RAPPORT TP4 ESEC

Réalisation d'un modem clé de voiture

par Y. KIM, X. LEGASPI & Y. MOREL

17 décembre 2023

Table des matières

1 Contexte **1**

2 Premières observations **2**

2.1 Outils 2

2.2 Interception d'un signal digital 4

3 Démodulation manuelle **10**

4 Démodulation automatisée **16**

5 Interpretation **17**

6 Modulation **19**

1. Contexte

L'objectif du TP4 est de pouvoir interpréter et simuler le signal radio d'une clé de voiture fictive. En général, les clés de voiture émettent dans la bande de fréquence LPD433. Ça sera notre cas ici : 433.92 MHz précisément. Pour simuler notre clé, une *HackRF One* enverra un signal toutes les n secondes en salle de TP : un signal de fermeture, n secondes de pause, un signal d'ouverture, etc. On va d'abbord démoduler manuellement et automatiquement ce signal, avec *Inspectrum* et *GNU Radio*, respectivement. Puis on va décoder les données des *IQ* fournis. Finalement, on réalisera un émetteur avec *GNU Radio* pour émuler le signal de la clé observée.

Si un signal avait déjà été préparé pour l'acquisition avec une clé, nous avons décidé d'essayer, en parallèle avec l'acquisition de celle-ci, d'interpréter ce qui est envoyé par nos

propres clés de voitures, le plus possible.

Environnement.– Dans la suite, on va travailler avec (1) une machine Kali et (2) une HackRF One avec antenne.

2. Premières observations

L'objectif de cette section est d'apprendre à intercepter un signal digital.

2.1. Outils

HackRF One (figure 1, figure 2) est un émetteur-récepteur de radio logicielle (SDR), conçu pour permettre aux utilisateurs d'explorer et de manipuler une grande variété de signaux radiofréquences (RF) à l'aide de logiciels. En particulier, il va nous permettre de intercepter le signal de la clé d'une voiture fictive. Ce dispositif a été développé par Great Scott Gadgets, et il est largement utilisé par les amateurs, les chercheurs en sécurité informatique, les radioamateurs, et d'autres personnes intéressées par l'exploration et l'expérimentation dans le domaine des communications sans fil.

Figure 1

Voici quelques caractéristiques clés du HackRF One :

(i) *Radiofréquences* : Le HackRF One est un SDR qui permet de recevoir et de transmettre des signaux RF sur une large plage de fréquences, de 1 MHz à 6 GHz.

Figure 2

(ii) *Reprogrammabilité* : Il est basé sur un convertisseur analogique-numérique (CAN) et un convertisseur numérique-analogique (CNA) programmables.

(iii) *Large bande passante de* 250 *Mhz* : Il offre une large bande passante, permettant aux utilisateurs d'explorer différents types de signaux radio, tels que la transmission et la réception de signaux FM, AM, TV, GSM, GPS, Wi-Fi, Bluetooth, et bien d'autres.

(iv) *Open Source.*

Voici quelques inconvénients du HackRF One :

(i) *Prix* : HackRF One est relativement cher par rapport à d'autres SDR (bladeRF, Airspy, LimeSDR, USRP, RTL-SDR).

(ii) *Puissance de sortie* : La puissance de sortie de HackRF One est limitée à 20 dBm, ce qui peut être insuffisant pour certaines applications.

(iii) *Consommation d'énergie* : HackRF One consomme beaucoup d'énergie, ce qui peut être un problème pour une utilisation sur batterie.

Il faudra aussi déterminer la fréquence à lequel la signal de clé de voiture opère. Pour cela on peut utiliser CubicSDR, après avoir connecté HackRF à notre machine Kali. On peut vérifier que la clé de voiture fourni pour le TP a bien la fréquence 433.92 MHz. Si on veut essayer avec nos proprés clés de voiture, il faudra faire attention a cette étape. On peut l'installer avec la commande :

```
$ sudo apt install cubicsdr
```

CubicSDR
Cross-Platform and Open-Source Software-Defined Radio Application

FIGURE 3

Pour pouvoir éxécuter l'interception, on utilisera le logiciel GNU Radio Companion. On peut l'installer avec la commande :

```
$ sudo apt install gnuradio
```

GNURadio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

FIGURE 4

GNURadio a besoin du logiciel *xterm* pour générer des graphes (**??**). Pour résoudre le problème, on peut l'installer avec la commande suivante.

```
$ sudo apt-get install xterm
```

2.2. Interception d'un signal digital

Lancer GNU Radio Companion :

```
$ gnuradio-companion
```

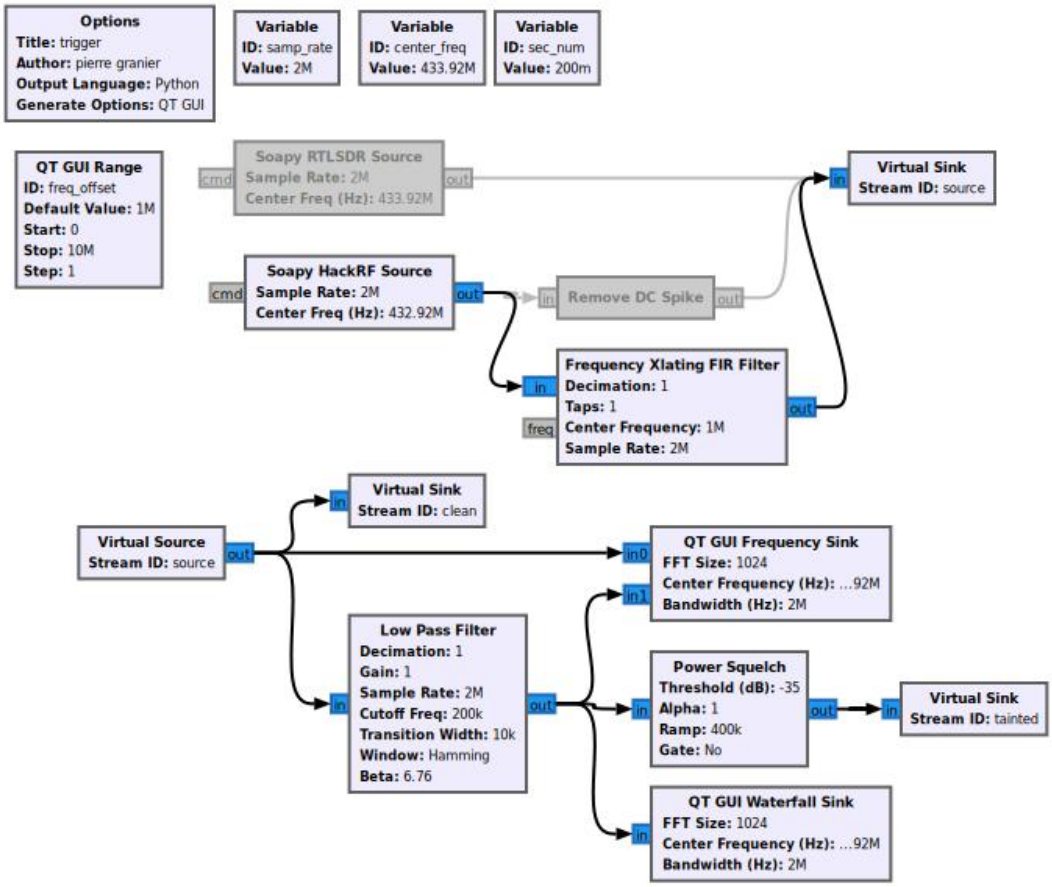

FIGURE 5 – *Première partie de trigger.grc*. Ouvrir le fichier de configuration *trigger.grc*. Ce schéma est fourni par le résponsable du TP : il correspond au paramétrage pour acquérir le signal radio capté par l'appareil HackRF One. Chaque bloc représente un point d'entrée, de sortie ou de modification de l'acquisition afin de pouvoir l'interpréter. On peut voir notamment le bloc "*Soapy HackRF Source*" qui correspond à l'acquisition par le HackRF, dont les fréquences vont être filtrées et affinées pour ensuite être sauvegardées en fichier .dat (bloc "Tagged File Sink", figure 6).

