ROS2 : bibliothèques et outils pour le développement logiciel en robotique

Culture Sciences de l'Ingénieur

La Revue 3E.I

Édité le 02/02/2026

école normale supérieure paris—saclay

Jules FARNAULT[1] – Sergio RODRIGUEZ[2] – Anthony JUTON[3]

[1] *Elève normalien à l'ENS Paris Saclay, DER Sciences de l'Ingénierie Électrique et Numérique*

[2] *Maître de conférences au laboratoire SATIE, ENS Paris Saclay*

[3] *Professeur agrégé à l'ENS Paris Saclay, DER Sciences de l'Ingénierie Électrique et Numérique*

Cette ressource fait partie du N°118 de La Revue 3EI du 1er trimestre 2026.

Cette ressource a pour but de présenter ROS (du sigle en anglais Robotics Operating System version 2), un ensemble de bibliothèques C/C++ et python et d'outils de développement open-source pour la robotique, drones compris. Les laboratoires de robotique, les fabricants de matériel et les industriels de la robotique partagent dans une communauté dynamique leurs développements ROS, ce qui permet de réutiliser des briques logicielles de qualité [modules d'acquisition de capteurs complexes (LiDAR-Light Detection And Ranging, caméra RGBD, GPS/GNSS...), de contrôle d'actionneurs, algorithmes complexes (filtres à particules, génération de trajectoire…)…] et faciles à interfacer.

Cette ressource présente ROS et ses différents composants et guide le lecteur à travers la documentation de la version ROS2 pour sa mise en œuvre sur deux premiers exemples. Elle est suivie de la ressource « *Mise en œuvre de ROS2 pour le contrôle d'une voiture autonome simulée sous Webots et réelle* » [15] présentant la mise en œuvre de ROS2 réelle et simulée.

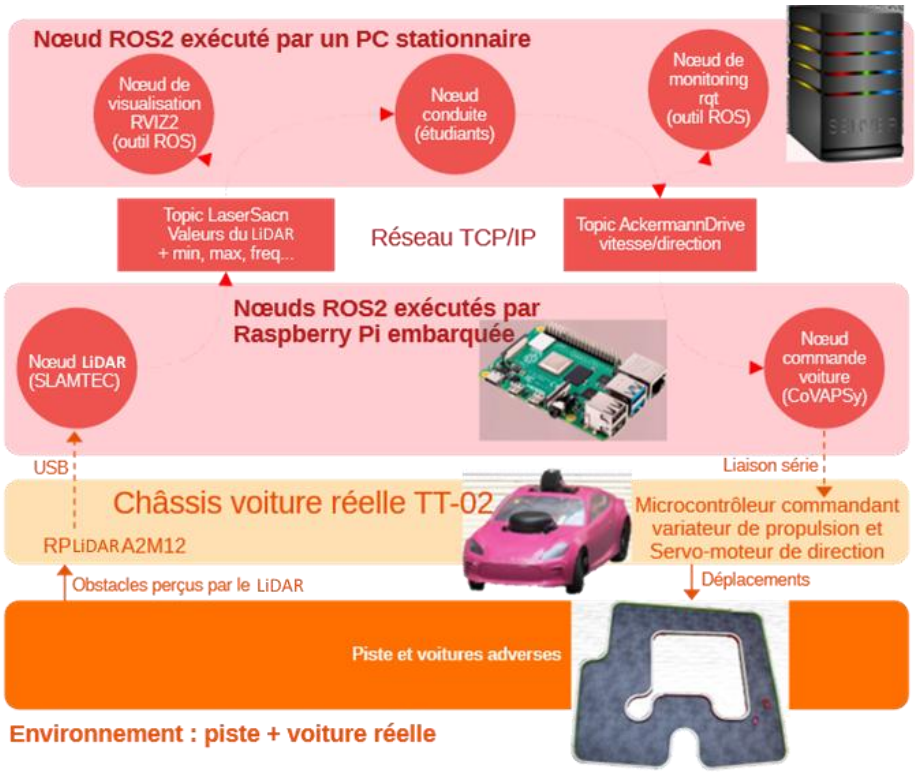

Figure 1 : Graphe ROS2 utilisé pour la conduite d'une voiture autonome 1/10ème

1 - ROS, un écosystème de bibliothèques et outils pour la robotique

ROS (Robot Operating System), né en 2007, [1] constitue une surcouche logicielle d'Ubuntu destiné au développement d'applications pour la robotique. Il offre des services standardisés essentiels tels que la gestion des capteurs, la gestion des actionneurs, la gestion de la navigation, ainsi que l'enregistrement et le rejeu de données. En outre, ROS met à disposition des outils de visualisation, de simulation, d'analyse et de débogage facilitant le développement et la maintenance des systèmes robotiques.

