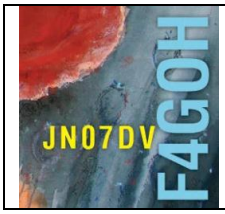
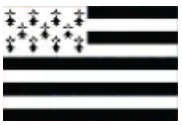


HAM Radio avec le Raspberry Pi
Partie 5



Prérequis : Découverte du Raspberry PI Partie 4

Sommaire :

Partie 5 : Commandes élémentaires, droits d'accès sur les fichiers et WSPR.

Version du 22/05/2020 V1.0

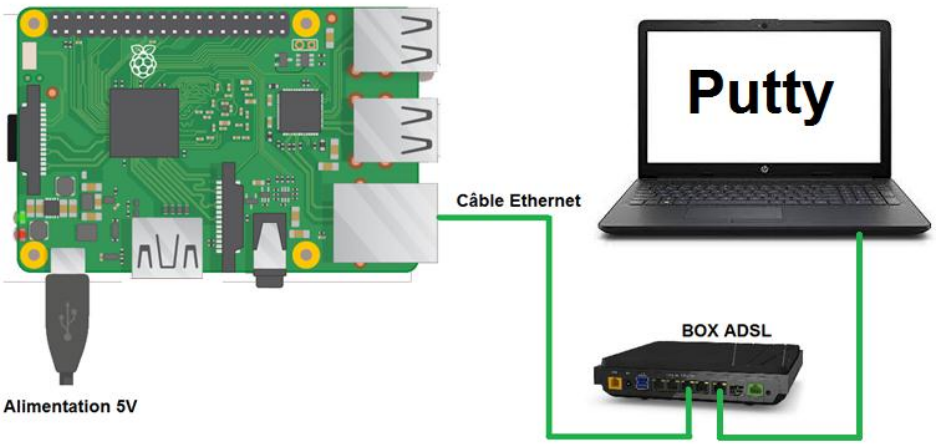
5.1 Introduction.

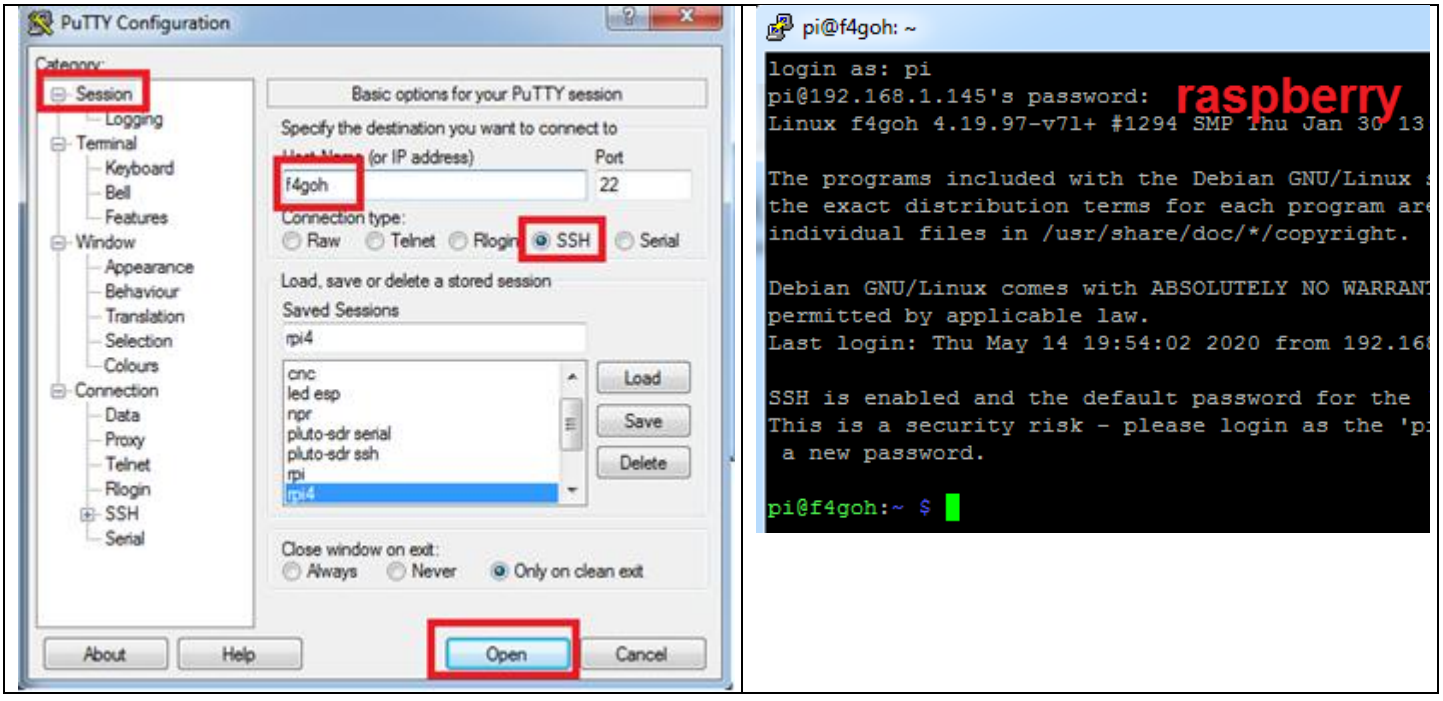
L'objectif de ce tutoriel est de pouvoir se débrouiller un minimum dans les lignes de commandes Linux, et donc :

- Accéder aux différents répertoires et les explorer ;
- Créer un répertoire, un fichier ;
- Créer, modifier un fichier ;
- Copier, déplacer ou supprimer un fichier,
- Changer les droits d'un fichier ;
- Rendre exécutable un fichier et savoir l'exécuter ;
- Passer en mode super utilisateur.

Ces commandes seront vues à travers des exemples concrets. Cela pourra vous servir de base pour la suite, l'intérêt étant de ne pas vous sentir démuni face à l'installation de projets autour de la clé de réception RTL-SDR.

Toutes les manipulations suivantes se feront en connexion SSh avec PuTTY. Plus besoin d'écran HDMI ni de clavier/souris.



