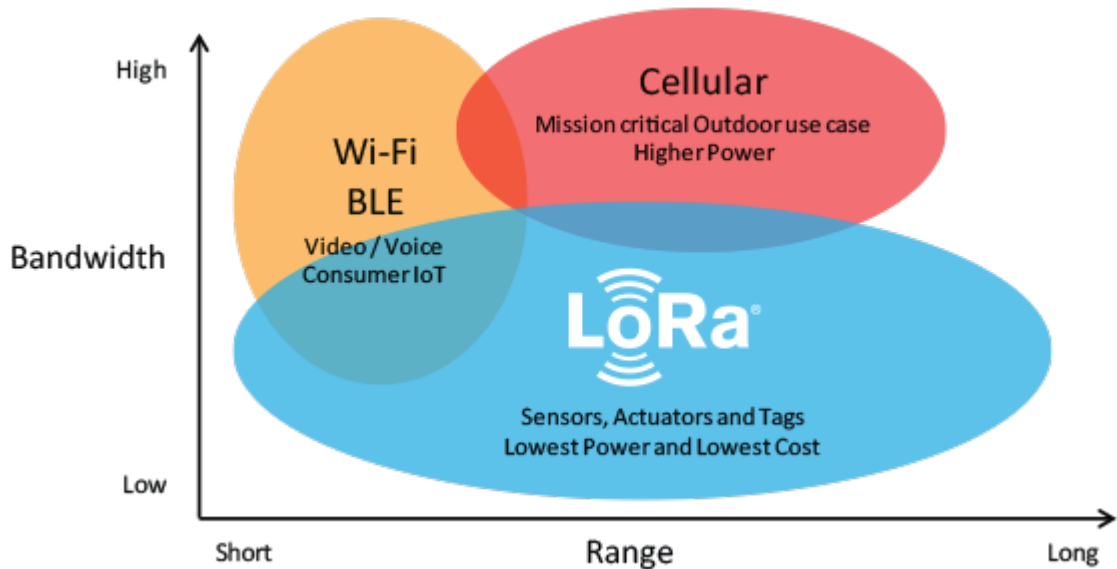


Identification de l'émetteur à partir du préambule LoRa

Note d'application :

**Configuration de GNU Radio et bladeRF 2.0 micro
pour la Réception des Signaux LoRa avec Calcul du
CFO**



Réalisation : ZHAO Yan

Ciente : Mme. EL RACHKIDY Nancy (LIMOS)

Tuteur industriel : M. FICKINGER Pascal

Professeur référent : M. LANDRAULT Alexis

Introduction

Dans le cadre de notre projet de fin d'études à Polytech Clermont, nous avons travaillé sur la configuration et l'exploitation de GNU Radio pour l'analyse et le traitement des signaux LoRa, en utilisant le bladeRF 2.0 micro comme récepteur SDR. Un défi majeur consistait à sélectionner et configurer les modules les plus adaptés pour la réception, le décodage et l'analyse des signaux LoRa, tout en intégrant une méthode efficace pour le calcul et l'affichage du Carrier Frequency Offset (CFO).

L'objectif principal était de configurer GNU Radio de manière optimale afin d'exploiter pleinement les capacités du bladeRF, en assurant une réception stable et un traitement efficace des signaux. Pour cela, nous avons étudié les différents blocs DSP disponibles, en intégrant notamment des modules externes comme gr-lora pour le décodage des trames LoRa.

Un aspect clé du projet a été l'estimation et l'affichage en temps réel du CFO, permettant d'analyser les écarts de fréquence entre l'émetteur et le récepteur. Pour ce faire, nous avons modifié le code source de gr-lora afin d'extraire le CFO à partir des signaux reçus et l'afficher dynamiquement dans GNU Radio QT GUI. Cette étape a permis d'améliorer la précision de la réception et d'optimiser la synchronisation des trames LoRa.

Matériel et logiciels requis

Pour la mise en œuvre de cette application, j'ai utilisé les outils suivants.