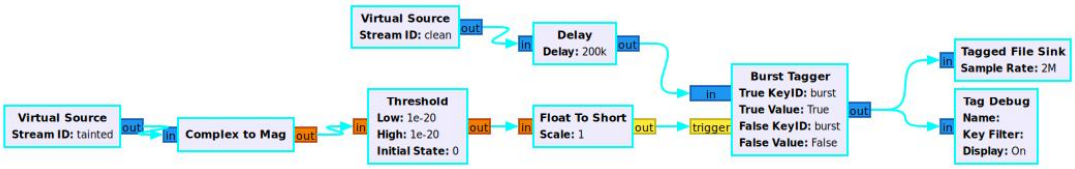

FIGURE 6 – *Deuxième partie de trigger.grc*

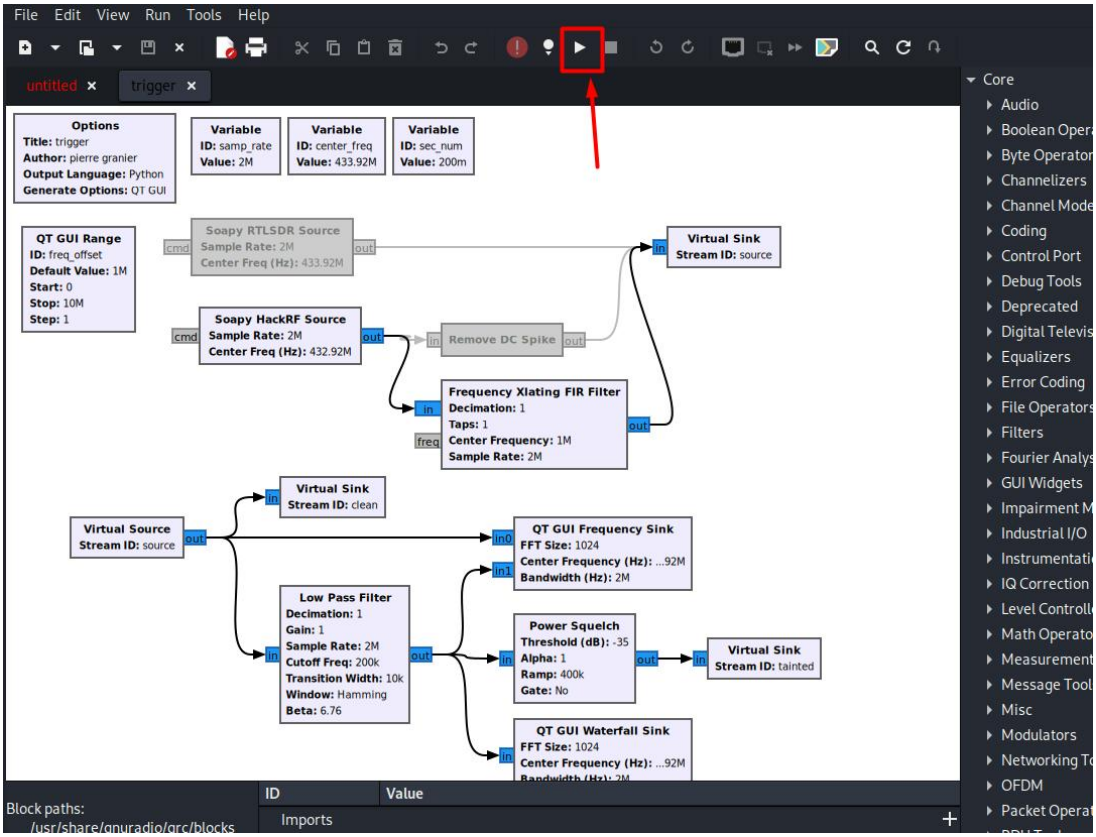

FIGURE 7 – Brancher le HackRF One et lancer la capture.

FIGURE 8 – On obtient une capture en temps réel.

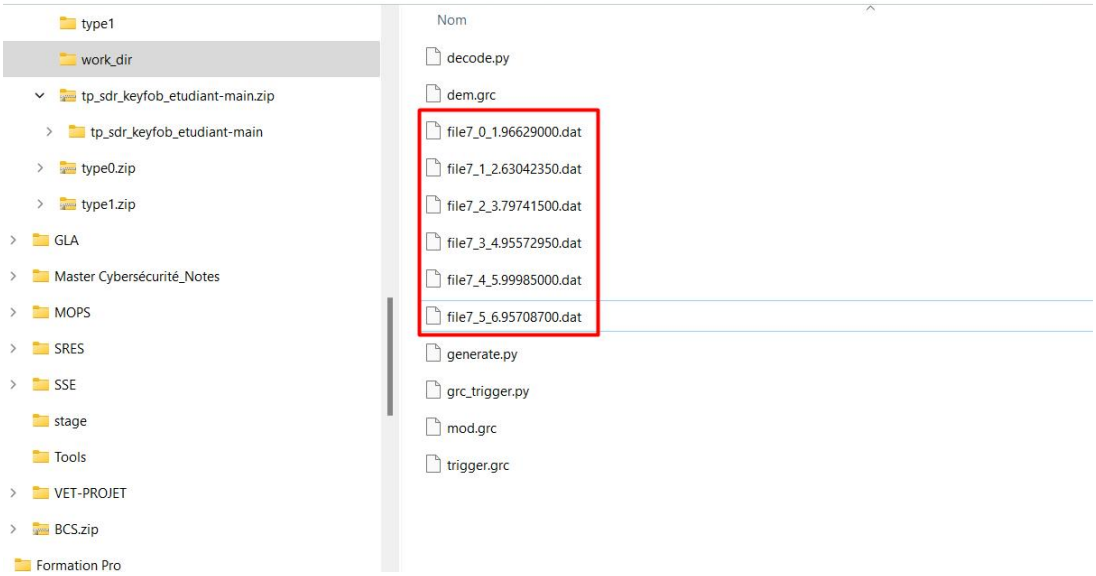

FIGURE 9 – Les fichiers contenant les données sont stockés au niveau du répertoire du fichier “trigger.grc” actuellement exécuté par GNU Radio.

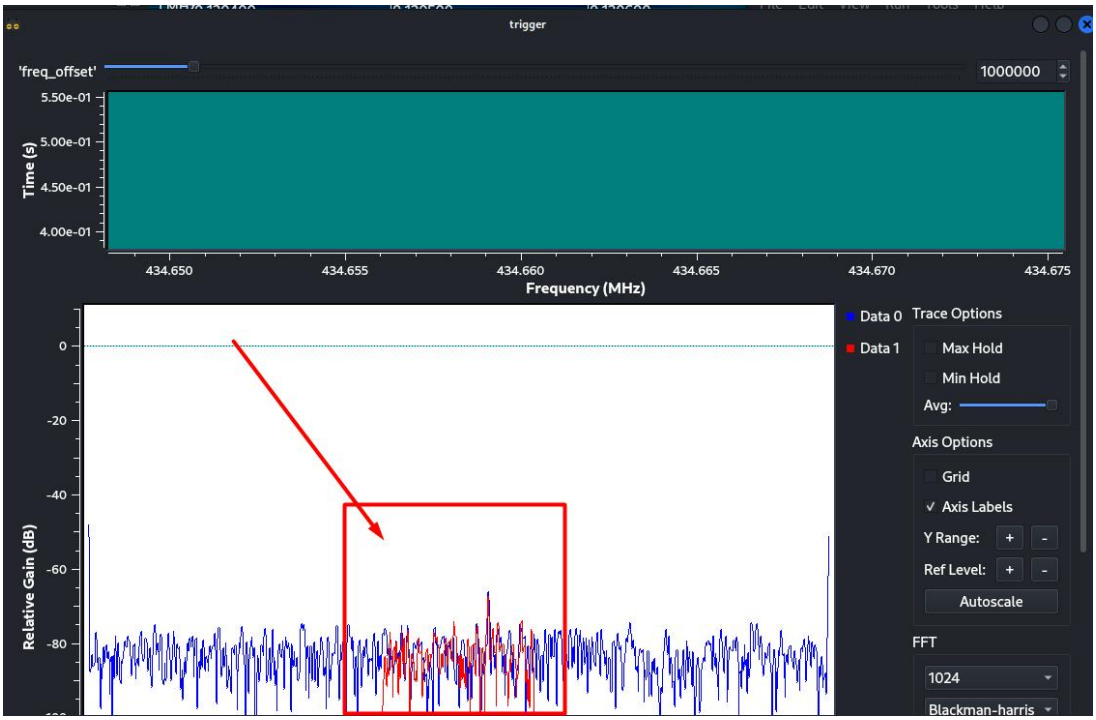

FIGURE 10 – Le fichier de configuration “trigger.grc” permet de paramétrer le signal radio capturé et de faire un focus sur une certaine plage de fréquence, en rouge.

