

Partez à l'exploration des ondes grâce à la SDR !

Débuter dans le monde des radio-amateurs grâce à la Software Defined Radio.

 Difficulté Moyen

 Durée 1 heure(s)

 Catégories Électronique, Machines & Outils

 Coût 10 EUR (€)

Sommaire

Introduction

Video d'introduction

Étape 1 - Première connexion du récepteur

Étape 2 - Empêcher le chargement du mauvais pilote

Étape 3 - Installer de quoi utilise le récepteur

Étape 4 - Partez à l'exploration des ondes !

Étape 5 - Ecouter les avions !

Étape 6 - ajout d'une connectique et d'une antenne pour écouter de 100 khz a 14 mhz

Étape 7 - installez une interface web pour votre recepteur sdr rtl2832 sur votre serveur raspberry 2

Étape 8 - La suite...

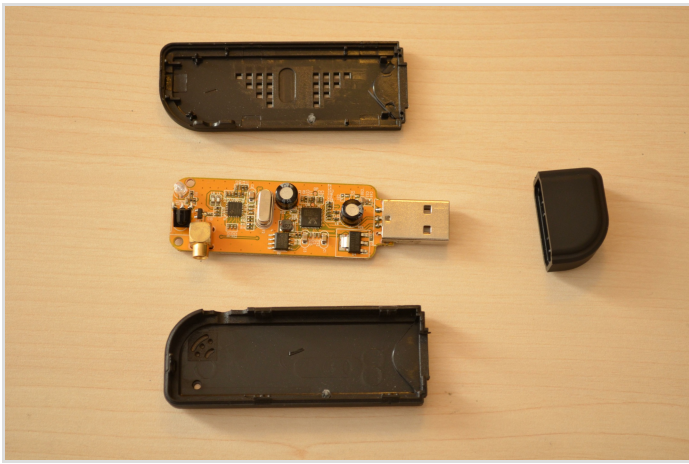
Notes et références

Commentaires

Introduction

Pour la plupart d'entre nous la radio est synonyme du "poste" qui se trouve dans la cuisine ou dans la voiture et qui nous permet d'écouter la musique, les infos... Ce dernier s'occupe de la réception puis la démodulation de signaux radios afin de les rendre audibles. Tous ce processus est mis en oeuvre de manière matérielle par les circuits qui le composent ce qui le rend dans bien des cas compliqués à reproduire. Mais ici on va intéresser à une autre manière moins commune de récupérer ces signaux radios: la SDR pour *Software Defined Radio*. Cette dernière se différencie par le fait que la majorité des circuits matériels sont remplacés par un logiciel installé sur un PC ce qui la rend bien plus simple à mettre en place. En effet il existe des périphériques USB destinés normalement à recevoir la télé TNT, qui permettent de numériser toutes les données radio qu'ils reçoivent puis de les transmettre à un ordinateur. Il s'avère que ces cartes sont majoritairement basées sur un RTL282U produit par le fabricant Realtek (c'est pour cela que l'on parle bien souvent de RTLSDR) qui est capable de recevoir bien plus que la télévision, s'il est utilisé avec les bons logiciels.

Nous verrons donc dans ce tutoriel comment détourner un de ces récepteurs pour partir à la conquête des ondes !!



Matériaux

- Un récepteur DVB-T: pour vous en procurer un, le plus simple reste une petite recherche avec les termes "dVB-t usb rtl" sur votre site d'enchère favori. Il est généralement vendu en kit avec une petite antenne qui suiffra bien dans un premier temps.

Pour l'étape 6:

- un fer à souder et de l'étain
- un long fil (20 a 30 mètres)
- un petit fil type: "connectique femelle pour Arduino"

Outils

- Un PC sous GNU/Linux (la mise en place de la SDR est aussi possible sous Windows et Mac, mais nous ne la décrierons pas dans ce tutoriel)



Étape 1 - Première connexion du récepteur

Juste après sa connexion à l'ordinateur utilisez la commande **dmesg** pour savoir ce qui a été détecté/chargé. Vous pourrez ainsi obtenir les identifiants de la clé (*VendorID, ProductID*).

