UTILISATION ----------

Demarage du path :

$ cd /home/eusapia/Documents/
$ python3 Ether.py

Accordage Cordes :

- aller dans l'onglet «Cordes »
- decocher « [ ] Radio on Cordes »
- pour chaque corde et chaque harmonique cocher case « amplitude x » et ajuster la fréquence pour avoir le maximum d'amplitude visuellement sur les cordes.

les fréquences sont enregistrées dans /home/eusapia/Documents/Gnu Radio/Harmonic_frequencies
faire des sauvegardes régulières de ce fichier

Bookmarks des fréquences Radios

Création :

Dans GQRX

Export :

home/.config/gqrx/bookmarks.csv

INSTALLATION ----------

Installation Ubuntu

sudo apt-get update

Installation Carte Son

elle est directement reconnue, pas besoin d'installation

Installation Geany pour avoir editeur python

sudo apt install geany

Instalation Gnu Radio companion :

Avec GQRX

gnuradio-companion est déjà installé avec GQRX avec en plus ce qu'il faut pour le recepteur radio
https://gqrx.dk/download/install-ubuntu

```
sudo add-apt-repository -y ppa:bladerf/bladerf
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

sudo apt-get install gqrx-sdr
```

par contre il faut corriger un bug de gnuradio lorsque l'on veut utiliser des variable sauvegardées

créer un fichier /usr/lib/python3/dist-packages/ConfigParser.py avec dedans :
from configparser import *

avec binaries : **n'installe pas ce qu'il faut pourle recepteur**

```
sudo apt install gnuradio

avec PyBOMBS : ne marche pas
sudo apt install python3-pip
sudo -H pip3 install PyBOMBS
pybombs auto-config
pybombs recipes add-defaults
pybombs prefix init ~/gnuradio -R gnuradio-default

genère erreure : PyBOMBS.install_manager - ERROR - Package has no
install method: python

https://github.com/gnuradio/pybombs/issues/485
```

DOCUMENTATION GNU RADIO COMPAGNON ----------

Automation et code Python

Exemples :

https://www.youtube.com/watch?v=LiAcNm9Rru8

prob signal : permet de recuperer volume d'un flux audio en exposant en python une fonction level()

ID : probSign

fonction prob : fait periodiquemnet appel à cette fonction level

ID : fun_prob

Block ID : probSign

Fonction Name : level

Qt GUI Label : permet d'afficher avec mise à jour automatique !

Defalut Value : fun_prob

https://www.youtube.com/watch?v=9bhmF7WRvMQ

assez bizarre comme structure...

Python Module

ID : python_mod

```
f1 = 400000000
f2 = 470000000
f = f1
step =1000000
def sweeper(prob_lvl ) :
    global f1,f2,f,step
    if prob_lvl :
        f+= step
if f >= f2 :
    f=f1
return f
```

variable :

La variable semble etre mise à jour automatiquement quand fun_prob mis à jour !?

ID : freq

value:python_mod.sweeper(fun_prob)

Prob Signal

Function Prob

https://wiki.gnuradio.org/index.php/Function_Probe

crée un thread qui boucle sur appel de la fonction , set de la variable et sleep
il appelle la function self.block_ID.Function_Name(Function_Ags)
problème on ne veut pas du self quand on veut appeler une fonction dans un Python Module.
Du coup il faut créer un variable qui va wrapper le module dans un attribut :
ID : epy_module
Value : epy_module_0
→ va generer le code self.epy_module = epy_module_0
https://youtu.be/9bhmF7WRvMQ?t=126

Python Module

https://wiki.gnuradio.org/index.php/Python_Module
bug de gnu radio companion : l'ID d'un Python Module n'est pas visible dans les paramètres → on ne peut le modifier. Mais dans le fichier grc on peut voir son ID qui est « epy_module_»

Embedded Python Block

https://wiki.gnuradio.org/index.php/Embedded_Python_Block
https://www.gnuradio.org/doc/doxygen-3.7.5/page_python_blocks.html
https://wiki.gnuradio.org/index.php/Types_of_Blocks
https://wiki.gnuradio.org/index.php/Guided_Tutorial_GNU_Radio_in_Python
https://www.gnuradio.org/doc/doxygen/classgr_1_1basic__block.html
pour les vecteurs :
https://lists.gnu.org/archive/html/discuss-gnuradio/2015-10/msg00374.html

