Universal Robot ROS2 Driver

Le Driver ROS2 pour les cobots Universal Robot a été développé en collaboration entre Universal Robots, le Forschungszentrum für Informatik (INRIA allemand) et PickNik Robotics. Plus précisément par nos voisins de Karlsruhe, en particulier Felix Exner (https://github.com/fmauch)). Le FZI est également à l'origine d'une proposition d'interface ROS standard pour les trajectoires Cartésiennes, qui est désormais implémentée dans les contrôleurs ROS Cartésiens d'Universal Robots. La proposition repose sur un état de l'art très intéressant des interfaces de programmation des marques de robot majeures.

Installation des paquets UR pour ROS2

- Réaliser l'installation de ROS2.
- Installer les paquets ros2 de Universal Robots

```
sudo apt install ros-humble-ur
```

- Installer rosdep pour installer automatiquement les paquets Debian dont dépendent les paquets ROS

```
sudo apt install python3-rosdep
```

- Si vous avez plusieurs versions de ROS2 installées, vérifiez que vous avez "sourcé" la bonne version de ROS2
- Mettre à jour les paquets dont ROS2 et les paquets UR dépendent

```
sudo rosdep init
rosdep update
sudo apt update
sudo apt dist-upgrade
```

Installation du simulateur hors-ligne URSim

https://github.com/UniversalRobots/Universal_Robots_ROS2_Driver#getting-started

https://github.com/UniversalRobots/Universal_Robots_ROS2_Driver#install-from-binary-packages

Le moyen le plus simple de découvrir et commencer à utiliser un robot UR avec ROS2 est d'utiliser la simulation de la console de programmation (teach panel) de son contrôleur. URSim est le simulateur hors-ligne de Universal Robots. Il permet de reproduire le comportement réel d'un robot quasiment à l'identique en se connectant à son adresse IP. Il est possible d'installer URSim 5 sur un Ubuntu <18 (non inclus!) ou dans une machine virtuelle (VirtualBox). Il faut se créer un compte pour télécharger le fichier URSim 5.13. Ubuntu 18 n'est plus supporté, et MoveIt2 tourne sur Ubuntu 22.

Sur Ubuntu 22

En attendant la sortie de URSim/Polyscope 6, **la manière la plus simple d'installer URSim 5 sur Ubuntu 22 est via conteneur Docker créé par UR Lab (pas maintenu officiellement)**.

Installer Docker

Depuis les dépôts officiels Ubuntu :

```
sudo apt update
sudo apt install docker-compose
```

Ajouter votre utilisateur au groupe docker afin de manipuler les containers sans avoir à utiliser sudo systématiquement :

```
sudo usermod -aG docker $USER
```

Fermer et rouvrir la session. Tester la bonne installation :

```
sudo service docker start
docker run hello-world
```

Installer le conteneur URSim

On suit le tutoriel fourni dans la documentation du driver UR ROS. Il existe une image docker fournie sur hub.docker.com, c'est très simple :

- Télécharger l'image docker

```
docker pull universalrobots/ursim_e-series
```

- Lancer l'image sur 192.168.56.101 avec l'URCap external_control préinstallé. Les programmes installés et les changements d'installation seront stockés de manière persistante dans `${HOME}/.ursim/programs`. Par exemple `-m ur5e`

```
ros2 run ur_robot_driver start_ursim.sh -m <ur_type>
```

- Ouvrir l'interface URSim en suivant les instructions qui s'affichent dans la fenêtre de terminal
  1. Voir URSim en utilisant une application VNC, en se connectant à `192.168.56.101:5900`
  2. Voir URSim en utilisant une application serveur web VNC

     http://192.168.56.101:6080/vnc.html

Sur Windows 10/11

Pour les systèmes non-Linux, UR fournit une VM LUbuntu 14.04 qui peut tourner sous VirtualBox (gratuit) ou sur VMware (gratuit pour usage non commercial).