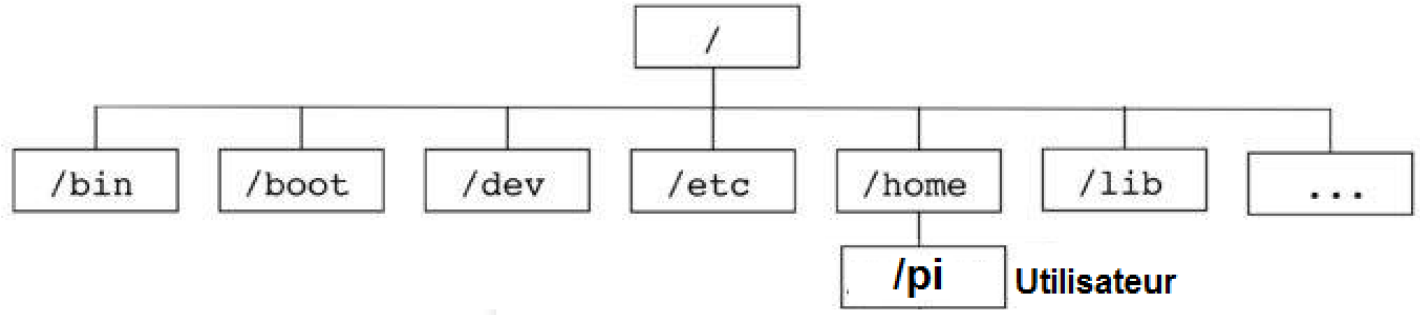


5.2 Arborescence Linux

Quand on parle d'arborescence, on parle de la hiérarchie et de la manière dont sont organisés les fichiers et les répertoires sur un système d'exploitation. L'organisation des dossiers sur un ordinateur est souvent comparée à un arbre. La base de votre arborescence est ce qu'on appelle la "racine".

Par exemple, sous Windows, la racine est souvent « C:\ », qui correspond au disque dur "C".

Sous Linux, la racine est « / ». Une suite de fichiers commençant par "/" démarre de la base de votre arborescence.




```
pi@f4goh:~ $ ls /
bin   dev  home  lost+found  mnt  proc  run   srv  tmp  var
boot  etc  lib   media       opt  root  sbin  sys  usr
```

Quand on installe Raspbian, l'utilisateur **pi** est le login par défaut. Il se situe toujours dans **home**. On peut ainsi créer d'autres utilisateurs.

- **/ =>** Racine, elle contient les répertoires principaux.
- **/bin =>** Exécutables essentiels au système, utilisables par tous les utilisateurs (ls pwd cp).
- **/boot =>** Fichiers permettant à Linux de démarrer.
- **/dev =>** Point d'entrée de tous les périphériques (disque dur, écran, partition, consoles TTY).
- **/etc =>** Contient les commandes et fichiers nécessaires à l'administrateur système.
- **/home =>** Répertoire personnel des utilisateurs.
- **/lib =>** Contient les bibliothèques partagées essentielles au système lors du démarrage.
- **/media =>** Contient les points de montage des partitions temporaires (clés USB, partitions de données).
- **/opt =>** Répertoire générique pour l'installation de programmes compilés par l'administrateur (logiciels spécifiques non présents dans les dépôts).
- **/proc =>** N'existe pas physiquement sur un disque, elle est créée par le noyau dans la mémoire. Cette partition permet de donner des informations sur le système.
- **/root =>** Répertoire personnel de l'administrateur.
- **/sbin =>** Contient les programmes du système essentiels utilisables par l'admin uniquement.
- **/srv =>** C'est un répertoire de données pour divers services (stockage des documents de comptes FTP ou pages de sites Web).
- **/tmp =>** Répertoire des fichiers temporaires.
- **/usr =>** Contient des programmes installés (**/usr/bin**) avec leurs librairies (**/usr/lib**).
- **/var =>** Contient les données variables (fichiers de log) mais parfois aussi les bases de données (/var/lib/mysql) et les pages de site Web (/var/www/html).

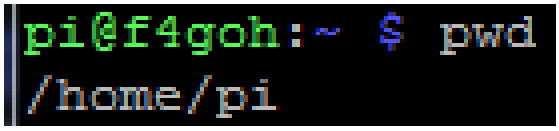
Dans la suite du tutoriel, on prendra toujours **pi** comme utilisateur.

5.3 Les commandes élémentaires

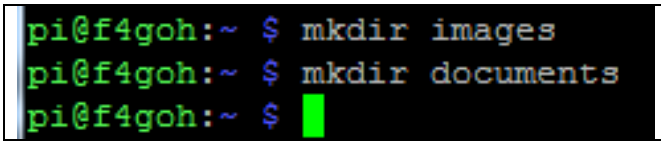
Les commandes Unix sont un mot ou une phrase qui indiquent une suite d'ordres d'exécution à l'ordinateur. Elles sont composées d'un nom, peuvent prendre une ou plusieurs options ainsi que des paramètres.

pwd permet d'afficher le chemin d'accès vers le répertoire où se situe l'utilisateur. Son nom signifie en anglais « print working directory ». Cette commande est très utile quand on ne sait plus dans quel répertoire on se trouve.

pwd

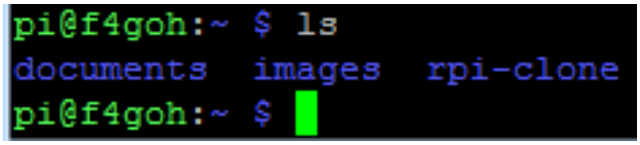
	Nom du répertoire courant :	pi
	Chemin absolu : (à partir de la racine)	/home/pi

mkdir permet de créer des répertoires. La commande est l'abréviation de make directory (termes anglais signifiant « créer répertoire »).

	Crée un répertoire images Crée un répertoire documents

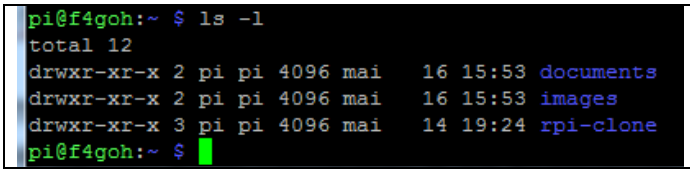
`ls` est une commande permettant de lister le contenu d'un répertoire (abréviation de list en anglais). On l'utilise sous la forme : `ls {options} {paramètres}`

ls

	Liste le contenu du répertoire courant (répertoire pi). On trouve les deux répertoires créés précédemment ainsi que l'utilitaire de sauvegarde rpi-clone installé dans la partie 4.

Néanmoins, cette commande liste les fichiers de mon répertoire courant sans aucune information complémentaire, il faut donc que je lui donne une option pour remédier à ce problème. Je vais lui donner l'option « `-l` » (Tiret Lima). A noter que toutes les options de toutes les commandes commencent toujours par un '-' (tiret du 6).

ls -l

	Liste le contenu du répertoire courant avec plus de détails.

cd (abréviation de l'anglais change directory) est une commande pour changer de répertoire courant.

cd documents

Remarque : En ligne de commande il faut utiliser au maximum l'auto-complétion avec la touche de tabulation TAB. Dans l'exemple ci-dessous, le but est de se déplacer dans le répertoire documents.

`pi@f4goh:~ $ cd d`	Je commence par taper cd, touche « espace », lettre d puis la touche de tabulation.	
`pi@f4goh:~ $ cd documents/` `pi@f4goh:~/documents $`	Le mot « documents » apparaît directement sans avoir besoin de le taper.	
	Il n'y a plus qu'à valider avec la touche entrée.	

pwd

`pi@f4goh:~/documents $ pwd`
`/home/pi/documents`
`pi@f4goh:~/documents $`
	Nom du répertoire courant :	documents

	Chemin absolu : (à partir de la racine)	/home/pi/documents

Pour revenir en arrière, on utilise toujours la commande cd, touche espace, point, point :

cd ..