Out of the Tree Object :

https://www.cgran.org/
https://wiki.gnuradio.org/index.php/OutOfTreeModules
https://github.com/gnuradio/gnuradio/blob/master/gr-blocks/python/blocks/qa_block_gateway.py

modification d'un objet avec interface graphique :

/usr/share/gnuradio/grc/blocks/qtgui_chooser.block.yml

DEVELOPPEMENT DU PATCH ----------

Septembre 2020

Pb accordage:il faut coler les cordes en haut du transducteur
Pb transducteurs qui crament : essayer en diminuant le volume
Pb blanc dans les transitions : à l'aproche d'une radio le volume se baisse
→ virer la compression ?
Je suis passé de 5 à 2 pour le gain max dans la compression
j'ai modifié les temps d'attaque et release
Pb transduteur qui font beaucoup de bruit dans le bruit entre les fréquences radios
→ addoucir les enveloppes generée à partir de la FFT :
j'ai fait un filtre pass_bas du premier ordre sur les coefficients des harmoniques
puis une interpolation spline sur 8 coefficiants

Accorder les cordes ?

Manuel :

À la main on voit des battements (résonne à fond , puis moins , puis re-résonne) , on ne sait pas si on est au dessus ou en dessous …
comment sauvegarde les valeurs ?

Problème import ConfigParser au lieu de import configparser :
version de gnu radio : demarer gnuradio-companion → help → about → 3.8.1.0
https://github.com/gnuradio/gnuradio/pull/3429
https://github.com/gnuradio/gnuradio/pull/3429/commits/5267d2653bb5eda9e08142b620ca84a27a836e12

https://github.com/gnuradio/gnuradio/issues/2782

Automatique

https://www.reddit.com/r/GNURadio/comments/2m2xzt/audio_pitch_estimation/
FFT ?
Fast Autocorrelation

Radio

Radio AM ?

24 Mhz à 1766 Mhz
la clef RTL-SDR n'a pas l'air de pouvoir descendre en desous de 24 MHz (à moins que ce soit GQRX qui ne le permet pas) . on ne peut donc pas capter les radio AM qui sont entre
En Europe, les radios publiques et commerciales émettent en AM en ondes longues (OL ou GO ou LW de 150 kHz - 281 kHz), moyennes (OM ou PO ou MW de 520 kHz - 1 620 kHz) et courtes (OC ou SW sur 12 bandes de 2 300 kHz à 26 100 kHz). En Amérique du Nord, en mode commercial, les stations de radio en modulation d'amplitude émettent dans la bande 530-1710 kHz.

Tutoriels

https://www.instructables.com/id/RTL-SDR-FM-radio-receiver-with-GNU-Radio-Companion/
https://wiki.gnuradio.org/index.php/WBFM_Receive_PLL
https://www.youtube.com/watch?v=CftDKOcH6CAn

problème [R82XX] PLL not locked!

https://www.rtl-sdr.com/forum/viewtopic.php?t=411

> PLL not locked is a normal message and happens under normal use. Basically it seems the driver tries to lock the PLL on several frequencies, and iterates until it finds one that it can actually lock at.

https://www.rtl-sdr.com/forum/viewtopic.php?t=2374

> PLL not locked is not an issue, you'll normally see that at any frequency as the PLL tries to lock on a few frequencies before finding one that it can lock on.
> Kalibrate doesn't seem to always work properly. If possible i'd recommend just manually calibrating it on a PC with a known signal on a program like SDR#. If it's a TCXO dongle the offset will usually be 0-2PPM anyway.

https://www.rtl-sdr.com/forum/viewtopic.php?t=442
https://www.reddit.com/r/RTLSDR/comments/3fjk35/what_is_r82xx_pll_not_locked/

Generartion son haut parleurs

alsa :

je n'ai pas réussit à balancer dans la carte en direct avec hw:CARD=UMC1820,DEV=0: j'ai du sortir sur plughw:CARD=UMC1820,DEV=0
il doit y'avoir un problème de format de bytes.

Scann des fréquences radios

A partir de Gnu Radio Compagnon (je n'ai pas réussit à fair un truc propre)

1) changer regulièrement de preset
   → pouvoir piloter l'objet choose
   j'ai essay avec Function_Prob, il suffirait d'appeller
   radio_frequency_center_menu_combo_box.setCurrentIndex(…)
   mais il ne veut pas compiler car il ne connaît pas « radio_frequency_center_menu_combo_box »
   → avoir un timer :
utiliser fonction_prob
2) sur un preset faire un scan ?

