Contrôle de l'UR3 avec ROS

Ressources

- http://wiki.ros.org/universal_robot/Tutorials/Getting%20Started%20with%20a%20Universal%20Robot%20and%20ROS-Industrial
- http://wiki.ros.org/action/show/universal_robots?action=show&redirect=universal_robot
- https://github.com/UniversalRobots/Universal_Robots_ROS_Driver

Configuration

- Robot : UR3e
- Logiciel Universal Robot : URSoftware 5.5

1. Installation du driver et des modèles

- Universal_Robots_ROS_Driver : pour contrôler les robots e-series

  Source : http://wiki.ros.org/ur_robot_driver

  Code source et tutoriel d'installation : https://github.com/UniversalRobots/Universal_Robots_ROS_Driver

2. Préparation du robot

Installation de URCap

Tutoriel pour les e-series : https://github.com/UniversalRobots/Universal_Robots_ROS_Driver/blob/master/ur_robot_driver/doc/install_urcap_e_series.md

Le fichier **externalcontrol-1.0.urcap** se trouve dans le répertoire src/Universal_Robots_ROS_Driver/ur_robot_driver/resources.

- Copie du fichier

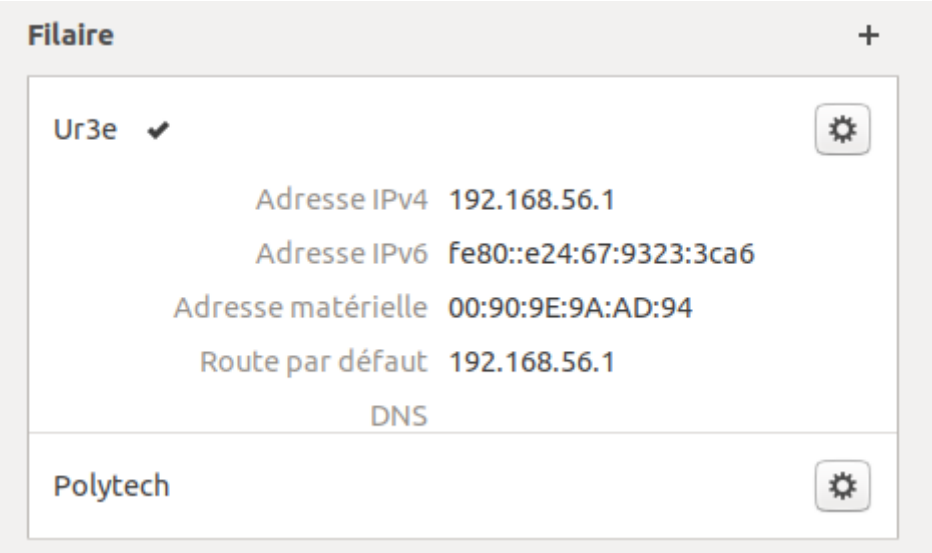
1. Connecter clavier
2. Ctrl + Alt + F1 pour accéder au terminal
3. Login : root , Password : easybot (attention clavier Qwerty, sudo loadkeys fr pour passer en azerty)
4. copie du fichiers dans /programs
5. exit
6. Ctrl + Alt + F7

2. Communication PC-Robot

Tutoriel : http://wiki.ros.org/universal_robot/Tutorials/Getting%20Started%20with%20a%20Universal%20Robot%20and%20ROS-Industrial

- Robot : adresse statique
  - Adresse IP : 192.168.56.2
  - Masque : 255.255.255.0
  - Passerelle : 192.168.56.1
- PC :

Adresse IP : 192.168.56.1 comme spécifié dans le paramétrage de l'URcap installé sur le robot



- Vérification :
  - avec `nmap`

```
$ nmap -n 192.168.56.0-255
```

  - avec `ping`

```
$ ping 192.168.56.2
PING 192.168.56.2 (192.168.56.2) 56(84) bytes of data.
64 bytes from 192.168.56.2: icmp_seq=1 ttl=64 time=0.399 ms
64 bytes from 192.168.56.2: icmp_seq=2 ttl=64 time=0.673 ms
^C
--- 192.168.56.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1011ms
rtt min/avg/max/mdev = 0.399/0.536/0.673/0.137 ms
```

3. Contrôle via ROS

Extraction des données de calibration

```
$ roslaunch ur_calibration calibration_correction.launch robot_ip:=192.168.56.2
target_filename:="${HOME}/catkin_ws_UR3e/my_robot_calibration.yaml"
```