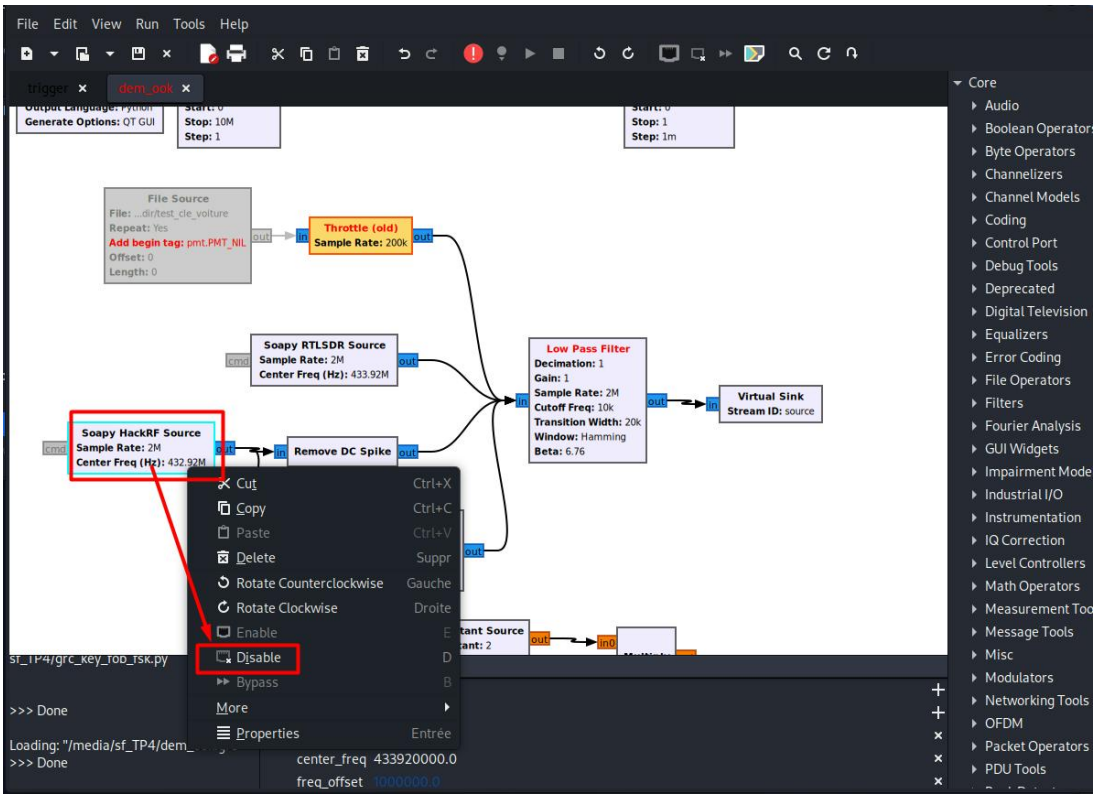

FIGURE 11 – *Il est possible de rejouer les données capturée en partant d'un fichier de configuration "dem_ook.grc"*. Désactiver la source : [click-droit → Disable] (ou [click-gauche et touche D]).

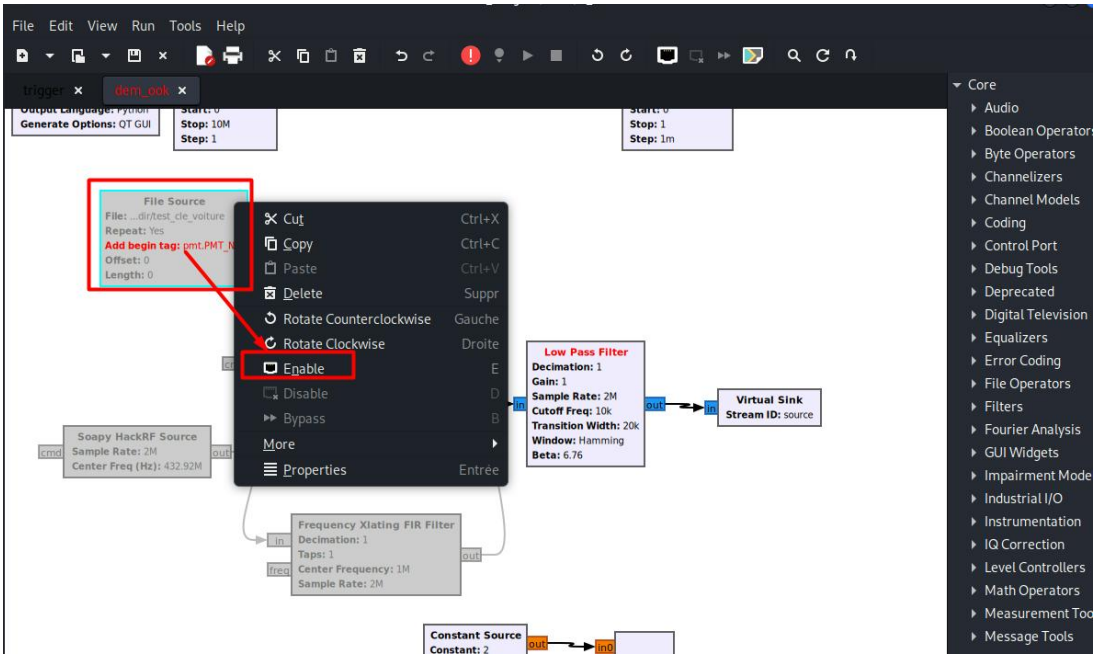

FIGURE 12 – *Il est possible de rejouer les données capturée en partant d'un fichier de configuration "dem_ook.grc"*. Activer le fichier : [click-droit → Enable] ou [click-gauche et touche E].

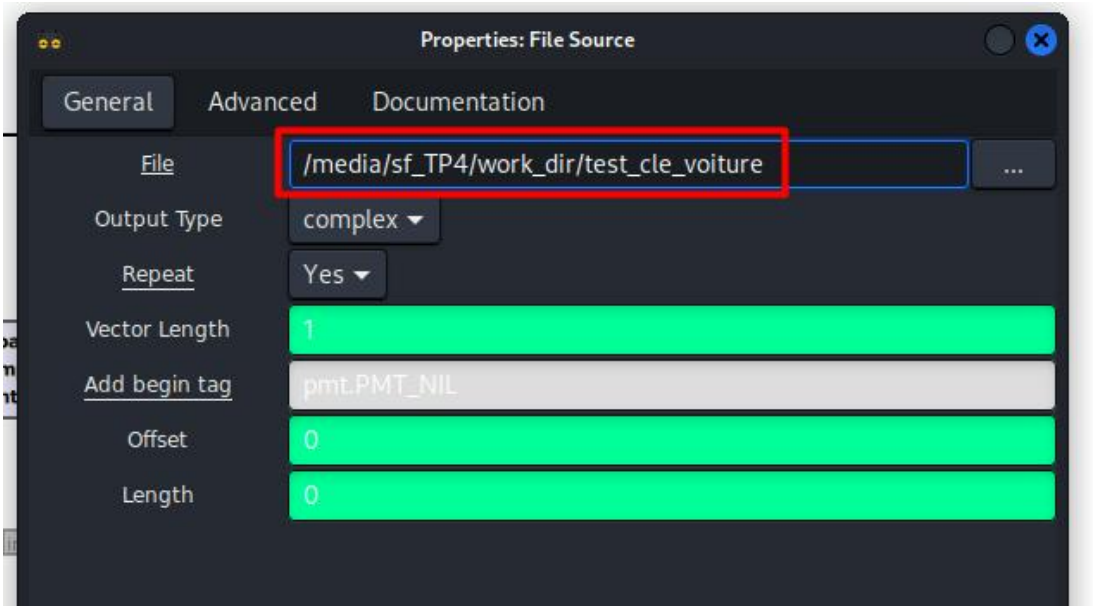

FIGURE 13 – Spécifier le fichier de capture, [double-click et renseigner le chemin du fichier contenant les données capturées].