`pi@f4goh:~/documents $ cd..`
`-bash: cd.. : commande introuvable`
`pi@f4goh:~/documents $ cd ..`
`pi@f4goh:~ $`
	J'ai oublié l'espace entre cd et les deux points !!!

	Voilà, cela fonctionne.

Je me déplace maintenant dans le répertoire images :

cd images

`pi@f4goh:~ $ cd images/` `pi@f4goh:~/images $`	cd i « touche TAB , suivi de la touche entrée ».

Maintenant je veux retourner directement dans le répertoire **documents**.

Première possibilité -> chemin relatif :

```
cd ../documents
```

`pi@f4goh:~/images $ cd ../documents/` `pi@f4goh:~/documents $`	Je retourne dans le répertoire précédent, puis je me déplace dans le répertoire documents en une seule ligne.

Deuxième possibilité -> chemin absolu :

```
cd /home/pi/documents
```

`pi@f4goh:~/images $ cd /home/pi/documents/` `pi@f4goh:~/documents $`	Je me déplace dans le répertoire documents en prenant comme référence la racine ***/***.

Nous allons maintenant créer un fichier texte dans le répertoire documents avec l'utilitaire **nano**.

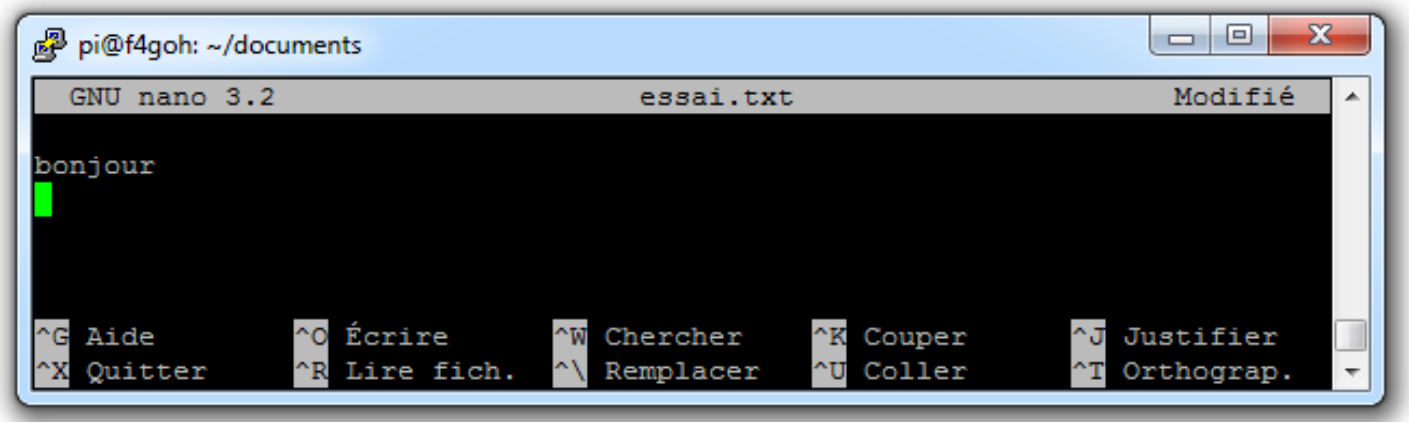
Nano est un éditeur de texte basique qui permet de modifier des fichiers de texte brut, sans mise en forme (gras, italique, souligné…).

Sous Windows, on dispose d'un éditeur de texte identique, c'est le Bloc-Notes.

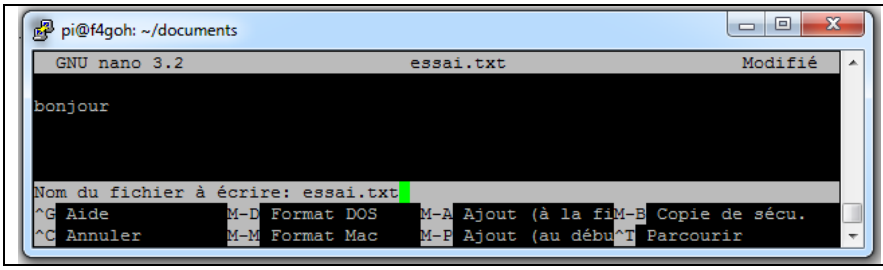
```
nano essai.txt
```

`pi@f4goh:~/documents $ nano essai.txt`	Crée et édite le fichier essai.txt

Tapez un texte quelconque, par exemple bonjour.



Enregistrez le fichier à l'aide des touches Ctrl + o.



Touche entrée pour valider l'enregistrement.

Sortez de l'éditeur à l'aide des touches Ctrl + x.

Listez le contenu du répertoire, le fichier **essai.txt** apparaît.

```
ls -l
```

```
pi@f4goh:~/documents $ ls -l
total 4
-rw-r--r-- 1 pi pi 10 mai   16 16:57 essai.txt
pi@f4goh:~/documents $
```

Liste le contenu du répertoire courant avec plus de détails.

`cp` (en référence au terme anglais copy, copier) est une commande permettant de copier des fichiers et répertoires.

Je désire copier le fichier essai.txt dans le répertoire images :

`pi@f4goh:~/documents $ cp essai.txt ../images`
`pi@f4goh:~/documents $ ls ../images/`
`essai.txt`
	Copie du fichier **essai.txt** en indiquant le répertoire de destination.
	Liste le contenu du répertoire **images** pour voir si le fichier est bien copié.

mv (en référence au terme anglais move, déplacer) permet de déplacer des fichiers et des répertoires. Il permet également de renommer un fichier ou un répertoire.

Je désire déplacer le fichier essai.txt dans le répertoire /home/pi en utilisant le chemin absolu.

`pi@f4goh:~/documents $ mv essai.txt /home/pi/`
`pi@f4goh:~/documents $ ls`
`pi@f4goh:~/documents $`
	Déplacement du fichier **essai.txt** en indiquant le répertoire de destination.
	Liste le contenu du répertoire **image** pour voir si le fichier est absent.

Je retourne maintenant dans le répertoire **/home/pi**

Pour revenir dans le répertoire de base « **/home/pi** » , on utilise toujours la commande cd, touche espace, point, point.

```
cd ..
```

OU

```
cd /home/pi
```

Ou revenir dans le répertoire par défaut

```
cd ~
```

rm (en référence au terme anglais remove, « supprimer ») est une commande permettant de supprimer des fichiers et répertoires.

Je supprime le répertoire documents et images

```
rm -r documents
rm -r images
```

`pi@f4goh:~ $ rm -r documents/`
`pi@f4goh:~ $ rm -r images/`
`pi@f4goh:~ $ ls`
`essai.txt  rpi-clone`
`pi@f4goh:~ $`
	Suppression des deux répertoires.
	Liste le contenu de **images** pour voir si les deux répertoires sont bien supprimés.