ROS est utilisé par de nombreux laboratoires de recherche et industriels pour le développement de logiciels de robots. Dans une optique open-source, ils développent des paquets (*ROS package*) que tout utilisateur peut ajouter à son installation afin de bénéficier de ses fonctionnalités. Des projets open-source pour les drones (PX4 autopilot), les bras robotisés (ROS-industrial), la navigation des robots (Nav2) sont basés sur ROS. De plus, les fabricants de capteurs (Intel Realsense pour les caméras RGBD, Slamtec ou Velodyne pour les LiDARs, ainsi que Analog Device (centrales inertielles) fournissent des paquets ROS permettant d'acquérir les données de leurs équipements. Par ailleurs, les fabricants de robots (Boston Dynamics, Unitree Husarion…) proposent également des paquets ROS permettant de s'interfacer avec leurs robots.

Un répertoire des paquets documentés disponibles est fourni par ROS : https://index.ros.org

La communauté des développeurs ROS [2] communique principalement à travers un forum [3], un canal Discord [4] et se rassemble chaque année lors de conférences ROScon nationales [6] et une conférence internationale [5]. L'accès public à ces discussions, ainsi qu'aux et les documentations [7], tutoriels et wiki [8] facilitent une prise en main rapide de ROS et des paquets complémentaires.

ROS2, dont la première distribution date de 2017, est une évolution majeure rendant les paquets ROS2 incompatibles avec les paquets ROS1. Aujourd'hui, la plupart des projets et paquets ont été migrés sur ROS2 et le nom générique ROS désigne souvent des travaux sur ROS2. Les évolutions mineures sont portées par des distributions (un ensemble de paquets compatibles entre eux). Pour cette ressource, la distribution LTS (Long Term Support) *Jazzy*, compatible avec *Ubuntu 24.04 LTS*, a été choisie. Elle est moderne et sera supportée jusqu'à mai 2029. Les paquets les plus populaires sont disponibles pour Jazzy.

ROS2 et la plupart des paquets complémentaires sont développés en C++ pour des critères de performances. La suite montrera que la modularité de ROS2 permet d'écrire des programmes en python s'interfaçant avec ces bibliothèques en C++.

2 - Nodes, Topics, Services... les différents composants de ROS

Un système robotique fonctionnant sous ROS possède une architecture construite en nodes qui communiquent via des topics ou des services. Cette modularité, en nodes, permet de faire cohabiter des nodes ROS officiels (par exemple, node fourni par le fabricant d'un capteur ou développé par un laboratoire…) et des nodes développés spécifiquement pour répondre à des besoins particuliers.

La messagerie (topics, services) s'appuie sur un middleware nommé DDS (de l'anglais *Data Distribution Service*) utilisant des mécanismes d'échange par mémoire partagée ou les protocoles IP [9]. Les nodes peuvent donc indifféremment être sur la même machine ou sur des machines différentes, ce qui simplifie le calcul déporté pour des systèmes embarqués aux ressources limitées.

Évidemment, si les nodes sont sur des machines différentes, la latence entre les envois et réceptions de message sera plus importante.