Matériel :

Récepteur SDR (bladeRF 2.0 micro).

Émetteur (ESP32 WiFi LoRa 32 V2).

PC avec environnement Linux.

Logiciels :

Gnu radio

Arduino

Configuration et mise en œuvre

Pour l'intégration de bladeRF micro 2.0 avec GNU Radio, voici un guide détaillé des étapes d'installation et de configuration.

1. installer le pilote bladeRF

```
git clone https://github.com/Nuand/bladeRF
cd bladeRF/host
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig
```

2. installation de GNU Radio et de gr-osmosdr

```
sudo apt install gnuradio
sudo apt install gr-osmosdr
```

3. installation de Gr-Lora

```
git clone https://github.com/rpp0/gr-lora.git
cd gr-lora
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig
```

Après avoir installé tous les logiciels nécessaires, y compris GNU Radio, gr-lora et les pilotes pour bladeRF, il est possible de configurer le système comme suit dans GNU Radio Companion:

Osmocom Source est configuré pour recevoir un signal LoRa à 868 MHz via bladeRF, avec un taux d'échantillonnage de 1M et une bande passante de 125 kHz.

LoRa Receiver est paramétré pour décoder les trames LoRa avec un Spreading Factor SF9, sans réduction de débit et avec correction de dérive de fréquence activée.

Ces paramètres, notamment le Spreading Factor (SF), la fréquence d'échantillonnage et la bande passante, peuvent être ajustés en fonction de vos besoins spécifiques. Par exemple, si votre émetteur utilise une SF différente ou une autre bande passante, il est possible de modifier ces valeurs dans GNU Radio pour assurer une réception correcte.