3. Démodulation manuelle

L'objectif de cette section est d'apprendre à démoduler un signal digital. La démodulation d'un signal est un processus qui consiste à extraire l'information d'un signal modulé.

Tout d'abord on va installer Inspectrum dans notre machine Kali. On peut utiliser la commande

```
$ sudo apt install inspectrum
```

C'est un outil qui permet analyser des signaux qui ont été capturés à partir de logiciels de réceptions d'onde. En particulier, il nous permettra de réaliser la démodulation manuelle.

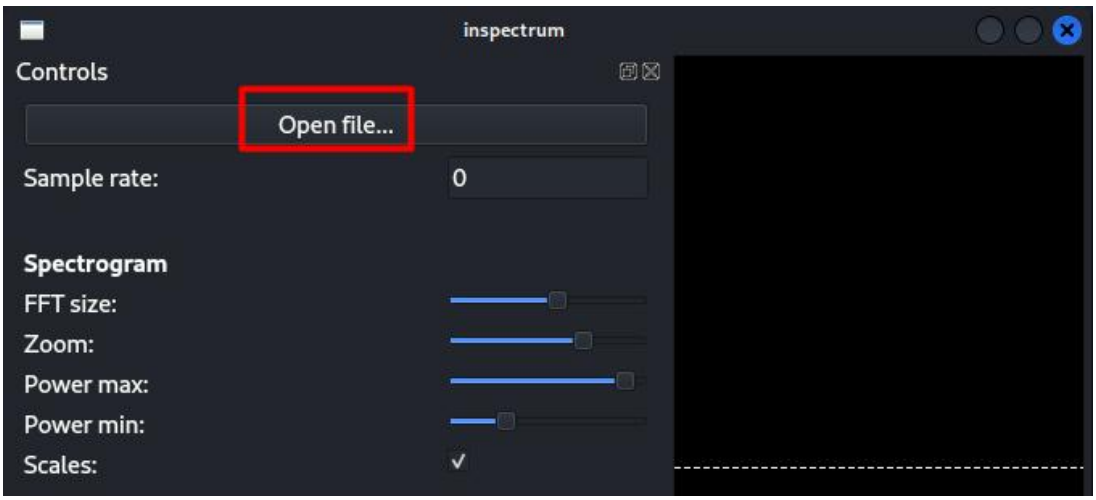

FIGURE 14 – Ensuite, on lance Inspectrum et on ouvre le fichier .dat du signal capturé.

Remarque.– Dans cette section on déduit les informations suivants sur la transmission. Le *sample rate* est $2.048 \cdot 10^6$. Le bit time est 51.9 μs. Le *bit rate* est 18.963 *kBd*. La forme de la modulation est ASK FSK. On voit que on peut décomposer la trame binaire en trois sous-trames de $\sim$ 800 bits, $\sim$ 1399 bits, et $\sim$ 234 bits respectivement. Ces trames correspondent au signal de fermeture de clé, quelques secondes de pause et au signal d'ouverture de clé.

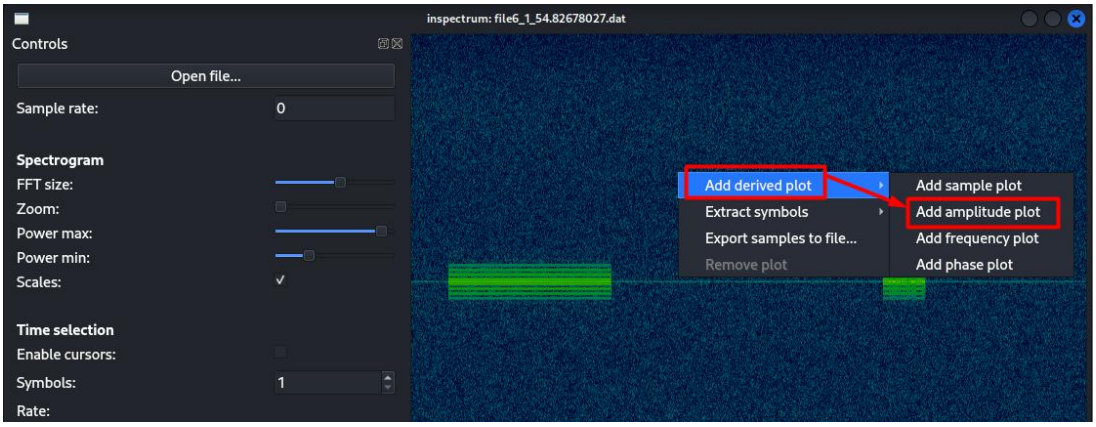

FIGURE 15 – Ajouter un point d'amplitude en faisant [clic-droit → Add derivated plot → Add amplitude plot].

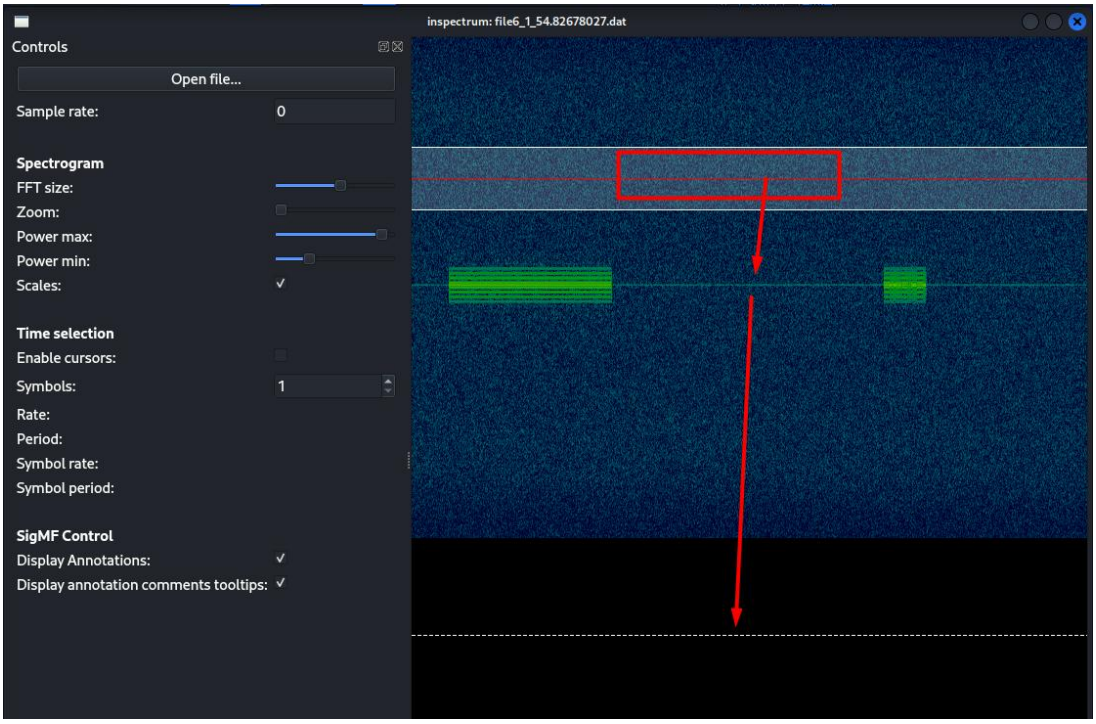

FIGURE 16 – Déplacer le seuil/threshold (trait rouge) jusqu'au milieu du signal jusqu'à obtenir une résolution correcte des signaux.