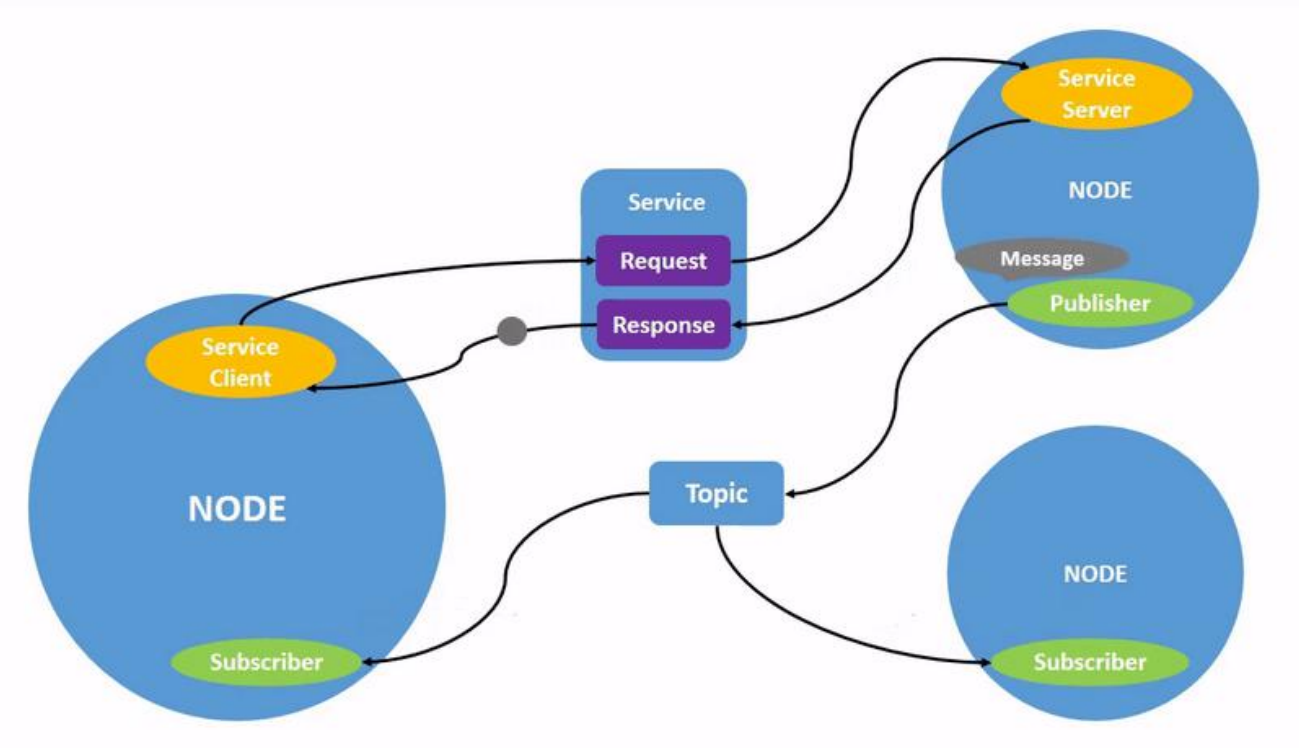

Figure 2 : Architecture d'un système robotique sous ROS, source : ROS.org

2.1 - Nodes (Nœuds)

Un node est un processus indépendant. Un nœud programmé en C++ peut donc cohabiter avec un nœud programmé en python. Les nodes s'exécutent en parallèle au sein du système d'exploitation, chaque node pouvant ainsi utiliser une CPU différente sur un microprocesseur multicœurs.

Pour communiquer entre eux, les nodes utilisent des topics ou des services.

2.2 - Topics (canaux)

Les topics sont une communication en mode publisher/subscriber. Ce sont des canaux de communication qui permettent à différents nodes d'échanger des messages. Un ou plusieurs nodes peuvent publier des messages dans un topic, tandis qu'un ou plusieurs nodes peuvent s'abonner à ce topic pour recevoir ces messages. La messagerie inter-processus TCP utilisée dans ROS1 et a été remplacée dans ROS2 par une communication nommée DDS. Cette solution combine l'utilisation conjointe des protocoles TCP, UDP et mémoire partagée selon les contraintes de l'application.

2.3 - Services et actions

Les services sont une communication en mode client serveur. Ils sont utilisés plutôt pour la modification de configuration d'un node, les topics étant plus adaptés pour les messages de process (valeurs des capteurs et commandes des actionneurs).

Les actions sont similaires aux services mais avec une différence clé : elles renvoient un feedback continu. C'est intéressant pour un service qui nécessite du temps pour s'exécuter. Par exemple si on demande à un robot d'atteindre une position absolue, il est intéressant de pouvoir suivre l'évolution de son déplacement.