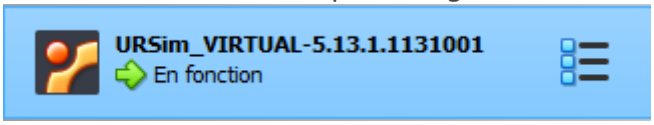
Installer VirtualBox

Télécharger et configurer la Machine Virtuelle

- Télécharger l'archive de la VM sur seafile ou en créant un compte chez UR
- Extraire l'archive dans `C:\Users\user\VirtualBox VMs\URSim_VIRTUAL-5.13.1.1131001`
- Dans l'interface VirtualBox cliquer sur `Ajouter`

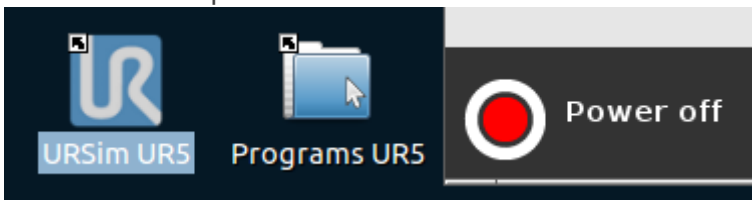


- Sélectionner le Fichier machine virtuelle (`.vbox`) `C:\Users\user\VirtualBox VMs\URSim_VIRTUAL-5.13.1.1131001\URSim_VIRTUAL-5.13.1.1131001.vbox`
- La VM est entièrement préconfigurée, il suffit de la lancer

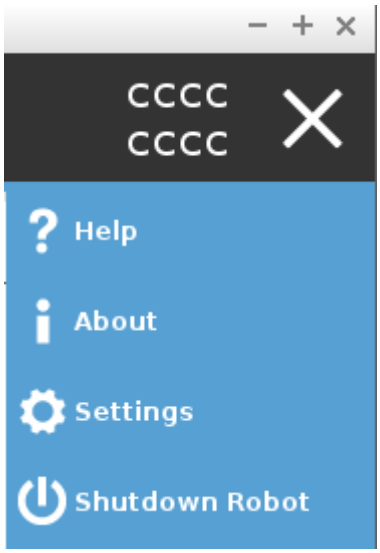


- La session Lubuntu est
  - user : `ur`
  - mdp : `easybot`

- Lancer URSim pour l'UR5e



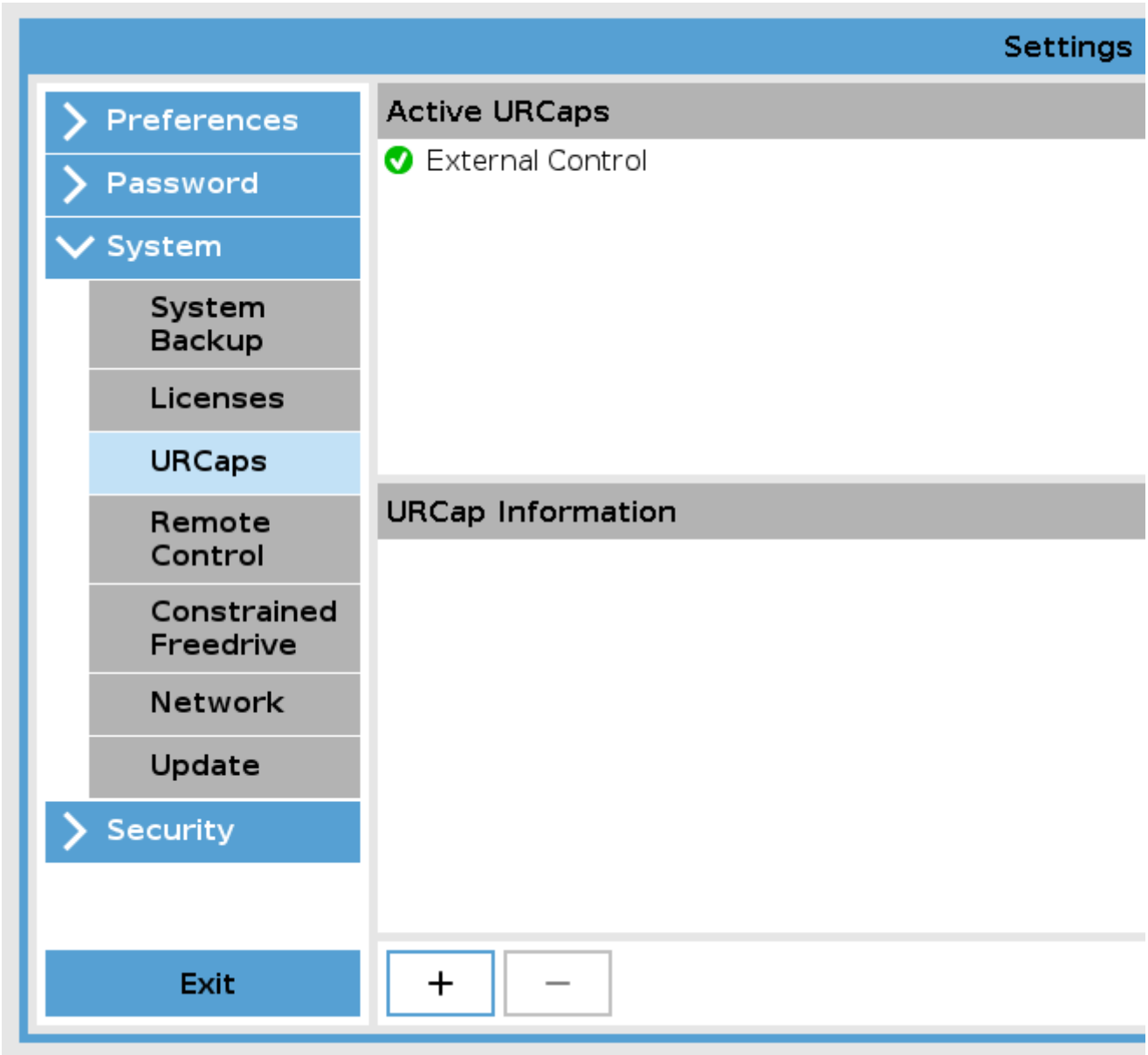
- Démarrer le robot en cliquant sur le bouton d'allumage "Power"
- Pour éteindre le robot



