First French Gnu Radio days Tutorial: Écriture de bloc GNU radio (*OOT block*)

Durée encadrée : 1h30

But du tutorial

L'objectif du tutorial est de comprendre le mécanisme permettant de développer de nouveaux blocs GNU radio, qui sont appelés "Out of tree modules" (OOT).

Ce tutorial a été monté à partir de l'ancienne page wiki GNURadio expliquant la création de modules "Out of Tree" (OOT), la nouvelle version se trouve ici: `https://wiki.gnuradio.org/index.php/OutOfTreeModules`.

Ce tutorial suppose que vous avez des connaissances basiques en python et C++ (Python et C devrait suffire). et que vous êtes famillier avec Linux et les commande `bash` de base.

1 Environnement de travail

Vous travaillez dans une salle du département Télécommunication de l'Insa de Lyon (`https://telecom.insa-lyon.fr/`), les machines installées dans cette salle sont dotées d'une distribution Linux `Suze` sur laquelle est installée une version de GnuRadio compatible avec le tutorial.

Vous pouvez suivre les tutorials sur votre propre machine mais dans ce cas:

- il faut vérifier que vous ayez une version compatible, notamment que vous ayez installé les pilotes pour le module RTL-SDR.
- Si vous faites le tutorial CorteXlab, il faut se créer préalablement un login pour pouvoir se logger sur la plateforme (et installer sa clef `ssh` sur le serveur CorteXlab)

2 Mise en place de l'environnement de travail

1. Loguez-vous sur la machine TC, avec le login `tc_invite` (le mot de passe vous sera donné par l'encadrant).
2. Lancer une fenêtre de commande
3. Lancer Gnuradio companion (commande: `gnuradio-companion`),
4. creez un flow graph très simple qui génére une sinusoide et l'affiche avec un bloc sink QT GUI Time sink.
5. Récupérez le fichier `oot-env.sh` disponible sur Moodle, copiez son contenu dans le fichier `$HOME/.bashrc`. Ce fichier contient les instruction bash nécessaire pour que gnuradio puisse installer des blocs localement (dans votre compte, répertoire `$HOME/.local`). Lancer une nouvelle fenêtre de comande et vérifier que les commandes ont été prises en compte (`echo $GNURADIO_INSTALL_PREFIX` par exemple)

Nous allons proposer la creation d'un nouveau bloc simple qui élève simplement le signal au carré (chaque échantillon sera élevé au carré). Dans Gnu Radio, Chaque bloc appartient à un module, nous allons donc créer un module intitulé `gr-grconf` et un bloc nommé `carre` dans ce module (évidement sans accent pour le nom du bloc).

3 Prise en main de l'outils `gr_modtool`

Gnu radio propose un outil d'aide à la conception de nouveaux modules: `gr_modtool`, cet outil va produire un certain nombre de fichiers que vous modifierez pour les adapter à vos besoins.

- Taper `gr_modtool help` puis `gr_modtool help newmod`

Une fois que `gr_modtool` a produit les fichiers pour le module et que vous les avez spécialisés, le module est compilé en utilisant l'outil `cmake`. `cmake` fournit un ensemble d'outils permettant de compiler un projet pour différentes plate-formes, de faire des tests et de créer des packages pour différents systèmes. Il est utilisé dans de nombreux projets tels que KDE et MySQL et est une bonne alternative aux *autotools*.

Le principe est que `cmake` procède en deux étapes: dans la première étape, les fichiers utilisés pour la construction du projet (par exemple des fichiers `Makefile`) sont produits, et dans la deuxième étape ces fichiers sont exécutés pour construire le projet. Chaque projet construit possède un fichier `CMakeLists.txt` ayant une syntaxe propre pour exprimer les commandes exécutées pour mettre en place les fichiers de construction produits.

4 Création d'un bloc simple

Nous allons créer un bloc simple qui élève simplement un signal au carré. Nous allons d'abord creer un repertoire `$HOME/.local` dans lequel nous allons ajouter nos bloc OOT (Out of Tree).

- Créez le répertoire `$HOME/.local`
- Créez un repertoire de travail pour le tutorial par exemple `$HOME/OOT`
- placez vous dans ce répertoire `OOT`

4.1 Création du module

- créez un nouveau module `grconf`:
  `gr_modtool newmod grconf`
- Allez dans le répertoire `gr-grconf`, familiarisez vous avec la hiérarchie.

