Robotique Open Source

    - Commander un robot UR avec le driver ROS2
    - Programmer un robot avec MoveIt2 - Jumeau Numérique

- Roomba ROS2
    - Connexion physique
    - ROS2
    - Application

- Bras Robot Arduino ROS2
    - Projet d'origine
    - Pilotage des servomoteurs : TTL, RS232, RS485

1 - ROS2 - Démarrage

Installation PC ROS2

ROS est un Middleware Open Source pour développer des applications robotiques. Originellement développé sous Linux (Ubuntu), il est maintenant disponible sur plusieurs systèmes d'exploitation dont Debian et Windows.

Installation des prérequis et liens importants

Pour des raisons de stabilité et légèreté du système, il y a tout à penser que les déploiements de ROS dans des milieux industriels se font (robotique autonome et mobile) et se feront à l'avenir sur Ubuntu et de plus en plus Debian. L'industrie des serveurs a déjà largement adopté Debian pour sa stabilité et sa modularité. C'est pourquoi plutôt que d'apprendre la ligne de commande Windows, nous recommandons d'apprendre la ligne de commande Bash, utilisée dans Ubuntu/Debian. Pour cela, il faut installer un système (noyau) Linux, plusieurs options s'offrent à nous:

- Machine virtuelle
  - Windows subsystem for Linux (WSL2)
  - Machine virtuelle Linux, par exemple via VirtualBox
- Machine physique
  - dual-boot Windows-Ubuntu -> Installation en quelques clics via une clé USB Live
  - PC sous Ubuntu 22.04
    - Pour une tour : Branchement d'un SSD SATA dédié au lieu du SSD Windows
    - Branchement d'un SSD USB3 type Transcend ESD310C

Notes importantes pour les installations virtuelles (deux premières options d'installation) :

- Ces installations sont suffisantes pour effectuer des simulations et du développement tant qu'il n'y a pas de Hardware à tester. VirtualBox fonctionne à peu près pour des TPs avec une VM URSim mais c'est loin d'être optimal (plantages,...)
- L'accélération graphique n'est pas supportée par la carte graphique (GPU) mais par le processeur (CPU) (voir ce bug)
- un PC avec 32Go de RAM est recommandé si des composants imposants de ROS doivent être compilés, par exemple pour utiliser la version de développement MoveIt 2 Rolling. En effet Windows consomme à lui seul près de 4-8Go, Ubuntu >2Go et la compilation >4Go,

on peut vite atteindre la saturation. 16Go peuvent suffire mais il faudra compiler sans parallélisation, et fermer des applications lourdes dans Windows comme Firefox.

Ubuntu via Windows SubSystem for Linux (WSL2)

WSL2 installe une machine virtuelle avec le noyau Linux complet, supporté et managé par Microsoft Windows. **Il n'y a pas besoin de droits administrateur car le logiciel est disponible dans le store Windows**.

Prérequis :

- Depuis le menu démarrer Windows, rechercher "A propos de", "Spécifications de Windows"
  - Version >22H2
  - Build >19041 (testé avec 19045.2486)
  - Si votre version est inférieure, demandez à votre administrateur de màj vers 22H2 et Build 19045.2486
  - Si vous ne pouvez màj, optez pour l'option d'installation d'Ubuntu via VirtualBox
- Exécuter Windows PowerShell en mode administrateur (connectez-vous avec un compte administrateur si vous n'avez pas les droits)
- Lancer `wsl --install` (si ça ne fonctionne pas, votre Windows n'est probablement pas à la bonne version)
- `wsl --update`
- Redémarrer l'ordinateur

Installation de Ubuntu 22 :

- Ouvrir Windows Store
- Rechercher et installer `Ubuntu` (c'est la version LTS actuelle qui sera installée, en ce moment 22.04.X)
- Depuis le menu démarrer Windows, Lancer l'application `Ubuntu`. Une Terminal s'ouvre (ligne de commande Linux Bash)
- Définir l'utilisateur principal, par exemple `ros2` et un mot de passe (8 caractères mini, majuscule, minuscule, chiffre, caractère spécial).
- Mettre à jour Ubuntu

```
sudo apt update
sudo apt upgrade
```

Depuis Windows, pour éteindre les Machines Virtuelles Ubuntu et ainsi libérer la mémoire RAM affectée :

- Lancer l'application `Windows PowerShell`
- `wsl --shutdown` Autres commandes WSL depuis `Windows PowerShell` :
- `wsl --status` : devrait retourner `Distribution par défaut : Ubuntu`, `Version par défaut : 2` (WSL2)
- `wsl --list` (ou `wsl -l -v`) : liste les Machines Virtuelles Linux installées via WSL (et la version WSL utilisée)

Docker dans une VM WSL2

Pour utiliser docker dans une VM WSL2, par exemple Ubuntu :

