2 - ROS2 - Robotique mobile

- Turtlesim et modèle de Dubins
- TurtleBot3 - Bases en Simulation
- TurtleBot3 - Piloter le Robot
- Calibration de la caméra
- Suivi de ligne ROS2 Humble
- Behavior Trees Demo
- Installation et démarrage du Turtlebot 3

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 :

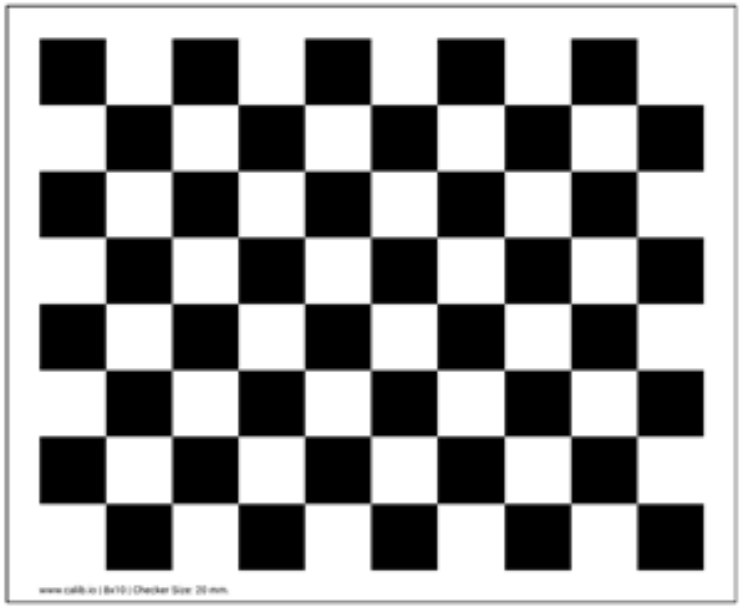
`ssh ubuntu@192.168.3.10` (burger) ou `ssh ubuntu@192.168.3.11` (waffle) avec le même mdp qu'en TP.

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