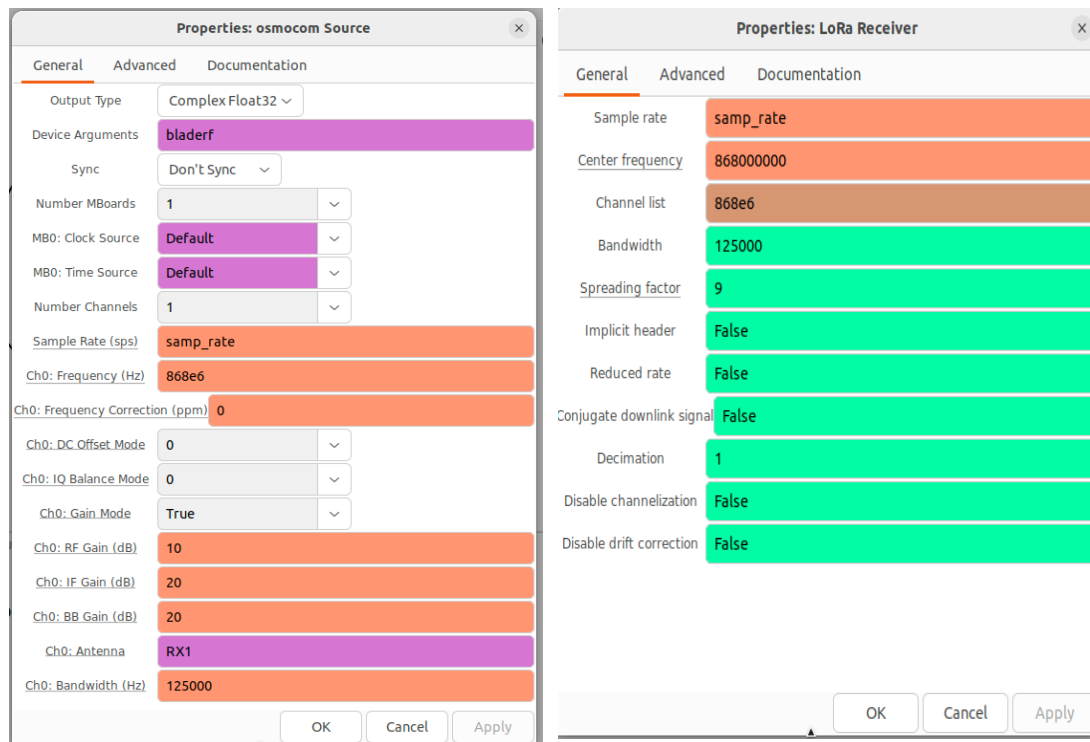


Figure 1 *osmocom source et lora receiver*

Comme illustré dans les captures d'écran ci-dessus, vous pouvez configurer vos blocs GNU Radio de la même manière afin de garantir la compatibilité avec votre environnement de test et vos équipements.

Une fois les deux modules principaux configurés – Osmocom Source (réception SDR) et LoRa Receiver (décodage LoRa). Nous pouvons ajouter des modules supplémentaires pour l'analyse et la visualisation des données.

Après avoir assuré la réception et le décodage des signaux LoRa, nous intégrons deux nouveaux blocs pour afficher et vérifier les données.

Le bloc Message Debug permet d'afficher les trames LoRa reçues et décodées directement dans le terminal de GNU Radio. Il est connecté à la sortie du LoRa Receiver, qui lui fournit les paquets de données interprétés. Son utilité principale est de vérifier que les données LoRa sont bien reçues et lisibles, facilitant ainsi l'analyse et le débogage du signal.

Le bloc QT GUI Frequency Sink est utilisé pour visualiser le spectre du signal reçu en temps réel. Il est connecté directement à la sortie de l'Osmocom Source, ce qui permet d'observer la présence d'un signal LoRa sur la fréquence cible 868 MHz avant même son décodage. Son utilité principale est de confirmer que le récepteur SDR capte bien un signal, facilitant ainsi l'analyse et le réglage des paramètres de réception.