- Désinstaller toute version précédente de docker installée sur votre VM Ubuntu. Dans Terminal(Ubuntu) :
  - `sudo apt remove docker*`
- Ajouter votre utilisateur au groupe docker
  - `sudo groupadd docker`
  - ```
    sudo usermod -aG docker $USER
    ```
- Passer sur une session administrateur_windows (.\install)
- Installer Docker Desktop for Windows https://docs.docker.com/desktop/windows/wsl/#turn-on-docker-desktop-wsl-2
  - Cocher WSL2 (devrait être coché par défaut si votre config WSL2 est OK)
- Ajouter votre utilisateur_windows (nom_de_famille) Windows au groupe docker
  - Dans CMD/Powershell :
    ```
    net localgroup docker-users "utilisateur_windows" /ADD
    ```
- Repasser sur votre session utilisateur_windows
- L'intégration Docker-WSL est activée sur la distribution WSL par défaut, normalement Ubuntu (22)
  - pour s'en assurer, `wsl --set-default ubuntu`
  - Au besoin il est possible de l'activer sur une distro spécifique dans **Settings** > **Resources** > **WSL Integration**
- Démarrer Terminal(Ubuntu)

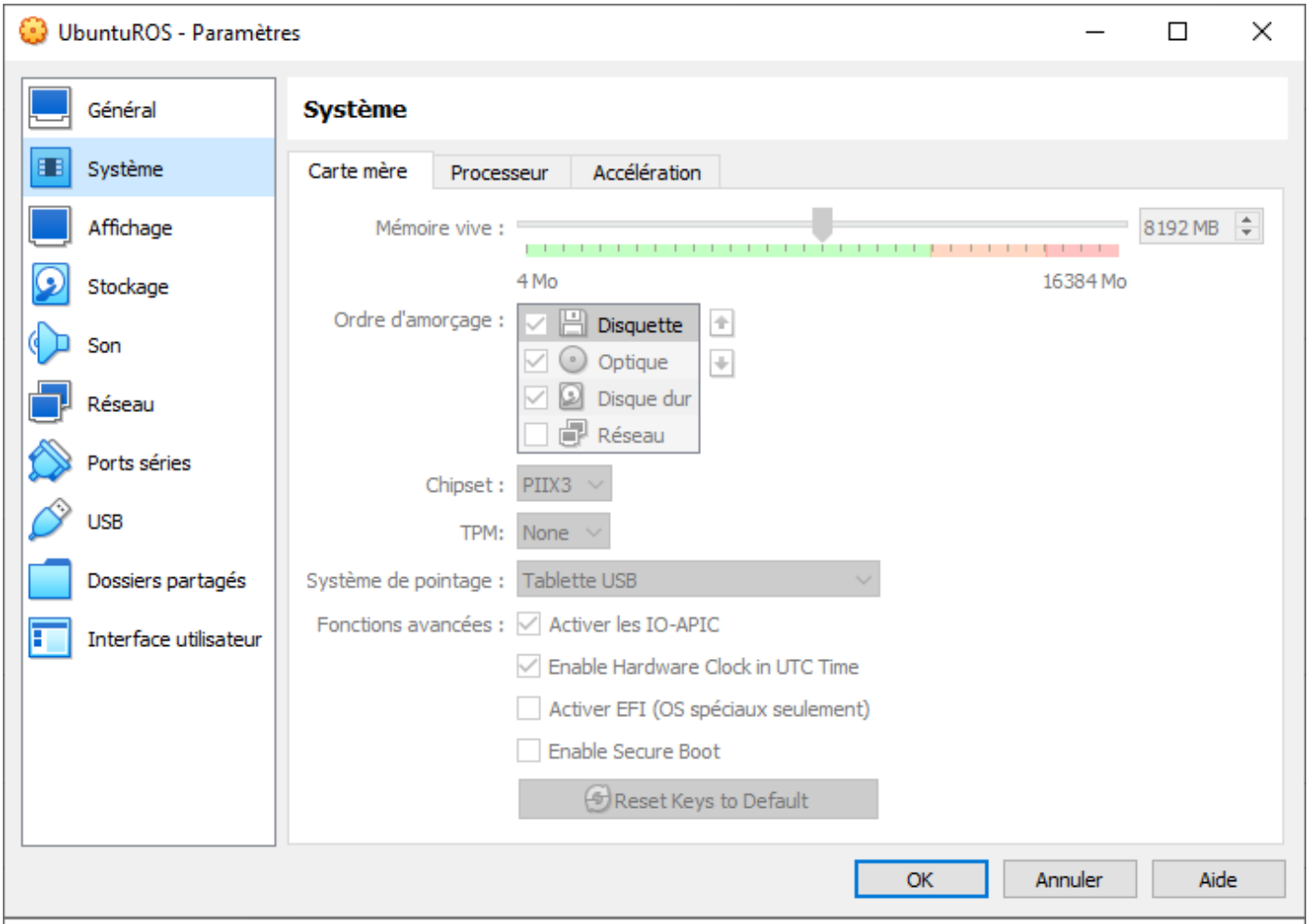
Ubuntu via VirtualBox

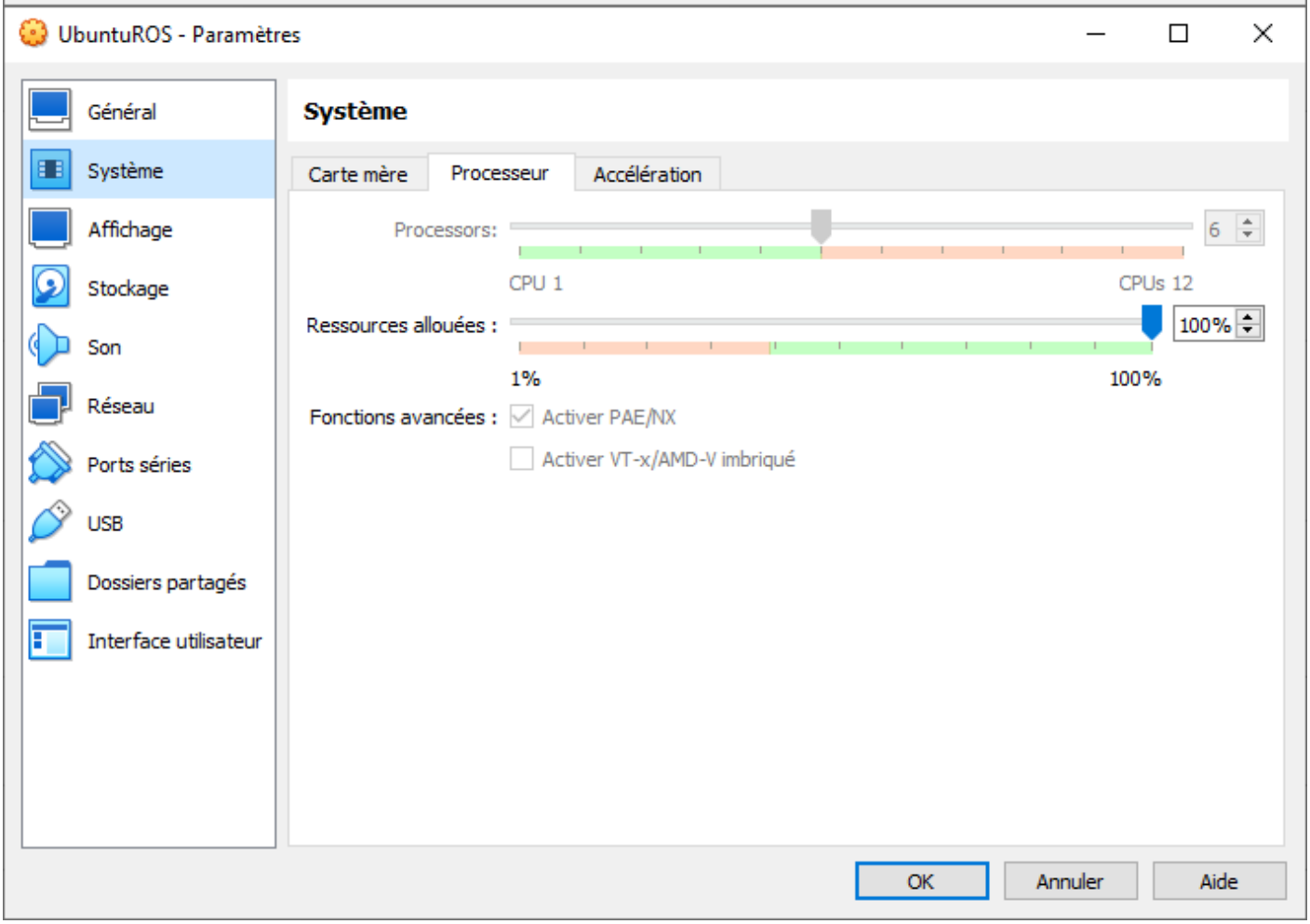
Télécharger et installer VirtualBox pour Windows :

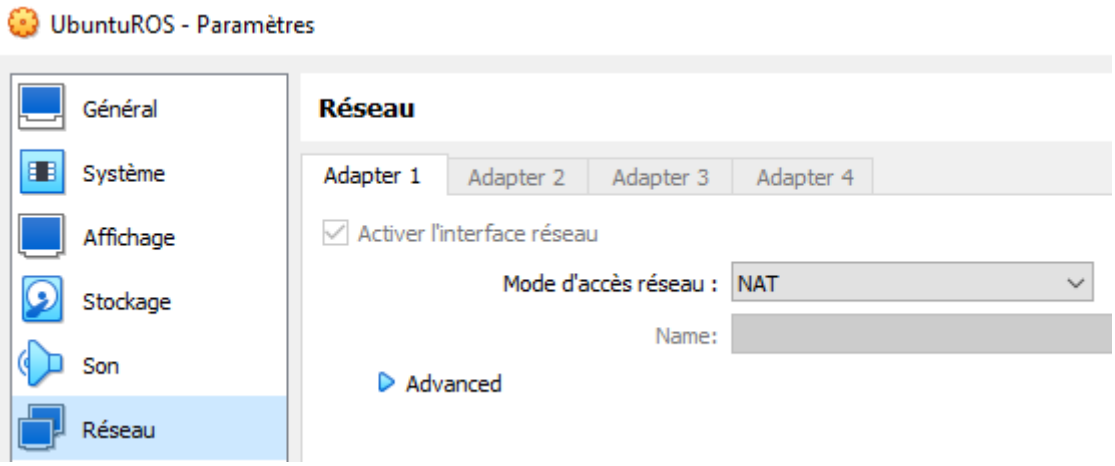
https://www.oracle.com/virtualization/technologies/vm/downloads/virtualbox-downloads.html

- Télécharger la VM depuis seafile (\Seafile\IHA-IDF\Smart_Prod\Formation_ROS2\UbuntuROS.ova)
  - Lien public de téléchargement
    ```

- Lancer VirtualBox
- Importer la VM : Outils -> Importer -> Rechercher le fichier UbuntuROS.ova
- Vérifier et adapter la configuration de la VM en ressources RAM, CPU, GPU et Réseau selon la configuration de votre PC


UbuntuROS - Paramètres
Général
Système
Affichage
Stockage
Son
Réseau
Ports séries
USB
Dossiers partagés
Interface utilisateur
Système
Carte mère
Processeur
Accélération
Mémoire vive :
8192 MB
4 Mo
16384 Mo
Ordre d'amorçage :
Disquette
Optique
Disque dur
Réseau
Chipset :
PIIX3
TPM:
None
Système de pointage :
Tablette USB
Fonctions avancées :
Activer les IO-APIC
Enable Hardware Clock in UTC Time
Activer EFI (OS spéciaux seulement)
Enable Secure Boot
Reset Keys to Default
OK
Annuler
Aide


UbuntuROS - Paramètres
Général
Système
Affichage
Stockage
Son
Réseau
Ports séries
USB
Dossiers partagés
Interface utilisateur
Système
Carte mère
Processeur
Accélération
Processors:
6
CPU 1
CPUs 12
Ressources allouées :
100%
1%
100%
Fonctions avancées :
Activer PAE/NX
Activer VT-x/AMD-V imbriqué
OK
Annuler
Aide

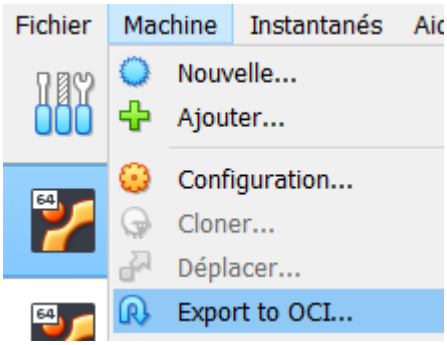


- Démarrer la VM
- Ignorer l'erreur sur le dossier partagé Linux-Windows

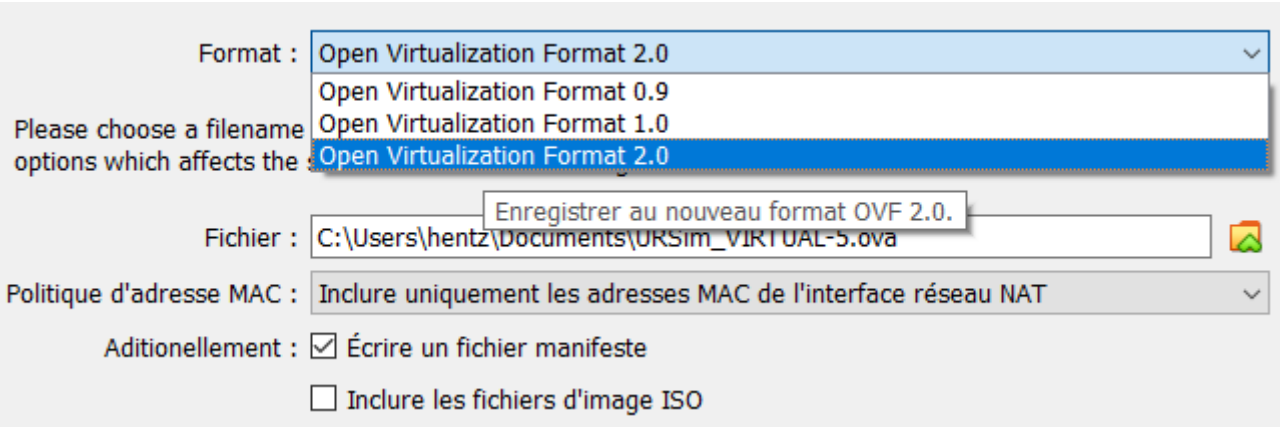
Exportation de VM au format OVF

Le système du TP est maintenu à jour et testé sur un PC Windows. Pour l'exporter sur les PC de salle TP, on veut avoir une image la plus petite possible.

- On commence par nettoyer Ubuntu puis on exporte un fichier `.ova`



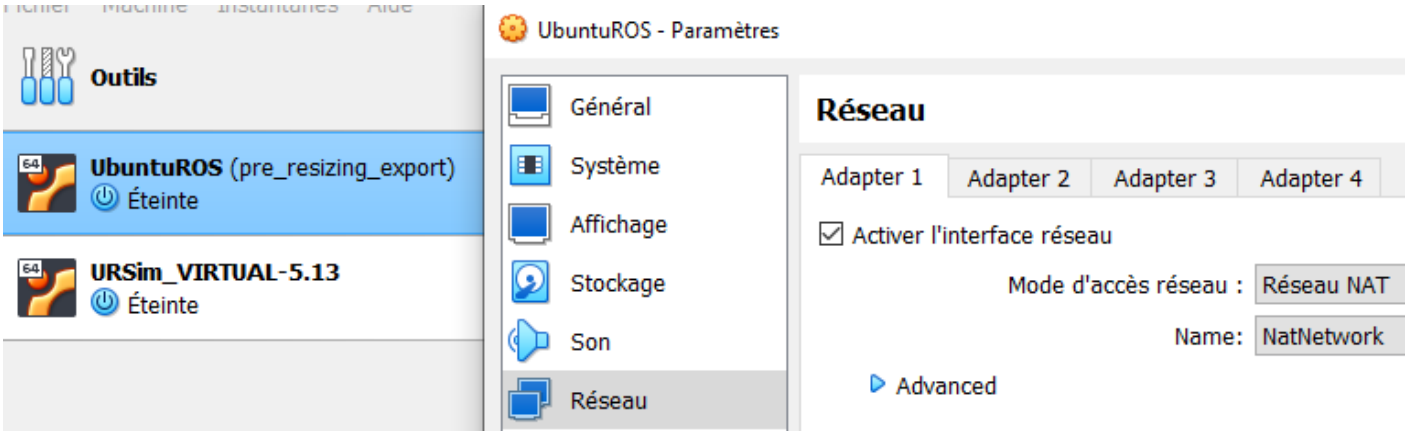
- Sélectionner le format `OVF 2.0` pour une meilleure compressions



Setup pour TP MoveIt2+URSim à l'IUT de Haguenau

La première année j'ai expérimenté avec des PC Windows et VirtualBox :

- Une VM contient Ubuntu 22, ROS et MoveIt
- Une seconde contient Xubuntu 14/16 avec URSim
- Les deux VMs en Réseau NAT



- Voir : https://innovation.iha.unistra.fr/books/robotique-open-source/page/programmer-un-robot-avec-moveit2-jumeau-numerique#bkmrk-sous-windows---virtu
- Il faut des machines de guerre, régler finement la quantité de RAM et de coeurs alloués aux VM et à Windows, et malgré cela les VM plantent.

En 2025 je change donc de fusil d'épaule et utilise la salle réseau de l'IUT :

- Avec des vieilles tours de 2013 : i5, 8G de RAM, petite carte graphique double écran, 60G de SSD
- Réseau isolé donc possibilité de mettre OS au choix sur les PC, d'isoler ou non les PC et d'éventuels robots à piloter

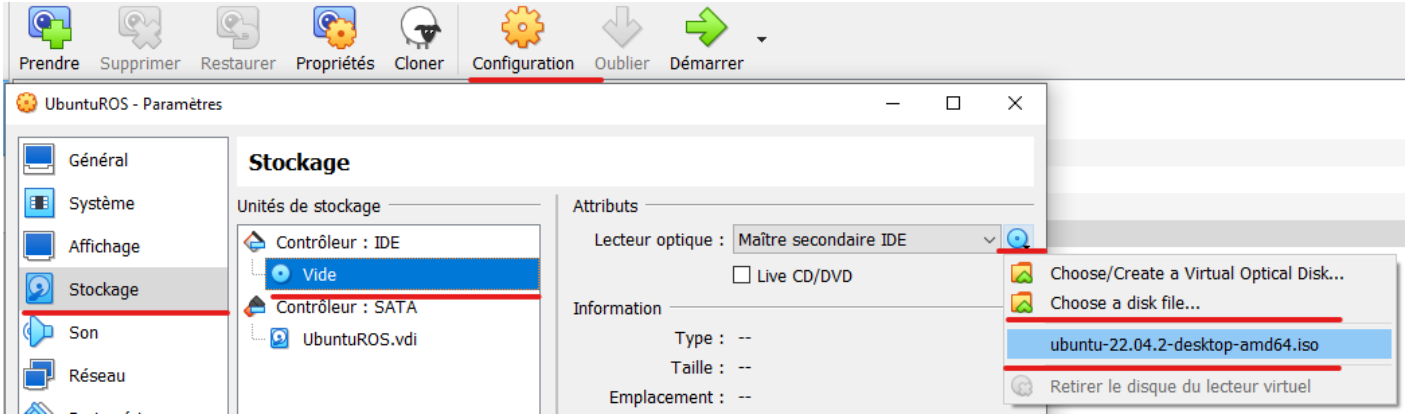
Migration VM vers disque physique

Entre 2024 et 2025 je suis passé de TPs en VM VirtualBox vers des PC physiques. Dans les deux cas, je maintiens l'environnement de TP sur VirtualBox de mon PC Windows. Ceci présente l'avantage de pouvoir maintenir des états de machines en fonction du type de TP.

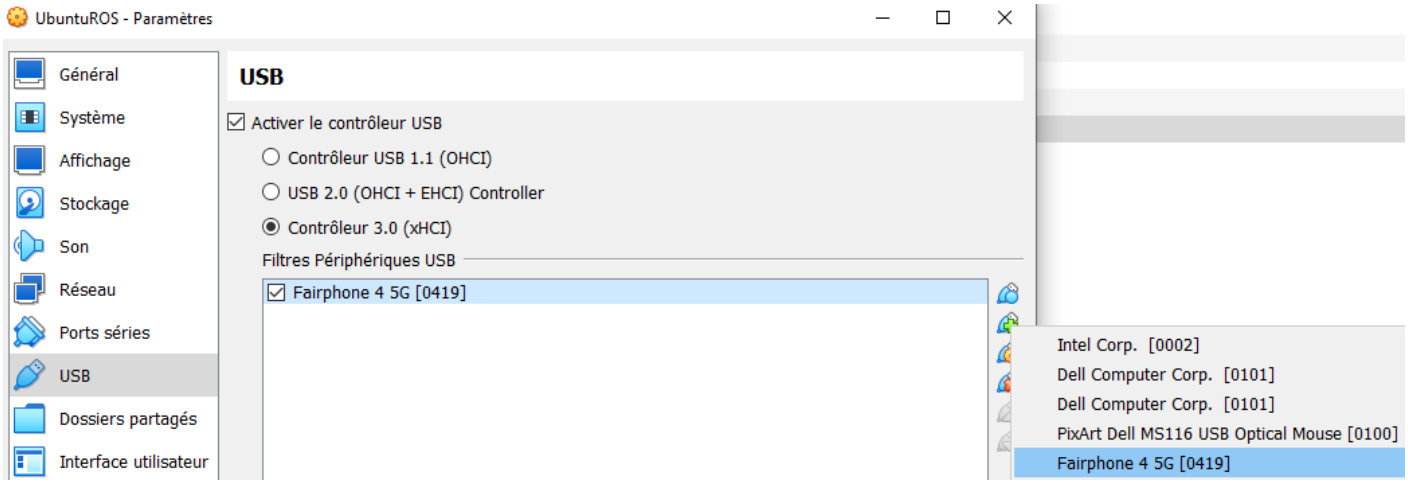
On peut déployer un disque virtuel de VM vers un disque physique :

- Nettoyer l'OS et éventuellement désactiver le SWAP pour encore gagner de l'espace

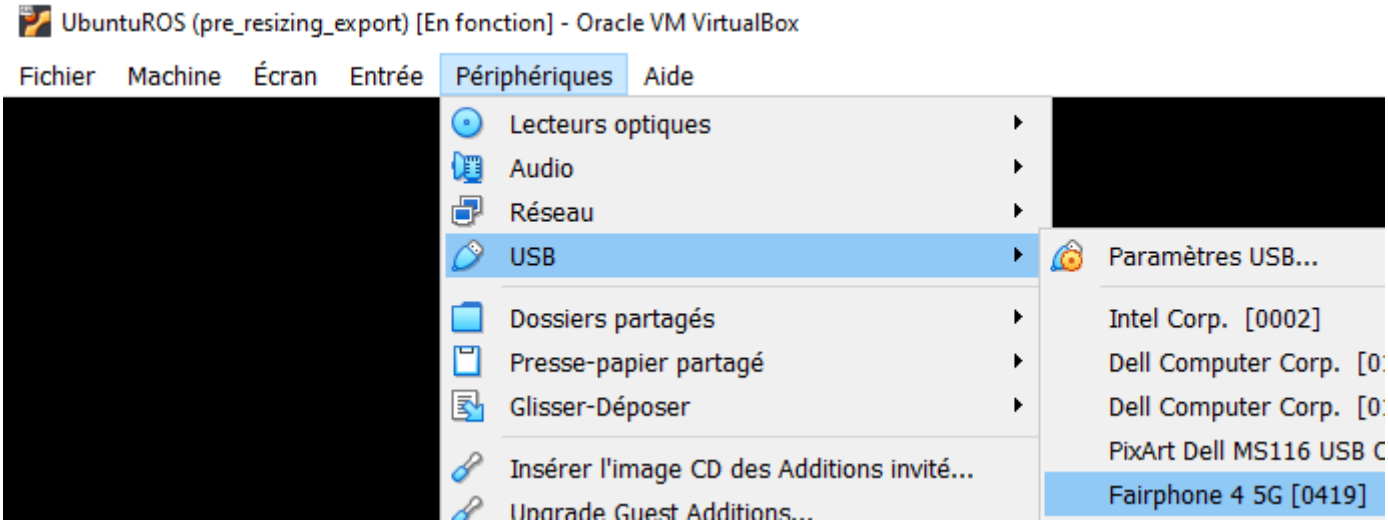
- Réduire la taille de la partition via l'application Disks. Celle-ci pourra être agrandie après copie de la VM. Garder quelques Go de marge.
- Démarrer la VM sur une `.iso` Live Ubuntu



- Brancher un SSD en USB3 au PC (utiliser un adaptateur SATA-USB3 si nécessaire)
- Passer le périphérique USB à la VM
  - Avant le démarrage



- ◦ Pendant que la VM tourne



- Ouvrir l'application Disks pour identifier les disques, en général :
  - disque virtuel de la VM : `/dev/sda`
  - SSD branché en USB : `/dev/sdb`
- Ouvrir un Terminal et lancer la commande de copie du disque virtuel vers le SSD physique :
- `sudo dd if=/dev/sda of=/dev/sdb bs=4096 status=progress && sync`
- Ouvrir Gparted (depuis une Live USB avec le SSD branché) pour vérifier que la partition principale, généralement `sdb3` est bien identifiée comme formatée en `ext4`. Agrandir la partition à la taille désirée.
- Si le SSD ne boot pas sur un PC, essayer de réparer le grub avec `boot-repair` depuis une Live USB

Windows 10/11

Une installation native sous Windows 10 avec Visual Studio 2019 (Version Community gratuite) est possible :

- ROS 1
- ROS 2

Installation de ROS2 Humble

Les distributions stables publiées (pré-compilées) de ROS2 sont nommées par ordre alphabétique.

Début 2023, on va installer **ROS 2 Humble** :

```
sudo apt update && sudo apt install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8
sudo apt install software-properties-common
sudo add-apt-repository universe
sudo apt update && sudo apt install curl
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o
/usr/share/keyrings/ros-archive-keyring.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-
keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME)
main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
sudo apt update && sudo apt upgrade
```

```
sudo apt install ros-humble-desktop-full
source /opt/ros/humble/setup.bash
echo 'source /opt/ros/humble/setup.bash' >> ~/.bashrc
```

Tester l'installation

https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html#try-some-examples

- Ouvrir un premier Terminal : `ros2 run demo_nodes_cpp talker`
- Ouvrir un second Terminal : `ros2 run demo_nodes_cpp listener`

Installation de Jazzy pour la Navigation et Manipulation

avec UR, Turtlebot3, Nav2, MoveIt2, etc.

```
  1  sudo apt update && sudo apt install locales
    2  sudo locale-gen en_US en_US.UTF-8
    3  sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
    4  export LANG=en_US.UTF-8
    5  sudo apt install software-properties-common
    6  sudo add-apt-repository universe
    7  sudo apt update && sudo apt install curl
    8  sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o
/usr/share/keyrings/ros-archive-keyring.gpg
    9  echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-
keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME)
main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
   10  sudo apt update && sudo apt upgrade
   11  sudo apt install ros-dev-tools
   12  exit
   13  locale
   14  sudo apt install ros-jazzy-desktop-full
   15  echo 'source /opt/ros/jazzy/setup.bash' >> ~/.bashrc
   16  exit
   17  ros2 run demo_nodes_cpp talker~
   18  ros2 run demo_nodes_cpp talker
```

```
   19  exit
   20  ros2 run demo_nodes_cpp listener
   21  exit
   22  sudo apt install docker-compose
   23  sudo usermod -aG docker $USER
   24  sudo service docker start
   25  docker run hello-world
   26  sudo service docker status
   27  docker run hello-world
   28  exit
   29  docker run hello-world
   30  sudo service docker restart
   31  docker run hello-world
   32  sudo usermod -aG docker robot
   33  docker run hello-world
   34  docker pull universalrobots/ursim_e-series
   35  exit
   36  docker run hello-world
   37  sudo service docker restart
   38  docker run hello-world
   39  sudo service docker start
   40  sudo service docker status
   41  docker run hello-world
   42  docker pull universalrobots/ursim_e-series
   43  sudo usermod -aG docker robot
   44  exit
   45  sudo apt install python3-argcomplete python3-colcon-common-extensions libboost-system-
dev build-essential
   46  sudo apt install ros-jazzy-hls-lfcd-lds-driver ros-jazzy-turtlebot3-msgs ros-jazzy-
dynamixel-sdk libudev-dev
   47  mkdir -p ~/turtlebot3_ws/src && cd ~/turtlebot3_ws/src
   48  git clone -b humble-devel https://github.com/ROBOTIS-GIT/turtlebot3.git
   49  git clone -b ros2-devel https://github.com/ROBOTIS-GIT/ld08_driver.git
   50  cd ~/turtlebot3_ws/src/turtlebot3
   51  rm -r turtlebot3_cartographer turtlebot3_navigation2
   52  cd ~/turtlebot3_ws/
   53  echo 'source /opt/ros/humble/setup.bash' >> ~/.bashrc
   54  source ~/.bashrc
   55  cd ..
```