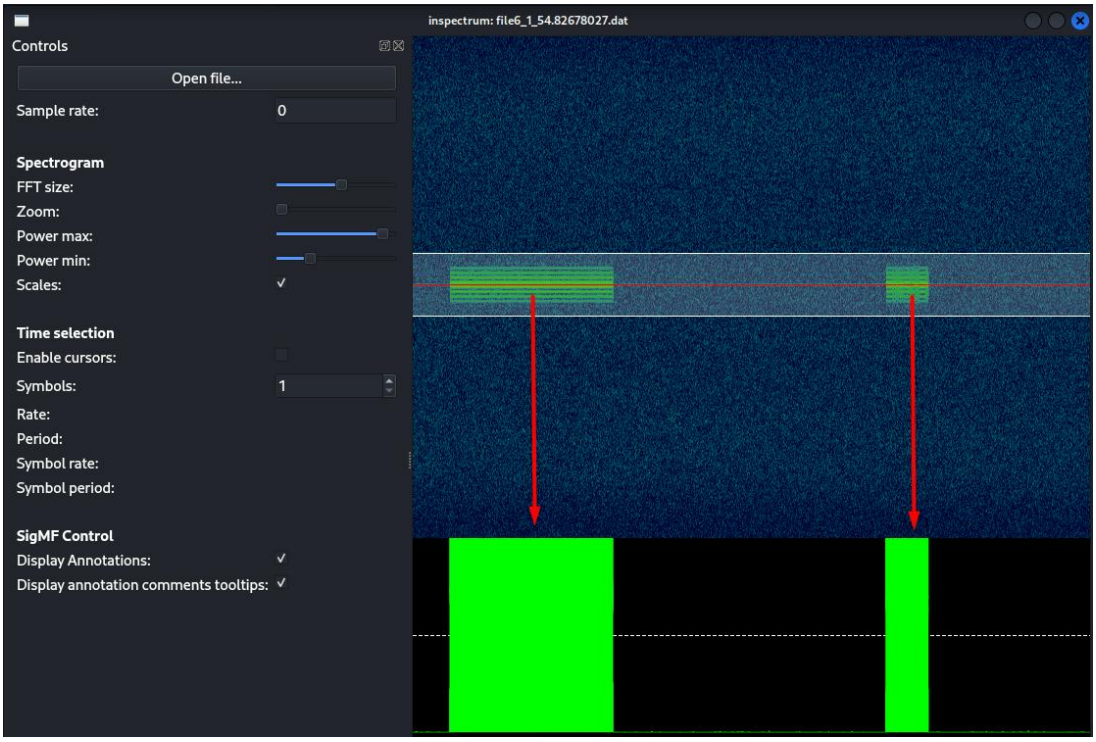

FIGURE 17 – Zoomer jusqu'à visualiser les signaux.

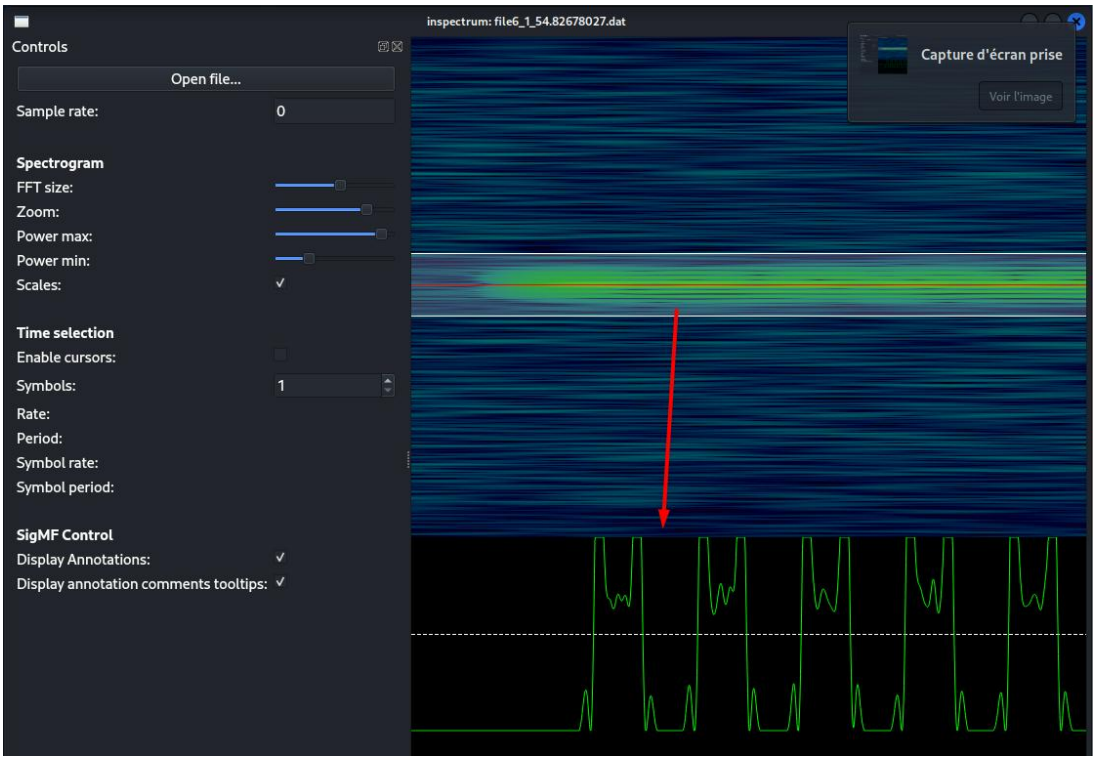

FIGURE 18 – Ajuster le power max pour minimiser les bruits.

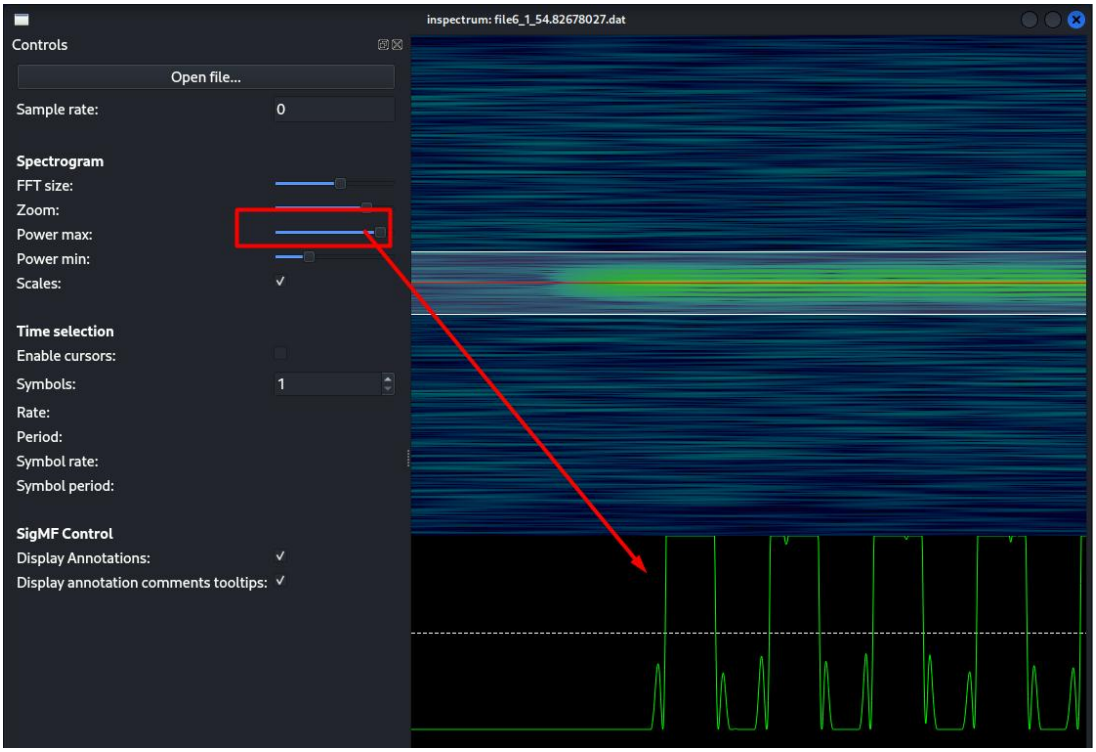

FIGURE 19 – Activer le cursor pour définir une plage d'échantillonnage des données.