4.2 Ajout d'une bloc dans le module

- Créez un bloc de type général intitulé `carre`, en répondant `cpp` à la première question et en tapant retour chariot aux autres questions:
  `gr_modtool add -t general carre`

- Le fivhoer `python/qa_carre.py` contient les test qui seront effectué en Python sur le nouveau bloc. Compléter ce fichier en créant une méthode de test de votre bloc, par exemple (vous pouvez récuperer ce code sur le site de la conférence):

```
def test_001_t (self):
    # set up fg
    src_data = (-3, 4, -5.5, 2, 3)
    expected_result = (9, 16, 30.2, 4, 9)
    src = blocks.vector_source_f(src_data)
    sqr = grconf.carre()
    dst = blocks.vector_sink_f()
    self.tb.connect(src, sqr)
    self.tb.connect(sqr, dst)
    self.tb.run()
    # check data
    result_data = dst.data()
    self.assertFloatTuplesAlmostEqual(expected_result, result_data, 6)
```

Note: `assertFloatTuplesAlmostEqual` est une méthode provenant de `gr_unittest` qui est une extension de python standard permettant de tester l'égualité *approximative* de t-uples de nombre flottants ou complexes.

- Comme le bloc `qa_carre` existait déjà, nous n'avons pas besoin de modifier le fichier `python/CMakeLists.txt`, visualisez le fichier `python/CMakeLists.txt`.
- créez le répertoire `gr-grconf/build`, allez dans ce répertoire
  - La commande habituelle est: `cmake ../`
  - mais commme nous créons des blocs OOT, nous allons utiliser la commande:
    `cmake -D CMAKE_INSTALL_PREFIX=$HOME/.local ../`
- **Note 2017:** Il semble qu'il y ait un bug sur la version installée sur `suze` , demandez à l'encadrant, vous pouvez avoir besoin de faire les opérations suivantes:
  - Editez le fichier `cmake/Modules/GrTest.cmake` et faites: *adding "" around $... at line 48 in file GrTest.cmake* (cf `https://forums.opensuse.org/showthread.php/511435-cmake-error-(gr_modtool)`).
    Plus précisément: remplacer la ligne 48 du fichier `cmake/Modules/GrTest.cmake`:
    `string(REGEX REPLACE "\\$\\(.*\\)" ${CMAKE_BUILD_TYPE} path ${path})`
    par:
    `string(REGEX REPLACE "\\$\\(.*\\)" "${CMAKE_BUILD_TYPE}" path "${path}")`
    i.e.: rajouter des guillement autour des variables d'environnement.
- Une fois que le `cmake` a fonctionné sans erreur, faites `ls`, juste pour visualiser les fichiers produits.

4.3 Compilation du bloc

- Tentez de la construire (commande `make`), essayez de comprendre le message d'erreur pour savoir quel fichier modifier pour que la compilation fonctionne.
- Complétez le fichier `gr-grconf/lib/carre_impl.cc` entre les chevron < et >:
  - combien de ports (1 en entrée, 1 en sortie) de quel type (float).
  - Complétez la méthode `forecast` selon le commentaire proposé.
  - Dans la méthode `general_work` remplacer les types entre chevron et ajouter le traitement effectué sur chaque échantillons (i.e. elevez le signal au carré):

```
for (int i = 0; i < noutput_items; i++) {
          out[i] = in[i] * in[i];
}
```

- Construisez le bloc (`make` dans le répertoire `build`)
- Testez le `make test`, le résultat du test peut être visualisé: `less Testing/Temporary/LastTest.log`
- débuggez votre bloc...

4.4 Création d'un bloc GRC pour le nouveau bloc

- Créez un bloc `gnuradio-companion`, dans le répertoire `gr-grconf`:
  `gr_modtool makexml carre`
- compilez le et installez le (`make` et `make install` dans `build`), vérifier que vous le voyez apparaître dans `gnuradio-companion`.
- Créez un flow graph simple `grc` avec ce nouveau bloc et testez que

5 Création d'un bloc pipeliné

Nous allons maintenant créer un filtre audio, `filter1` en répétant ces mêmes opérations, sauf qu'à la création le filtre héritera du bloc `gr_sync_block` pour avoir la methode `set_history(nbdata)` qui sert à avoir une fenetre glissante sur `nbdata` échantillons:
`gr_modtool add -t sync filter_noise`

L'objectif de cette section est d'implémenter un filtre FIR (et/ou IIR) qui permet de filtrer le bruit additif gaussaient. L'équation d'un filtre FIR est :

$$y(n) = \sum_{k=0}^{N-1} c_k x(n-k)$$

où y est le signal à la sortie du filtre et les c_k sont les coefficients du filtre. N est l'ordre ou la longueur du filtre.

Le filtre sera généré sous matlab (fdatool) et les coefficients c_k vous seront fournis. Le travail qu'il vous reste à faire est de créer un module OOT (out-of-tree) dans lequel vous implémenterez l'équation du filtre en utilisant bien évidement les coefficients c_k. Vérifiez l'éfficacité de votre filtre pour filtrer le bruit.

6 En fonction du temps restant...

Vous pouvez penser également, si vous avez le temps, à implémenter un filtre IIR qui fera le même travail que le filtre FIR. Comparez les deus filtres en termes de stabilité, difficulté d'implémentation,...?

L'équation d'un filtre IIR est :

$$y(n) = \sum_{k=0}^{N-1} b_k x(n-k) - \sum_{m=1}^{M-1} a_k y(n-m)$$