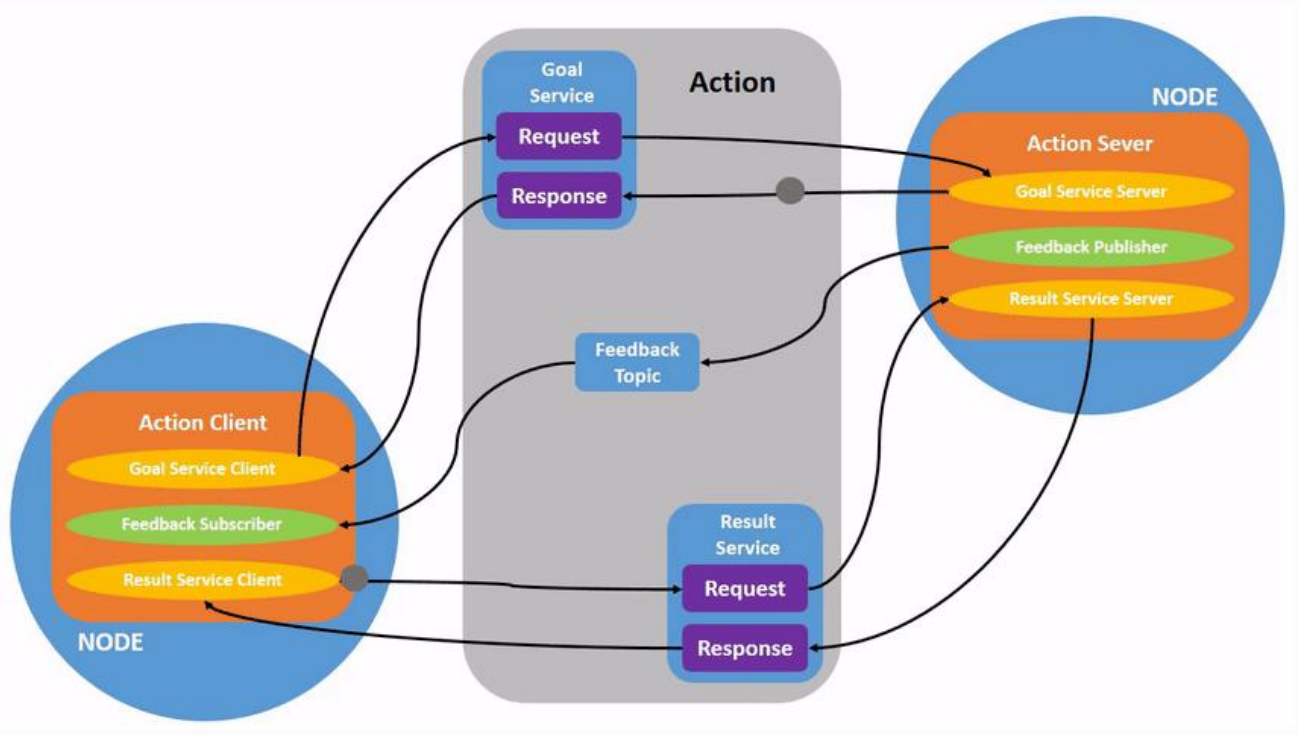

Figure 3 : Architecture d'un système robotique sous ROS utilisant les actions, source : ROS.org

2.4 - ROS bag, enregistrement de jeu de données

ROS permet d'enregistrer un jeu de données constitué des messages échangés sur les topics, services et actions dans un format standardisé, appelé ROS bag. Cette fonctionnalité facilite la collecte et rejoue des données. Par exemple, il est possible d'enregistrer des acquisitions capteurs pour les traiter hors ligne (segmentation d'image, reconstruction cartographique…) ou d'enregistrer des jeux de données (capteurs, actions) afin de réaliser de l'apprentissage supervisé.

3 - Premiers pas avec ROS2 sous Ubuntu 24.04

Un des points forts de ROS est la qualité de sa documentation. Celle-ci est conçue pour vous guider efficacement à travers l'ensemble de ressources disponibles, en insistant sur l'importance de suivre toutes les étapes pour acquérir une connaissance approfondie du potentiel de ROS et une maîtrise minimale de ses fonctionnalités.

La solution la plus simple, celle retenue ici, est d'installer ROS2 Jazzy sur une distribution Ubuntu 24.04. Pour débuter, il est préférable d'utiliser Ubuntu 24.04 Desktop, afin de disposer d'un environnement graphique pour les outils graphiques de ROS. Cet environnement peut être installé sur un PC, une machine virtuelle ou un nano-ordinateur Raspberry Pi. L'environnement graphique par défaut de Ubuntu 24, Gnome, n'est pas très fluide pour la Raspberry Pi4.

Pour le travail sur un système embarqué, il peut être intéressant d'utiliser Ubuntu 24 Server (sans interface graphique, accessible par ssh) sur le système embarqué (par exemple, un nano-ordinateur Raspberry Pi). En complément, Ubuntu 24 Desktop peut être installé sur un PC situé sur le même réseau, pour bénéficier des outils de visualisation/diagnostic.

3.1 - Installation d'une machine Ubuntu 24.04 - ROS2

L'installation de la machine virtuelle Ubuntu 24.04 est indiquée en annexe 1 et 2 [16].

L'installation Desktop ou Sever sur un nano-ordinateur Raspberry Pi se fait via Raspberry Imager :

