

Le Middleware MOOS-IvP

TP Robotique - Session MOOS-IvP #3 - Avril 2016

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

Une mission autonome dans l'anse du Moulin Blanc

Dans cet exercice, l'objectif est de rendre autonome le robot de surface préparé lors du TP 2.

Les applications suivantes seront de nouveau utilisées :

- `pMarineViewer` pour l'affichage du robot sur une carte
- `uSimMarine` pour la simulation du robot
- `pNodeReporter` pour faire une synthèse de l'état du robot

Nous ajouterons les MOOSApp dédiées à l'autonomie :

- `pMarinePID` pour la régulation du robot en vitesse / cap
- `pHelmIvP` pour la partie intelligence

L'architecture suivante est proposée :

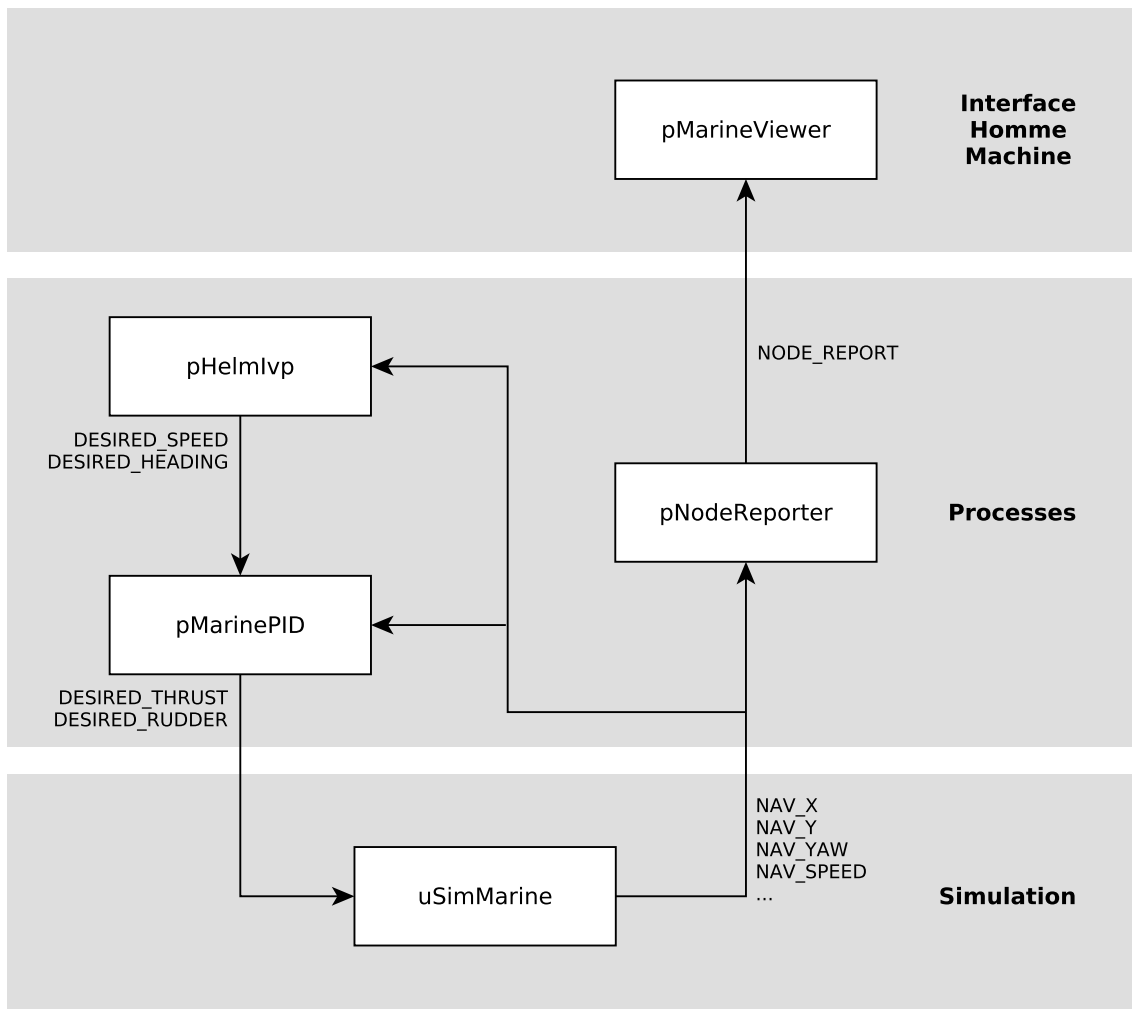


Figure 1: Organisation de la communauté MOOS — les échanges de variables MOOS entre les applications (représentées par des flèches) se font en réalité *via* la MOOSDB, non représentée ici. Hors simulation, des capteurs et algorithmes de localisation fourniraient une estimation de l'état du robot à `pHelmIvP` et `pMarinePID`.