Remarque : l'option tiret -r est obligatoire pour un répertoire. Pour un fichier, il n'y a pas besoin de tiret -r. Le fichier **essai.txt** est utilisé pour la suite et il sera effacé plus tard.

5.4 Les droits d'accès sur les fichiers

Sur un système Linux, tous fichiers et tous dossiers disposent d'un propriétaire et de droits. Les utilisateurs sont réunis en trois groupes : **Propriétaire (u), groupe (g) et autre utilisateur (o)**. Ces groupes permettent de donner des droits différents à chaque utilisateur. Les différents droits pouvant être octroyés sont : **la lecture (r, 4), l'écriture (w, 2) et l'exécution (x, 1)**.

Les différents droits		
	Fichier	**Répertoire**
Lecture	Voir le contenu	Lister le contenu
Écriture	Modifier le contenu	Ajouter ou supprimer un élément
Exécution	Exécuter	Passer au travers

Observons les droits d'accès du fichier essai.txt :

```
pi@f4goh:~ $ ls -l essai.txt
-rw-r--r-- 1 pi pi 10 mai   16 16:57 essai.txt
pi@f4goh:~ $
```

- (Fichier)	**read**	**write**	**not execute**	**read**	**not write**	**not execute**	**read**	**not write**	**not Execute**
	r	w	-	r	-	-	r	-	-
d (Répertoire)	**4**	**2**	**0**	**4**	**0**	**0**	**4**	**0**	**0**
	Propriétaire (u) User			**Groupe (g)** Group			**Autre utilisateur (o)** Other		

Le fichier **essai.txt** est en accès lecture et écriture pour son propriétaire, c'est-à-dire l'utilisateur dont le login est **pi**. Par contre, le fichier est en lecture seule pour le groupe et les autres utilisateurs.

La modification des droits d'un fichier ou d'un répertoire se fait de deux façons : **absolue** ou **relative**.

Exemple de changement **absolu** :

```
chmod 660 essai.txt
```

```
pi@f4goh:~ $ ls -l essai.txt
-rw-r--r-- 1 pi pi 10 mai   16 16:57 essai.txt
pi@f4goh:~ $ chmod 660 essai.txt
pi@f4goh:~ $ ls -l essai.txt
-rw-rw---- 1 pi pi 10 mai   16 16:57 essai.txt
pi@f4goh:~ $
```

Chaque digit du nombre 660 est codé en octal.

- (Fichier)	read	write	not execute	read	write	not execute	Not read	Not write	not execute
	r	w	x	r	w	x	-	-	x
d (Répertoire)	4	2	0	4	2	0	0	0	0
	Propriétaire (u) Owner			**groupe (g)** Group			**Autre utilisateur (o)** Other		

Exemple de changement **relatif** : (on ajoute « **+** » les droits d'exécution au propriétaire)

```
chmod u+x essai.txt
```

```
pi@f4goh:~ $ ls -l essai.txt
-rw-rw---- 1 pi pi 10 mai   16 16:57 essai.txt
pi@f4goh:~ $ chmod u+x essai.txt
pi@f4goh:~ $ ls -l essai.txt
-rwxrw---- 1 pi pi 10 mai   16 16:57 essai.txt
pi@f4goh:~ $
```

Le fichier **essai** est devenu exécutable. Lors de l'utilisation de la commande **ls** celui-ci est colorié en vert (même s'il ne peut pas réellement s'exécuter, car c'est toujours un fichier texte, cela est pris juste comme exemple)

Remarque : il arrive fréquemment que l'on télécharge un fichier et que l'on ne puisse pas l'exécuter, car celui-ci ne possède pas les droits nécessaires. La commande **chmod** résout le problème.

Effacez le fichier essai.txt :

```
rm essai.txt
```

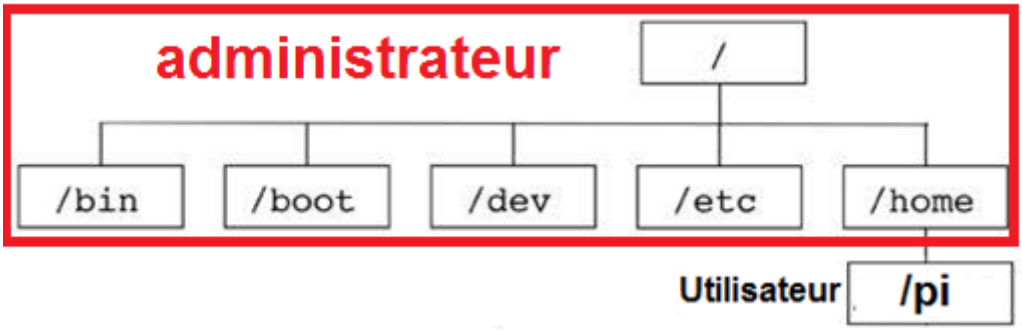
Exercice : à partir du tableau ci-dessous, donnez les commandes Linux associées afin de modifier les droits de **fichier.txt**.

- (Fichier)	read	write	execute	read	not write	execute	Not read	Not write	Not Execute
	r	w	x	r	-	x	-	-	-
d (Répertoire)	4	2	1	4	0	1	0	0	0
	Propriétaire (u) Owner			**Groupe (g)** Group			**Autre utilisateur (o)** Other		

chmod 750 fichier.txt

5.5 Le super utilisateur

Sur le Raspberry Pi, lorsque l'utilisateur utilise le login « **pi** », celui-ci n'a accès qu'au répertoire **/home/pi** et aux sous répertoires qu'il a lui-même créés. L'utilisateur pi n'a pas accès aux autres répertoires (par exemple /bin, /boot, etc..).

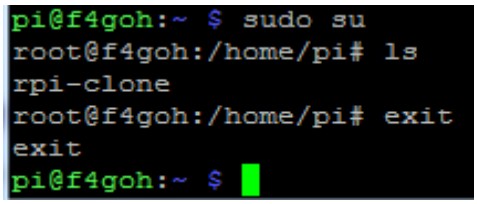


Pour modifier des fichiers, par exemple dans le répertoire /bin ou installer des nouveaux programmes, l'utilisateur « **pi** » devra systématiquement saisir la commande **sudo** .

sudo (abréviation de substitute user do) est une commande qui permet à un utilisateur de lancer une commande en tant qu'administrateur.

Il est cependant possible de passer facilement en mode super utilisateur, et il n'y aura plus besoin de taper systématiquement la commande **sudo**.

sudo su

	En mode super utilisateur ou mode root, il n'a plus de couleurs, et l'invite de commande se termine par # au lieu de **$**. Pour quitter le mode super utilisateur, tapez **exit**

exit

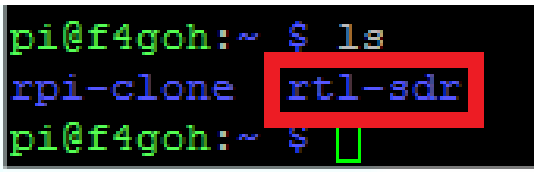
Attention : En mode super utilisateur, il faut vraiment savoir ce que l'on fait :

Une modification ou une suppression de fichier est irréversible.