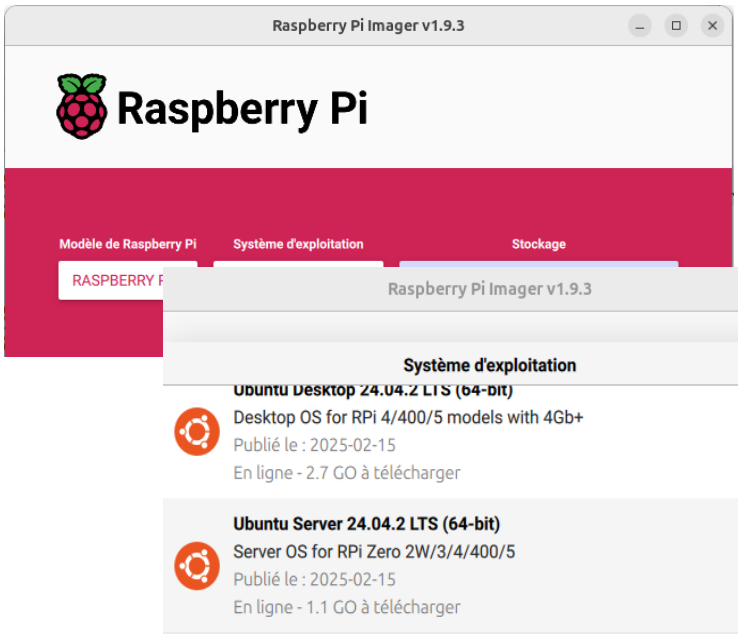

Figure 4 : Raspberry Imager : logiciel de création d'image disque pour Raspberry Pi

3.2 - Installation de ROS2 sur la machine Ubuntu 24.04

Les différentes étapes pour installer ROS2 sur Ubuntu 24.04 sont décrites sur la page web officielle : https://docs.ros.org/en/Jazzy/Installation/Ubuntu-Install-Debs.html [1]

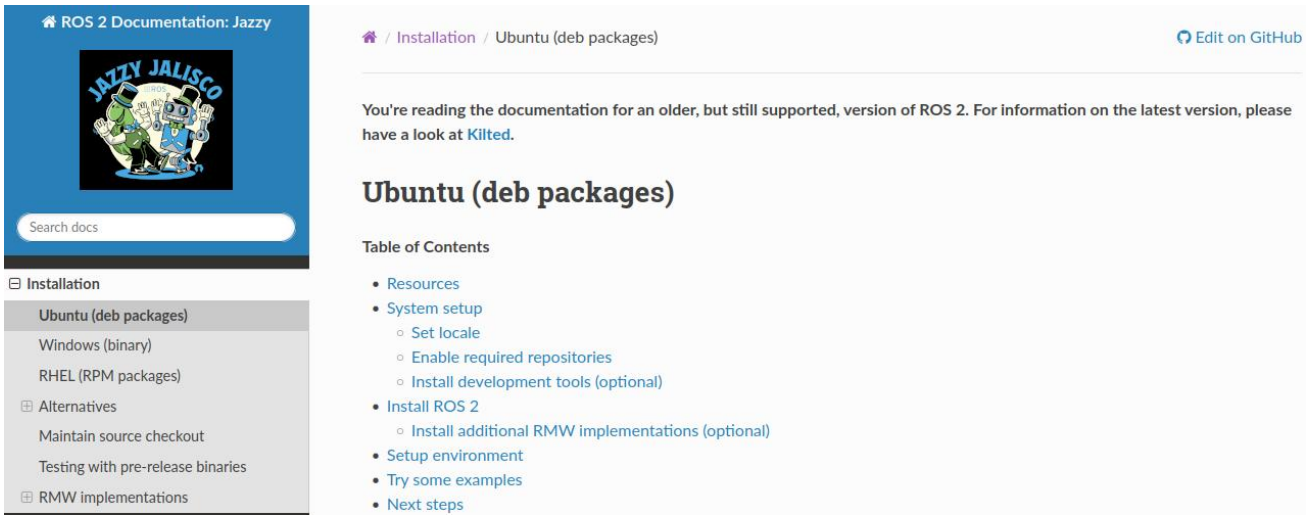

Figure 5 : Page d'accueil de l'installation de ROS2 Jazzy

Suivre scrupuleusement ce qui est demandé permet d'être efficace. Pour ces premiers pas, sur le PC comme sur la raspberry Pi, il est conseillé d'installer la version Desktop, ce qui permet d'avoir quelques outils intéressants. Pour le bon fonctionnement d'un robot par la suite, il suffit d'installer sur le système embarqué la version ROS-Base et de garder sur un PC la version Desktop pour les outils de visualisation/diagnostic.

[1] Consulté le 19/01/2026

Desktop Install (Recommended): ROS, RViz, demos, tutorials.

```
$ sudo apt install ros-jazzy-desktop
```

ROS-Base Install (Bare Bones): Communication libraries, message packages, command line tools. No GUI tools.

```
$ sudo apt install ros-jazzy-ros-base
```

Figure 6 : Lignes de commande pour installer la version desktop ou la version légère (ROS-base) de ROS2 Jazzy

Une fois ROS installé, il est proposé de tester sur un exemple :