Si le retour de la commande s'arrête après la ligne *SerialNumber* alors vous avez de la chance cela signifie que le système n'a pas tenté de prendre le périphérique en charge, sinon cela signifie que linux à belle est bien tenté que chargé un pilote et ce dernier ne fonctionne surement pas. Débranchez le récepteur et utilisez la commande **lsmod**. Vous verrez surement plusieurs modules *DVB* de chargés. Il va donc falloir corriger ce problème

```
214.040272 usb 1-3: new high-speed USB device number 5 using xhci_hcd
214.042518 usb 1-3: New USB device found, idVendor=0bda, idProduct=2838
214.042529 usb 1-3: New USB device strings: Mfr=1, Product=2, SerialNumber=3
214.042535 usb 1-3: Product: RTL2838U(HID)
214.042540 usb 1-3: Manufacturer: Realtek
214.042544 usb 1-3: SerialNumber: 00000001
214.100110 usb 1-3: dvb_usb_v2: found a 'Realtek RTL2832U reference design' in warm state
214.150230 usb 1-3: dvb_usb_v2: will pass the complete MPEG2 transport stream to the software demuxer
214.150250 DVB: registering new adapter (Realtek RTL2832U reference design)
214.170832 l2c l2c-0: Added multiplexed l2c bus 7
214.170842 rtl2832 0-0010: Realtek RTL2832 successfully attached
214.170863 usb 1-3: DVB: registering adapter 0 frontend 0 (Realtek RTL2832 (DVB-T))...
214.178074 r820t 7-0010: creating new instance
214.180930 r820t 7-0010: Rafael Micro r820t successfully identified
214.196776 rtl2832_sdr rtl2832_sdr.0.auto: Registered as sdrapi00
214.196782 rtl2832_sdr rtl2832_sdr.0.auto: Realtek RTL2832 SDR attached
214.196786 rtl2832_sdr rtl2832_sdr.0.auto: SDR API is still slightly experimental and functionality changes may follow
214.205413 Registered IR keymap rc-empty
214.205682 input: Realtek RTL2832U reference design as /devices/pci0000:00/0000:00:14.0/usb/l-3/rc/rc0/inputs
214.206029 rc0: Realtek RTL2832U reference design as /devices/pci0000:00/0000:00:14.0/usb/l-3/rc/rc0
214.213723 IR NEC protocol handler initialized
214.214002 IR RC6 protocol handler initialized
214.215474 IR RC5(s/z) protocol handler initialized
214.222202 IR JVC protocol handler initialized
214.226303 IR Sony protocol handler initialized
214.226377 IR Sharp protocol handler initialized
214.228206 usb 1-3: dvb_usb_v2: schedule remote query interval to 200 msecs
214.228416 IR SANYO protocol handler initialized
214.229600 input: MCE IR Keyboard/Mouse (dvb_usb_rtl28xxu) as /devices/virtual/input/input16
214.230017 IR MCE Keyboard/Mouse protocol handler initialized
214.231211 IR XMP protocol handler initialized
214.232393 lirc_dev: IR Remote Control driver registered, major 245
214.234079 rc rc0: lirc_dev: driver ir-lirc-codec (dvb_usb_rtl28xxu) registered at minor = 0
214.234084 IR lirc bridge handler initialized
214.237885 usb 1-3: dvb_usb_v2: 'Realtek RTL2832U reference design' successfully initialized and connected
214.239399 usbcore: registered new interface driver dvb_usb_rtl28xxu
nathieu@marvix-linux:~$
```

```
rtl2832_sdr 0-0010: 36864 0
r820t 7-0010: 28672 1
rtl2832 0-0010: 28672 1
l2c-0: 16384 1 rtl2832
dvb_usb_rtl28xxu 0-0010: 36864 1
dvb_usb_v2 0-0010: 36864 1 dvb_usb_rtl28xxu
dvb_core 122880 2 rtl2832,dvb_usb_v2
lirc_core 28672 14 ir_sharp_decoder,ir_xmp_decoder,lirc_dev,ir_lirc_codec,dvb_usb_rtl28xxu,ir_rc5_decoder,ir_nec_decoder,ir_sony_decoder,ir_mce_decoder,ir_jvc_decoder,dvb_usb_v2,ir_rc6_decoder,ir_sanyo_decoder
lircnec 88672 0
```

Étape 2 - Empêcher le chargement du mauvais pilote

Pour cela il faut interdire au système de charger automatiquement ces modules. Commencer donc par créer un fichier **sdrblacklist.conf** dans **/etc/modprobe.d/** avec la commande:

```
sudo nano /etc/modprobe.d/sdrblacklist.conf
```

et y ajouter les lignes suivantes:

```
blacklist dvb_usb_rtl28XXU
blacklist rtl2832
blacklist rtl2830
```