Démarrage avec le fichier de calibration

```
$ roslaunch ur_robot_driver ur3e_bringup.launch robot_ip:=192.168.56.2
kinematics_config:="${HOME}/catkin_ws_UR3e/my_robot_calibration.yaml"
```

Contrôle avec `MoveIt`

1. Exécuter les mouvements sur le robot réel

```
$ roslaunch ur_robot_driver ur3e_bringup.launch robot_ip:=192.168.56.2
kinematics_config:="${HOME}/catkin_ws_UR3e/my_robot_calibration.yaml"
```

2. Lancer `MoveIt`

```
$ roslaunch ur3_e_moveit_config ur3_e_moveit_planning_execution.launch
```

3. Lancer `RViz` pour générer les trajectoires

```
$ roslaunch ur3_e_moveit_config moveit_rviz.launch config:=true
```

Configuration à postériori (si pas config:=true)

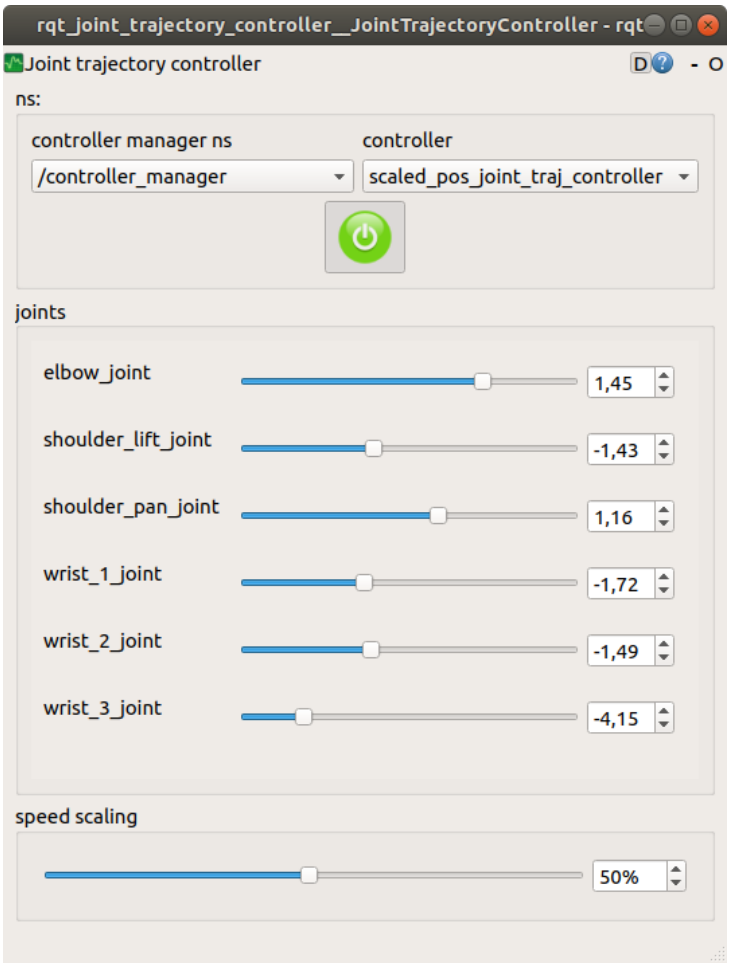
- Fixed Frame : *base_link*
- Ajouter le modèle du robot
- Ajouter le plugin `MotionPlanning`
- Ajouter les affichages des robots
  - Planning Request > Planning Group : sélectionner `manipulator`
  - Tuto MoveIt : http://docs.ros.org/en/melodic/api/moveit_tutorials/html/index.html

Remarques :

- Vérifier la configuration de moveit pour qu'il communique avec les bons contrôleurs (tuto : http://docs.ros.org/en/melodic/api/moveit_tutorials/html/doc/controller_configuration/controller_configuration_tutorial.html)

Contrôle avec `rqt_joint_trajectory_controller`

```
$ rosrun rqt_joint_trajectory_controller rqt_joint_trajectory_controller
```



Contrôle au joystick

1. Robot réel
    - Exécuter le programme `ros-control` sur le robot

      Exécuter > Charger Programme > Ouvrir ros-control.urp
    - Ecrire sur le topic `/joy`

      ```
      $ rosrun joy joy_node
      ```

    - Choisir le mode de contrôle du robot

      doc : https://github.com/UniversalRobots/Universal_Robots_ROS_Driver/blob/master/ur_robot_driver/doc/ROS_INTERFACE.md
        - Quel est le mode de contrôle actuel du robot ?