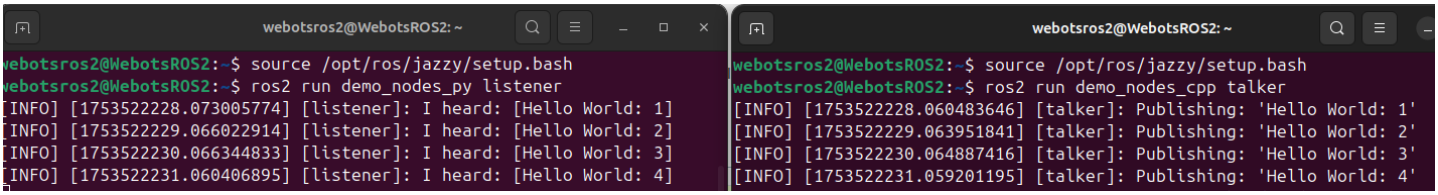

Figure 7 : Exemple de système ROS2 minimaliste lancé sur deux consoles : une pour le récepteur du message (listener) et une pour l'émetteur du message (talker)

Pour permettre l'utilisation de plusieurs versions de ROS2 ou de différents jeux de paquets, il est possible de créer plusieurs workspaces. Cependant, pour commencer, on se limite à un seul workspace. Avant de lancer l'exemple sur chaque terminal, on configure l'espace de travail en exécutant le fichier *setup.bash*.

Pour éviter de lancer cette ligne à chaque ouverture de terminal sur notre système dédié à ROS2, il est conseillé de l'ajouter au fichier *.bashrc*. Ce fichier est automatiquement exécuté à chaque ouverture de terminal, ce qui permet de générer une configuration automatique pour l'environnement ROS2 :

```
echo "source /opt/ros/Jazzy/setup.bash" >> ~/.bashrc
```

Pour que les nodes de différents équipements ROS2 puissent dialoguer, ils doivent être dans le même domaine. Il est donc nécessaire d'ajouter une ligne de configuration pour utiliser domaine spécifique, par exemple 94 :

```
echo "export ROS_DOMAIN_ID=94" >> ~/.bashrc
```

Les terminaux doivent alors tous être fermés puis réouverts pour prendre en compte la modification de configuration.

3.3 - Découverte des fonctionnalités de ROS2 avec le tutoriel *Beginner : CLI tools*

Une fois ROS2 installé, la documentation propose plusieurs tutoriels pour débuter. Il est important de suivre scrupuleusement les deux premiers :

- *Beginner: CLI* (*Command Line Interface*) *tools* permet de découvrir les fonctionnalités de ROS2 ;
- *Beginner: Client libraries* enseigne comment créer des nodes et les faire communiquer.

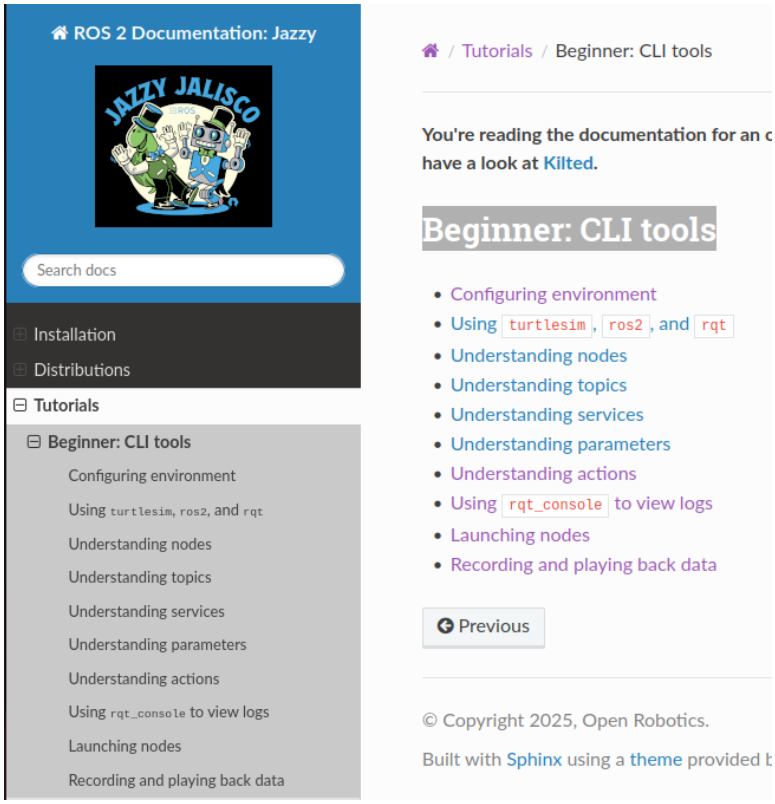

Figure 8 : Page d'accueil du tutoriel Beginner: CLI tools de ROS2 Jazzy

L'exemple *turtlesim* proposé permet de tester différents outils de ROS2, notamment pour lister les nodes, topics, actions et services. Il offre aussi la possibilité d'installer et d'expérimenter l'outil *rqt*, qui permet d'afficher le diagramme des nodes et topics actifs, ainsi que d'interagir avec les nodes en utilisant les services ou messages.