5.6 Installation de la clé RTL-SDR

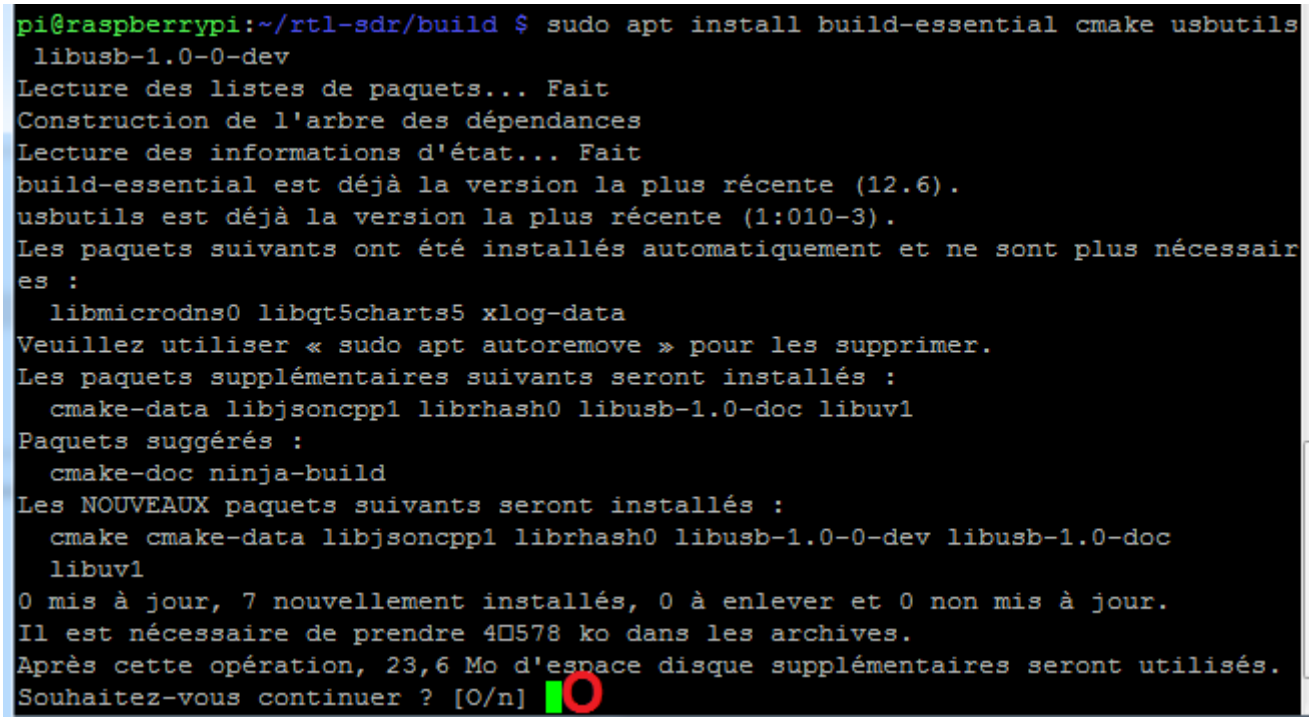
Récupérez les fichiers officiels :

```
git clone https://github.com/osmocom/rtl-sdr.git
```



Installez les bibliothèques :

```
sudo apt install build-essential cmake usbutils libusb-1.0-0-dev
```



Tapez ces commandes dans l'ordre :

```
cd rtl-sdr

mkdir build

cd build

cmake -DINSTALL_UDEV_RULES=ON -DDETACH_KERNEL_DRIVER=ON ../
```

```
pi@raspberrypi:~ $ cd rtl-sdr/
pi@raspberrypi:~/rtl-sdr $ mkdir build
pi@raspberrypi:~/rtl-sdr $ cd build/
pi@raspberrypi:~/rtl-sdr/build $ cmake -DINSTALL_UDEV_RULES=ON -DDETACH_KERNEL_D
RIVER=ON ../
-- The C compiler identification is GNU 8.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Build type not specified: defaulting to release.
-- Extracting version information from git describe...
-- Looking for pthread.h
-- Looking for pthread.h - found
```

sudo make install

```
pi@raspberrypi:~/rtl-sdr/build $ sudo make install
Scanning dependencies of target convenience_static
[  3%] Building C object src/CMakeFiles/convenience_static.dir/convenience/conve
nience.c.o
[  6%] Linking C static library libconvenience_static.a
[  6%] Built target convenience_static
Scanning dependencies of target rtlsdr
[  9%] Building C object src/CMakeFiles/rtlsdr.dir/librtlsdr.c.o
[ 12%] Building C object src/CMakeFiles/rtlsdr.dir/tuner_e4k.c.o
[ 15%] Building C object src/CMakeFiles/rtlsdr.dir/tuner_fc0012.c.o
[ 18%] Building C object src/CMakeFiles/rtlsdr.dir/tuner_fc0013.c.o
[ 21%] Building C object src/CMakeFiles/rtlsdr.dir/tuner_fc2580.c.o
[ 25%] Building C object src/CMakeFiles/rtlsdr.dir/tuner_r82xx.c.o
[ 28%] Linking C shared library librtlsdr.so
```

sudo ldconfig

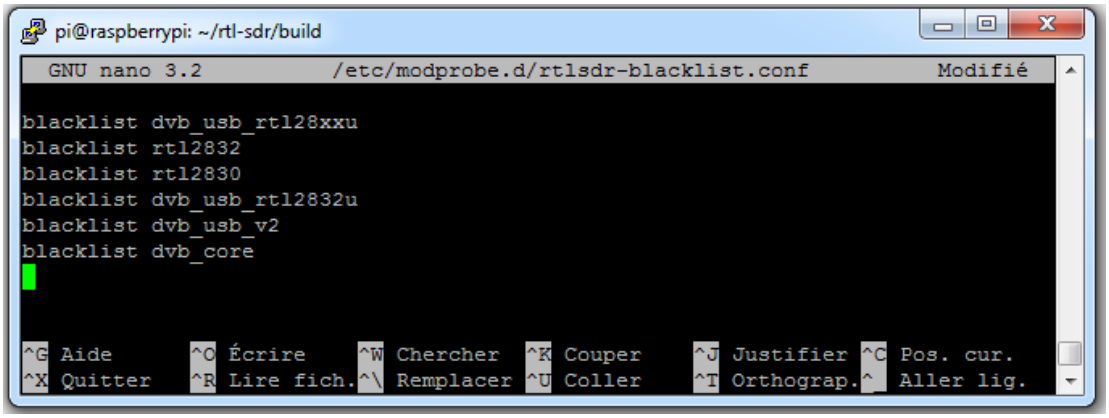
```
-- Set runtime path of "/usr/local/bin/rtl_fm" to ""
-- Installing: /usr/local/bin/rtl_eeprom
-- Set runtime path of "/usr/local/bin/rtl_eeprom" to ""
-- Installing: /usr/local/bin/rtl_adsb
-- Set runtime path of "/usr/local/bin/rtl_adsb" to ""
-- Installing: /usr/local/bin/rtl_power
-- Set runtime path of "/usr/local/bin/rtl_power" to ""
pi@raspberrypi:~/rtl-sdr/build $ sudo ldconfig
pi@raspberrypi:~/rtl-sdr/build $
```

sudo nano /etc/modprobe.d/rtlsdr-blacklist.conf

```
pi@raspberrypi:~/rtl-sdr/build $ sudo nano /etc/modprobe.d/rtlsdr-blacklist.conf
```

```
blacklist dvb_usb_rtl28xxu
blacklist rtl2832
blacklist rtl2830
blacklist dvb_usb_rtl2832u
blacklist dvb_usb_v2
blacklist dvb_core
```