```
   56  sudo nano .bashrc
   57  exit
   58  cd turtlebot3_ws/
   59  source install/setup.bash
   60  ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
   61  ls src/
   62  colcon build --symlink-install
   63  rosdep update && rosdep install --ignore-src --from-paths src -y
   64  vcs --help
   65  vcs status
   66  sudo apt list ros-jazzy-gazebo-ros-pkgs
   67  sudo apt list ros-jazzy-ros-gz
   68  sudo apt install ros-jazzy-ros-gz
   69  colcon build --symlink-install
   70  ros2 launch nav2_bringup tb3_simulation_launch.py slam:=True nav:=True headless:=False
use_sim_time:=True
   71  sudo apt install ros-jazzy-navigation2 ros-jazzy-nav2-bringup
   72  ros2 launch nav2_bringup tb3_simulation_launch.py headless:=False
   73  exit
   74  cd ..
   75  cd turtlebot3_ws/
   76  colcon build --symlink-install --parallel-workers 1
   77  cd src/
   78  git clone -b humble https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
   79  cd ..
   80  colcon build --symlink-install --parallel-workers 1
   81  sudo nano .bashrc
   82  sudo nano ~/.bashrc
   83  source install/setup.bash
   84  ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
   85  sudo apt list ros-jazzy-turtlebot3-*
   86  sudo apt install ros-jazzy-turtlebot3-fake-node
   87  sudo apt install ros-jazzy-gazebo-msgs
   88  cd src/
   89  sudo apt update && rosdep install -r --from-paths . --ignore-src --rosdistro
$ROS_DISTRO -y
   90  rosdep update
   91  sudo apt update && rosdep install -r --from-paths . --ignore-src --rosdistro
$ROS_DISTRO -y
```

```
   92  sudo curl https://packages.osrfoundation.org/gazebo.gpg --output
/usr/share/keyrings/pkgs-osrf-archive-keyring.gpg
   93  echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/pkgs-osrf-
archive-keyring.gpg] http://packages.osrfoundation.org/gazebo/ubuntu-stable $(lsb_release -cs)
main" | sudo tee /etc/apt/sources.list.d/gazebo-stable.list > /dev/null
   94  sudo apt-get update
   95  sudo apt-get install gz-harmonic
   96  cd ..
   97  colcon build --symlink-install --parallel-workers 1
   98  cd ..
   99  ls
  100  mkdir -p ur_ws/src
  101  cd ur_ws/src
  102  cd ..
  103  git clone https://github.com/UniversalRobots/Universal_Robots_ROS2_Tutorials.git
src/ur_tutorials
  104  rosdep update && rosdep install --ignore-src --from-paths src -y
  105  git clone -b ros2
https://github.com/UniversalRobots/Universal_Robots_ROS2_GZ_Simulation.git
src/ur_simulation_gz
  106  rosdep update && rosdep install --ignore-src --from-paths src -y
  107  colcon build --symlink-install
  108  exit
  109  docker run hello-world
  110  docker pull universalrobots/ursim_e-series
  111  ros2 run ur_robot_driver start_ursim.sh -m ur5e
  112  sudo apt install ros-jazzy-ur
  113  sudo apt list python3-rosdep
  114  sudo rosdep init
  115  rosdep update
  116  sudo apt update
  117  sudo apt dist-upgrade
  118  ros2 run ur_robot_driver start_ursim.sh -m ur5e
  119  sudo apt list python3-colcon*
  120  colcon mixin add default https://raw.githubusercontent.com/colcon/colcon-mixin-
repository/master/index.yaml
  121  colcon mixin update default
  122  sudo apt list python3-vcstool
  123  mkdir -p ~/ws_moveit/src
```

```
  124  cd ~/ws_moveit/src
  125  git clone -b main https://github.com/moveit/moveit2_tutorials
  126  vcs import --recursive < moveit2_tutorials/moveit2_tutorials.repos
  127  cd moveit2
  128  git checkout jazzy
  129  ls
  130  ls moveit2.repos
  131  cat moveit2.repos
  132  cd ..
  133  cd src/
  134  sudo apt update && rosdep install -r --from-paths . --ignore-src --rosdistro
$ROS_DISTRO -y
  135  cd ..
  136  colcon build --mixin releasee
```

Installation d'autres versions de ROS2

Pour avoir accès à toutes les dernières fonctionnalités en cours de développement (partiellement publiées), il faut installer **ROS2 Rolling**, qui est une distribution en développement continu "rolling release". Par exemple en Avril 2023, l'API Python de MoveIt2 et son tutoriel ne sont disponibles que sous rolling.

On peut installer plusieurs versions de ros en parallèle. Chaque version sera installée dans `/opt/ros/version`. Pour faire cohabiter les deux versions, il faut "sourcer" le bon répertoire avant de lancer un programme `ros2 launch ...` ou de compiler un workspace `colcon build ...`. Deux options s'offrent à nous :

- Si on bascule souvent de version : commenter les lignes `source /opt/ros/humble/setup.bash` en bas du fichier `~/.bashrc`
  - Il faudra alors lancer la commande `source /opt/ros/humble/setup.bash` **à chaque nouvelle ouverture de Terminal Bash**.
- Si on travaille principalement avec une version : commenter la ligne correspondant à la version principale `source /opt/ros/humble/setup.bash` en bas du fichier `~/.bashrc` lorsqu'on veut utiliser la version secondaire.

Gestion de version avec Ansible

L'idéal serait de gérer l'état des VM/PC de TP avec ansible plutôt que des snapshot VirtualBox

https://github.com/richlamdev/ansible-desktop-ubuntu

Outils utiles

Terminal multi-fenêtres Terminator

- Installer Terminator : c'est un logiciel de Ligne de commande pratique pour programmer avec ROS
  - Depuis Windows Store : Rechercher et installer `Terminator (Ubuntu)`
  - Depuis la ligne de commande Linux : `sudo apt install terminator`
- Depuis le menu démarrer Windows, Lancer `Terminator (Ubuntu)`

Visual Studio Codium

Pour éviter d'alourdir la VM avec de la télémétrie Microsoft, on installe la version sans tracker de Visual Studio Code depuis un dépôt debian :

- Lancer la VM VirtualBox ou WSL (`Terminator (Ubuntu)`)
- Dans Terminator, lancer les commandes suivantes :

```
wget https://gitlab.com/paulcarroty/vscodium-deb-rpm-repo/raw/master/pub.gpg
sudo mv pub.gpg /usr/share/keyrings/vscodium-archive-keyring.asc
echo 'deb [ signed-by=/usr/share/keyrings/vscodium-archive-keyring.asc ]
https://paulcarroty.gitlab.io/vscodium-deb-rpm-repo/debs vscodium main' \
    | sudo tee /etc/apt/sources.list.d/vscodium.list
sudo apt update
sudo apt install codium
```

- Lancer VSCodium dans la VM VirtualBox ou directement depuis Windows, lancer `VSCodium (Ubuntu)`
- Ouvrir le **dossier contenant le code source** `/src` du projet dont vous voulez étudier/modifier le code : `File --> Open Folder --> ~/ws_moveit/src`

Installer Firefox dans WSL

https://askubuntu.com/questions/1444962/how-do-i-install-firefox-in-wsl-when-it-requires-snap-but-snap-doesnt-work

```
sudo snap remove firefox
sudo apt remove firefox
sudo add-apt-repository ppa:mozillateam/ppa

# Create a new file, it should be empty as it opens:
sudo gedit /etc/apt/preferences.d/mozillateamppa

# Insert these lines, then save and exit
Package: firefox*
Pin: release o=LP-PPA-mozillateam
Pin-Priority: 501

# after saving, do
sudo apt update
sudo apt install firefox-esr
```

Alléger Ubuntu (pour VM ou clonage)

Désinstaller snap :

- Vérifier qu'on n'a pas de paquet snap important avec `snap list`
- Purger snap et tous ses paquets

```
sudo rm -rf /var/cache/snapd/

sudo apt autoremove --purge snapd gnome-software-plugin-snap

rm -fr ~/snap

# sudo apt-mark hold snapd
```

- Empêcher snap d'être réinstallé par Ubuntu

```
cat <<EOF | sudo tee /etc/apt/preferences.d/nosnap.pref
# To prevent repository packages from triggering the installation of Snap,
# this file forbids snapd from being installed by APT.
# For more information: https://linuxmint-user-guide.readthedocs.io/en/latest/snap.html
```

```
Package: snapd
Pin: release a=*
Pin-Priority: -10
EOF
```

Installer le magasin d'applications de gnome sans snap/flatpak `sudo apt install gnome-software --no-install-recommends`

Supprimer les paquets apt plus nécessaires `sudo apt autoremove --purge`

Supprimer le cache de compilation de VSCode `~/.cache/vscode-cpptools`

Supprimer les fichiers de compilation des workspaces qui ne seront pas utilisés en TP. Attention à conserver les paquets qui devront être compilés en TP (en utilisant `colcon build --package-select`).

Réduire la taille de la partition via l'application Disks. Celle-ci pourra être agrandie après copie de la VM.

Sources

- Installation de MoveIt2 Humble sur Ubuntu 22.04
- Tutoriels débutant

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

Tutoriels de base

Pour comprendre les concepts de ROS2 par la pratique, il existe des tutoriels pour débutant. Ils reposent sur la simulation d'un robot mobile à deux roues principales développé par les développeurs de ROS en 2010 : Turtlebot. Le TurtleBot 3 est vendue par Robotis et peut être couplé à un bras manipulateur 5 axes OpenMANIPULATOR-X. Il est possible de simuler des applications de manipulation mobile avec Gazebo.

Pour réaliser les tutoriels de base, il nous faut un environnement de développement, deux options :

- Une machine virtuelle (VM) VirtualBox, ou disponible au téléchargement ici
  - Le plus simple et rapide pour démarrer
  - Si on n'a pas de Hardware ou qu'on ne souhaite travailler qu'en simulation
- Une installation simple ou dual-boot sur un PC
  - Il faut alors installer de zéro
  - Indispensable si on veut travailler avec du Hardware

Supposons que vous avez installé et testé votre environnement comme celui de la VM.

Connexion à la VM

- nom : ubuntu22ros2
- utilisateur : `etudiant`
- mdp : département de l'IUT

Nous pouvons directement passer aux tutoriels sur les outils ROS 2 en ligne de commande :

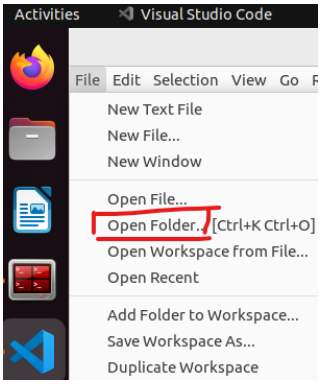
- Configuring environment
- Using `turtlesim`, `ros2`, and `rqt`
- Understanding nodes
- Understanding topics
- Understanding services
- Understanding parameters

Ces premiers tutoriels ne nécessitent qu'une installation basique de ROS 2, on n'y regarde pas le code source.

Ensuite on passe aux tutoriels sur les bibliothèques clientes de ROS 2 en C++ et Python :

- Using `colcon` to build packages
- Creating a workspace
- Creating a package
- Writing a simple publisher and subscriber (C++) --> Correction
- Writing a simple publisher and subscriber (Python)
- Writing a simple service and client (C++)
- Writing a simple service and client (Python)
- Creating custom msg and srv files

Ces tutoriels vous engagent à copier et analyser du code source en C++ et Python. Les fichiers créés sont placés dans le dossier de travail (workspace) `/home/etudiant/ros2_ws/src`, à ouvrir avec Visual Studio Code :



Vous trouverez des fichiers de correction commentés dans `ros2_ws/src/cpp_pubsub/src/`, en particulier :

- publisher_member_function.cpp
- subscriber_member_function.cpp

Pour les tester il faut lancer :

```
cd ~/ros2_ws
colcon build --packages-select cpp_pubsub
source install/setup.bash
ros2 run cpp_pubsub talker
```

Le noeud se met à publier/parler :

```
[INFO] [minimal_publisher]: Publishing: "Hello World: 0"
[INFO] [minimal_publisher]: Publishing: "Hello World: 1"
[INFO] [minimal_publisher]: Publishing: "Hello World: 2"
[INFO] [minimal_publisher]: Publishing: "Hello World: 3"
[INFO] [minimal_publisher]: Publishing: "Hello World: 4"
```

Puis dans un second Terminal :

```
ros2 run cpp_pubsub listener
```

Le noeud se met à écouter :