- Ou simplement éteindre la VM

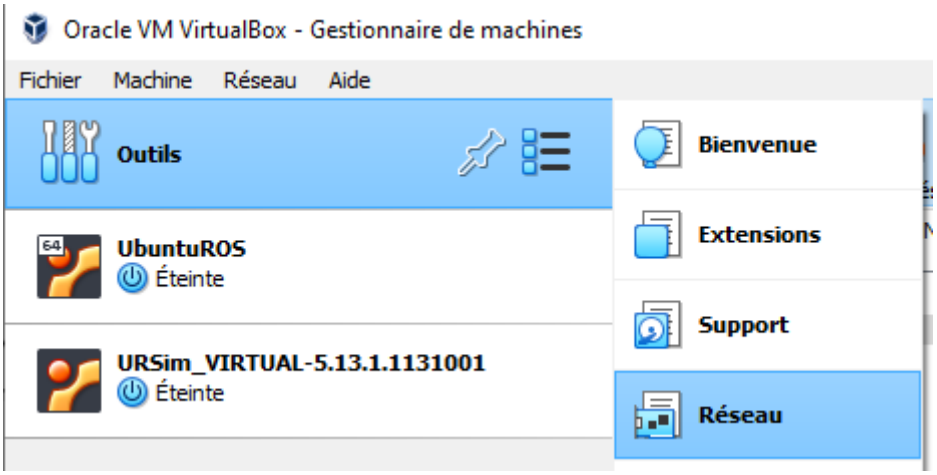
Ajouter l'URCap External Control

- Installer Terminator qui permet de faire des copier-coller
  `sudo apt install terminator`
- Se placer dans le dossier d'échange qui fonctionne comme une clé USB
  `cd /home/ur/ursim-current/programs.UR5`
- Lancer Terminator et télécharger le fichier `.urcap`
  `wget https://github.com/UniversalRobots/Universal_Robots_ExternalControl_URCap/releases/download/v1.0.5/externalcontrol-1.0.5.urcap`
- Démarrer URSim, ajouter l'URCap et redémarrer URSim