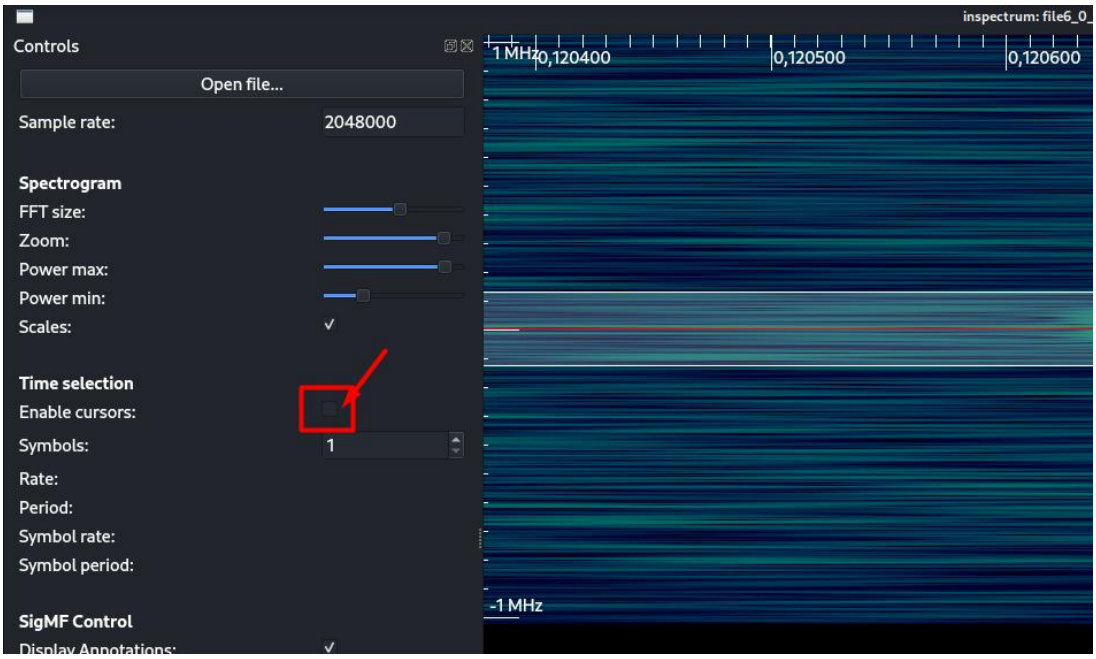

FIGURE 20 – Il est aussi possible, pour plus de précision, d'afficher le graphe retourné par l'utilisation du treshold afin d'avoir, lorsqu'il est bien placé, uniquement une variation de 1 et de 0, plus facile à échantilloner que la capture brute.

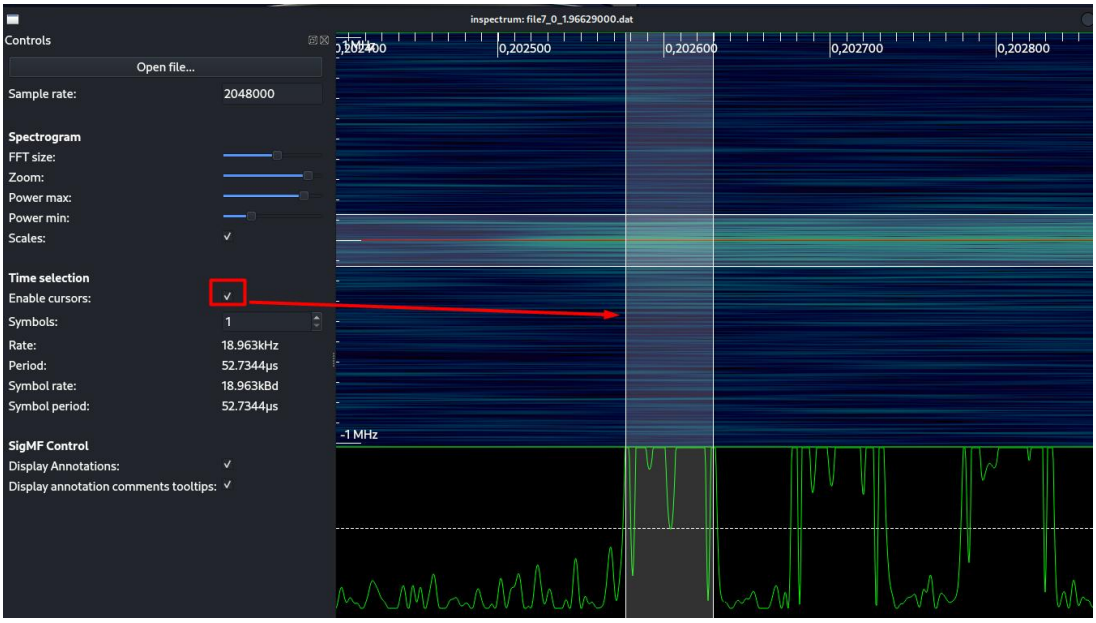

FIGURE 21 – Ajuster la plage de capture.

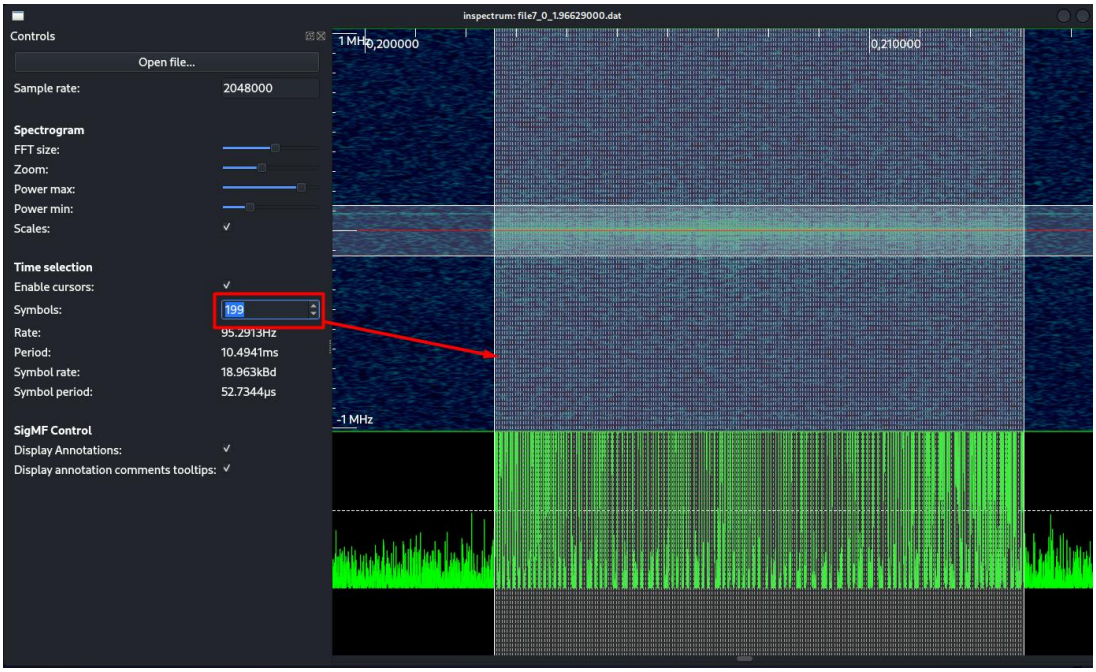

FIGURE 22 – Augmenter le nombre d'échantillons.

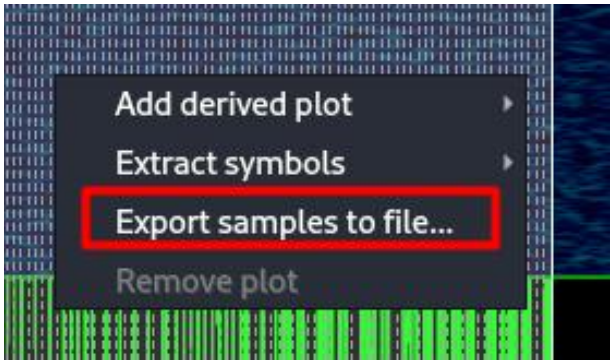

FIGURE 23 – Faire un click-droit Export samples to file . . .