```
[INFO] [minimal_subscriber]: I heard: "Hello World: 10"
[INFO] [minimal_subscriber]: I heard: "Hello World: 11"
[INFO] [minimal_subscriber]: I heard: "Hello World: 12"
[INFO] [minimal_subscriber]: I heard: "Hello World: 13"
[INFO] [minimal_subscriber]: I heard: "Hello World: 14"
```

Tapez `Ctrl+C` dans chaque Terminal pour arrêter les noeuds ("stop spinning").

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

Découverte d'Ubuntu Linux et son Terminal Bash

Navigating the Ubuntu GUI

> In this exercise, we will familiarize ourselves with the graphical user interface (GUI) of the Ubuntu operating system.

Task 1: Familiarize Yourself with the Ubuntu Desktop

At the log-in screen, click in the password input box, enter `rosindustrial` for the password, and hit enter. The screen should look like the image below when you log in:

There are several things you will notice on the desktop:

1. The gear icon on the top right of the screen brings up a menu which allows the user to log out, shut down the computer, access system settings, etc…
2. The bar on the left side shows running and “favorite” applications, connected thumb drives, etc.
3. The top icon is used to access all applications and files. We will look at this in more detail later.
4. The next icons are either applications which are currently running or have been “pinned” (again, more on pinning later)

5. Any removable drives, like thumb drives, are found after the application icons.
6. If the launcher bar gets “too full”, clicking and dragging up/down allows you to see the applications that are hidden.
7. To reorganize the icons on the launcher, click and hold the icon until it “pops out”, then move it to the desired location.

Task 2: Open and Inspect an Application

Click on the filing-cabinet icon in the launcher. A window should show up, and your desktop should look like something below:

./_images/ubuntu_folder_browser.png

Things to notice:

1. The close, minimize, and maximize buttons typically found on the right-hand side of the window title bar are found on the left-hand side.
2. The menu for windows are found on the menu bar at the top of the screen, much in the same way Macs do. The menus, however, only show up when you hover the mouse over the menu bar.
3. Notice that there are menu highlights of the folder icon. The dots on the left show how many windows of this application are open. Clicking on these icons when the applications are open does one of two things:

- If there is only one window open, this window gets focus.
- If more than one are open, clicking a second time causes all of the windows to show up in the foreground, so that you can choose which window to go to (see below):

./_images/ubuntu_inspect.png

Task 3: Start an Application & Pin it to the Launcher Bar

Click on the launcher button (top left) and type gedit in the search box. The “Text Editor” application (this is actually gedit) should show up (see below):

./_images/ubuntu_start_application.png

Click on the application. The text editor window should show up on the screen, and the text editor icon should show up on the launcher bar on the left-hand side (see below):

1. Right-click on the text editor launch icon, and select “Lock to Launcher”.
2. Close the gedit window. The launcher icon should remain after the window closes.
3. Click on the gedit launcher icon. You should see a new gedit window appear.

Le Terminal Linux

> In this exercise, we will familiarize ourselves with the Linux terminal.

Starting the Terminal

1. Pour ouvrir le Terminal, recherchez le programme "terminator" ou cliquez sur l'icône:
   
2. Create a second terminal window, either by:
   - Right-clicking on the terminal and selecting the “Open Terminal” or
   - Selecting “Open Terminal” from the “File” menu
3. Create a second terminal within the same window by pressing “Ctrl+Shift+T” while the terminal window is selected.
4. Close the 2nd terminal tab, either by:
   - clicking the small ‘x’ in the terminal tab (not the main terminal window)
   - typing `exit` and hitting enter.
5. The window will have a single line, which looks like this:
   `ros-industrial@ros-i-humble-vm:~$`
6. This is called the prompt, where you enter commands. The prompt, by default, provides three pieces of information:
   1. *ros-industrial* is the login name of the user you are running as.
   2. *ros-i-humble-vm* is the host name of the computer.
   3. ~ is the directory in which the terminal is currently in. (More on this later).
7. Close the terminal window by typing `exit` or clicking on the red ‘x’ in the window’s titlebar.

Navigating Directories and Listing Files

Prepare your environment

1. Open your home folder in the file browser.
2. Double-click on the `ex0.3` folder we created in the previous step.
   - *We'll use this to illustrate various file operations in the terminal.*
3. Right click in the main file-browser window and select "Open in Terminal" to create a terminal window at that location.
4. In the terminal window, type the following command to create some sample files that we can study later:
   - `cp -a ~/industrial_training/exercises/0.3/. .`

ls Command

1. Enter `ls` into the terminal.
   - You should see `test.txt`, and `new` listed. (If you don't see 'new', go back and complete the previous exercise).
   - Directories, like `new`, are colored in blue.
   - The file `sample_job` is in green; this indicates it has its "execute" bit set, which means it can be executed as a command.
2. Type `ls *.txt`. Only the file `test.txt` will be displayed.
3. Enter `ls -l` into the terminal.
   - Adding the `-l` option shows one entry per line, with additional information about each entry in the directory.
   - The first 10 characters indicate the file type and permissions
   - The first character is `d` if the entry is a directory.
   - The next 9 characters are the permissions bits for the file
   - The third and fourth fields are the owning user and group, respectively.
   - The second-to-last field is the time the file was last modified.
   - If the file is a symbolic link, the link's target file is listed after the link's file name.
4. Enter `ls -a` in the terminal.
   - You will now see one additional file, which is hidden.
5. Enter `ls -a -l` (or `ls -al`) in the command.
   - You'll now see that the file `hidden_link.txt` points to `.hidden_text_file.txt`.

`pwd` and `cd` Commands

1. Enter `pwd` into the terminal.
   - This will show you the full path of the directory you are working in.
2. Enter `cd new` into the terminal.
   - The prompt should change to `ros-industrial@ros-i-humble-vm:~/ex0.3/new$`.
   - Typing `pwd` will show you now in the directory `/home/ros-industrial/ex0.3/new`.

3. Enter `cd ..` into the terminal. * In the previous exercise, we noted that `..` is the parent folder. * The prompt should therefore indicate that the current working directory is `/home/ros-industrial/ex0.3`.
4. Enter `cd /bin`, followed by `ls`.
   - This folder contains a list of the most basic Linux commands.
     Note that `pwd` and `ls` are both in this folder.
5. Enter `cd ~/ex0.3` to return to our working directory.
   - Linux uses the `~` character as a shorthand representation for your home directory.
   - It's a convenient way to reference files and paths in command-line commands.
   - You'll be typing it a lot in this class… remember it!

If you want a full list of options available for any of the commands given in this section, type `man <command>` (where `<command>` is the command you want information on) in the command line. This will provide you with built-in documentation for the command. Use the arrow and page up/down keys to scroll, and `q` to exit.

Altering Files

mv Command

1. Type `mv test.txt test2.txt`, followed by `ls`.
   - You will notice that the file has been renamed to `test2.txt`.
     This step shows how `mv` can rename files.
2. Type `mv test2.txt new`, then `ls`.
   - The file will no longer be present in the folder.
3. Type `cd new`, then `ls`.
   - You will see `test2.txt` in the folder.
     These steps show how `mv` can move files.
4. Type `mv test2.txt ../test.txt`, then `ls`.
   - `test2.txt` will no longer be there.
5. Type `cd ..`, then `ls`.
   - You will notice that `test.txt` is present again.
     This shows how `mv` can move and rename files in one step.

cp Command

1. Type `cp test.txt new/test2.txt`, then `ls new`.
   - You will see `test2.txt` is now in the `new` folder.
2. Type `cp test.txt "test copy.txt"`, then `ls -l`.
   - You will see that `test.txt` has been copied to `test copy.txt`.
     Note that the quotation marks are necessary when spaces or other special

characters are included in the file name.

rm Command

1. Type `rm "test copy.txt"`, then `ls -l`.
   - You will notice that `test copy.txt` is no longer there.

mkdir Command

1. Type `mkdir new2`, then `ls`.
   - You will see there is a new folder `new2`.

touch Command

1. Type `touch ~/Templates/"Untitled Document"`.
   - This will create a new Document named **“Untitled Document”**

You can use the `-i` flag with `cp`, `mv`, and `rm` commands to prompt you when a file will be overwritten or removed.

Editing Text (and Other GUI Commands)

1. Type `gedit test.txt`.
   - You will notice that a new text editor window will open, and `test.txt` will be loaded.
   - The terminal will not come back with a prompt until the window is closed.
2. There are two ways around this limitation. Try both…
3. **Starting the program and immediately returning a prompt:**
   1. Type `gedit test.txt &`.
      - The `&` character tells the terminal to run this command in “the background”, meaning the prompt will return immediately.
   2. Close the window, then type `ls`.
      - In addition to showing the files, the terminal will notify you that `gedit` has finished.
4. **Moving an already running program into the background:**
   1. Type `gedit test.txt`.
      - The window should open, and the terminal should not have a prompt waiting.
   2. In the terminal window, press Ctrl+Z.
      - The terminal will indicate that `gedit` has stopped, and a prompt will appear.
   3. Try to use the `gedit` window.
      - Because it is paused, the window will not run.

4. Type `bg` in the terminal.
    - The `gedit` window can now run.
5. Close the `gedit` window, and type `ls` in the terminal window.
    - As before, the terminal window will indicate that `gedit` is finished.

Running Commands as Root

1. In a terminal, type `ls -a /root`.
    - The terminal will indicate that you cannot read the folder `/root`.
    - Many times you will need to run a command that cannot be done as an ordinary user, and must be done as the “super user”
2. To run the previous command as root, add `sudo` to the beginning of the command.
    - In this instance, type `sudo ls -a /root` instead.
    - The terminal will request your password (in this case, `rosindustrial`) in order to proceed.
    - Once you enter the password, you should see the contents of the `/root` directory.

***Warning**: `sudo` is a powerful tool which doesn’t provide any sanity checks on what you ask it to do, so be **VERY** careful in using it*

https://industrial-training-master.readthedocs.io/en/humble/_source/prerequisites/Navigating-the-Ubuntu-GUI.html

Usage avancé du bash Linux

Job management

Stopping Jobs

1. Type `./sample_job`.
   - The program will start running.
2. Press Control+C.
   - The program should exit.
3. Type `./sample_job sigterm`.
   - The program will start running.
4. Press Control+C.
   - This time the program will not die.

Stopping “Out of Control” Jobs

1. Open a new terminal window.
2. Type `ps ax`.
3. Scroll up until you find `python ./sample_job sigterm`.
   - This is the job that is running in the first window.
   - The first field in the table is the ID of the process (use `man ps` to learn more about the other fields).
4. Type `ps ax | grep sample`.
   - You will notice that only a few lines are returned.
   - This is useful if you want to find a particular process
   - *Note: this is an advanced technique called “piping”, where the output of one program is passed into the input of the next. This is beyond the scope of this class, but is useful to learn if you intend to use the terminal extensively.*
5. Type `kill <id>`, where `<id>` is the job number you found with the `ps ax`.
6. In the first window, type `./sample_job sigterm sigkill`.
   - The program will start running.
7. In the second window, type `ps ax | grep sample` to get the id of the process.
8. Type `kill <id>`.
   - This time, the process will not die.
9. Type `kill -SIGKILL <id>`.
   - This time the process will exit.

Showing Process and Memory usage

1. In a terminal, type `top`.
   - A table will be shown, updated once per second, showing all of the processes on the system, as well as the overall CPU and memory usage.
2. Press the Shift+P key.
   - This will sort processes by CPU utilization.
     This can be used to determine which processes are using too much CPU time.
3. Press the Shift+M key.
   - This will sort processes by memory utilization
     This can be used to determine which processes are using too much memory.
4. Press q or Ctrl+C to exit the program.

2 - ROS2 - Robotique mobile

Turtlesim et modèle de Dubins

https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Introducing-Turtlesim/Introducing-Turtlesim.html

TurtleBot3 - Bases en Simulation

Astuces

Gazebo

- Réinitialiser la pose du robot
  - `ctrl+Shift+R`
  - Edit --> Reset Model Poses

Le robot ne spawn pas

On a remarqué que parfois certains processus de gazebo continuent à tourner ou sont redémarrés malgré l'arrêt du noeud ROS principal. Il faut alors tuer le processus avecla commande `kill 1234`, voir commandes utiles ci-dessous.

Commandes utiles

- Lister les processus système (programmes) qui tournent actuellement
  `ps -ef`
- Lister les processus système (programmes) qui tournent actuellement sous forme d'arbre hiérarchisé : un processus enfant est rattaché à une branche d'un processus parent dont il dépend
  `ps -ef --forest`
- Parmi ces processus, sélectionner ceux qui contiennent le mot-clé `gazebo`
  `ps -ef --forest | grep ros`
- Repérer l'ID du processus
- Envoyer le signal d'arrêt du processus `SIGTERM` "mode gentil" : on demande au programme de s'arrêter
  `kill 1234`
- Si le processus continue tout de même à tourner, envoyer le signal de destruction du processus `SIGKILL` "mode méchant" :
  `kill -9 1234`

Ressources

- https://www.classes.cs.uchicago.edu/archive/2022/fall/20600-1/warmup_project.html
- https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/#gazebo-simulation

TurtleBot3 - Piloter le Robot

https://emanual.robotis.com/docs/en/platform/turtlebot3/navigation/#navigation

Les 2 ordinateurs dans le FabLab sur le mur de gauche sont installés sous Ubuntu 22.04 avec ROS2 Humble et les paquets nécessaires au pilotage des TurtleBot3. Il faut utiliser le compte étudiant (et non le compte fablab qui se login automatiquement au démarrage) avec le même mdp que celui utilisé en TP.

Vous pouvez emprunter un des 2 TurtleBot quand vous voulez en demandant à M. Carron ou M. Hentz dans le bureau en face :
- TurtleBot3 Burger
- TurtleBot3 Waffle + OpenManipulator X
Pensez bien à les recharger pour les suivants.

Un réseau Wifi fab-lab-5g est disponible et les TurtleBot configurés pour s'y connecter. **Attention ce réseau est limité à 20Go de données, donc ne pas l'utiliser pour autre-chose que la robotique**. Les PC doivent être connectés sur le même réseau que les TurtleBot. Vous devez ensuite vous connecter au Robot via ssh pour démarrer le noeud ROS2 côté robot :

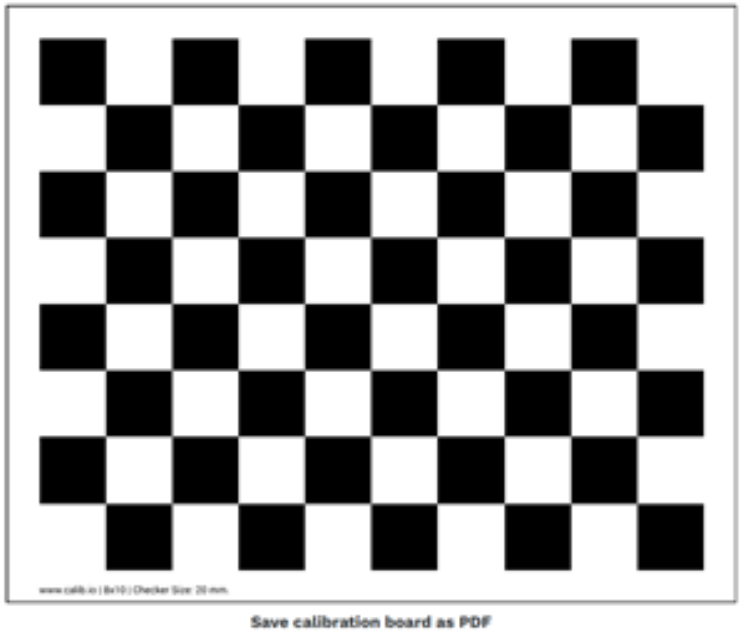
`ssh ubuntu@192.168.3.10` (burger) ou `ssh ubuntu@192.168.3.11` (waffle) avec le même mdp qu'en TP.

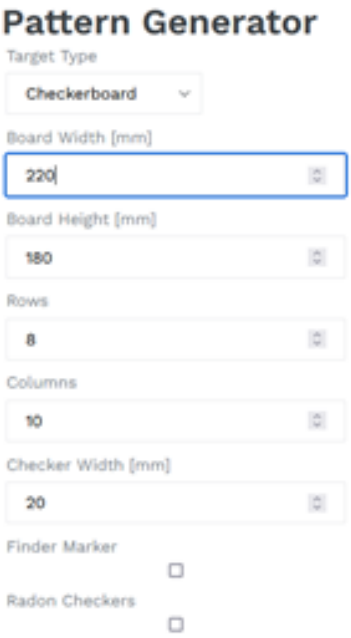
Les autres noeuds sont lancés sur le PC : télémanipulation, cartographie, navigation autonome, planification de trajectoire avec évitement d'obstacle etc.

Calibration de la caméra

https://docs.nav2.org/tutorials/docs/camera_calibration.html

- Générer un damier de calibration : 8 x 10 carrés de 20m
- avec https://calib.io/pages/camera-calibration-pattern-generator





- Ce sont les sommets intérieurs des carrés qui sont utilisés, donc 7x9 sommets

Ressources

- Noeud ROS2 pour Raspberry Cam https://github.com/christianrauch/camera_ros
- https://index.ros.org/p/camera_ros/
- https://medium.com/swlh/raspberry-pi-ros-2-camera-eef8f8b94304
-

Suivi de ligne ROS2 Humble

Le suivi de ligne repose surtout sur du traitement de l'image avec OpenCV, et l'envoi de commandes de vitesse au robot. Le code sous ROS1 devrait donc être portable assez directement sous ROS2.

Environnement de simulation Gazebo

Modélisation

Créer un paquet

- Créer un paquet ROS2 python « autonomy » qui implémente follower_p.py
- Adapter le CMakeLists.txt et package.xml en vous inspirant de :
  - turtlebot3_behavior_demos (/tb3_autonomy/scripts/test_vision.py)
  - turtlebot3/turtlebot3_example/turtlebot3_example/turtlebot3_position_control/turtlebot3_position_control.py
- Lancer ros2 launch turtlebot3_gazebo turtlebot3_circuit_competition.launch.py
- Démarrer la simulation en plaçant le robot au début de la piste
  - Caméra en vue de la ligne
- Lancer votre nœud python avec ros2 run autonomy follower_p.py

Ressources

Pour le suivi d'une ligne blanche :

- https://github.com/gabrielnhn/ros2-line-follower/blob/main/follower/follower/follower_node.py
  - `sudo apt install python3-cv-bridge python3-opencv`
  - ajouter au `~/.bashrc` : `export GAZEBO_MODEL_PATH=~/turtlebot3_ws/src/ros2-line-follower/follower/models`
- Le TP2 de robotique de Loïc Cuvillon donné à Telecom Physique Strasbourg

- ROS1 / OpenCV :
  https://github.com/osrf/rosbook/blob/master/followbot/follower_line_finder.py

Behavior Trees Demo

Concepts

https://docs.nav2.org/concepts/index.html#behavior-trees

https://docs.nav2.org/behavior_trees/overview/nav2_specific_nodes.html

https://docs.nav2.org/behavior_trees/overview/detailed_behavior_tree_walkthrough.html

Démo avec le Turtlebot3

https://github.com/sea-bass/turtlebot3_behavior_demos

Usage sans docker

- Installation

  https://github.com/sea-bass/turtlebot3_behavior_demos?tab=readme-ov-file#local-setup

- Vérifier que Gazebo fonctionne en lançant le noeud de base : `ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py`
- Éteindre le noeud

On lance l'environnement de simulation lié à la démo de Behavior Tree :

- `ros2 launch tb3_worlds tb3_demo_world.launch.py`
  Le robot navigue en des positions connues avec pour but de trouver un cube d'une couleur spécifiée (rouge, vert ou bleu). La détection d'objets est faite par un simple seuillage en couleurs HSV avec des valeurs calibrées.

Démos de Behavior Trees en Python

On regarde le fichier `turtlebot3_behavior_demos/docker-compose.yaml` pour déterminer les commandes Bash correspondant aux commande docker indiquées dans le dépôt.

Dans un second terminal, on lance une des démos suivantes :

- `ros2 launch tb3_autonomy tb3_demo_behavior_py.launch.py tree_type:=queue enable_vision:=true target_color:=green`
  Démo avec `py_trees`

Les fichiers source de la démo sont :

- tb3_demo_behavior_py.launch.py
- autonomy_node.py
  - navigation.py
    - test_move_base.py
  - vision.py
    - test_vision.py

Installation et démarrage du Turtlebot 3

Remarques sur les architectures ROS pour un robot mobile

L'architecture utilisée par Robotis pour son Turtlebot3, qui est également plébiscitée par la communauté de robotique mobile ROS Navigation 2, repose sur un PC embarqué (ARM64, Raspberry 4) sur le robot pour les calculs temps réel, et un PC de calcul et développement logiciel (AMD64, PC portable ou fixe) pour les calculs lourds et ayant une moindre contrainte temporelle. Les deux PC communiquent via un réseau wifi. On installe typiquement Ubuntu Server (sans interface graphique donc plus léger) sur la Raspberry et Ubuntu Desktop sur le PC.

Avec l'utilisation d'une Raspberry >=4, les problèmes de ressources sont moins importants et on peut envisager d'installer un environnement de bureau (interface graphique) et faire les calculs lourds sur la Raspberry. Dans cette architecture, on peut envisager de se connecter à l'environnement de bureau de la Raspberry depuis un PC (Linux ou Windows) via le wifi et VNC.

> Il est déconseillé d'utiliser les applications graphiques de ROS comme RViz et Gazebo sur architecture ARM64. Par exemple, Gazebo 11 n'est pas disponible sur ARM64 sous ROS Humble. Il l'est depuis peu sous Jazzy.

Préconisation

Nous préconisons l'architecture suivante pour les TP et projets :

- ROS2 Humble
  - version LTS
  - Robotis ne maintient les paquets du Turtlebot3 que pour ROS2 Humble (et non Jazzy)
- version Desktop sur un PC Ubuntu 22
- version Server sur une Raspberry Pi 4 (et non Pi 5 qui nécessite de compiler Humble depuis les sources)

- Connexion du PC et de la Raspberry via un hotspot wifi émis par un routeur sur lequel on a les droits administrateur
- Routeur wifi avec DHCP et possibilité de fixer les IP du PC et de la Raspberry. Accès internet via ethernet ou 4G/5G
- Développement sur le PC dans Visual Studio Code, configuré pour gérer les fichiers sur la Raspberry (via ssh a priori)

Pour simplifier l'expérience utilisateur :

- Si nécessaire, pour débogage du réseau par exemple, installation d'un environnement de bureau sur la Raspberry en suivant les instructions ci-dessous
  - Il est dès lors possible de se connecter à la Raspberry via VNC pour développer directement dessus
- Si nécessaire, configuration du PC pour qu'une connexion Ethernet avec le robot permette le partage de sa connexion internet.

Installation Ubuntu Server

Configuration clavier en Français

- `sudo dpkg-reconfigure keyboard-configuration`
- Choisir toutes les options françaises par défaut

Version rapide

- Télécharger l'image compressée de la carte SD préinstallée

  https://seafile.unistra.fr/smart-link/dcc9e405-88a0-41d2-ab5c-3e796a6cebf3/
- Insérer la carte SD et démonter les partitions existantes