→ avoir un timer

en modifiant directemetn le fichier radio.py j'arrive à passer le self à la fonction du python_module, leproblème c'estque je n'arrive pas à recuper self._radio_frequency__delta_Mhz_win à cause de fait qu'il passer par __getattrib__

Avec Thread rajouté à la main dans le code à la fin du __main__:

```
# v1 (avec thread)--------------------
if False :
    #self._radio_frequency_center_menu_combo_box.setCurrentIndex(1)
    step_float =  2000/(self.time_between_frequencies*update_frequency)
    def _radio_frequency_delta_probe():
        value = 0
        while True:
            if self.get_radio_scann():
                if int(value) != self._radio_frequency__delta_MHz_win.d_widget.value():
                    value = self._radio_frequency__delta_MHz_win.d_widget.value()

                if (value <= 1000) and ((value + step_float) >= 1000):
                    print(value)
                    time.sleep(self.time_stop_on_frequency)
                    value += step_float
                value += step_float
                if value >= 2000:
                    self._radio_frequency_center_menu_combo_box.setCurrentIndex( self._radio_frequency_center_menu_combo_box.currentIndex()+ 1)
                    value = 0
                self._radio_frequency__delta_MHz_win.d_widget.setValue(int(value))
                time.sleep(1.0 / update_frequency)
    _radio_frequency_delta_thread = threading.Thread(target=_radio_frequency_delta_probe)
    _radio_frequency_delta_thread.daemon = True
    _radio_frequency_delta_thread.start()
```

Avec Qtimer ajouté à la main dans le code à la fin du __main__:

```
if True :
    self.between_frequencies_position  = 0
    def _radio_scann_tick():
        if self.get_radio_scann():
            self.timer.setInterval(int(1000/update_frequency))
            current_index  = self._radio_frequency_center_menu_combo_box.currentIndex()
            next_index = current_index+1
            if next_index >= self._radio_frequency_center_menu_combo_box.count() :
                next_index = 0
                self._radio_frequency_center_menu_combo_box.setCurrentIndex(next_index)
                self.between_frequencies_position = 0
                self.epy_block_0_1_0.set_mode(STATIC)
                radio_frequency = self._radio_frequency_center_menu_options[next_index]
                self.set_radio_frequency_center(radio_frequency)
                self.timer.setInterval(int(self.time_stop_on_frequency*1000))
            else :

                radio_frequency_1 = self._radio_frequency_center_menu_options[current_index]
                radio_frequency_2 = self._radio_frequency_center_menu_options[next_index]
                radio_frequency = radio_frequency_1 + (radio_frequency_2 - radio_frequency_1) * self.between_frequencies_position
                self.set_radio_frequency_center(radio_frequency)
                self.between_frequencies_position += 1/(self.time_between_frequencies*update_frequency)
                if self.between_frequencies_position >= 1 :
                    self._radio_frequency_center_menu_combo_box.setCurrentIndex(next_index)
                    self.between_frequencies_position = 0
                    self.epy_block_0_1_0.set_mode(STATIC)
                    self.timer.setInterval(int(self.time_stop_on_frequency*1000))
                else :
                    self.epy_block_0_1_0.set_mode(SCANNING)

    self.timer  = Qt.QTimer(parent = qapp)
    self.timer.setInterval(int(1000/update_frequency))
    self.timer.timeout.connect(_radio_scann_tick)
    self.epy_block_0_1_0.set_mode(SCANNING)
    self.timer.start()
```

Mapping

Log Power FFT :

l'objet est bugé : https://github.com/gnuradio/gnuradio/issues/2158

FFT :

https://wiki.gnuradio.org/index.php/FFT

je n'arrive pas à avoir exactemetn la meme chose qu'avec frequency sink..

sur partie FM :

changement tres rapides.
Moduler avec le volume ne donne rien car changement trop rapides , laisse pas le temps à la corde de s'arreter et reprendre en suivant le volume .

Trop pour laisser le temsp à des ondes de s'installer, après on peut faire une moyenne temporelle longe sur fft

enveloppe detector http://play.fallows.ca/wp/radio/software-defined-radio/gnuradio-envelope-detector-in-python/

Pb son haché quand on scanne

1) passer directment sur carte son
2) diminuer audio rate
2) virer les plots
3)