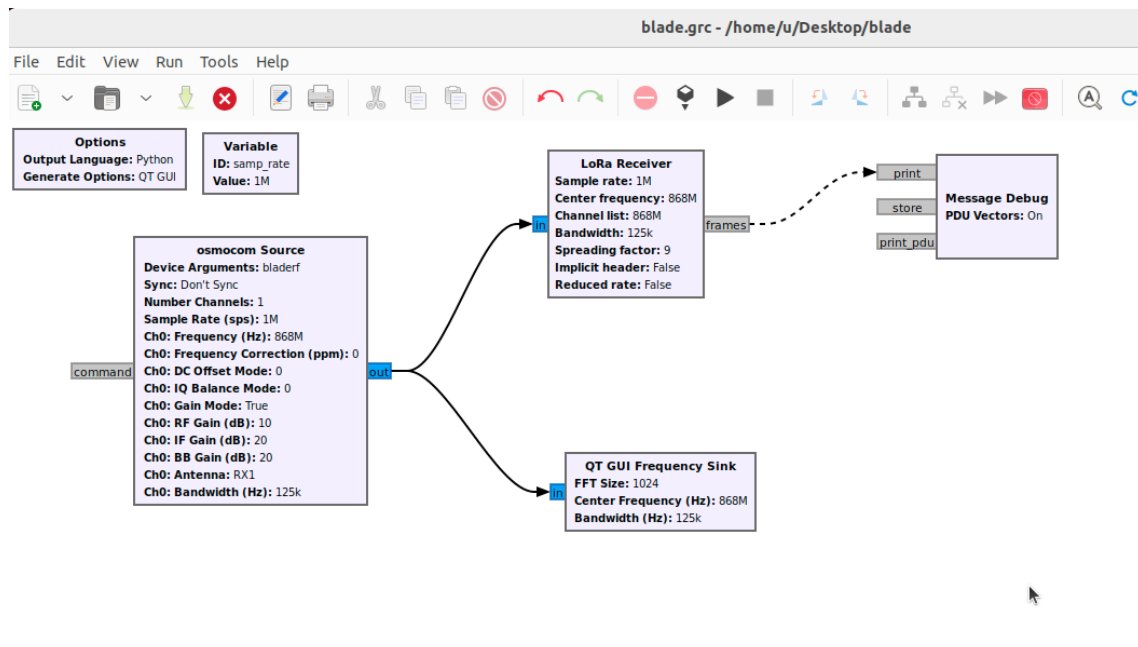


Figure 2 Gnu-radio

Analyse et résultats

Après avoir terminé la configuration, j'ai commencé l'analyse et constaté que le CFO ne s'affichait pas dans le terminal comme prévu ; seules les trames de données étaient visibles. Pour remédier à cela, il est possible de modifier le fichier `decoder_impl.cc` du module `gr-lora`.

Il suffit d'ouvrir ce fichier et de localiser les lignes 805 et 806, puis de supprimer les caractères `//` au début de ces lignes. Une fois cette modification effectuée, le CFO sera correctement affiché dans le terminal via le bloc Message Debug de GNU Radio.

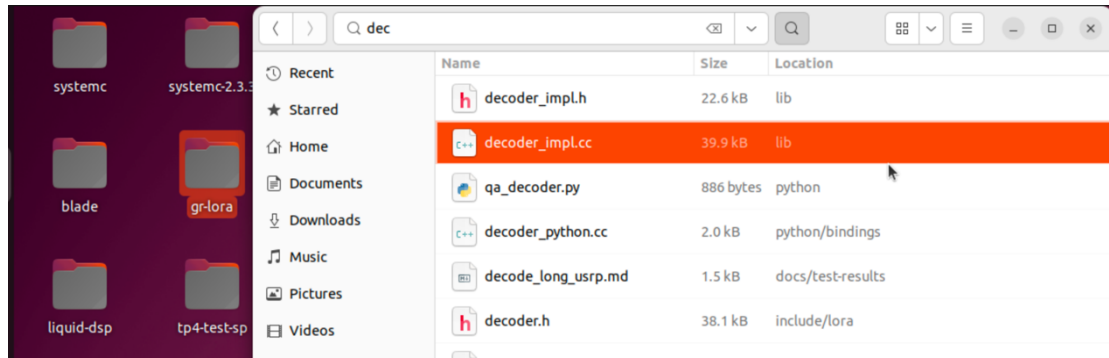


Figure 3 *Fichier de decoder*

Première méthode : Cette approche est rapide mais imprécise, car elle se base uniquement sur un seul échantillon (`mult_ifreq[256]`) pour estimer le Carrier Frequency Offset (CFO). En utilisant une seule valeur, cette méthode est très sensible au bruit et aux variations instantanées du signal. En cas de distorsion, d'interférences ou d'un signal faible, l'estimation du CFO peut être instable et peu représentative de la fréquence réelle du signal. Son principal avantage est sa simplicité et son efficacité en termes de calcul, ce qui la rend adaptée aux environnements où la rapidité d'exécution est prioritaire et où une précision fine n'est pas essentielle.