```
sudo umount /media/user/writable /media/user/system-boot
```

> Attention, bien vérifier le disque associé à la carte SD avant de lancer la commande dd. Sinon on risque d'écraser le disque dur. En général disque dur = /dev/sda et carte SD = /dev/sdb

- Flasher la carte SD avec l'utilitaire `dd`

```
sudo gunzip -c ~/turtlebot3-manipulator-humble.img.gz | sudo dd of=/dev/sdb status=progress
```

- Configurer la connexion automatique au réseau wifi et donner une IP fixe au robot (dans la plage DHCP autorisée par le routeur) :
  - Éjecter et réinsérer la carte SD pour qu'elle se monte
  - Modifier la `CONFIGURATION_RESEAU` (les éléments en majuscule) dans le fichier suivant :
    `sudo nano /media/user/writable/etc/netplan/99-hotspot-fablab.yaml`

```
            addresses: [IP_TURTLEBOT/24]
            gateway4: IP_BOX
            nameservers:
                addresses: [DNS_BOX_OPERATEUR, 9.9.9.9, 89.234.141.66]
            access-points:
                "SSID_WIFI":
                    password: PASSWORD_WIFI
```

Remarque : l'image a été créée après avoir suivi les instructions longues ci-dessous (et quelques workspace et package ros installés en plus) en lançant la commande suivante :

```
sudo dd if=/dev/sda status=progress | gzip -9 > ~/turtlebot3-manipulator-humble.img.gz
```

Reinitialiser le mot-de-passe

Voir la section 4 ici

- Connecter la carte SD sur un PC
- naviguer au point de montage, par exemple :
  `cd /media/user/writable`
- ouvrir le fichier
  `sudo nano /etc/passwd`
- modifier la ligne qui concerne votre user du système Ubuntu du turtlebot, par exemple ici `ubuntu`
  Avant : `ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash`
  Après (on supprime le `x`) : `ubuntu::1000:1000:Ubuntu:/home/ubuntu:/bin/bash`
- Redémarrer le système dans le TurtleBot
- Se connecter avec le login `ubuntu`
- Aucun MDP n'est demandé
- Lancer le programme de changement de MDP
  `sudo passwd`
- Rentrer le nouveau MDP 2 fois

Installation d'un environnement graphique

Pour installer un environnement graphique sur la Raspberry Pi 4 préalablement installée en Ubuntu server. On choisit Mate qui est léger et assez moderne à la fois :

- `sudo apt install ubuntu-mate-desktop`
- `sudo reboot`

cf. https://phoenixnap.com/kb/how-to-install-a-gui-on-ubuntu#ftoc-heading-4

Version longue

https://emanual.robotis.com/docs/en/platform/turtlebot3/sbc_setup/#sbc-setup

Depuis un ordinateur sous Ubuntu 22.04

- Insérer la carte SD dans le navigateur
- Installer rpi-manager `sudo apt install rpi-imager`
- Sélectionner `CHOOSE OS`
  - autre système Linux (Other general-purpose OS)
  - Ubuntu Server 22.04 LTS (64bits)
- Sélectionner `CHOOSE STORAGE` la carte micro SD
- Cliquer sur `WRITE`

Depuis une VM WSL Ubuntu 22

A faire

Configuration réseau

Configurer la connexion automatique au réseau wifi et donner une IP fixe au robot (dans la plage DHCP autorisée par le routeur) :

- Éjecter et réinsérer la carte SD pour qu'elle se monte
- Créer le fichier suivant : `sudo nano /media/user/writable/etc/netplan/99-hotspot-fablab.yaml`

```
network:
    renderer: networkd
    ethernets:
        eth0:
            dhcp4: true
            dhcp6: true
            optional: true
    wifis:
        wlan0:
            dhcp4: false
```

```
            dhcp6: false
            addresses: [192.168.100.40/24]
            gateway4: 192.168.100.1
            nameservers:
                addresses: [192.168.100.1, 9.9.9.9, 89.234.141.66]
            access-points:
                fablab:
                    password: ...
    version: 2
```

- Désactiver la configuration du réseau par cloud-init en créant le fichier suivant : `sudo nano /media/user/writable/etc/cloud/cloud.cfg.d/99-disable-network-config.cfg`

```
network: {config: disabled}
```

Connexion au robot en ssh

Insérer la carte dans le robot, le démarrer assez proche du hotspot wifi configuré, se connecter en ssh depuis l'ordinateur :

- Utiliser l'adresse IP précédemment configurée `ssh ubuntu@192.168.100.40`
- mdp : `ubuntu`
- changer le mdp par un suffisamment sécurisé
- se connecter en ssh avec le nouveau mdp
- pour lancer des commandes en root utiliser `sudo` avec le même mdp
- Ne pas attendre la connexion réseau pour démarrer : `systemctl mask systemd-networkd-wait-online.service`
- Désactiver la veille et l'hibernation : `sudo systemctl mask sleep.target suspend.target hibernate.target hybrid-sleep.target`

Installation de ROS embarqué

Installer **ROS 2 Humble** sans interfaces graphiques (`ros-humble-desktop`) qui seront lancées sur l'ordinateur externe :

```
sudo apt update && sudo apt install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8
sudo apt install software-properties-common
sudo add-apt-repository universe
sudo apt update && sudo apt install curl
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o
```

```
/usr/share/keyrings/ros-archive-keyring.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
sudo apt update && sudo apt upgrade
sudo apt install ros-humble-ros-base ros-dev-tools
source /opt/ros/humble/setup.bash
echo 'source /opt/ros/humble/setup.bash' >> ~/.bashrc
```

Installer le workspace du turtlebot et les dépendances :

```
sudo apt install python3-argcomplete python3-colcon-common-extensions libboost-system-dev build-essential
sudo apt install ros-humble-hls-lfcd-lds-driver
sudo apt install ros-humble-turtlebot3-msgs
sudo apt install ros-humble-dynamixel-sdk
sudo apt install libudev-dev
mkdir -p ~/turtlebot3_ws/src && cd ~/turtlebot3_ws/src
git clone -b humble-devel https://github.com/ROBOTIS-GIT/turtlebot3.git
git clone -b ros2-devel https://github.com/ROBOTIS-GIT/ld08_driver.git
cd ~/turtlebot3_ws/src/turtlebot3
rm -r turtlebot3_cartographer turtlebot3_navigation2
cd ~/turtlebot3_ws/
echo 'source /opt/ros/humble/setup.bash' >> ~/.bashrc
source ~/.bashrc
colcon build --symlink-install --parallel-workers 1
echo 'source ~/turtlebot3_ws/install/setup.bash' >> ~/.bashrc
source ~/.bashrc
```

Configurer le Port USB pour OpenCR :

```
sudo cp `ros2 pkg prefix turtlebot3_bringup`/share/turtlebot3_bringup/script/99-turtlebot3-cdc.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules
sudo udevadm trigger
```

ID de domain ROS pour la communication DDS :

```
echo 'export ROS_DOMAIN_ID=30 #TURTLEBOT3' >> ~/.bashrc
source ~/.bashrc
```

Configurer le modèle du LIDAR :

```
echo 'export LDS_MODEL=LDS-02' >> ~/.bashrc
source ~/.bashrc
```

Raspberry Pi 5 - Ubuntu Noble 24.04 - ROS2 Jazzy

ROS2 Jazzy est la nouvelle version LTS. Elle est installable sur Raspberry Pi 5.

Pour installer ROS2 Humble sur Ubuntu 24.04 il faut compiler depuis les sources :

```
sudo apt install -y git colcon python3-rosdep2 vcstool wget python3-flake8-docstrings python3-pip python3-pytest-cov python3-flake8-blind-except python3-flake8-builtins python3-flake8-class-newline python3-flake8-comprehensions python3-flake8-deprecated python3-flake8-import-order python3-flake8-quotes python3-pytest-repeat python3-pytest-rerunfailures python3-vcstools libx11-dev libxrandr-dev libasio-dev libtinyxml2-dev

mkdir -p ~/ros2_iron/src

cd ~/ros2_iron

vcs import --input https://raw.githubusercontent.com/ros2/ros2/iron/ros2.repos src

sudo rm /etc/ros/rosdep/sources.list.d/20-default.list

sudo apt upgrade

sudo rosdep init

rosdep update

rosdep install --from-paths src --ignore-src --rosdistro iron -y --skip-keys "fastcdr rti-connext-dds-6.0.1 urdfdom_headers python3-vcstool"

colcon build --symlink-install
```

cf. https://forums.raspberrypi.com/viewtopic.php?t=361746

3 - ROS2 - Manipulation Mobile

Assemblage du Turtlebot et OpenManipulator-X et configuration initiale

https://www.classes.cs.uchicago.edu/archive/2022/fall/20600-1/turtlebot_assembly_setup.html

A Word of Caution

This page is NOT intended for student use. This page is designed for course staff to build and maintaing the turtlebots. **If you are a student, please ensure you have permission from course staff before making any of the changes detailed on this page to any of the turtlebots**.

Setup Checklist

Turtlebot Assembly Tips

Turtlebot3

- DO NOT TIGHTEN the screws too much.
- Page 10: When connecting one pair of waffle-plates to another, make that the seams connecting each waffle-plate are parallel to one other.
- Page 17: When attaching the wheels to the dynamixels, do not tighten the screws too much.
- Page 28: Mount the LiDAR further up than the manual specifies so that the arm can fit on (see completed Turtlebots for reference).

OpenMANIPULATOR Arm

- OpenMANIPULATOR Dynamixel 12: when putting the back piece onto the servo, you will need to break off a plastic piece where the cable should be threaded through.
- Make sure the placement of the marking on the servo horns is identical to the diagram PRIOR to screwing anything down onto it.

OpenMANIPULATOR Arm First-Time Configuration

After assembling an OpenMANIPULATOR arm, its servos must first be configured properly before it can be used. By default, all the arm motors are set to the same ID (1) and the wrong baud rate, which causes collisions when trying to detect them.

- Setup OpenCR with Arduino IDE using OpenCR 1.0 manual, “Arduino IDE” section

- Download Dynamixel Wizard 2.0 if not already installed.

1. Connect a **SINGLE** motor (no daisy-chains in the arm) to the OpenCR module and **DISCONNECT ALL OTHER MOTORS** (the wheel motors!!)
2. Open up Dynamixel Wizard 2.0 and update the firmware for that motor by following this tutorial. The arm Dynamixel model is XM430-**W350**, and the wheel motors are XM430-**W210**.
3. Scan for connected Dynamixels using the “Scan” button on the top menu. If the scan does not turn up any results, you may need to change the scan options in the "Options" menu. By default, an unconfigured arm motor will have ID 1, be on Protocol 2.0, and have a baud rate of 57600 bps.
4. Change the ID for the detected motor from 1 to 11/12/13/14/15 (whichever you're doing the procedure for). Click on the “ID” item, and find the ID # you want in the lower right corner. Click it and press “Save”.
5. Change the baud rate to 1M (if not already 1M). Click on the “Baud Rate (Bus)” item, and find the 1 Mbps option. Click it and press “Save”.
6. Disconnect the motor (both in the wizard by clicking “Disconnect” up top and physically disconnecting from the board) and repeat the steps for the remaining ones.

Attaching/Detaching the OpenMANIPULATOR Arm

Physical Attachment/Detachment

Follow these steps below:

1. The arm can be attached and detached by first removing the top layer of the the Turtlebot. There are 10 screws around the perimeter of the top layer that you will need to remove with a Phillips head / cross screwdriver.
2. Once the top layer of the Turtlebot is loose, disconnect the cable that is attached to the LiDAR. This cable is multi-colored with several pins.

> Note: ONLY remove this LiDAR cable in this step.

Turtlebot3 with an arm.

Once the top layer is fully disconnected, flip the top layer upside-down for easy access.

3. To connect the arm, there are **four screws** that you will need to insert through the **bottom side** of the top layer that connect to **Dynamixel #11**. The exact location of the screws are marked in the picture below. **Use an hex/allen key to secure them.**

Turtlebot3 with an arm.

4. Once the arm is securely attached, thread the cable connected to DynaMIXEL #11 through the top layer (if not already threaded) and attach the other end to the remaining **3-pin** slot on the OpenCR board. It should be the open slot adjacent to the already occupied slots.
5. Reconnect the LiDAR cable and re-screw the top layer onto the robot.

To disconnect the arm, follow the same steps as above, but remove the four screws instead and disconnect the DynaMIXEL.

Updating Firmware and Files

Every time the arm is attached or detached, you will need to reconfigure environment variables and update the OpenCR firmware.

If you are *attaching* the arm, set the environment variable `OPENCR_MODEL=om_with_tb3` and ensure that `OPENCR_PORT=/dev/ttyACM0` in the `~/.bashrc` file.

> Note: You can also set those environment variables in the command line by running:

```
$ export OPENCR_PORT=/dev/ttyACM0
$ export OPENCR_MODEL=om_with_tb3
```

Source the file and update the firmware with:

```
$ source ~/.bashrc
```

```
$ cd ~/opencr_update && ./update.sh $OPENCR_PORT $OPENCR_MODEL.opencr
```

If you are *detaching* the arm, set the environment variable `OPENCR_MODEL=waffle` and ensure that `OPENCR_PORT=/dev/ttyACM0` in the `~/.bashrc` file.

Enter the OpenCR board into firmware recovery mode by pressing and holding SW2, and simultaneously pressing RESET (you will need to remove the top layer of the turtlebot in order to access those OpenCR buttons). If you want to read more about the OpenCR bootloader, please refer to this page in the OpenCR 1.0 manual.

buttons on OpenCR

Image not found or type unknown

Source the file and update the firmware with:

```
$ source ~/.bashrc
```

```
$ cd ~/opencr_update && ./update.sh $OPENCR_PORT $OPENCR_MODEL.opencr
```

Common Errors During Setup

Firmware Recovery Mode and Device Firmware Update for OpenCR Board

We found the **firmware recovery mode** not clearly detailed in the Turtlebot3 manual. Here are the steps for firmware recovery:

1. Push and hold SW2
2. Press RESET
3. Release SW2
4. Arduino code should upload successfully with `jump_with_fw`

Firmware recovery is different from **device firmware update (DFU)**, where the steps are:

1. Push and hold BOOT
2. Press RESET
3. Release BOOT

Troubleshooting `/dev/ttyACM0` Connection Issues

If the `/dev/ttyACM0` port isn't found or running `bringup` on the Pi is hanging:

- Make sure `$OPENCR_MODEL` is correct (`waffle` for no arm, `om_with_tb3` for arm)
- Disconnect and reconnect the USB cable connecting the Pi and OpenCR board
- Enter firmware recovery (SW2 + RESET)
- Reinstall the firmware with `./update.sh $OPENCR_PORT $OPENCR_MODEL.opencr` (see more detailed steps above in the prior section on attaching/detaching the arm)
- [If the above doesn’t work] Restart `roscore`
- [If the above doesn’t work] Reupload the sketch using the Arduino IDE (install instructions) in `Examples > OpenCR > Etc > usb_to_dxl` to the OpenCR board (which you'll need to directly connect to your computer)

If the hydro rosserial / groovy Arduino error appears:

- Configure the arm motors as described in the previous section on attaching/detaching the arm
- Enter firmware recovery (SW2 + RESET)
- Reinstall the firmware with `./update.sh $OPENCR_PORT $OPENCR_MODEL.opencr`
- Set up the wheel motors by uploading the sketch using the Arduino IDE (install instructions) in `Examples > TurtleBot3 > turtlebot3_setup > turtlebot3_setup_motor` to the OpenCR board (which you'll need to directly connect to your computer)

Installation et démarrage du OpenManipulator-X & Turtlebot 3

Suivre d'abord https://innovation.iha.unistra.fr/books/robotique-open-source/page/installation-et-demarrage-du-turtlebot-3

Montage et Configuration des Dynamixels

https://www.classes.cs.uchicago.edu/archive/2022/fall/20600-1/turtlebot_assembly_setup.html

- Installer Arduino IDE : `sudo apt install arduino`
- https://emanual.robotis.com/docs/en/parts/controller/opencr10/#install-on-linux
- Connecter OpenCR
  https://emanual.robotis.com/docs/en/platform/turtlebot3/manipulation/#arduino-ide
- Installer Dynamixel Wizard
  https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/
- Lancer Dynamixel Wizard
  ```
  cd ~/ROBOTIS/DynamixelWizard2
  bash DynamixelWizard2.sh
  ```
- Si une erreur de dépendance apparait, désinstaller/réinstaller/upgrader dynamixelWizard via l'executable `~/ROBOTIS/DynamixelWizard2/maintenancetool`
  https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_wizard2/#uninstall-linux

Pour l'Open-Manipulator

Configurer les dynamixel (baud et ID 11 à 15)

https://www.classes.cs.uchicago.edu/archive/2022/fall/20600-1/turtlebot_assembly_setup.html#arm-first-time :

- Connect a **SINGLE** motor (no daisy-chains in the arm) to the OpenCR module and **DISCONNECT ALL OTHER MOTORS** (the wheel motors!!)
- Open up Dynamixel Wizard 2.0 and update the firmware for that motor by following this tutorial. The arm Dynamixel model is XM430-**W350**, and the wheel motors are XM430-**W210**.
- Scan for connected Dynamixels using the “Scan” button on the top menu. If the scan does not turn up any results, you may need to change the scan options in the "Options" menu. By default, an unconfigured arm motor will have ID 1, be on Protocol 2.0, and have a baud rate of 57600 bps.
- Change the ID for the detected motor from 1 to 11/12/13/14/15 (whichever you're doing the procedure for). Click on the “ID” item, and find the ID # you want in the lower right corner. Click it and press “Save”.
- Change the baud rate to 1M (if not already 1M). Click on the “Baud Rate (Bus)” item, and find the 1 Mbps option. Click it and press “Save”.
- Disconnect the motor (both in the wizard by clicking “Disconnect” up top and physically disconnecting from the board) and repeat the steps for the remaining ones

Pour le Turtlebot

Via Arduino IDE ou DynamixelWizard en s'inspirant de :

https://emanual.robotis.com/docs/en/platform/turtlebot3/faq/#setup-dynamixels-for-turtlebot3

- Moteur gauche : ID=1
- Moteur droit : ID=2
- Baud rate : 1M

Test depuis un PC sans la raspberry

Téléopération du OpenManipulator-X seul

Suivre le tutoriel Foxy en remplaçant `foxy` par `humble` et `foxy-devel` par `ros2` en utilisant l'interface de communication OpenCR :

https://emanual.robotis.com/docs/en/platform/openmanipulator_x/quick_start_guide/

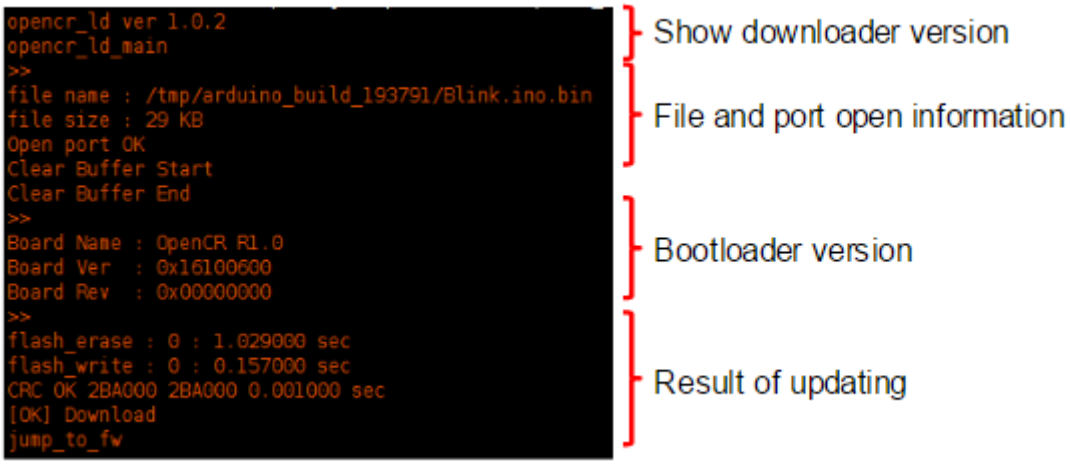
Pour tester le bon fonctionnement du bras et de sa pince, on connecte la carte OpenCR directement à un PC ayant ROS Humble préinstallé :

- Installer et compiler le workspace