Pendant que vous y êtes porfitez en pour autoriser tous les utilisateurs à accéder au périphérique en créant un fichier **99_rtlsdr.rules** dans **/etc/udev/rules.d/** avec la commande:

```
sudo nano /etc/udev/rules.d/99_rtlsdr.rules
```

et ensuite ajouter sur une seule ligne:

```
SUBSYSTEM=="usb",ATTRS{idVendor}=="Obda",ATTRS{idProduct}=="2838",MODE="0666",GROUP="adm",SYMLINK+="rtl_sdr"
```

Redémarrez maintenant votre ordinateur. Branchez de nouveau le récepteur et vérifiez avec les commandes **dmesg** et **lsmod** que tout est en ordres.

Étape 3 - Installer de quoi utilise le récepteur

Maintenant que votre récepteur est accessible il va falloir installer les outils nécessaires à son utilisation.

On commence par installer les paquets qui nous seront utiles pour la suite; git, cmake (configurateur de sources), libusb-1.0-0.dev (gestion des périphériques usb) et build-essential (gestion de la compilation):

```
sudo apt-get install git cmake libusb-1.0-0.dev build-essential
```

Maintenant on peut passer à l'installation de RTLSDR qui nous permettra de nous attaquer véritablement aux ondes. On commence par créer un répertoire dans son dossier personnel puis on s'y rend:

```
mkdir SDR
cd SDR
```

On récupère les sources de RTLSDR:

```
git clone git://git.osmocom.org/rtl-sdr.git
```

On crée un nouveau dossier pour compiler proprement:

```
cd rtl-sdr
```

```
mkdir compil
```

```
cd compil
```

On configure ensuite les sources:

```
cmake -DCMAKE_INSTALL_PREFIX=PATH=/usr ..
```

Puis on provoque la construction:

```
make
```

Et on finit par installer tout ça;

```
sudo make install
```

Normalement si tout s'est déroulé sans accro les outils de RTLSDR sont bel et bien installés et fonctionnels !! Pour vous assurer d'essayer la commande **rtl_test -l** qui vous renvoie des infos sur votre récepteur.

```

mathieu@Marvix-Linux:~/SDR/rtl-sdr/build$ cmake -DCMAKE_INSTALL_PREFIX=PATH=/usr ..
-- The C compiler identification is GNU 5.3.1
-- 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...
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.1")
-- Checking for module 'libusb-1.0'
-- Found libusb-1.0, version 1.0.20
-- Looking for libusb_handle_events_timeout_completed
-- Looking for libusb_handle_events_timeout_completed - found
-- Looking for libusb_error_name
-- Looking for libusb_error_name - found
-- Looking for include file pthread.h - found
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Udev rules not being installed, install them with -DINSTALL_UDEV_RULES=ON
-- Building with kernel driver detaching disabled, use -DDETACH_KERNEL_DRIVER=ON to enable
-- Building for version: v0.5.3-12-ge3c0 / 0.5git
-- Using install prefix: /usr
-- Configuring done
-- Generating done
-- Build files have been written to: /home/mathieu/SDR/rtl-sdr/build