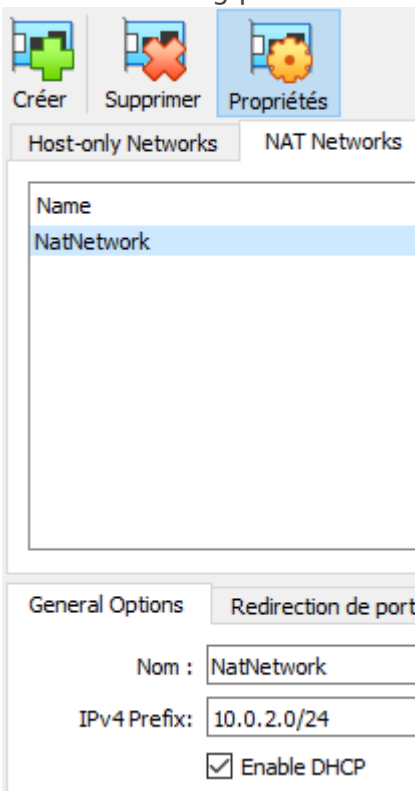


Configurer le réseau pour que les VMs communiquent

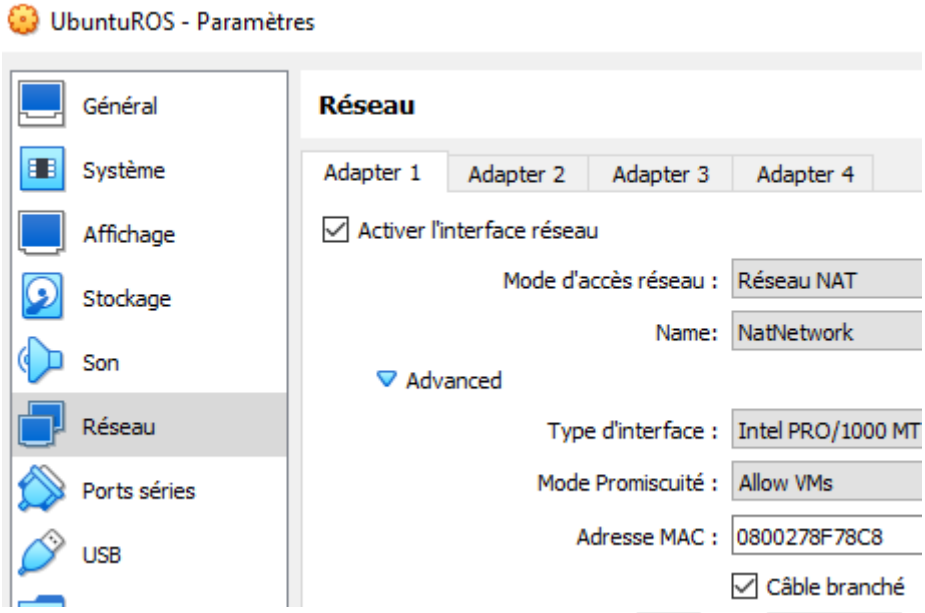
- Créer un réseau NAT dans les outils réseau



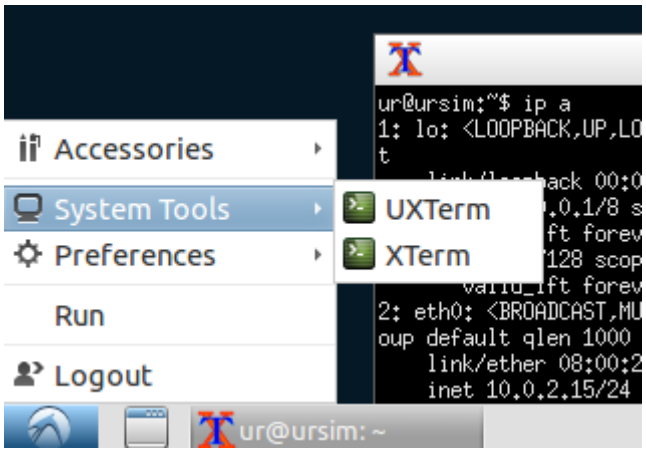
- Laisser la config par défaut



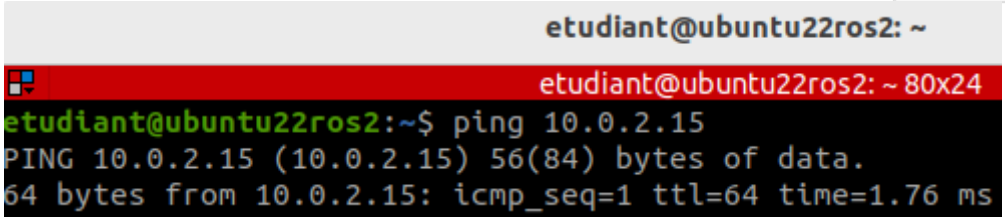
- Configurer chaque VM qui doit communiquer en `Réseau NAT`, par défaut elles seront en DHCP sur le réseau `10.0.2.0/24`