```
sudo apt install ros-humble-rqt* ros-humble-joint-state-publisher
mkdir -p ~/openmanipulator_ws/src/
cd ~/openmanipulator_ws/src/
```

```
git clone -b ros2 https://github.com/ROBOTIS-GIT/DynamixelSDK.git
git clone -b ros2 https://github.com/ROBOTIS-GIT/dynamixel-workbench.git
git clone -b ros2 https://github.com/ROBOTIS-GIT/open_manipulator.git
git clone -b ros2 https://github.com/ROBOTIS-GIT/open_manipulator_msgs.git
git clone -b ros2 https://github.com/ROBOTIS-GIT/open_manipulator_dependencies.git
git clone -b ros2 https://github.com/ROBOTIS-GIT/robotis_manipulator.git
cd ~/openmanipulator_ws && colcon build --symlink-install
```

- Corriger le bug de compilation et re-compiler. Dans `src/open_manipulator/open_manipulator_x_controller/src/open_manipulator_x_controller.cpp`, lignes 67-68, remplacer :
  `this->declare_parameter("sim");`
  `this->declare_parameter("control_period");`
  par :
  `this->declare_parameter("sim", false);`
  `this->declare_parameter("control_period", 0.010);`
- Lancer `arduino`
- Uploader l'exemple `File > Examples > OpenCR > 10.Etc > usb_to_dxl` vers OpenCR



- Si cela réussit, jump_to_fw apparaît, sinon essayer d'uploader une seconde fois
- Lancer le contrôleur du robot. **Attention les moteurs vont bouger et se bloquer dans la position initiale**
  `ros2 launch open_manipulator_x_controller open_manipulator_x_controller.launch.py usb_port:=/dev/ttyACM0`
- Dans un second terminal, lancer le noeud de téléopération :
  `ros2 run open_manipulator_x_teleop teleop_keyboard`
- Piloter le robot dans l'espace Cartésien ou articulaire avec les touches indiquées

Programmation hors-ligne du OpenManipulator-X et TurtleBot3 depuis MoveIt

On suit le tutoriel https://emanual.robotis.com/docs/en/platform/turtlebot3/manipulation/#operate-the-actual-openmanipulator en installant tout ce qui est censé être installé sur le raspberry **[SBC]**/ **[TurtleBot3]** sur le PC **[Remote PC]**.

- Installer le workspace et compiler :

```
sudo apt install ros-humble-dynamixel-sdk ros-humble-ros2-control ros-humble-ros2-controllers
ros-humble-gripper-controllers ros-humble-moveit
cd ~/turtlebot3_ws/src/
git clone -b humble-devel https://github.com/ROBOTIS-GIT/turtlebot3_manipulation.git
cd ~/turtlebot3_ws && colcon build --symlink-install
```

- Ajouter au `~/.bashrc` :

```
export ROS_DOMAIN_ID=30 #TURTLEBOT3
export LDS_MODEL=LDS-02
export TURTLEBOT3_MODEL=waffle_pi
export OPENCR_PORT=/dev/ttyACM0
export OPENCR_MODEL=turtlebot3_manipulation
```

- AVANT TOUT FLASHAGE DE OPENCR, se mettre en **mode debug** en
  - Rester appuyé sur le bouton SW2
  - Appuyer quelques secondes sur RESET
  - Relacher RESET
  - Relacher SW2
- ATTENTION SI LE MODE DEBUG n'est pas activé, il se peut `jump_fw` soit affiché mais que le flashage ait échoué.
- Configurer OpenCR pour turtlebot3_manipulation depuis Arduino `File > Examples > Turtlebot3 ROS2 > turtlebot3_manipulation` ou avec le prebuild :

```
rm -rf ./opencr_update.tar.bz2
wget https://github.com/ROBOTIS-GIT/OpenCR-
Binaries/raw/master/turtlebot3/ROS2/latest/opencr_update.tar.bz2
tar -xvf opencr_update.tar.bz2
cd ./opencr_update
./update.sh $OPENCR_PORT $OPENCR_MODEL.opencr
```

- Démarrer ROS Control : `ros2 launch turtlebot3_manipulation_bringup hardware.launch.py`
- **Le setup a fonctionné si le robot apparaît dans la bonne configuration dans RViz !**
- Dans un second terminal démarrer au choix :

- MoveIt pour la programmation hors-ligne et planification de trajectoire : `ros2 launch turtlebot3_manipulation_moveit_config moveit_core.launch.py`
  - Piloter le robot en bougeant les flèches dans RViz et en cliquant sur "Plan and Execute"
- MoveIt servo `ros2 launch turtlebot3_manipulation_moveit_config servo.launch.py`
  - et la téléopération avec le clavier (dans un 3ème terminal) `ros2 run turtlebot3_manipulation_teleop turtlebot3_manipulation_teleop`
  - Piloter le robot dans l'espace Cartésien ou articulaire avec les touches indiquées

Configuration OpenCR

Pour le Turtlebot : https://emanual.robotis.com/docs/en/platform/turtlebot3/opencr_setup/#opencr-setup

Pour l'OpenManipulator-X : https://emanual.robotis.com/docs/en/platform/turtlebot3/manipulation/#opencr-setup

Dépendances manquantes :

```
sudo apt install ros-humble-hardware-interface
ros-humble-ros2-control ?
ros-humble-joint-state-publisher ?
```

Dépendances manquantes côté Raspberry :

```
sudo apt install rros-humble-gripper-controllers ros-humble-xacro
```

Dépendances manquantes côté PC :