```

```

mathieu@Marvix-Linux:~/SDR/rtl-sdr/build$ make
Scanning dependencies of target rtlsdr_shared
[ 3%] Building C object src/CMakeFiles/rtlsdr_shared.dir/librtlsdr.c.o
[ 6%] Building C object src/CMakeFiles/rtlsdr_shared.dir/tuner_e4k.c.o
[ 10%] Building C object src/CMakeFiles/rtlsdr_shared.dir/tuner_fc0012.c.o
[ 13%] Building C object src/CMakeFiles/rtlsdr_shared.dir/tuner_fc0013.c.o
[ 16%] Building C object src/CMakeFiles/rtlsdr_shared.dir/tuner_fc2580.c.o
[ 20%] Building C object src/CMakeFiles/rtlsdr_shared.dir/tuner_r82xx.c.o
[ 23%] Linking C shared library librtlsdr.so
[ 23%] Built target rtlsdr_shared
Scanning dependencies of target convenience_static
[ 26%] Building C object src/CMakeFiles/convenience_static.dir/convenience/convenience.c.o
[ 30%] Linking C static library libconvenience_static.a
[ 30%] Built target convenience_static
Scanning dependencies of target rtl_sdr
[ 33%] Building C object src/CMakeFiles/rtl_sdr.dir/rtl_sdr.c.o
[ 36%] Linking C executable rtl_sdr
[ 36%] Built target rtl_sdr
Scanning dependencies of target rtlsdr_static
[ 40%] Building C object src/CMakeFiles/rtlsdr_static.dir/librtlsdr.c.o
[ 43%] Building C object src/CMakeFiles/rtlsdr_static.dir/tuner_e4k.c.o
[ 46%] Building C object src/CMakeFiles/rtlsdr_static.dir/tuner_fc0012.c.o
[ 50%] Building C object src/CMakeFiles/rtlsdr_static.dir/tuner_fc0013.c.o
[ 53%] Building C object src/CMakeFiles/rtlsdr_static.dir/tuner_fc2580.c.o
[ 56%] Building C object src/CMakeFiles/rtlsdr_static.dir/tuner_r82xx.c.o
[ 60%] Linking C static library librtlsdr.a
[ 60%] Built target rtlsdr_static
Scanning dependencies of target rtl_power
[ 63%] Building C object src/CMakeFiles/rtl_power.dir/rtl_power.c.o
[ 66%] Linking C executable rtl_power
[ 66%] Built target rtl_power
Scanning dependencies of target rtl_tcp
[ 70%] Building C object src/CMakeFiles/rtl_tcp.dir/rtl_tcp.c.o
[ 73%] Linking C executable rtl_tcp
[ 73%] Built target rtl_tcp
Scanning dependencies of target rtl_fm
[ 76%] Building C object src/CMakeFiles/rtl_fm.dir/rtl_fm.c.o
[ 80%] Linking C executable rtl_fm
[ 80%] Built target rtl_fm
Scanning dependencies of target rtl_test
[ 83%] Building C object src/CMakeFiles/rtl_test.dir/rtl_test.c.o
[ 86%] Linking C executable rtl_test
[ 86%] Built target rtl_test
Scanning dependencies of target rtl_adsb
[ 90%] Building C object src/CMakeFiles/rtl_adsb.dir/rtl_adsb.c.o
[ 93%] Linking C executable rtl_adsb
[ 93%] Built target rtl_adsb
Scanning dependencies of target rtl_eepron
[ 96%] Building C object src/CMakeFiles/rtl_eepron.dir/rtl_eepron.c.o
[ 100%] Linking C executable rtl_eepron
[ 100%] Built target rtl_eepron

```

```

mathieu@Marvix-Linux:~/SDR/rtl-sdr/build$ sudo make install
[ 23%] Built target rtlsdr_shared
[ 30%] Built target convenience_static
[ 36%] Built target rtl_sdr
[ 60%] Built target rtlsdr_static
[ 66%] Built target rtl_power
[ 73%] Built target rtl_tcp
[ 80%] Built target rtl_fm
[ 86%] Built target rtl_test
[ 93%] Built target rtl_adsb
[100%] Built target rtl_eepron
Install the project...
-- Install configuration: "Release"
-- Installing: /usr/lib/pkgconfig/librtlsdr.pc
-- Installing: /usr/include/rtl-sdr.h
-- Installing: /usr/include/rtl-sdr_export.h
-- Installing: /usr/lib/librtlsdr.so.0.5git
-- Installing: /usr/lib/librtlsdr.so.0
-- Installing: /usr/lib/librtlsdr.so
-- Installing: /usr/lib/librtlsdr.a
-- Installing: /usr/bin/rtl_sdr
-- Set runtime path of "/usr/bin/rtl_sdr" to ""
-- Installing: /usr/bin/rtl_tcp
-- Set runtime path of "/usr/bin/rtl_tcp" to ""
-- Installing: /usr/bin/rtl_test
-- Set runtime path of "/usr/bin/rtl_test" to ""
-- Installing: /usr/bin/rtl_fm
-- Set runtime path of "/usr/bin/rtl_fm" to ""
-- Installing: /usr/bin/rtl_eepron
-- Set runtime path of "/usr/bin/rtl_eepron" to ""
-- Installing: /usr/bin/rtl_adsb
-- Set runtime path of "/usr/bin/rtl_adsb" to ""
-- Installing: /usr/bin/rtl_power
-- Set runtime path of "/usr/bin/rtl_power" to ""

```

Étape 4 - Partez à l'exploration des ondes !