- Récupérer l'adresse IP de la VM URSim avec `ip a`, ici `10.0.2.15`



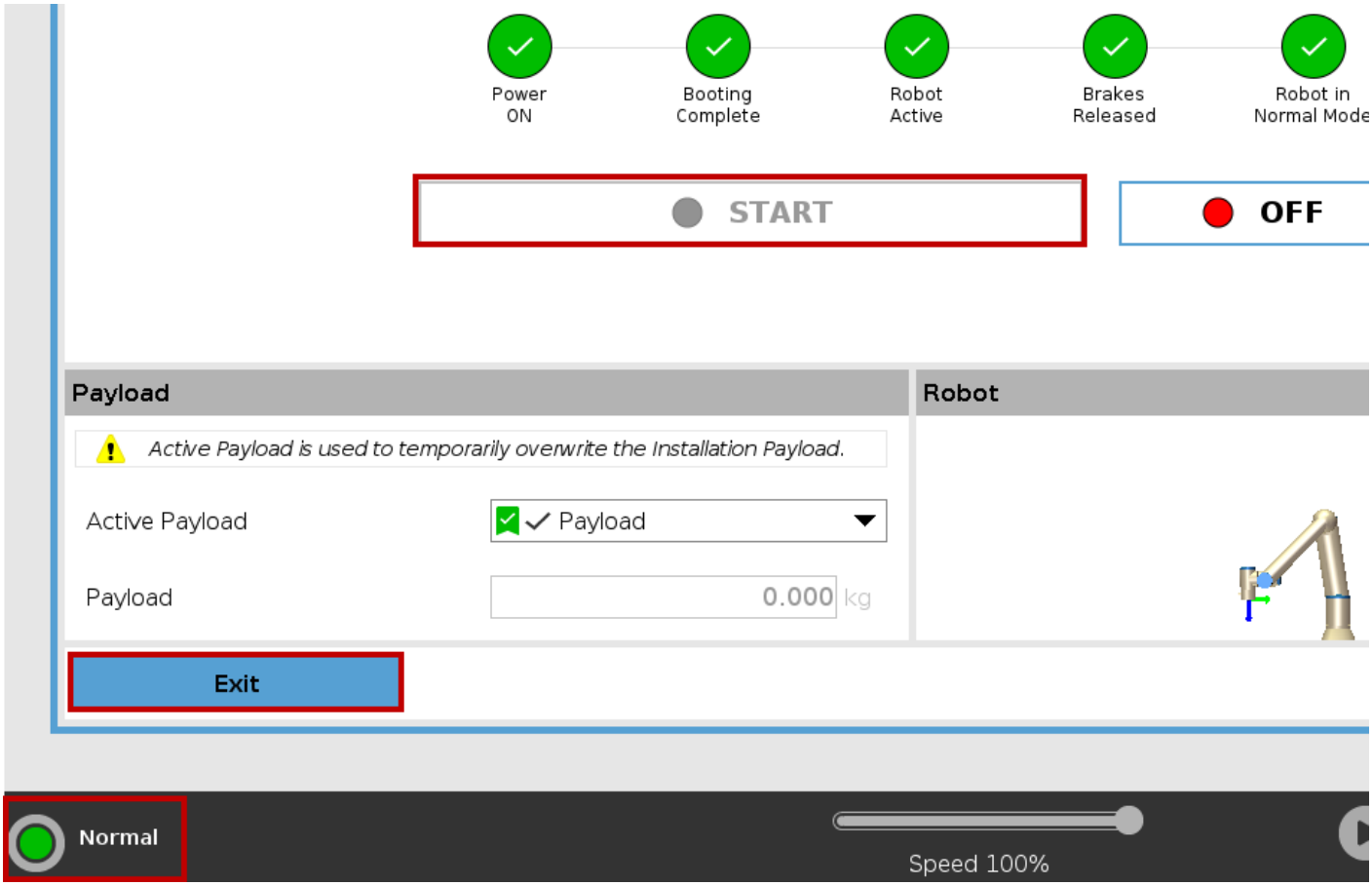
- Tester la communication de la VM ROS vers la VM URSim avec `ping 10.0.2.15`