```
sudo apt install ros-humble-moveit-servo
```

Issues :

https://forum.robotis.com/t/ros-2-foxy-openxmanuipaltor-bringup-issues/2142/9

https://github.com/ROBOTIS-GIT/open_manipulator/issues/212

https://github.com/ROBOTIS-GIT/open_manipulator/issues/209

https://www.classes.cs.uchicago.edu/archive/2022/fall/20600-1/turtlebot_assembly_setup.html#arm-first-time

---

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

4 - ROS2 - Manipulation Cobot

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

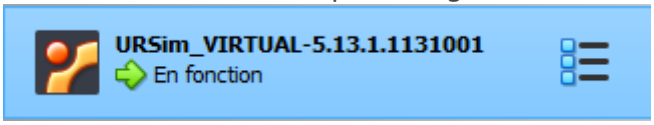
Pour les systèmes non-Linux, UR fournit une VM LUbuntu 14.04 qui peut tourner sous VirtualBox (gratuit) ou sur VMware (gratuit pour usage non commercial).

Installer VirtualBox

Télécharger et configurer la Machine Virtuelle

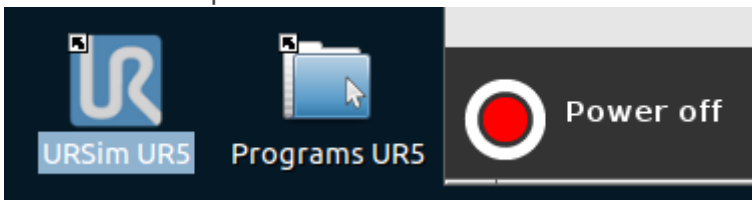
- Télécharger l'archive de la VM sur seafile ou en créant un compte chez UR
- Extraire l'archive dans `C:\Users\user\VirtualBox VMs\URSim_VIRTUAL-5.13.1.1131001`
- Dans l'interface VirtualBox cliquer sur `Ajouter`

- 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

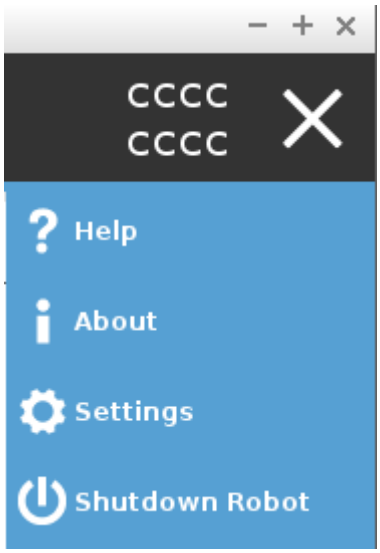


- La session Lubuntu est
  - user : `ur`
  - mdp : `easybot`

- Lancer URSim pour l'UR5e



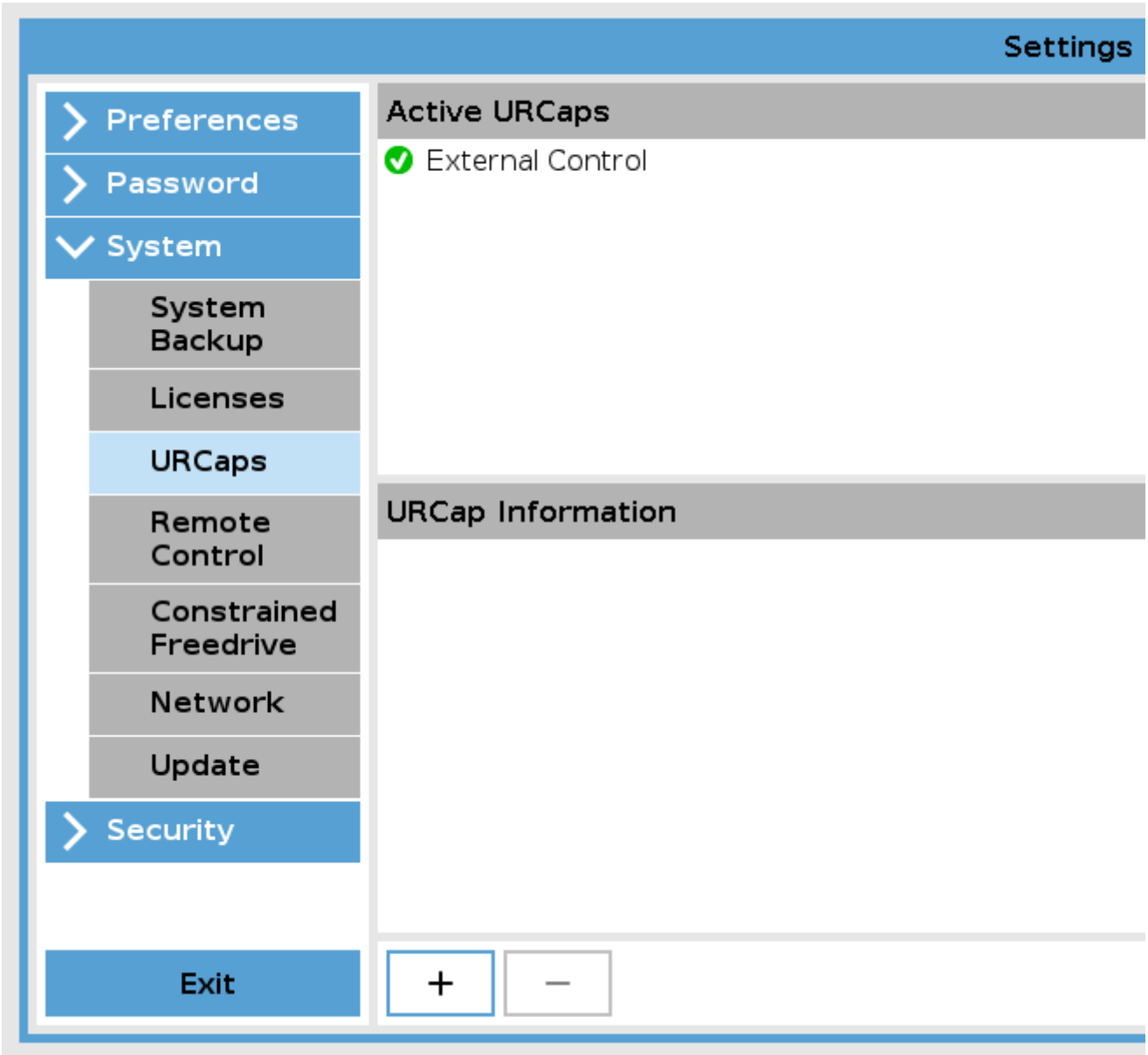
- Démarrer le robot en cliquant sur le bouton d'allumage "Power"
- Pour éteindre le robot



- Ou simplement éteindre la VM

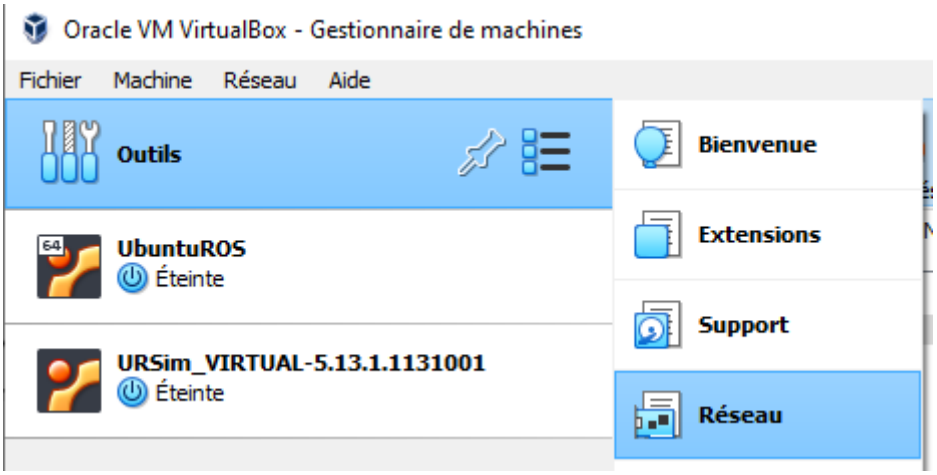
Ajouter l'URCap External Control

- 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

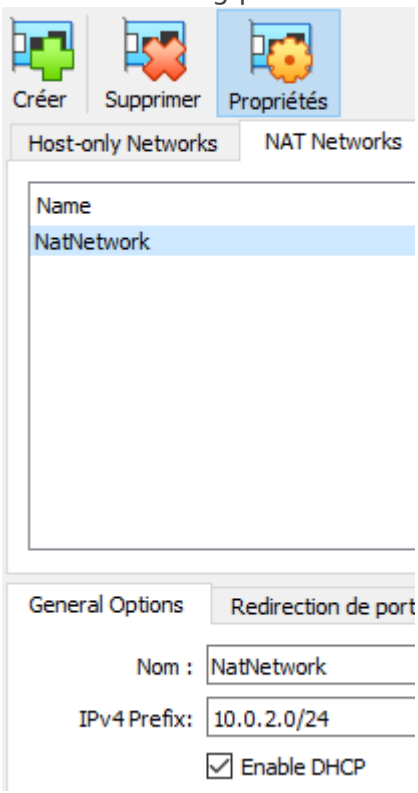


Configurer le réseau pour que les VMs communiquent

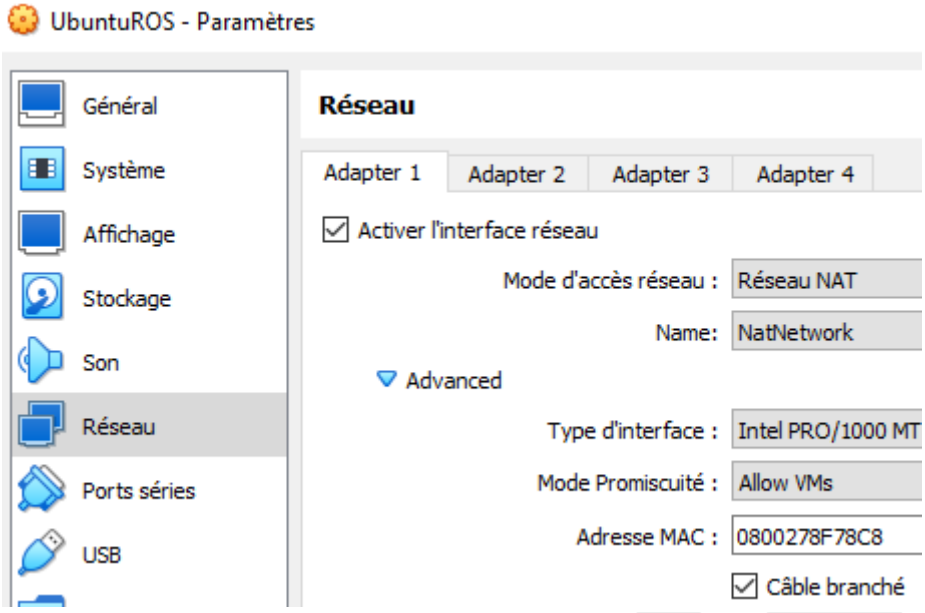
- Créer un réseau NAT dans les outils réseau



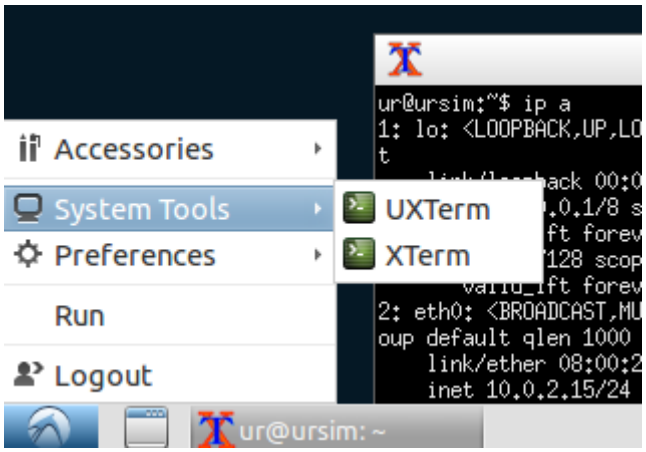
- Laisser la config par défaut



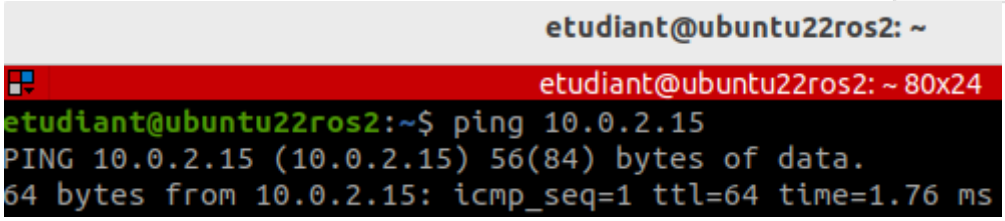
- Configurer chaque VM qui doit communiquer en `Réseau NAT`, par défaut elles seront en DHCP sur le réseau `10.0.2.0/24`



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



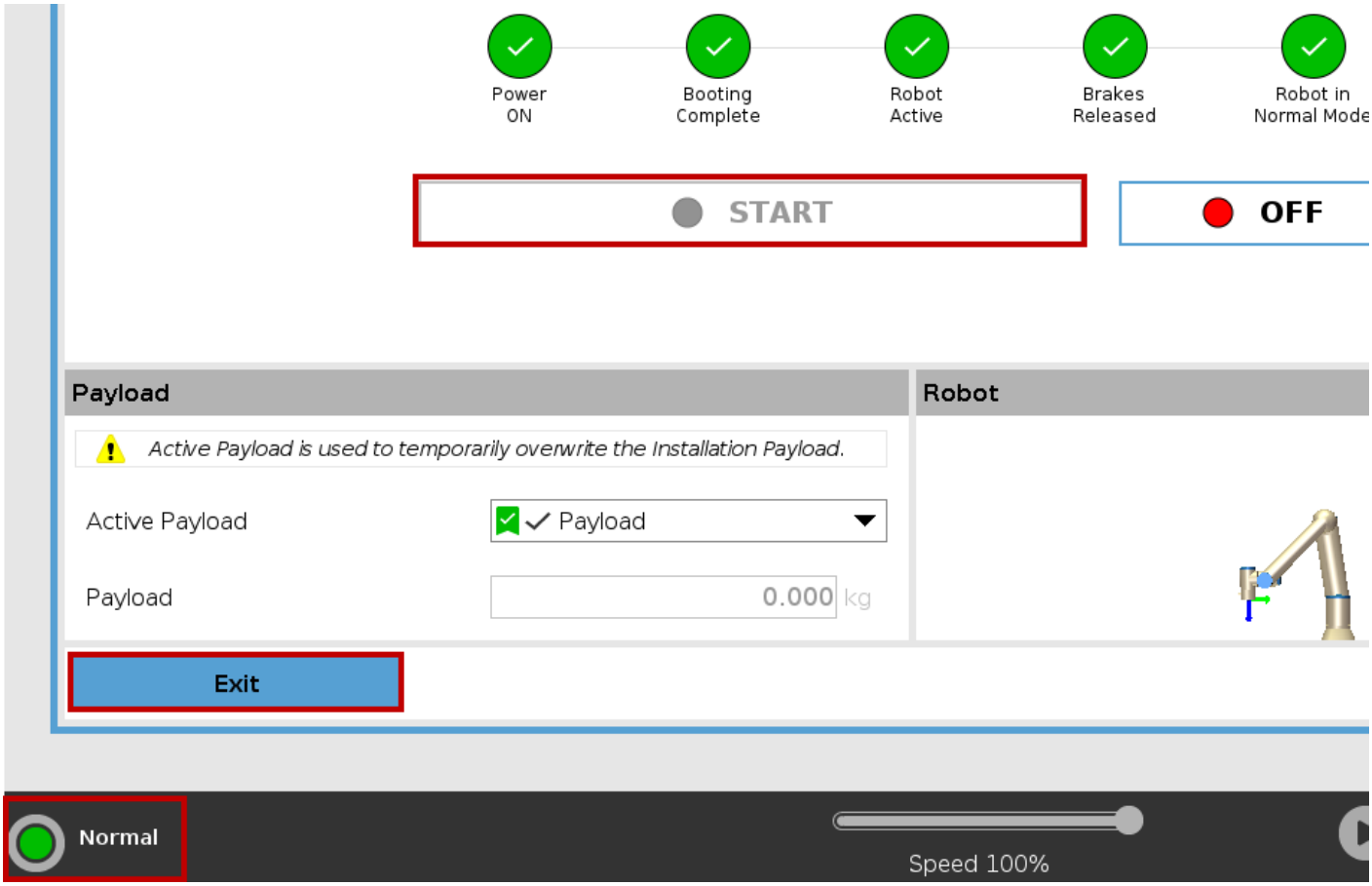
- Tester la communication de la VM ROS vers la VM URSim avec `ping 10.0.2.15`



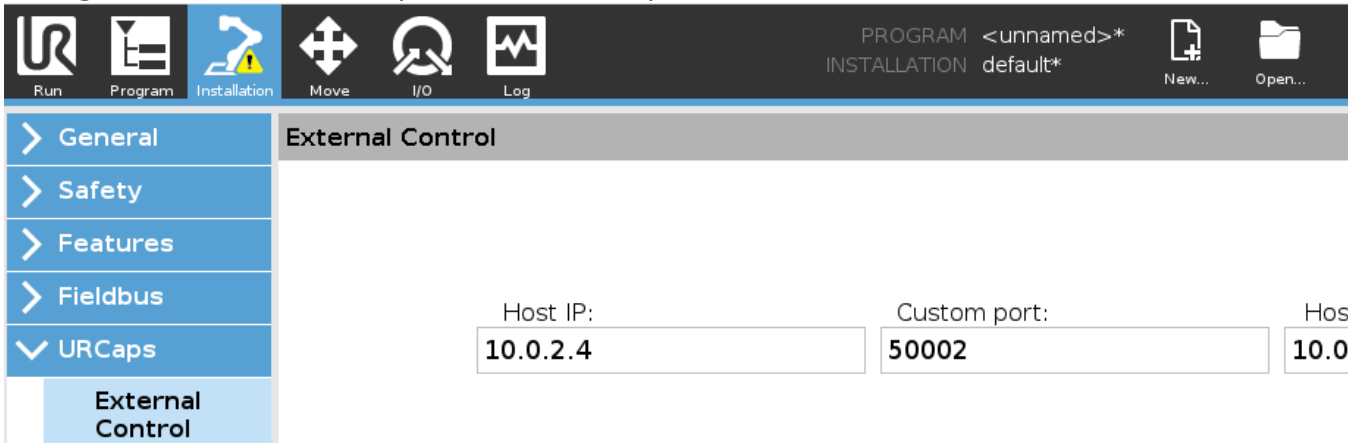
- Tester la communication de la VM URSim vers la VM ROS avec `ping 10.0.2.4`
- Tester la communication entre ROS et URSim, voir la section dédiée

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

Commander un robot UR avec le driver ROS2

Installation d'Ubuntu avec des capacités temps-réel

Pour un usage basique, un Ubuntu (ou Linux Mint) classique permet de piloter le robot :

- Installer Ubuntu LTS 22.04
-

Pour éviter des instabilités il est conseillé d'installer un noyau Linux avec des capacités temps-réel (PREEMPT_RT kernel). En particulier avec un robot de la Série-E, la fréquence de contrôle plus élevée peut entraîner des trajectoires non fluides sans noyau temps-réel.

- Vérifier qu'il reste au moins 25Go d'espace disque
- On se place dans un dossier pour la compilation

```
mkdir -p ~/rt_kernel_build
cd ~/rt_kernel_build
```

```
tar xf linux-4.14.139.tar
cd linux-4.14.139
xzcat ../patch-4.14.139-rt66.patch.xz | patch -p1
make oldconfig
```

Récupérer les sources du noyau temps-réel

- Regarder les versions LTS du noyau Linux : https://www.kernel.org/category/releases.html

- Regarder la version actuelle et les versions proposées par Ubuntu : Update Manager -> view -> Linux Kernel
- Regarder les versions maintenues activement du noyau PREEMPT_RT -> https://wiki.linuxfoundation.org/realtime/preempt_rt_versions
- Choisir la version du noyau Linux PREEMPT_RT maintenue activement correspondant à une version LTS et proposée par Ubuntu
- Exemple pour un Lenovo Thinkpad T480
  - Actuellement installé : 5.19 (non-LTS)
  - Proposé : 5.19 et 5.15 (LTS)
  - On choisit 5.15 car LTS et activement maintenu en PREEMPT_RT
- Récupérer les sources pour `5.15.107-rt62`

```
wget https://cdn.kernel.org/pub/linux/kernel/projects/rt/5.15/patch-5.15.107-rt62.patch.xz
wget https://cdn.kernel.org/pub/linux/kernel/projects/rt/5.15/patch-5.15.107-rt62.patch.sign
wget https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.15.107.tar.xz
wget https://www.kernel.org/pub/linux/kernel/v5.x/linux-5.15.107.tar.sign
```

- Décompresser les fichiers téléchargés

```
xz -dk patch-5.15.107-rt62.patch.xz
xz -d linux-5.15.107.tar.xz
```

- Importer les clés publiques des développeurs du noyau et du patch (5.15-rt Joseph Salisbury 2026-10)

```
gpg2 --locate-keys torvalds@kernel.org gregkh@kernel.org
gpg2 --keyserver hkp://keys.gnupg.net --search-keys salisbury
```

- Vérifier l'intégrité des fichiers source

```
gpg2 --verify linux-5.15.107.tar.sign
```

> gpg: Good signature from "Greg Kroah-Hartman <gregkh@kernel.org>" [unknown]
> gpg: WARNING: This key is not certified with a trusted signature!
> gpg: There is no indication that the signature belongs to the owner.

```
gpg2 --verify patch-5.15.107-rt62.patch.sign
```

```
gpg: Good signature from "Tom Zanussi <tom.zanussi@linux.intel.com>"
[unknown]
gpg:                 aka "Tom Zanussi <zanussi@kernel.org>" [unknown]
gpg:                 aka "Tom Zanussi <tzanussi@gmail.com>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
```

Compiler le noyau temps réel

- Extraire l'archive tar, appliquer la patch et configurer le noyau temps-réel

```
tar xf linux-5.15.107.tar
cd linux-5.15.107
xzcat ../patch-5.15.107-rt62.patch.xz | patch -p1
make oldconfig
```

- Choisir l'option `4. Fully Preemptible Kernel (RT) (PREEMPT_RT_FULL) (NEW)` pour l'option `Preemption Model`

```
Preemption Model
  1. No Forced Preemption (Server) (PREEMPT_NONE)
> 2. Voluntary Kernel Preemption (Desktop) (PREEMPT_VOLUNTARY)
  3. Preemptible Kernel (Low-Latency Desktop) (PREEMPT)
  4. Fully Preemptible Kernel (RT) (PREEMPT_RT_FULL) (NEW)
choice[1-4]: 4
```

- Compiler le noyau

```
make -j `getconf _NPROCESSORS_ONLN` deb-pkg
```

- Après la compilation, installer les paquets `linux-headers` et `linux-image` dans le dossier parent (pas les paquets `-dbg`)

```
sudo apt install ../linux-headers-5.15.107-rt62_*.deb ../linux-image-5.15.107-rt62_*.deb
```

Définir les permissions utilisateur pour exécuter des tâches temps-réel

- Le Driver ROS2 va planifier des threads avec les permissions de votre utilisateur. Il faut donc autoriser votre utilisateur à utiliser lancer des threads avec une priorité temps-réel. On créé le groupe des utilisateurs temps-réel et on y ajoute l'utilisateur :

```
sudo groupadd realtime
sudo usermod -aG realtime $(whoami)
```

- S'assurer que `/etc/security/limits.conf` contient :

```
@realtime soft rtprio 99
@realtime soft priority 99
@realtime soft memlock 102400
@realtime hard rtprio 99
@realtime hard priority 99
@realtime hard memlock 102400
```

Note: Pour que ces changements soient pris en compte il faut se déconnecter et se reconncter. On redémarrera plus tard.

https://github.com/HowardWhile/Ubunutu22.04-RT-Kernel

Configurer GRUB pour toujours booter le noyau temps-réel

- Lister les noyaux disponibles

```
awk -F\' '/menuentry|submenu / {print $1 $2}' /boot/grub/grub.cfg
```

> menuentry Ubuntu
> submenu Advanced options for Ubuntu
>
> menuentry Ubuntu, with Linux 5.15.107-rt62
> menuentry Ubuntu, with Linux 5.15.107-rt62 (recovery mode)

- Définir le noyau `5.15.107-rt62` par défaut avec un pattern `"submenu_name>entry_name"`

```
sudo sed -i 's/^GRUB_DEFAULT=.*/GRUB_DEFAULT="Advanced options for Ubuntu>Ubuntu, with Linux
5.15.107-rt62"/' /etc/default/grub
$ sudo update-grub
```

Vérification de la capacité de préemption temps-réel

```
uname -v |  cut -d" " -f1-4
```

```
#1 SMP PREEMPT RT
```

Optionnel : Désactiver le CPU speed scaling

Les threads planifiés en temps-réel s'exécutent sans problème. Cependant, des composants externes tels que les systèmes de visual servoing, non planifiés pour le temps réel peuvent être interrompus par les fonctionnalités d'économie d'énergie des processeurs récents qui changent leur fréquence d'horloge de manière dynamique.

- Pour vérifier et modifier les modes d'économie d'énergie :

```
sudo apt install cpufrequtils
cpufreq-info
```

```
current CPU frequency is XXX MhZ
```

- Désactiver le changement de fréquence automatique :