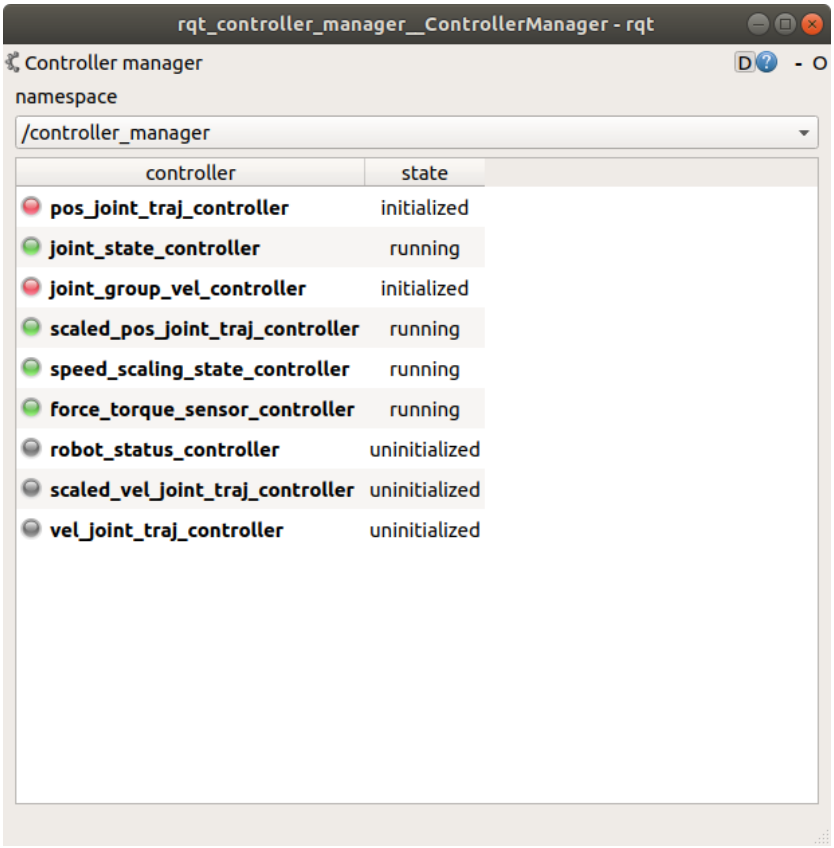
          ```
          $ rosservice call /ur_hardware_interface/dashboard/get_robot_mode
          robot_mode:
            mode: 7
          answer: "Robotmode: RUNNING"
          success: True
          ```

        - Quel est le mode de sécurité du robot ?

          ```
          $ rosservice call /ur_hardware_interface/dashboard/get_safety_mode
          safety_mode:
            mode: 1
          answer: "Safetymode: NORMAL"
          success: True
          ```

        - Lancer le `controller_manager` pour arrêter/démarrer des contrôleurs

          ```
          $ rosrun rqt_controller_manager rqt_controller_manager
          ```



- `pos_joint_traj_controller` :

  Type : position_controllers/JointTrajectoryController

  Contrôle en position de chacune des articulations.

- `joint_state_controller` :

  Type : joint_state_controller/JointStateController

  Publie l'état des articulations du robot.

- `joint_group_vel_controller` :

  Type : velocity_controllers/JointGroupVelocityController

  Contrôle en vitesse de chacune des articulations.

- `scaled_pos_joint_traj_controller` :

  Type : position_controllers/ScaledJointTrajectoryController

  Contrôle en position de chacune des articulations.

- `speed_scaling_state_controller` :

  Type : ur_controllers/SpeedScalingStateController

  Contrôle en vitesse de chacune des articulations.

- `force_torque_sensor_controller` :

  Type : force_torque_sensor_controller/ForceTorqueSensorController

- `robot_status_controller` :

  Type : industrial_robot_status_controller/IndustrialRobotStatusController

- `scaled_vel_joint_traj_controller` :

  Type : velocity_controllers/ScaledJointTrajectoryController

  Contrôle en vitesse de chacune des articulations.

- `vel_joint_traj_controller` :

Type : velocity_controllers/JointTrajectoryController

Contrôle en vitesse de chacune des articulations.

- Lancer le noeud de commande

```
$ rosrun control_ur3e joy_control.py
```