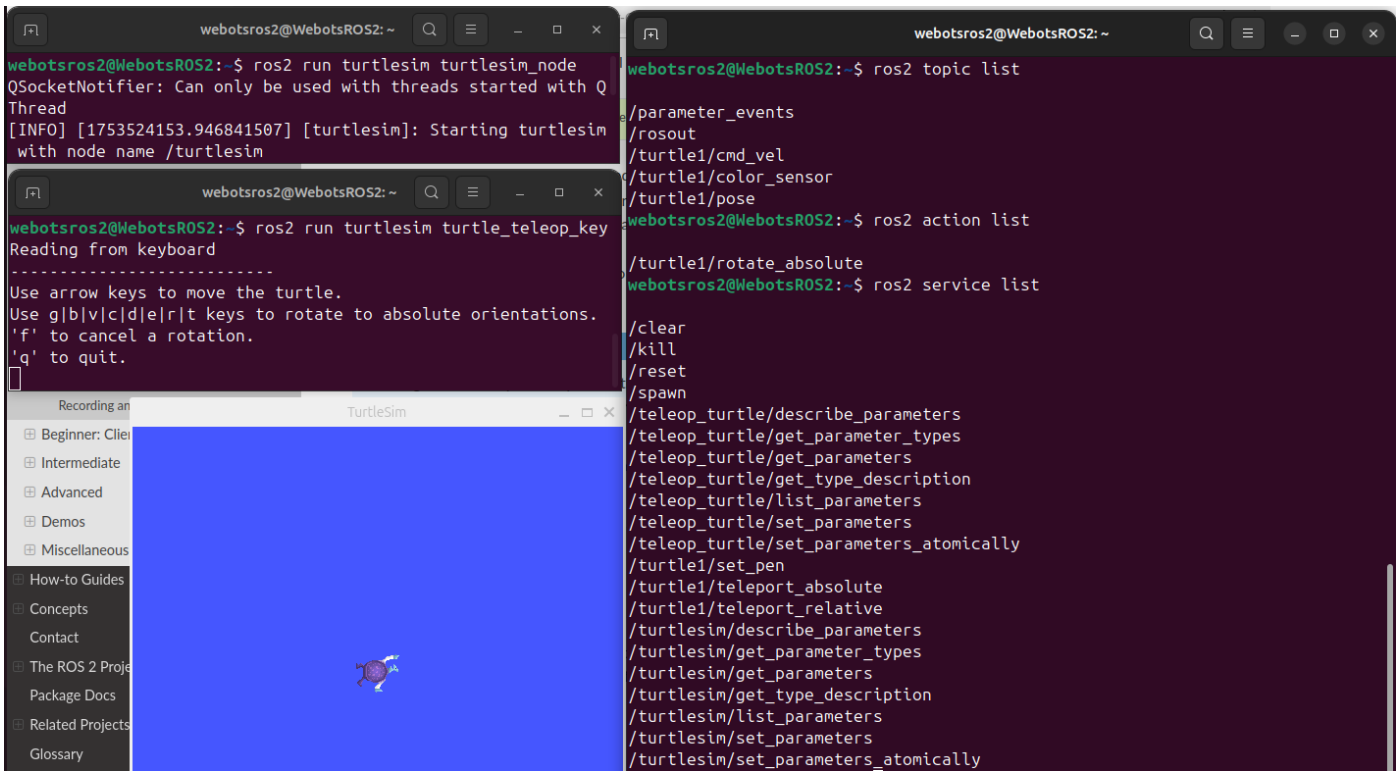

Figure 9 : Exemple Turtlesim lancé sur trois consoles : un node pour la simulation et l'affichage de la tortue (turtlesim_node), un node pour la commande (turtle_teleop_key) et une dernière console pour l'affichage des services ROS2 disponibles