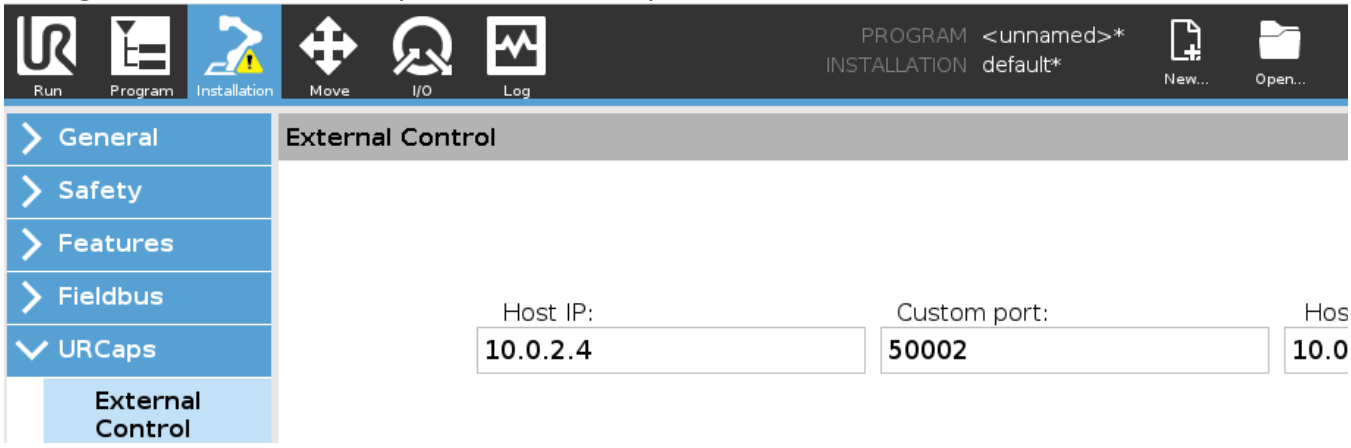
- Tester la communication de la VM URSim vers la VM ROS avec `ping 10.0.2.4`
- Tester la communication entre ROS et URSim, voir la section dédiée

Démarrage et configuration URSim

- démarrer le robot



- Configuer l'IP de la VM ROS pour l'autoriser à prendre le contrôle externe



-

Programmation hors-ligne avec URSim et MoveIt2/RViz

- Réaliser l'installation de ROS2.
- Installer rosdep pour installer automatiquement les paquets Debian dont dépendent les paquets ROS

```
sudo apt install python3-rosdep
```

- Si vous avez plusieurs versions de ROS2 installées, vérifiez que vous avez "sourcé" la bonne version de ROS2
- Mettre à jour les paquets dont ROS2 dépend

```
sudo rosdep init
rosdep update
sudo apt update
sudo apt dist-upgrade
```

- Installer colcon qui est le système de compilation de ROS2 avec mixin.

```
sudo apt install python3-colcon-common-extensions
sudo apt install python3-colcon-mixin
colcon mixin add default https://raw.githubusercontent.com/colcon/colcon-mixin-repository/master
colcon mixin update default
```

NOTES IMPORTANTES

- si vous utilisez Linux Mint 21.1 VERA (Mate ou Cinammon, basée sur Ubuntu 22 Jammy), il faudra toujours préciser la version d'Ubuntu `--os=ubuntu:jammy` dont on veut récupérer les paquets avec la commande `rosdep` :

```
sudo apt update && rosdep install -r --from-paths . --ignore-src --rosdistro $ROS_DISTRO -y --os=ubuntu:jammy
```

- Si vous travaillez dans une VM d'un ordinateur ayant moins de 20GO de RAM (WSL par exemple). Ou sur un ordinateur Ubuntu ayant moins de 10GO de RAM, il faudra lancer la compilation `colcon` sans parallélisation des tâches (1 paquet compilé à la fois) `--parallel-workers 1` :

```
colcon build --mixin release --parallel-workers 1
```

