visualise l'état du processeur et autres composants du Raspberry.



7) Installation des pilotes du dongle wifi:

Le dongle usb que nous avons utilisé n'a pas nécessité d'installation de pilote, cependant on a vérifié s'il était bien détecté par le Raspberry grâce à la commande lsusb.

```
pi@raspberrypi ~ $ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 148f:5370 Ralink Technology, Corp. RT5370 Wireless Adapte
```

8) Configuration du point d'accès wifi:

Turn any computer into a wireless access point with Hostapd
Linux hotspotDo you want to make a computer function as a WLAN base station, so that other computers can use as it as their wifi access point? This can easily be done using the open source software Hostapd and compatible wifi hardware.

This is a useful thing to do if computer acting as a firewall or as a server in the local network, and you want to avoid adding new appliances that all require their own space and cables in you already crowded server closet. Hostapd enables you to have full control of your WLAN access point and also enhances security. By using Hostapd the system will be completely in your control, every line of code can be audited and the source of all software can be verified and all software can be updated easily. It is quite common that active network devices like wifi access points are initially fairly secure small appliances with Linux inside, but over time their vendors don't provide timely security updates and local administrators don't care to install them via some clumsy firmware upgrade mechanism. With a proper Linux server admins can easily SSH into it and run upgrades using the familiar and trusted upgrade channels that Linux server distributions provide.

The first step in creating wireless base station with Hostapd is to make sure the WLAN hardware supports running in access point mode. Examples are listed in the hostapd documentation. A good place to shop for WLAN cards with excellent Linux drivers is thinkpenguin.com and in their product descriptions the WLAN card supported operation modes are nicely listed.

The next step is to install the software called Hostapd by Jouni Malinen and others. This is a very widely used software and it most likely is available in your Linux distribution by default. Many of the WLAN router appliances available actually are small Linux computers running hostapd inside, so by running hostapd on a proper Linux computer will give you at least all the features available in the WIFI routers, including advanced authentication and logging.

Our example commands are for Ubuntu 14.04. You need to have access to install hostapd and dnsmasq Dnsmasq is a small DNS/DHCP server which we'll use in this setup. To start simply run:

sudo apt-get install hostapd dnsmasq
After that you need to create and edit the configuration file:

```
zcat /usr/share/doc/hostapd/examples/hostapd.conf.gz | sudo tee -a /etc/hostapd/hostapd.conf
```

The configuration file /etc/hostapd/hostapd.conf is filled with configuration examples and documentation in comments. The relevant parts for a simple WPA2 protected 802.11g network with the SSID 'Example-WLAN' and password 'PASS' are:

```
interface=wlan0
ssid=Example-WLAN
hw_mode=g
wpa=2
```

```
wpa_passphrase=PASS
wpa_key_mgmt=WPA-PSK WPA-EAP WPA-PSK-SHA256 WPA-EAP-SHA256
```

Next you need to edit the network interfaces configuration to force the WLAN card to only run in the access point mode. Assuming that the access point network will use the address space 192.168.8.* the file /etc/network/interfaces should look something like this:

```
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto wlan0
iface wlan0 inet static
hostapd /etc/hostapd/hostapd.conf
address 192.168.8.1
netmask 255.255.255.0
```

Then we need to have a DNS relay and DHCP server on our wlan0 interface so the clients actually get a working Internet connection, and this can be accomplished by configuring dnsmasq. Like hostapd it also has a very verbose configuration file /etc/dnsmasq.conf, but the relevant parts look like this:

```
interface=lo,wlan0
no-dhcp-interface=lo
dhcp-range=192.168.8.20,192.168.8.254,255.255.255.0,12h
```

Next we need to make sure that the Linux kernel forwards traffic from our wireless network onto other destination networks. For that you need to edit the file /etc/sysctl.conf and make sure it has lines like this:

```
net.ipv4.ip_forward=1
```

We need to activate NAT in the built-in firewall of Linux to make sure the traffic going out uses the external address as its source address and thus can be routed back. It can be done for example by appending the following line to the file /etc/rc.local:

```
iptables -t nat -A POSTROUTING -s 192.168.8.0/24 ! -d 192.168.8.0/24  -j MASQUERADE
```

Some WLAN card hardware might have a virtual on/off switch. If you have such hardware you might need to also run rfkill to enable the hardware using a command like rfkill unblock 0.

The same computer also runs Network Manager (as for example Ubuntu does by default) you need to edit it's settings so that if won't interfere with the new wifi access point. Make sure file /etc/NetworkManager/NetworkManager.conf looks like this:

```
[main]
plugins=ifupdown,keyfile,ofono
dns=dnsmasq

[ifupdown]
managed=false
```

Now all configuration should be done. To be sure all changes take effect, finish by rebooting the computer.

If everything is working, a new WLAN network should be detected by other devices.
On the WLAN-server you'll see similar output from these commands:

```
$ iw wlan0 info
Interface wlan0
        ifindex 3
```

```
        type AP
        wiphy 0

$ iwconfig
wlan0     IEEE 802.11bgn  Mode:Master  Tx-Power=20 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Power Management:off

$ ifconfig
wlan0     Link encap:Ethernet  HWaddr f4:ec:38:de:c8:d2
          inet addr:192.168.8.1  Bcast:192.168.8.255  Mask:255.255.255.0
          inet6 addr: fe80::f6ec:38ff:fede:c8d2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5463040 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8166528 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:861148382 (861.1 MB)  TX bytes:9489973056 (9.4 GB)
```