On se propose la mission suivante. Le bateau, stationné dans le port du Moulin Blanc, doit sortir sans encombre pour effectuer une mission de reconnaissance dans l'anse du même nom. La mission consiste à maintenir une position donnée au centre de la zone¹. Toutefois, le robot devra faire face à de forts courants marins. À la fin de la mission, on souhaite le retour du robot au port, en toute autonomie.

Le découpage suivant nous permet de structurer la mission :

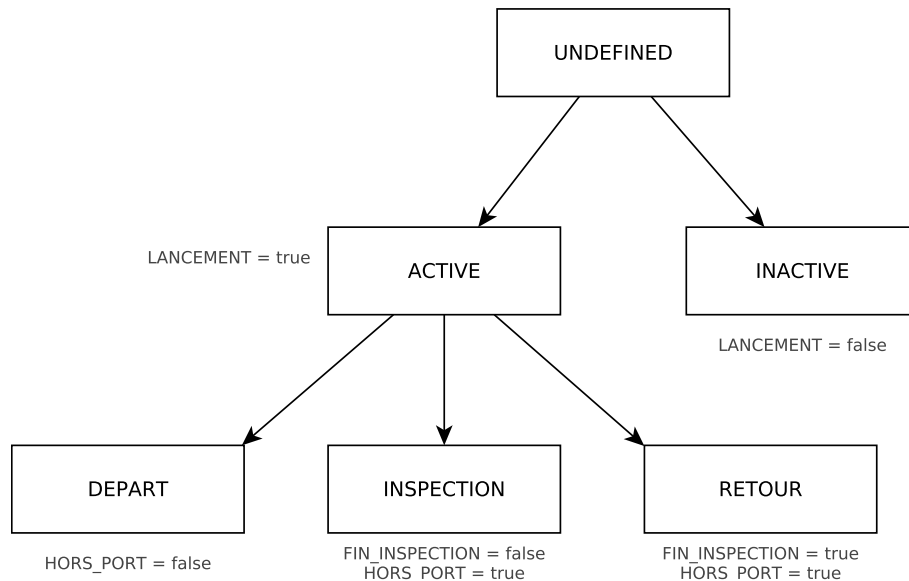


Figure 2: Les différents modes à configurer pour pHelmIvP dans un fichier .bhv.

1. Mise en place

Reprendre le répertoire projet du TP 2 (correction disponible sur www.github.com/SimonRohou/td-moos-ivp).

- conserver uniquement les MOOSApp listées précédemment
- dans uSimMarine, positionner le robot en $x = 112$, $y = -128$, $\theta = 90$
- dans pMarineViewer, configurer `trails_length` à 1000 et `trails_connect_viewable` à true
- enfin, ajouter la configuration globale `MOOSTimeWarp = 10` dans l'en-tête du fichier .moos. Cette instruction permet d'accélérer 10 fois la simulation.

Lancer la mission : le robot doit normalement être immobile dans la marina.

2. Régulation

Ajouter le bloc pMarinePID dans le fichier de mission. On conservera les paramètres donnés par la commande pMarinePID -e.