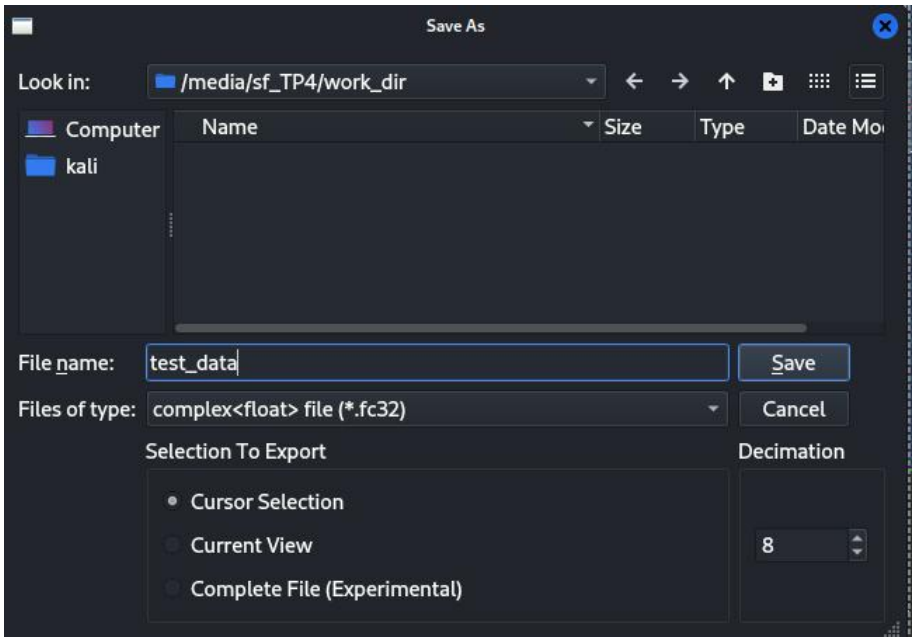

FIGURE 24 – Donner un nom au fichier.

4. Démodulation automatisée

En utilisant un schéma de configuration avancé “dem_ook.grc”, il est possible d’automatiser ce qu’on a fait dans la partie précédente.

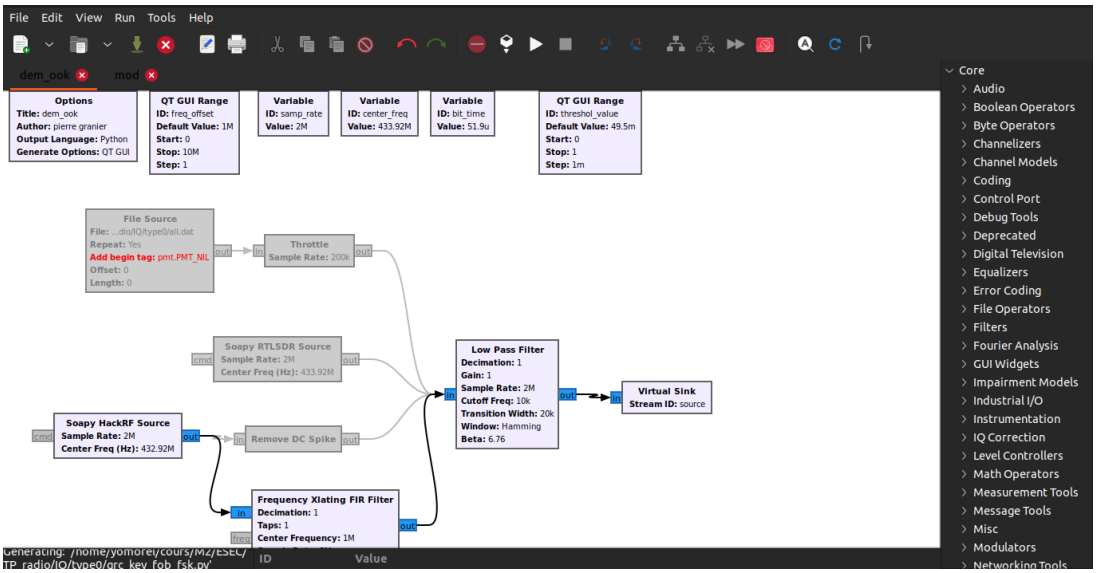

FIGURE 25

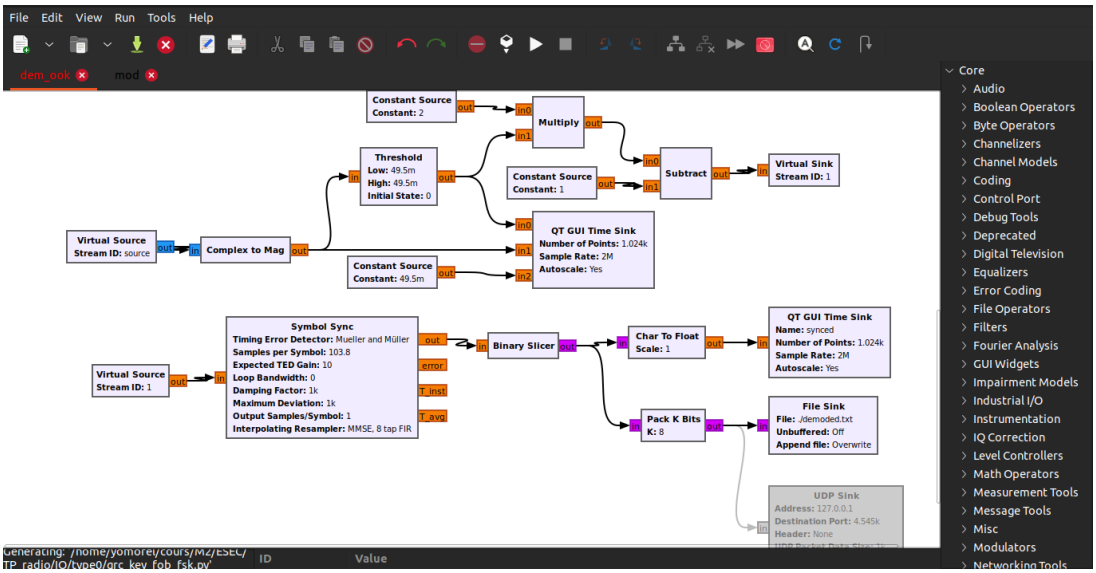

FIGURE 26

Il faut cependant déterminer “à la main” la valeur de treshold, jusqu’à avoir une valeur pour laquelle le “synced” est le plus unifié possible, c’est à dire qu’il y aura le moins de “00” et “11” à la suite (les valeurs undefined du codage Manchester).

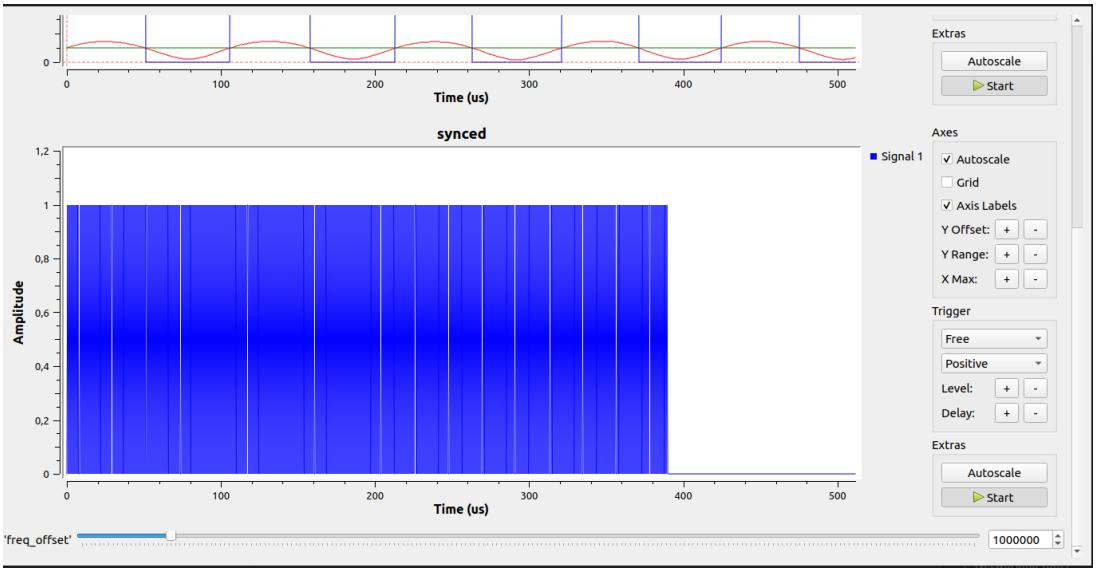

FIGURE 27

A noter que nous avons dans un premier temps réalisé l'acquisition dans l'optique d'interpréter un fichier (cf. section Interprétation), mais nous avons ensuite opté pour un envoi par UDP lorsque le professeur nous a donné son fichier d'interprétation.