Pour pouvoir écouter les ondes nous allons utiliser un logiciel de SDR appelé Gqrx. Commencez son installation en ajoutant les dépôts suivants puis actualisez la liste des paquets:

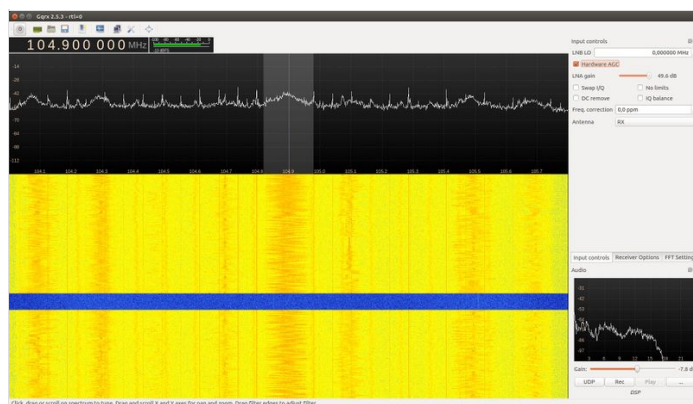
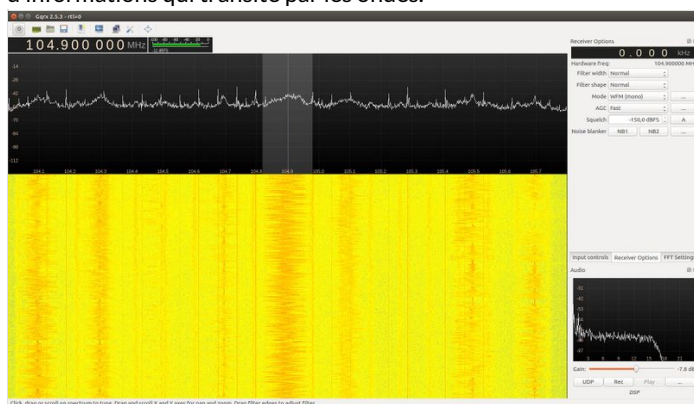
```
sudo add-apt-repository -y ppa:ettusresearch/uhd
sudo add-apt-repository -y ppa:myriadrf/drivers
sudo add-apt-repository -y ppa:myriadrf/gnuradio
sudo add-apt-repository -y ppa:gqrx/gqrx-sdr
sudo apt-get update
```

Et enfin installer le avec:

```
sudo apt-get install gqrx-sdr
```

Lancez ensuite le logiciel, sélectionnez votre récepteur et laissez les autres réglages sur les valeurs par défaut. Pour votre première écoute le mieux est de tester sur station FM favorite. Il vous faudra donc tout d'abord entrer sa fréquence en haut à gauche ou en cliquant sur le graphique. Puis cocher *Hardware AGC* dans l'onglet *Input Control* et enfin régler la démodulation (*mode*) sur *WFM* (*stereo ou mono*) dans l'onglet *Receiver Option*.

En vous promenant sur les différentes fréquences et en jouant sur les différents types de démodulation, vous pourrez ainsi voir la foule d'informations qui transite par les ondes.



Étape 5 - Ecouter les avions !

Dans cette dernière partie nous allons voir un exemple étonnant de ce que permet de faire la SDR.

Si vous êtes curieux comme moi vous vous demandez peut-être lorsque vous voyez passer un de ces bijoux de technologie que sont les avions passer au-dessus de votre tête où il va? à quelle vitesse il vole? à quelle altitude?... Et bien la SDR permet de répondre à ces questions. En effet les avions transmettent en permanence ces informations par l'intermédiaire de leur transpondeur sur la fréquence 1090MHz (système ADS-B pour *Automatic Dependant Surveillance-Broadcast*). Et ces signaux sont émis en continue et en claire, il nous suffit donc de les recevoir puis de les décoder avec un logiciel développé à cet effet: **dump1090**.