Copiez-collez la liste ci-dessus dans l'éditeur **nano**. Rappel clic droit : pour coller le texte dans l'éditeur **nano**.

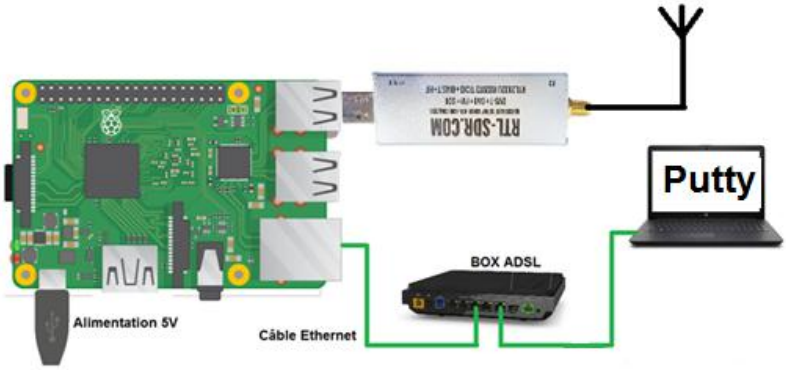


Enregistrez le fichier à l'aide des touches Ctrl + o, puis la touche entrée pour valider l'enregistrement.	**Sortez de l'éditeur à l'aide des touches Ctrl + x.**
	

Redémarrez le Raspberry PI : `pi@raspberrypi:~/rtl-sdr/build $ sudo reboot`

```
sudo reboot
```

Vérification : Branchez la clé RTL-SDR sur un port USB libre du Raspberry Pi.



Vérifiez la présence de la clé RTL-SDR :

Avec LXTerminal ou Putty, tapez la ligne de commande suivante :

```
lsusb
```

Localisez la clé RTL-SDR : **RTL238 DVB-T**

Cas du Raspberry Pi 3 :

```
pi@raspberrypi:~ $ lsusb
Bus 001 Device 004: ID 0bda:2838 Realtek Semiconductor Corp. RTL2838 DVB-T
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp. SMSC9512/9514 Fast
Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp. SMC9514 Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Cas du Raspberry Pi 4 :

```
pi@raspberrypi:~ $ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 0bda:2838 Realtek Semiconductor Corp. RTL2838 DVB-T
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Avec LXTerminal ou Putty, tapez la ligne de commande suivante : (attention : **tiret du bas**)

```
rtl_test
```

La clé RTL-SDR devrait être reconnue. Si ce n'est pas le cas, débranchez et rebranchez la clé et recommencez le test.

```
pi@raspberrypi:~ $ rtl_test
Found 1 device(s):
  0:  Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Supported gain values (29): 0.0 0.9 1.4 2.7 3.7 7.7 8.7 12.5 14.4 15.7 16.6 19.7
 20.7 22.9 25.4 28.0 29.7 32.8 33.8 36.4 37.2 38.6 40.2 42.1 43.4 43.9 44.5 48.0
 49.6
[R82XX] PLL not locked!
Sampling at 2048000 S/s.

Info: This tool will continuously read from the device, and report if
samples get lost. If you observe no further output, everything is fine.

Reading samples in async mode...
^CSignal caught, exiting!
ctrl+c

User cancel, exiting...
Samples per million lost (minimum): 0
pi@raspberrypi:~ $ ^C
```

Quittez le programme en appuyant simultanément sur les touches Ctrl et la touche c (Ctrl+c)

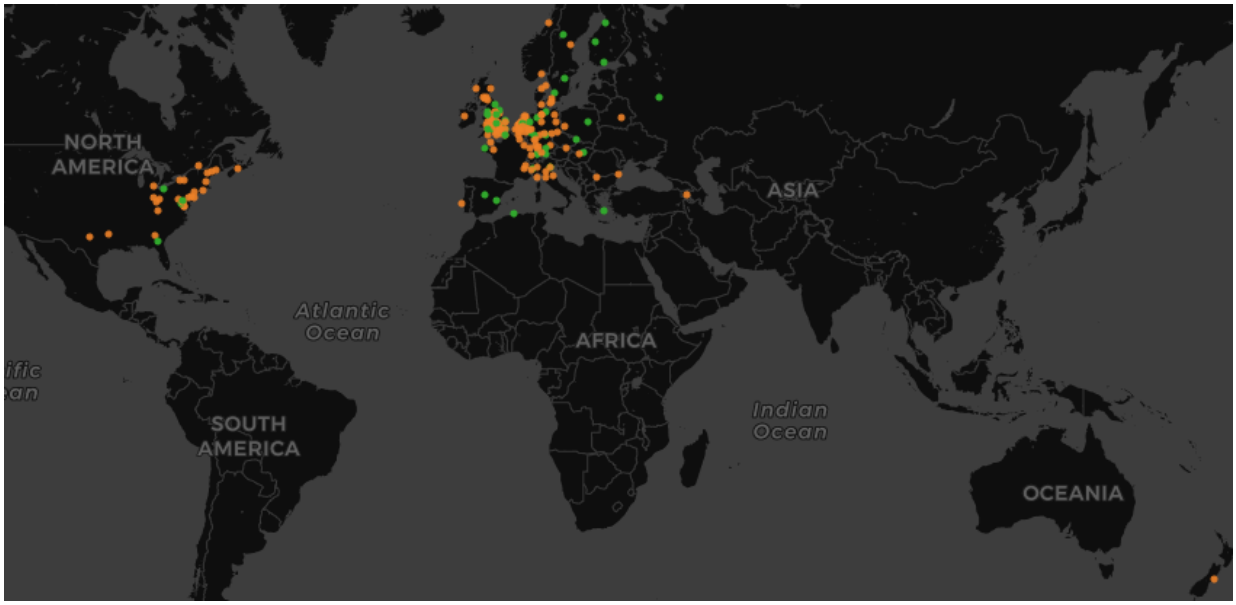
Si **`rtl_test`** affiche **en boucle** ce message `lost at least 112 bytes` réinstallez le programme en utilisant la méthode manuelle.