```
sudo systemctl disable ondemand
sudo systemctl enable cpufrequtils
sudo sh -c 'echo "GOVERNOR=performance" > /etc/default/cpufrequtils'
sudo systemctl daemon-reload && sudo systemctl restart cpufrequtils
```

Configuration du robot UR

https://docs.ros.org/en/ros2_packages/rolling/api/ur_robot_driver/installation/robot_setup.html

Récupération de la calibration usine

Les robots sont calibrés en usine. Pour que les calculs cinématiques effectués dans ROS soient exacts, il faut récupérer les données de calibration. Sinon la précision des trajectoires envoyées depuis ROS et exécutées sur le robot risquent d'être de l'ordre du centimètre (au lieu du dixième de millimètre en temps normal).

https://docs.ros.org/en/ros2_packages/rolling/api/ur_robot_driver/installation/robot_setup.html#extract-calibration-information

---

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

Programmer un robot avec MoveIt2 - Jumeau Numérique

Prérequis : être arrivé au bout du tutoriel sur le Driver UR ROS2

Comment fonctionne la manipulation avec MoveIt ?

MoveIt2 est la plateforme de manipulation robotique pour ROS2. Il implémente un nombre important des dernières innovations en termes de :

- Planification de trajectoire
- Manipulation
- Perception 3D
- Cinématique
- Contrôle
- Navigation

https://moveit.picknik.ai/humble/doc/concepts/concepts.html

Premiers pas avec MoveIt dans RViz

Pour débuter avec MoveIt, on peut utiliser ses fonctionnalités de planification de trajectoire via le **plugin MoveIt Display** du logiciel de visualisation 3D de ROS **RViz**. C'est un outils très puissant pour débuguer des applications robotiques ROS. On verra que RViz est alors un jumeau numérique du vrai robot.

Les tutoriels pour débuter et approfondir ses compétences avec MoveIt sont en Anglais et fonctionnent avec le robot Panda de Franka Emika.

Nous reprenons ici le tutoriel pour débuter avec MoveIt, et l'appliquons à un UR5e.

Avec un hardware simulé par ROS

- Faire tourner le driver UR ROS2

```
ros2 launch ur_robot_driver ur_control.launch.py ur_type:=ur5e robot_ip:=yyy.yyy.yyy.yyy
fake_hardware:=true launch_rviz:=false
```

Avec la simulation URSim

Voir https://innovation.iha.unistra.fr/books/robotique-open-source/page/universal-robot-ros2-driver#bkmrk-installation-du-simu

Sous Ubuntu 22 - docker

- Démarrer la simulation URSim pour 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
launch_rviz:=false
```

Sous Windows - VirtualBox

- Télécharger la VM URSim avec External Control préinstallé
- Configurer le réseau NAT VirtualBox, récupérer les adresses IP avec `ip a` et tester la communication avec `ping 10.0.2.X`, cf. :
  - https://innovation.iha.unistra.fr/books/robotique-open-source/page/universal-robot-ros2-driver#bkmrk-configurer-le-r%C3%A9seau
- Démarrer URSim
- Démarrer le robot virtuel
- Tester la communication entre ros_control et l'URCap external control, cf. https://innovation.iha.unistra.fr/books/robotique-open-source/page/universal-robot-ros2-

driver#bkmrk-https%3A%2F%2Fgithub.com%2Fr

- `ros2 launch ur_robot_driver ur_control.launch.py ur_type:=ur5e robot_ip:=10.0.2.5 initial_joint_controller:=joint_trajectory_controller launch_rviz:=true`

Avec le vrai robot

- Faire tourner le driver UR ROS2

```
ros2 launch ur_robot_driver ur_control.launch.py ur_type:=ur5e robot_ip:=192.168.0.10
launch_rviz:=false
```

Lancer MoveIt et RViz

```
ros2 launch ur_moveit_config ur_moveit.launch.py ur_type:=ur5e launch_rviz:=true
```

https://ur-documentation.readthedocs.io/en/latest/index.html ?

Déplacer le Robot avec le Plugin MoveIt dans RViz

- On veut lancer une requête de planification de trajectoire
- On utilise le plugin MotionPlanning qui permet de configurer la requête via une interface graphique

Résultat

ros2_moveit2_ros2control_commande_externet_UR5e.mp4

Planification de trajectoire avec OMPL

Ajouter un obstacle

- Choisir une configuration cible faisable, sans collision

ros2_moveit2_OMPL_RRTconnect_evitement_collision_automatique

Tester différents algorithmes d'OMPL

- Algorithmes RRT d'exploration rapide stochastique d'arbre
-

ros2_moveit2_OMPL_RRTstar_evitement_collision_longueur_optimisee

Résultat

En optimisant la trajectoire avec RRTstar, on obtient un mouvement fluide, qui évite les collision avec l'environnement et de longueur minimisée. Voir la vidéo réalisée en salle robotique de l'IUT.

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

Roomba ROS2

Connexion physique

http://www.robotappstore.com/Knowledge-Base/3-Serial-Port-Baud-Rate-Configuration/17.html

https://fr.aliexpress.com/item/1005004346786275.html

https://www.yoctopuce.com/FR/article/piloter-un-roomba-avec-un-module-yoctopuce

https://www.adafruit.com/product/2438

https://www.crc.id.au/hacking-the-roomba-600/

ROS2

Roomba ROS2
https://hackaday.io/project/178565-roomba-rpi
https://github.com/process1183/roomba-rpi
https://github.com/process1183/roomba-ros2

https://hackaday.io/project/183524-controll-yer-roomba-600-series-with-esp8266

CRASHbot, Ubuntu 18 for NVidia Jetson nano, ROS Melodic, AutonomyLab create-robot
https://www.hackster.io/andrewrgross/crashbot-part-1-planning-a-ros-roomba-helper-robot-d1c8cc

Roomblock Ubuntu 16 ROS Kinetic
https://www.instructables.com/Roomblock-a-Platform-for-Learning-ROS-Navigation-W/
https://github.com/tork-a/roomblock

ROS navigation Ubuntu 10
https://www.ctralie.com/Teaching/DukeDusty2/

Ressources

https://github.com/AutonomyLab/create_robot/tree/humble

https://github.com/process1183/roomba-ros2

https://github.com/process1183/roomba-rpi

Application

https://www.hackster.io/search?i=projects&q=roomba

https://www.hackster.io/andrewrgross/crashbot-part-1-planning-a-ros-roomba-helper-robot-d1c8cc

https://www.hackster.io/ssbaker/making-roomba-smarter-800-series-40c5e2

https://www.hackster.io/FANUEL_CONRAD/automatic-soap-dispenser-75abd6

Bras Robot Arduino ROS2

Projet d'origine

https://github.com/AntoBrandi/Robotics-and-ROS-2-Learn-by-Doing-Manipulators

Pilotage des servomoteurs : TTL, RS232, RS485

https://esp32io.com/tutorials/esp32-rs485

Modèles Open Source

Open Manipulator-X

- 5 DOF + Pince
- 6x Dynamixel XM430-W350 https://emanual.robotis.com/docs/en/dxl/x/xm430-w350/
  - 4.1 [N.m] (at 12.0 [V], 2.3 [A])
  - 46 [rev/min] (at 12.0 [V])
  - 10.0 ~ 14.8 [V]
- Operating Modes
  - Current Control Mode
  - Velocity Control Mode
  - Position Control Mode (0 ~ 360 [°])
  - Extended Position Control Mode (Multi-turn)
  - Current-based Position Control Mode
  - PWM Control Mode (Voltage Control Mode)
- baud rate 9,600 [bps] ~ 4.5 [Mbps]
- TTL Half Duplex Asynchronous Serial Communication with 8bit, 1stop, No Parity
- RS485 Asynchronous Serial Communication with 8bit, 1stop, No Parity

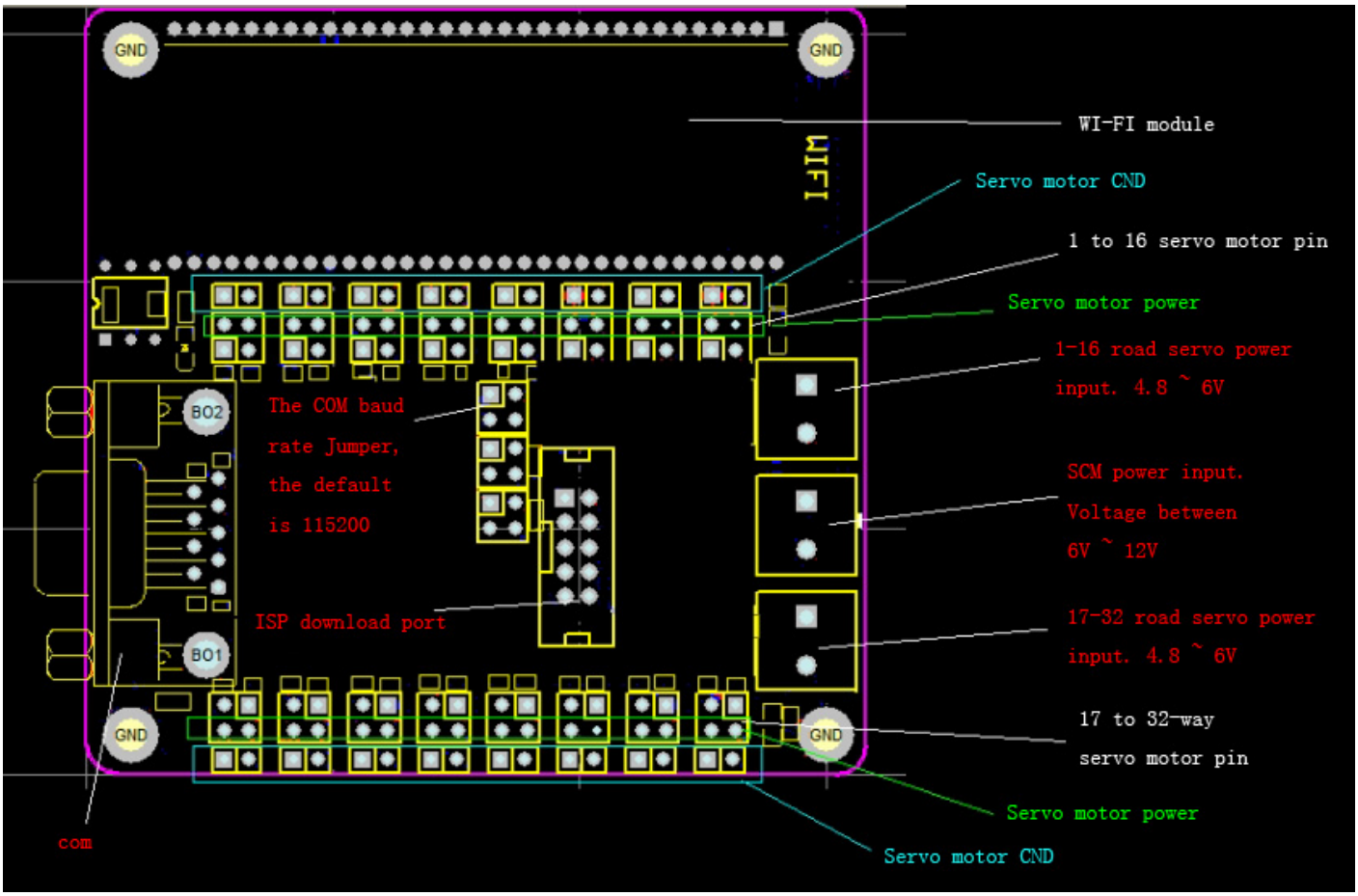
Carte de contrôle OpenCR1.0 :

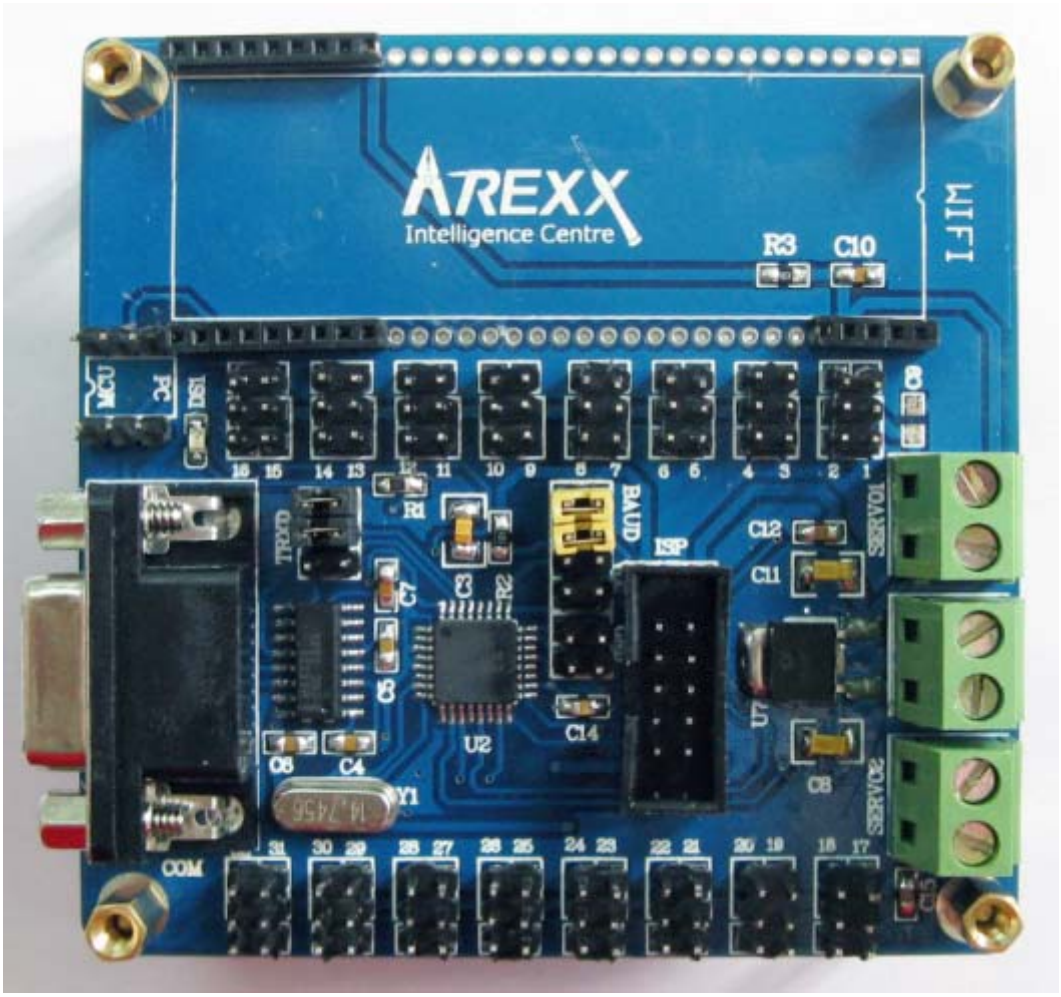
- STM32F746ZGT6 / 32-bit ARM Cortex®-M7 with FPU (216MHz, 462DMIPS)
  Reference Manual, Datasheet
- Programmer : ARM Cortex 10pin JTAG/SWD connector
  USB Device Firmware Upgrade (DFU)
  Serial
- Digital I/O
  - 32 pins (L 14, R 18) *Arduino connectivity
  - 5Pin OLLO x 4
  - GPIO x 18 pins

    - PWM x 6
    - I2C x 1
    - SPI x 1
- Communication Ports
    - USB x 1 (Micro-B USB connector/USB 2.0/Host/Peripheral/OTG)
    - TTL x 3 (B3B-EH-A / DYNAMIXEL)
    - RS485 x 3 (B4B-EH-A / DYNAMIXEL)
    - UART x 2 (20010WS-04)
    - CAN x 1 (20010WS-04)

DAGU Six-servo Robot Arm

- 5DOF + Pince
- 6 servos
    - 3x 13 kg.cm torque metal gear, 40.4 * 19.8 * 36 mm, 48g, 0.22s/60°
    - 1x 3.2 kg.cm, 39.5 x20.0x35.5mm, 41g, 0.27s/60°
    - 2x 2.3 kg.cm, 28 x14x29.8mm, 18g, 0.13/60°
- Carte de contrôle AREXX Intelligence Centre
    - atmega168 MCU
    - RS232
    - default baud rate is 115.2k
    - Wifi wireless control reserve the ISP downloaded, you can download the MCU controller program using the STK500 ISP cable


GND
GND
WI-FI module
WIFI
Servo motor CND
1 to 16 servo motor pin
Servo motor power
1-16 road servo power
input. 4.8 ~ 6V
BO2
The COM baud
rate Jumper,
the default
is 115200
SCM power input.
Voltage between
6V ~ 12V
ISP download port
17-32 road servo power
input. 4.8 ~ 6V
BO1
GND
GND
17 to 32-way
servo motor pin
Servo motor power
com
Servo motor CND


AREXX
Intelligence Centre
WIFI
R3
C10
R1
C12
C11
C7
C3
C2
C14
C6
C4
C8
U2
Y1
BAUD
ISP
TRIG
COM
SERVOS
SERVOS

- dual - Power Supply
  - 6 ~ 12 V SCM power
  - 4.8 ~ 6 V, 1.2A servo motor power [servo motor power supply Road 1-16 respectively, a 17-32 road supply port])

https://seafile.unistra.fr/d/693101e6046d4819a3af/

https://arexx.com/product/robot-arm/

www.arexx.com.cn

SO-ARM100

- 5DOF + Pince
- Waveshare Serial Bus Servo Driver Board
  - Supports connecting to a host or MCU
  - up to 253 ST/SC series serial bus servos
  - RS485
  - UART pour contrôle depuis Arduino, ESP32, STM32 (RX-RX, TX-TX)
  - USB pour contrôle via Raspberry, Jetson ou PC
  - 9~12.6V voltage input (the input voltage and the servo voltage must be matched)

https://www.waveshare.com/wiki/Bus_Servo_Adapter_(A)

https://github.com/huggingface/lerobot/blob/main/examples/10_use_so100.md

https://medium.com/@sarohapranav/my-experiences-and-tips-for-creating-a-robotic-so100-arm-3df779a4aae7

Cartes de contrôle

https://emanual.robotis.com/docs/en/parts/controller/opencr10/

Servomoteurs