Pour l'installer rendez-vous sur cette page et télécharger le fichier .Zip, puis décompressez-le dans votre dossier personnel:

<https://github.com/antirez/dump1090>

Rendez-vous ensuite dans le répertoire de dump1090-master puis compilez le programme:

```
cd dump1090-master
make
```

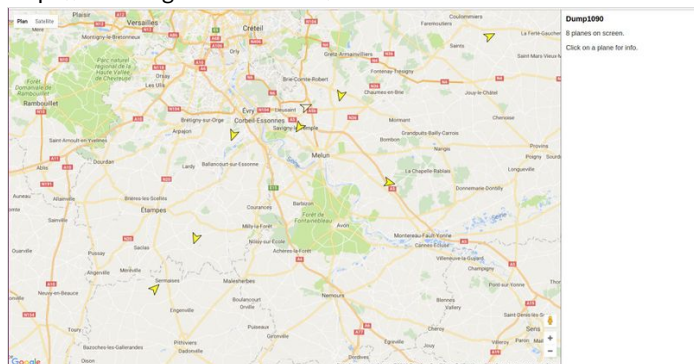
Exécuter ensuite le programme avec la commande (les options permettent d'utiliser des mètres, de décoder en temps réel les données et d'activer les fonctionnalités réseaux):

```
./dump1090 --metric --interactive --net
```

Vous obtenez alors des informations sur les différents avions qui volent à proximité de vous. Vous pouvez également afficher ces informations sur une carte en vous rendant sur l'adresse locale: <http://127.0.0.1:8080>

Profitez-en aussi pour aller jeter un coup d'œil au site [Flightradar24](https://www.flightradar24.com/) qui collecte les infos reçues via l'ADS-B dans le monde entier:

<https://www.flightradar24.com/>



Hex	Flight	Altitude	Speed	Lat	Lon	Track	Messages Seen	
484135		12489	0	0.000	0.000	0	34	0 sec
344292	VLG2KT	2482	572	48.621	2.551	67	435	1 sec
4ca2a9		10509	829	48.501	1.947	22	43	36 sec
4066e5		12184	0	0.000	0.000	0	31	52 sec
393d85	AF248WC	3746	529	48.477	2.379	173	122	58 sec
02001b	RAM225R	11880	796	48.403	2.330	200	702	12 sec
405632	EZY41CE	11567	818	48.830	2.966	327	313	59 sec

Étape 6 - ajout d'une connectique et d'une antenne pour écouter de 100 khz à 14 mhz

- Vous devez souder à l'étain un petit fil "connectique femelle" derrière la plaque, du côté où il n'y a pas les composants.

⚠ toute soudure s'effectue sur un matériel déconnecté électriquement

⚠ exactement au même emplacement que sur la photo, ou est souder le **fil vert**

- pour pouvoir par la suite insérer un cable très long (de 20 à 30 mètres) qui nous servira d'antenne

- Sous Ubuntu lancer le programme GQRX

cliquez sur l'icône **Configure I/O devices**

- à la ligne **device** choisissez:

other

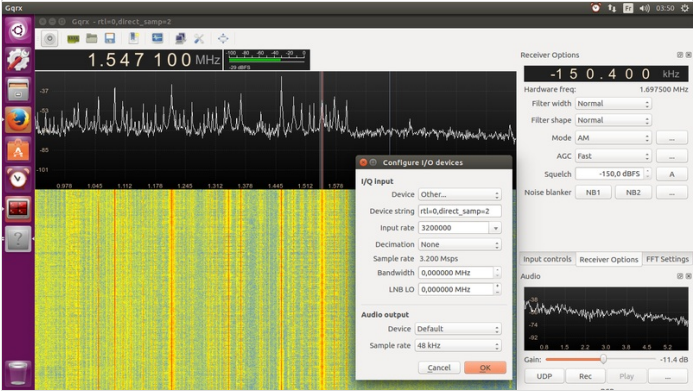
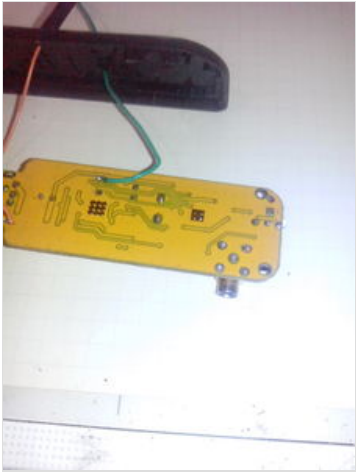
- puis à la ligne **device string** écrivez:

rtl=0,direct_samp=2

cliquez sur ok

- fermez le programme et redémarrez le

💡 Vous pourrez ainsi recevoir le morse, les images-météo wefax, écouter certains radioamateurs français à 7.14 mhz