Installez des bibliothèques complémentaires (dépendances).

5.7 Décodage WSPR

L'objectif est de décoder les trames reçues en WSPR sur 40 mètres et de les reporter sur le serveur WSPR sans utiliser le logiciel WSJT-X.

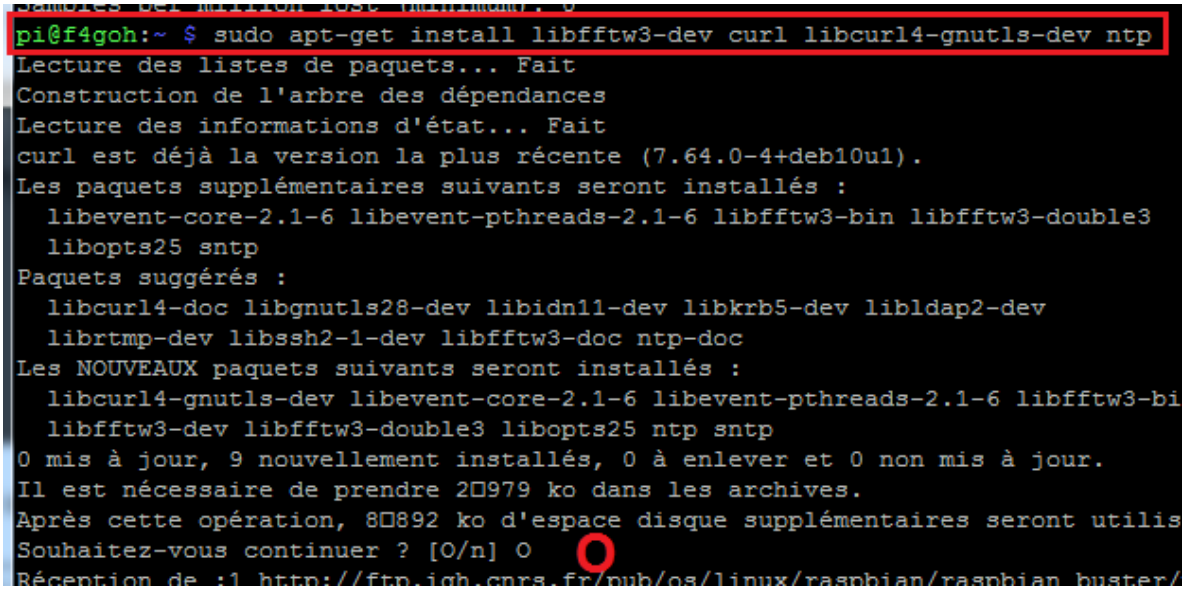
Ci-dessous, un exemple de réception sur une durée de 24 heures.



Prérequis : avoir précédemment installé les pilotes de la clé RTL-SDR .

```
sudo apt-get install libfftw3-dev curl libcurl4-gnutls-dev ntp
```

Il vaut mieux faire un copier-coller, plutôt que de tout retaper. (Voir partie 2, page 4)



Téléchargez avec git le programme de Guenael VA2GKA.

```
git clone https://github.com/Guenael/rtlsdr-wsprd
```

```
pi@f4goh:~ $ git clone https://github.com/Guenael/rtlsdr-wsprd
Clonage dans 'rtlsdr-wsprd'...
remote: Enumerating objects: 123, done.
remote: Total 123 (delta 0), reused 0 (delta 0), pack-reused 123
Réception d'objets: 100% (123/123), 217.19 KiB | 774.00 KiB/s, fait.
Résolution des deltas: 100% (73/73), fait.
pi@f4goh:~ $
```

```
cd rtlsdr-wsprd
```

Quels sont les fichiers du répertoire **rtlsdr-wsprd** ?

```
ls
```

Il n'y a pas de fichier exécutable sinon il serait de couleur verte :

```
pi@f4goh:~ $ cd rtlsdr-wsprd/
pi@f4goh:~/rtlsdr-wsprd $ ls
fano.c           nhash.c         rtlsdr_wsprd.h  wsprd_utils.c
fano.h           nhash.h         tab.c           wsprd_utils.h
Makefile         README.md       wsprd.c         wsprsim_utils.c
metric_tables.h  rtlsdr_wsprd.c  wsprd.h         wsprsim_utils.h
pi@f4goh:~/rtlsdr-wsprd $
```

Compilation du programme :

```
make
```

```
pi@f4goh:~/rtlsdr-wsprd $ make
gcc -Wall -O3 -ffast-math -std=gnu99 -c rtlsdr_wsprd.c -o rtlsdr_wsprd.o
rtlsdr_wsprd.c: In function 'main':
rtlsdr_wsprd.c:750:34: warning: '%02d' directive writing between 2 and 11 bytes int
o a region of size 7 [-Wformat-overflow=]
         sprintf(rx_options.date,"%02d%02d%02d", gtm->tm_year - 100, gtm->tm_mon +
1, gtm->tm_mday);
```

Un fichier exécutable est-il généré ?

```
ls
```

Oui, le fichier est en vert (rtlsdr_wsprd)

```
pi@f4goh:~/rtlsdr-wsprd $ ls
fano.c           nhash.c       rtlsdr_wsprd.c  wsprd.c        wsprd_utils.o
fano.h           nhash.h       rtlsdr_wsprd.h  wsprd.h        wsprsim_utils.c
fano.o           nhash.o       rtlsdr_wsprd.o  wsprd.o        wsprsim_utils.h
Makefile         README.md     tab.c           wsprd_utils.c  wsprsim_utils.o
metric_tables.h  rtlsdr_wsprd  tab.o           wsprd_utils.h
pi@f4goh:~/rtlsdr-wsprd $
```

Exécution du fichier : point, slash, rtlsdr_wsprd

```
./rtlsdr_wsprd
```

```
pi@f4goh:~/rtlsdr-wsprd $ ./rtlsdr_wsprd
rtlsdr_wsprd, a simple WSPR daemon for RTL receivers

Use:    rtlsdr_wsprd -f frequency -c callsign -l locator [options]
        -f dial frequency [(,k,M) Hz], check http://wsprnet.org/ for freq.
        -c your callsign (12 chars max)
        -l your locator grid (6 chars max)
Receiver extra options:
        -g gain [0-49] (default: 29)
        -a auto gain (default: off)
        -o frequency offset (default: 0)
        -p crystal correction factor (ppm) (default: 0)
        -u upconverter (default: 0, example: 125M)
        -d direct dampling [0,1,2] (default: 0, 1 for I input, 2 for Q input)
        -n max iterations (default: 0 = infinite loop)
        -i device index (in case of multiple receivers, default: 0)
Decoder extra options:
        -H use the hash table (could caught signal 11 on RPi)
        -Q quick mode, doesn't dig deep for weak signals
        -S single pass mode, no subtraction (same as original wsprd)
Example:
        rtlsdr_wsprd -f 144.489M -c A1XYZ -l AB12cd -g 29 -o -4200
pi@f4goh:~/rtlsdr-wsprd $
```

Fréquences usuelles :

Frequencies

USB dial (MHz): 0.136, 0.4742, 1.8366, 3.5686, 5.2872, 5364.7, 7.0386, 10.1387, 14.0956, 18.1046, 21.0946, 24.9246, 28.1246, 50.293, 70.091, 144.489, 432.300, 1296.500

Il faut passer des paramètres au programme pour l'exécuter avec un indicatif.