3. Interface

Configurer pMarineViewer pour ajouter un bouton *Lancement* (se référer à la documentation ou au cours 2). Un clic sur ce bouton publiera `LANCEMENT=true` et `MOOS_MANUAL_OVERRIDE=false`. Ajouter un second bouton permettant de publier manuellement `FIN_INSPECTION=true`.

4. IvP

Ajouter un bloc pHelmIvP s'intéressant aux domaines de décision `speed` (de 0 à 5, sur 20 valeurs) et `course` (de 0 à 359, sur 360 valeurs). La configuration se fait dans le fichier .moos tandis que celle des behaviors se fera dans le fichier moulin_blanc.bhv, à créer.

5. Modes

Dans ce nouveau fichier, définir les variables communes suivantes :

- `LANCEMENT = false`
- `HORS_PORT = false`
- `FIN_INSPECTION = false`

¹Probablement pour surveiller des étudiants en Hydrographie, se trouvant non loin de là.

Puis, en s'aidant de la Fig. 2, configurer les différentes sous-missions (*modes*) qui organiseront les behaviors. S'aider du cours et des exemples disponibles sur la documentation IvP, Section 7. *IvP Helm Autonomy*.

6. Sortir du port

Pour le mode DEPART, ajouter la behavior BHV_Waypoint permettant au robot de sortir de la marina en suivant les waypoints suivants :

```
(195,-128) (300,-18) (360,115) (600,115)
```

- configurer la vitesse du robot à 5
- une fois les waypoints atteints, la behavior devra lever le *endflag* HORS_PORT = true, ce qui permettra à pHelmIvP de passer au mode suivant
- tester et vérifier le comportement du robot

7. Retourner au port

Pour le mode RETOUR et de la même manière que pour la question 6, ajouter la behavior BHV_Waypoint permettant au robot de retourner à sa position initiale. Tester et vérifier que le robot fasse correctement l'aller-retour. Cliquer sur "Fin Inspection" dans pMarineViewer pour passer l'étape de l'inspection.

8. Simulation de courants marins en dehors du port

L'application uSimMarine s'abonne aux variables DRIFT_X, DRIFT_Y décrivant un courant marin². Ainsi, en publiant DRIFT_X=2.0 dans la MOOSDB, le robot sera soumis à un courant de force 2 selon x . Ici, on souhaite un courant égal à (1.2, 0.3) et nul lorsque le robot est abrité dans le port.

En ajoutant 2 instances de uTimerScript³ et en utilisant les conditions sur les variables, simuler ce courant marin :

- première instance : courant nul lorsque le robot est dans le port : NAV_X < 400
- seconde instance : courant de (1.2, 0.3) lorsque le robot est hors du port : NAV_X > 400

9. Maintien en position au centre de la carte

Une fois le robot en dehors du port, il est soumis au courant. Dans le mode INSPECTION, ajouter la behavior BHV_StationKeep permettant au robot de se maintenir en $x = 800$, $y = 0$ malgré le courant. La position doit être maintenue par le robot pendant 300s. On prendra la configuration suivante :

```
inner_radius=10.0  outer_radius=30.0  hibernation_radius=60.0
outer_speed=8.0   transit_speed=5.5
```

Au bout des 300s, la behavior devra lever le *endflag* FIN_INSPECTION = true.
Tester le déroulement complet de la mission.

Questions complémentaires

10. Rails d'acquisition

Ajouter un nouveau mode nommé EXPLORATION. Dans ce mode, le robot devra effectuer une série d'acquisition (par sonar) en faisant des rails réguliers.

- ajouter une nouvelle behavior permettant de faire des rails dans l'anse du Moulin Blanc
- cette sous-mission doit avoir lieu après le maintien en position et avant le retour au port
- le robot est toujours soumis aux courants...

11. Visualisation de la zone explorée

En utilisant l'application pSearchGrid, afficher une grille sur la zone d'exploration et visualiser la zone couverte par le robot.

²La simulation de courants peut aussi être faite par uSimCurrent si l'on dispose d'un jeu de données modélisant une perturbation.

³Se référer au cours 1 pour planifier plusieurs instances d'une même MOOSApp.