Deuxième méthode : Cette approche est plus fiable et plus robuste, car elle prend en compte tous les échantillons disponibles dans une fenêtre d'analyse donnée. Plutôt que de s'appuyer sur une seule valeur, elle applique une moyenne pondérée, ce qui permet de réduire l'influence du bruit et des variations soudaines du signal. La pondération est effectuée en fonction de l'amplitude du signal, ce qui permet de donner plus d'importance aux échantillons présentant un niveau de puissance plus élevé, généralement plus représentatifs du signal utile. Grâce à cette technique, l'estimation du CFO est plus stable et plus précise, même dans un environnement où le signal est sujet à des fluctuations ou à des interférences. Toutefois, cette méthode est plus coûteuse en calcul, car elle nécessite plus d'opérations mathématiques que la première approche. Elle est donc recommandée lorsque la précision est essentielle, notamment dans des applications où une correction fine du CFO est nécessaire pour garantir une bonne synchronisation du signal.

```

727      /**
728       * New method to determine CFO.
729       */
730      /*float decoder_impl::experimental_determine_cfo(const gr_complex *samples, uint32_t window) {
731          gr_complex mult>window];
732          float mult_ifreq>window];
733
734          volk_32fc_x2_multiply_32fc(mult, samples, &d_downchirp[0], window);
735          instantaneous_frequency(mult, mult_ifreq, window);
736
737          return mult_ifreq[256] / (2.0 * M_PI) * d_samples_per_second;
738      }*/
739      float decoder_impl::experimental_determine_cfo(const gr_complex *samples, uint32_t window) {
740          gr_complex mult>window];
741          float mult_ifreq>window];
742          float sum_ifreq = 0.0f;
743          float sum_weight = 0.0f;
744          volk_32fc_x2_multiply_32fc(mult, samples, &d_downchirp[0], window);
745          instantaneous_frequency(mult, mult_ifreq, window);
746          for (uint32_t i = 0; i < window; i++) {
747              float weight = std::abs(mult[i]);
748              sum_ifreq += mult_ifreq[i] * weight;
749              sum_weight += weight;
750          }
751          float avg_ifreq = sum_ifreq / sum_weight;
752          float cfo = avg_ifreq / (2.0f * M_PI) * d_samples_per_second;
753          return cfo;
754      }

```

Figure 4 Code modifié

Les deux méthodes permettent de calculer le CFO, mais chacune répond à des besoins différents. La première méthode est adaptée aux applications nécessitant une exécution rapide, tout en tolérant une certaine marge d'erreur. En revanche, la deuxième méthode est plus précise et convient aux cas où une grande exactitude est requise.

Étant donné que mon objectif final est d'utiliser le CFO pour identifier l'origine des transmissions et distinguer les différents nœuds, une estimation précise est essentielle. C'est pourquoi j'ai choisi d'implémenter la deuxième méthode, qui offre une meilleure fiabilité pour cette application.

Une fois cette modification effectuée, il suffit de recompiler gr-lora, puis d'exécuter à nouveau le programme pour afficher le CFO dans le terminal.

```

01 20 10 37 (7)
***** MESSAGE DEBUG PRINT *****
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 54 0 1 32 16 55]
*****
etting CFO 8027.26
etting CFO -7889.96
01 20 10 38 (8)
***** MESSAGE DEBUG PRINT *****
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 55 0 1 32 16 56]
*****
etting CFO 7702.44
etting CFO -5058.59
02 20 e0 32 33 (23)
***** MESSAGE DEBUG PRINT *****
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 52 0 2 32 224 50 51]
*****
etting CFO 4584.65

```

Figure 5 résultat

Conclusion et perspectives

Ce projet nous a permis d'exploiter GNU Radio et bladeRF pour analyser les signaux LoRa, en mettant l'accent sur l'estimation et l'affichage du Carrier Frequency Offset (CFO). Après avoir configuré le système et intégré les outils d'analyse, nous avons apporté des modifications au module gr-lora afin de permettre l'affichage du CFO dans le terminal. Deux méthodes d'estimation ont été comparées : une méthode rapide mais moins précise, et une méthode basée sur une moyenne pondérée, que nous avons choisie pour sa meilleure précision et robustesse.

Cette étude valide l'utilisation du CFO comme une signature unique pour l'identification des nœuds émetteurs LoRa et ouvre la voie à des applications avancées, telles que l'optimisation des transmissions dans les réseaux IoT et l'amélioration des mécanismes de gestion des réseaux LoRaWAN.

Dans l'avenir, cette méthode pourrait être étendue à des environnements plus complexes en testant sur des réseaux plus larges et diversifiés. Il serait également intéressant d'intégrer des algorithmes de machine learning plus avancés pour améliorer encore la précision de l'identification et explorer des approches en temps réel pour un déploiement pratique dans des systèmes IoT.