5. Interpretation

Le but de ce section est le décryptage des données avec Python.

A présent nous avons un fichier contenant une suite de 0 et de 1, sous l'encodage Manchester. Nous avons donc fait une fonction Python permettant de le décoder pour obtenir une suite binaire "traditionnelle" et interprétable.

```python
def decode_manchester_file(filename):
    print("decode machester here")
    manchester_bits = []
    with open(filename, 'rb') as file:
        byte = file.read(1)
        while byte:
            manchester_bits.extend([int(bit) for bit in
            ↪  f'{ord(byte):08b}'])
            byte = file.read(1)

    decoded_bits = []
    i = 0
    while i < len(manchester_bits) - 1:
        bit1 = manchester_bits[i]
        bit2 = manchester_bits[i + 1]
        if bits[i] == 0 and bits[i+1] == 0:
            while bits[i] == 0 and bits[i+1] == 0:
                i+=1
            continue
        elif bit1 == 1 and bit2 == 0:
            decoded_bits.append(0)
            i += 1
        elif bit1 == 0 and bit2 == 1:
            decoded_bits.append(1)
            i += 1
        else:
            decoded_bits.append(-1)
            i += 1

    #print(decoded_bits)
    return "".join(map(str, decoded_bits))
```

L'idée de cette fonction de décodage est de transformer les “01” et “10”, respectivement en 1 et 0 en langage binaire (On peut se les représenter comme étant une montée ou descente de tension, donc un passage à 1 ou à 0). Le plus difficile étant de savoir quand commencer l'interprétation (entre les suites de 0 avant l'envoi de données).

Le lendemain nous avons donc obtenu le script plus propre et complet fourni, nous avons donc continué à partir de celui-ci, avec l'envoi en UDP par gnuradio.

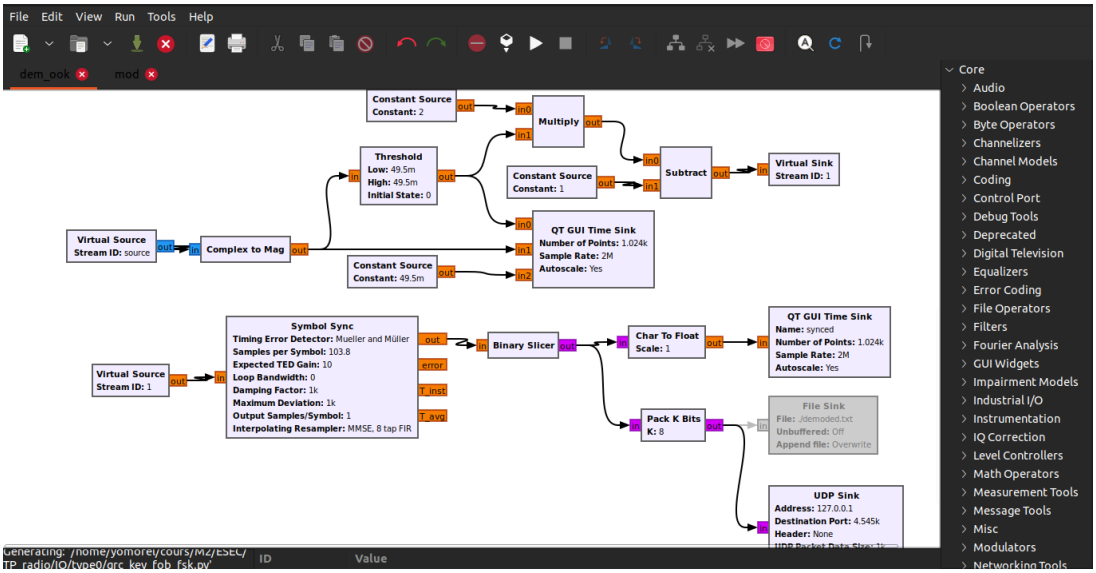

FIGURE 28

Cependant, l'utilisation de nos propres clés de voitures pour l'interprétation s'arrête là : cela demanderait beaucoup de temps supplémentaire de faire du "reverse-engeeniring" sur le format des données envoyées par notre clé.

Nous avons donc continué avec le fichier mask_clear.txt, qui correspond à un fichier encodé en manchester, dans le contexte de la voiture du TP.

FIGURE 29

6. Modulation

Pour cette partie nous avons reproduit l'encodage manchester à partir d'un fichier binaire classique (les données qu'on veut envoyer à la voiture, dans notre contexte, celles

qu'on a intercepté).

Voir le fichier generate.py, qui contient les fonctions permettant de récupérer le padding à utiliser entre les envois entre les envois (plusieurs envois pour la redondance, afin d'assurer la bonne transmission), d'encoder au format Manchester, et plus globalement de "forger" le fichier binaire manchester à partir d'une trame binaire classique.

Une fois ce fichier généré, on va l'envoyer à Gnuradio et le faire passer dans des blocs spécifiques qui vont permettre de transformer les bits en fréquence à envoyer.

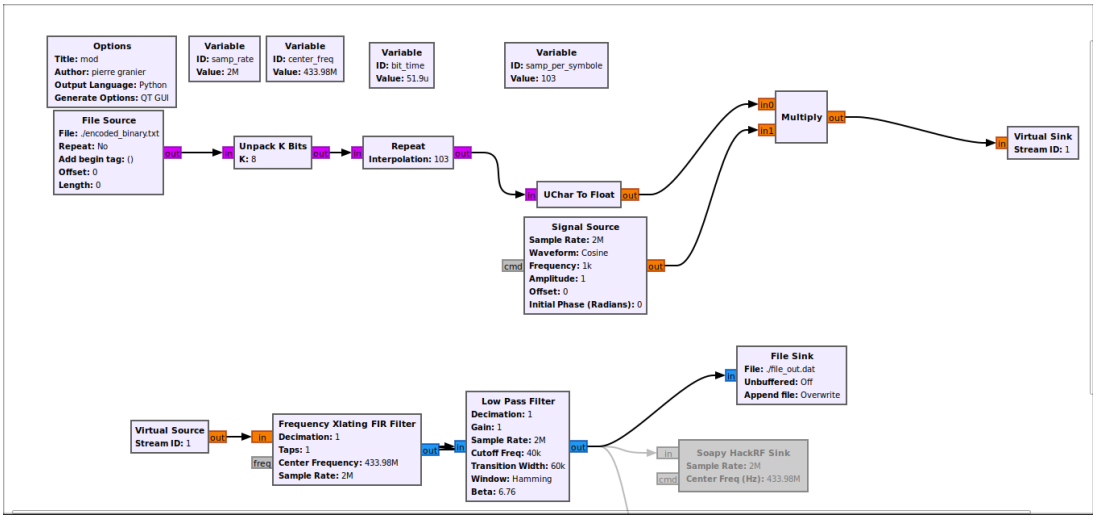

FIGURE 30

Après avoir valué samp_rate, center_freq, et bit_time avec les valeurs que nous avions trouvé/utilisé pour la démodulation, nous avons décidé (conseillés par un camarade ayant fait le module ISR), d'utiliser un signal sinusoidal (cosinus, avec le bloc "Signal Source"), qui sera multiplié à nos binaires (encodés en Manchester) afin d'obtenir les variations de fréquences du transport des données. Après cela il n'y a plus qu'à adapter la fréquence de la courbe (par défaut à 1 MHz), à celle utilisée par notre clé (le bloc "Frequency Xlating FIR Filter" avec la valeur center_freq).

On a donc pu le renvoyer en utilisant le HackRF, mais nous n'avons pas eu l'occasion de vérifier sa réception par manque de temps. Cependant, on peut le charger en fichier .dat, et en refaisant les étapes depuis le début, le re-décoder.

Quelques liens

1. Samy Kamkar. Drive it like you Hacked it: New Attacks and Tools to Wireless. DEF CON 23.
2. I/Q Signals 101: Neither Complex Nor Complicated.
3. La réception radiofréquence définie par logiciel (Software Defined Radio - SDR).
4. Outils, logiciels et matériels opensource de Radio-Fréquence.
5. Rapid Radio Reversing.