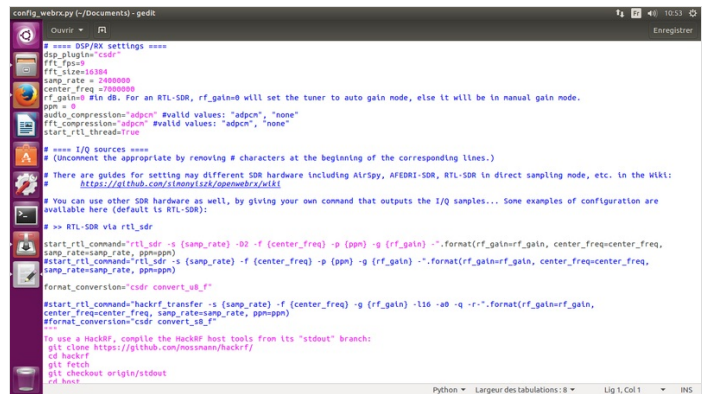
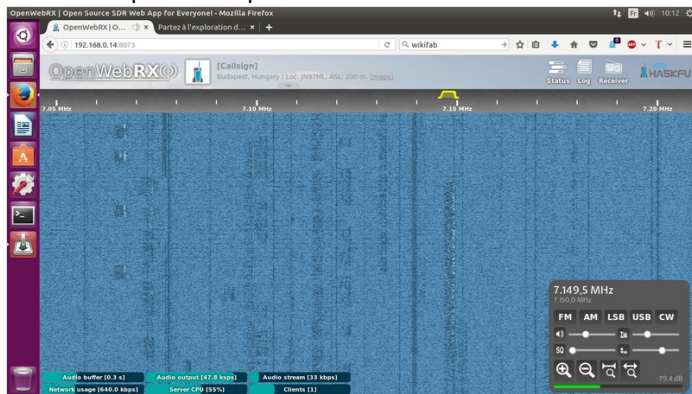
Étape 7 - installez une interface web pour votre recepteur sdr rtl2832 sur votre serveur raspberry 2

suivez les commande d'écrite dans ce lien : <http://blog.sdr.hu/2015/06/30/quick-setup-openwebbrx.html>

si vous avez modifier votre rtl2832 (étape 6) suivez les commande d'écrite dans ce lien :

<https://github.com/simonyiszk/openwebbrx/wiki/Using-RTLSDR-in-direct-sampling-mode-with-OpenWebRX>

sources : <http://sdr.hu/openwebbrx>



Étape 8 - La suite...

Voilà, mes connaissances dans le domaine étant limitées, cette petite initiation au monde de la SDR se termine. Au travers de ce tutoriel vous n'avez put découvrir qu'une infime partie de ce domaine.

Continuez à scrutez les ondes vous aurez peut-être des surprises (certain arrive à entendre les voix de l'espace...) et intéressez vous aussi à *la science des antennes* qui vous permettra de capter toujours plus de choses. (si quelqu'un veut faire des tutos dessus je suis preneur ☺)

Edit: Merci à Tartempion pour son astuce permettant d'augmenter la plage de fréquence du récepteur.



Notes et références

Logiciel Gqrx: <http://gqrx.dk/>

Dump-1090: <https://github.com/antirez/dump1090>

Installation: <http://nobru54.blogspot.fr/2014/01/sdr-gqrx-analyseur-de-spectre-sous.html>

ADS-B: <http://www.framboise314.fr/un-raspberry-pi-pour-suivre-les-avions-sur-flightradar24-2/>

identifiez les signaux radio: http://www.sigidwiki.com/wiki/Signal_Identification_Guide

installez un serveur raspberry 2 sdr: <http://sdr.hu/openwebbrx> + config pour l'étape 6 et 7 :

<https://github.com/simonyiszk/openwebbrx/wiki/Using-RTLSDR-in-direct-sampling-mode-with-OpenWebRX>

logiciel de décodage morse, wexaf etc: <http://doc.ubuntu-fr.org/fldigi> + sstv: <https://doc.ubuntu-fr.org/qsstv>

+ Magazine Hackable n°2 Septembre-Octobre 2014