Introduction au Middleware MOOS

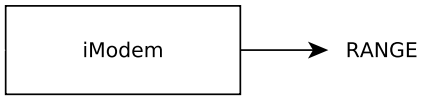
TP Robotique - Session MOOS-IvP #1 - Mars 2016

Supports de cours disponibles sur
`www.simon-rohou.fr/cours/moos-ivp`

1 Implémentation d'une application MOOS

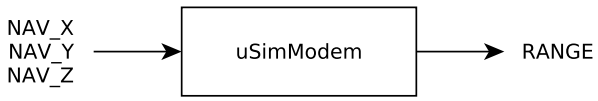
On se place dans le cas d'un véhicule sous-marin cherchant à se localiser par rapport à une balise fixe émettant des signaux acoustiques. Les données perçues par le sous-marin sont des informations de distance (approche *range-only*).

Un modem acoustique est un capteur permettant, notamment, de calculer des distances sous l'eau. Dans la pratique, une application MOOS nommée `iModem` ferait l'interface entre ce capteur et des algorithmes de localisation embarqués. Une variable `RANGE` serait publiée dans la MOOSDB à chaque réception de signal.



En simulation, il faut représenter cette acquisition de données.

Objectif : création d'une nouvelle application MOOS nommée `uSimModem` et.. simulant un modem. Le capteur simulé est fixé à un véhicule sous-marin qui se déplace. On simulera ce déplacement en changeant les valeurs des variables `NAV_X`, `NAV_Y`, `NAV_Z` à l'aide de `uPokeDB`. Ces trois variables représentent la position exacte du sous-marin et ne sont connues qu'en simulation. On renseignera la position (x, y, z) de la balise émettrice en précisant de nouveaux paramètres dans le fichier `.moos`.



Le paramètre `AppTick` permettra au robot de recevoir une nouvelle variable `RANGE` toutes les 2 secondes.

2 Intégration d'un observateur d'état par intervalles dans une communauté MOOS

On se propose d'intégrer les résultats du TD sur CMake dans des applications MOOS. La correction de la `q2` est disponible sur `https://github.com/benEnsta/TD_CMake`.

- proposer une architecture séparant les différentes briques de simulation (la MOOSApp `uSimModem` ainsi que les MOOSVar `NAV_X`, `NAV_Y`, `NAV_Z`, `RANGE` pouvant être réutilisées) ;
- écrire le fichier `.moos` correspondant dans un nouveau répertoire de `mission` ;
- implémenter les applications MOOS avec la correction du TD CMake disponible sur Github.

Notes pour les utilisateurs de Linux

Installation de MOOS-IvP et du répertoire projet

Pour installer MOOS-IvP, vous aurez besoin de SVN :

```
> sudo apt-get install subversion
```

En cas de proxy, editer le fichier `/etc/subversion/servers` :

```
[Global]
http-proxy-host=my.proxy.com
http-proxy-port=3128
```

Installation de `moos-ivp` :

```
> sudo apt-get install g++ subversion xterm cmake
                       libfltk1.3-dev freeglut3-dev libpng12-dev
                       libjpeg-dev libxft-dev libxinerama-dev libtiff5-dev
> svn co https://oceanai.mit.edu/svn/moos-ivp-aro/releases/moos-ivp-15.5 ~/moos-ivp
> cd ~/moos-ivp
> ./build.sh
```

Configuration des chemins d'accès : les applications MOOS doivent être accessibles depuis le `PATH` du système. Pour cela, ajouter à la fin de `~/.bashrc` :

```
export PATH=$PATH:~/moos-ivp/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/moos-ivp/lib
export PATH=$PATH:~/moos-ivp-extend/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:~/moos-ivp-extend/lib
```

Préparation du répertoire de travail `moos-ivp-extend` :

```
> source ~/.bashrc
> svn co https://oceanai.mit.edu/svn/moos-ivp-extend/trunk ~/moos-ivp-extend
> cd ~/moos-ivp-extend
> ./build.sh
```

Création d'une nouvelle MOOSApp

Les nouvelles MOOSApp sont à créer dans `~/moos-ivp-extend/src/` à l'aide de `GenMOOSApp_AppCasting`
Une ligne doit alors être ajoutée dans `~/moos-ivp-extend/src/CMakeLists.txt`

Configuration d'une mission

Par convention, l'exécution des applications MOOS se fait dans le répertoire `mission`. Un fichier `test_modem.moos` est à créer dans `~/moos-ivp-extend/missions/test_modem/`. Son exécution avec `pAntler` exécutera les MOOSApp renseignées dans cette configuration.