Déploiement du code source des Tutoriels de MoveIt

- Créer un répertoire de travail "workspace" pour la compilation du projet avec colcon

```
mkdir -p ~/ws_moveit2/src
```

- Se déplacer dans le workspace colcon et récupérer le code source des tutoriels MoveIt :

```
cd ~/ws_moveit2/src
git clone https://github.com/ros-planning/moveit2_tutorials -b humble --depth 1
```

Optionnel : Installation d'un environnement de développement MoveIt2 avec les dernières améliorations et résolutions de bug

Installation de MoveIt2 Humble sur Ubuntu 22.04 :

- Installer vcstool qui est un outil Python pour gérer les dépôts git composant un paquet ROS.

```
sudo apt install python3-vcstool
```

- Récupérer les autres dépôts git de MoveIt2 dont dépend moveit2_tutorials

```
cd ~/ws_moveit2/src
vcs import < moveit2_tutorials/moveit2_tutorials.repos
```

Note : si `vcs import` vous demande vos identifiants GitHub, tapez Entrer jusqu'à ce que ça continue. Pas besoin d'identifiant pour récupérer un dépôt GitHub public.

Installation des dépendances et compilation

- Installer les dépendances ROS2 Humble (paquets debian stables).

```
sudo apt update && rosdep install -r --from-paths . --ignore-src --rosdistro $ROS_DISTRO -y
```

- Compiler MoveIt Tutorials (20+ minutes si vous avez choisi de déployer l'environnement de développement)

```
cd ~/ws_moveit2
colcon build --mixin release
```

- Sourcer les paquets compilés :

```
cd ~/ws_moveit2
source ~/ws_moveit2/install/setup.bash
```

- Optionnel, si vous utilisez principalement ce workspace : Sourcer automatiquement le workspace colcon au lancement d'un Terminal

```
echo 'source ~/ws_moveit2/install/setup.bash' >> ~/.bashrc
```

Tester la communication entre ROS et URSim

https://docs.ros.org/en/ros2_packages/rolling/api/ur_robot_driver/usage.html#usage-with-official-ur-simulator

- Lancer URSim, par exemple avec un UR5e

```
ros2 run ur_robot_driver start_ursim.sh -m ur5e
```

- Ouvrir l'interface URSim dans le navigateur : http://192.168.56.101:6080/vnc.html --> cliquer sur Connect
- Faire tourner le driver UR ROS2

```
ros2 launch ur_robot_driver ur_control.launch.py ur_type:=ur5e robot_ip:=192.168.56.101
```

- Si vous bougez le robot dans URSim vous le verrez bouger dans RViz.

https://docs.ros.org/en/ros2_packages/rolling/api/ur_robot_driver/usage.html#example-commands-for-testing-the-driver

Envoyer des commandes au contrôleur

Avant d'envoyer des commandes, vérifier l'état des contrôleurs en utilisant `ros2 control list_controllers`

- Envoyer un objectif (goal) au contrôleur de trajectoire articulaire (Joint Trajectory Controller) en utilisant le noeud de démonstration du paquet ros2_control_demos. Dans nouveau Terminal (sans oublier de sourcer) :

  ```
  ros2 launch ur_robot_driver test_scaled_joint_trajectory_controller.launch.py
  ```

  Après quelques secondes le robot devrait bouger.

- Pour tester un autre contrôleur, il suffit de l'ajouter en argument de la commande `initial_joint_controller`, par exemple en utilisant joint_trajectory_controller :

```
ros2 launch ur_robot_driver ur_control.launch.py ur_type:=ur5e
robot_ip:=192.168.56.101 initial_joint_controller:=joint_trajectory_controller
launch_rviz:=true
```

- Et envoyer la commande avec le noeud de démo :

```
ros2 launch ur_robot_driver test_joint_trajectory_controller.launch.py
```

  Après quelques secondes le robot devrait bouger (ou sauter si la simulation est utilisée).

Auteur: Gauthier Hentz, sur le wiki de l'innovation de l'IUT de Haguenau

Attribution-NonCommercial-PartageMemeConditions 4.0 International (CC BY-NC-SA 4.0)

Revision #21
Created 20 April 2023 09:57:27 by admin_idf
Updated 13 December 2024 00:12:19 by admin_idf