Exemple pour la bande des 40 mètres : fréquence 70386 MHz, indicatif f4goh, locator JN07dv, gain 29 db, offset en fréquence 10 hertz, direct sampling pour la HF (-d 2) Q input.

```
sudo ./rtlsdr_wsprd -f 7.0386M -c F4GOH -l JN07DV -g 29 -o -10 -d 2
```

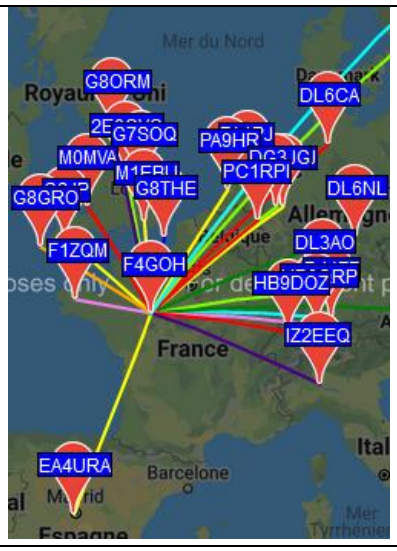
Résultat :

```
pi@f4goh: ~/rtlsdr-wsprd
pi@f4goh:~/rtlsdr-wsprd $ sudo ./rtlsdr_wsprd -f 7.0386M -c F4GOH -l JN07DV -g 2
9 -o -10 -d 2
Found 1 device(s):
  0:  Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Enabled direct sampling mode, input 2

Starting rtlsdr-wsprd (2020-05-22, 18:53z) -- Version 0.2
  Callsign      : F4GOH
  Locator       : JN07DV
  Dial freq.    : 7038599 Hz
  Real freq.    : 7038589 Hz
  PPM factor    : 0
  Gain          : 29 dB
Wait for time sync (start in 43 sec)

Spot : -11.27 -0.42   7.040109  0  <...> JN47VQ 37
Spot : -12.00 -2.09   7.040065  0  <...> JO31OL 23
Spot : -12.58 0.60   7.040140  1  G8THE JO00 20
Spot : -20.83 -0.31   7.040119  0  <...> JO50IP 23
Spot : -21.66 1.67   7.040058  0  G7SOQ IO92 23
Spot : 0.47 -1.49   7.040102  0  G2JP IO70 37
```



5.8 Décodage des stations météo environnantes

Prérequis : avoir précédemment installé les pilotes de la clé RTL-SDR.

Téléchargez avec **git** le programme de Guenael VA2GKA.

```
git clone https://github.com/merbanan/rtl_433.git
```

```
pi@f4goh:~ $ git clone https://github.com/merbanan/rtl_433.git
Clonage dans 'rtl_433'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 14841 (delta 3), reused 1 (delta 0), pack-reused 14827
Réception d'objets: 100% (14841/14841), 5.08 MiB | 1.47 MiB/s, fait.
Résolution des deltas: 100% (11480/11480), fait.
pi@f4goh:~ $ ls
rpi-clone  rtl_433  rtl-sdr  rtlsdr-wsprd
```

```
cd rtl_433/
mkdir build
cd build
cmake ../
```

```
pi@f4goh:~ $ ls
rpi-clone  rtl_433  rtl-sdr  rtlsdr-wsprd
pi@f4goh:~ $ cd rtl_433/
pi@f4goh:~/rtl_433 $ mkdir build
pi@f4goh:~/rtl_433 $ cd build/
pi@f4goh:~/rtl_433/build $ cmake ../
-- The C compiler identification is GNU 8.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
```

```
make
```

```
pi@f4goh:~/rtl_433/build $ make
Scanning dependencies of target r_433
[  1%] Building C object src/CMakeFiles/r_433.dir/abuf.c.o
[  1%] Building C object src/CMakeFiles/r_433.dir/am_analyze.c.o
[  2%] Building C object src/CMakeFiles/r_433.dir/baseband.c.o
[  2%] Building C object src/CMakeFiles/r_433.dir/bitbuffer.c.o
[  3%] Building C object src/CMakeFiles/r_433.dir/compat_paths.c.o
```

sudo make install

```
pi@f4goh:~/rtl_433/build $ sudo make install
[ 88%] Built target r_433
[ 89%] Built target rtl_433
[ 92%] Built target data
[ 93%] Built target style-check
[ 94%] Built target baseband-test
[ 95%] Built target test_bitbuffer
[ 96%] Built target data-test
[ 97%] Built target test_fileformat
[ 98%] Built target test_optparse
[100%] Built target test_util
Install the project...
```

Reliez une antenne VHF/UHF sur la clé RTL-SDR, puis exécutez le programme (attention : **tiret du bas**)

rtl_433

Voilà quelques stations météo de mon quartier.

```
pi@f4goh:~/rtl_433/build $ rtl_433
rtl_433 version 20.02-56-gd4ce64b branch master at 202005201829 inputs file rtl_tcp RTL-SDR
Use -h for usage help and see https://triq.org/ for documentation.
Trying conf file at "rtl_433.conf"...
Trying conf file at "/home/pi/.config/rtl_433/rtl_433.conf"...
Trying conf file at "/usr/local/etc/rtl_433/rtl_433.conf"...
Trying conf file at "/etc/rtl_433/rtl_433.conf"...
Registered 124 out of 152 device decoding protocols [ 1-4 8 11-12 15-17 19-21 23 25-26 29-36 38
-60 63 67-71 73-100 102-105 108-116 119 121 124-128 130-149 151-152 ]
Found Rafael Micro R820T tuner
Exact sample rate is: 250000.000414 Hz
[R82XX] PLL not locked!
Sample rate set to 250000 S/s.
Tuner gain set to Auto.
Tuned to 433.920MHz.

time      : 2020-05-22 21:48:13
model     : Nexus-T      House Code: 162
Channel   : 1            Battery   : 0            Temperature: 21.10 C

time      : 2020-05-22 21:48:17                   brand     : OS
model     : Oregon-THGR122N                       House Code: 26
Channel   : 1            Battery   : 1            Temperature: 22.00 C
Humidity  : 35 %

time      : 2020-05-22 21:48:17                   brand     : OS
model     : Oregon-THGR122N                       House Code: 26
Channel   : 1            Battery   : 1            Temperature: 22.00 C
Humidity  : 35 %
^CSignal caught, exiting!
pi@f4goh:~/rtl_433/build $
```

ctrl+c pour sortir

Prochaine partie : installation et configuration d'un serveur d'écoute SDR (openwebrx)

A suivre : https://github.com/projecthorus/radiosonde_auto_rx/wiki