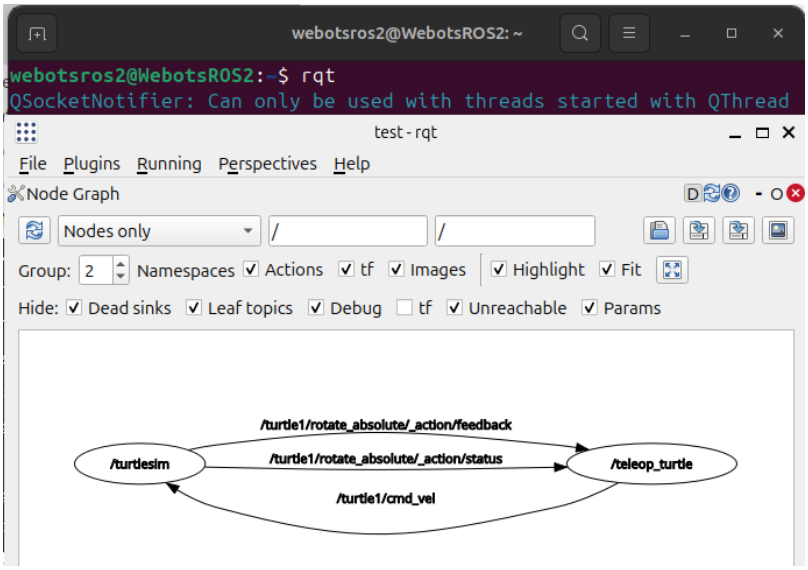

Figure 10 : Outil ROS2 rqt de visualisation des nodes actifs

La suite du tutoriel guide à l'utilisation d'outils de diagnostic complémentaires, la mise en œuvre des services et actions, la modification des paramètres d'un node, toujours sur l'exemple *turtlesim*, ainsi que la création un fichier de lancement de nodes.

Les fonctions de sauvegarde et de chargement des paramètres d'un node via des fichiers *YAML* seront bien utiles pour tout utilisateur de ROS.

Enfin, le tutoriel aborde l'enregistrement et la relecture d'un jeu de données avec *ROS bag*.

3.4 - Programmation de premiers nodes ROS2 avec le tutoriel *Beginner : CLI libraries*

Ce second tutoriel ROS2 amène à écrire 2 nodes (en C++ ou en python), un publiant un message dans un topic et l'autre s'y abonnant pour l'afficher. Le tutoriel guide ensuite à « construire » ces nodes avec l'outil colcon de ROS pour les lancer.

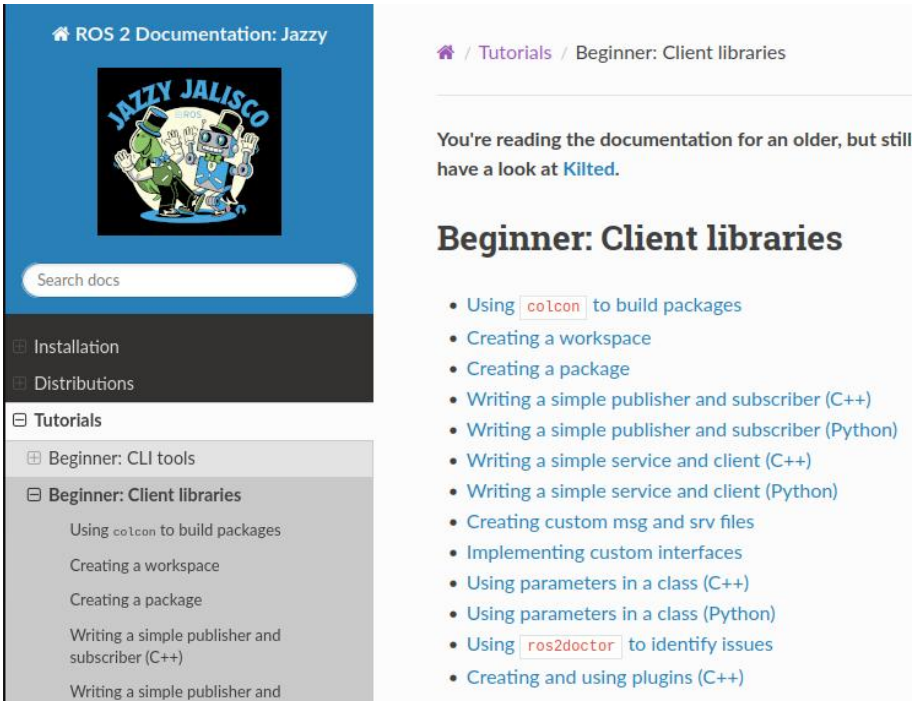

Figure 11 : Page d'accueil du tutoriel Beginner: Client libraries de ROS2 Jazzy

On peut se limiter aux quatre premières étapes :

- La découverte de l'outil colcon, nécessaire pour construire les nodes ;
- La découverte des environnements de travail, surcouche (overlay) au-dessus des paquets de l'installation de base de ROS2 (underlay) ;
- La structure et la création d'un paquet avec un node simple, en C++ ou en python ;
- L'écriture (en C++ ou en python) d'un node *talker* publiant sur un topic et d'un node *listener* souscrivant à ce topic pour l'afficher.

Les plus intéressés feront la suite du tutoriel avec la création de nodes communiquant via des services et l'écriture de type de messages et services personnalisés, de nodes avec des paramètres.

L'avant-dernier item *Using ros2doctor* est rapide et utile pour la suite.

On trouve sur le web des mémos regroupant les principales commandes ROS. On donne ici un exemple pertinent, sur deux pages [13].

4 - Quelques outils ROS complémentaires

ROS propose aux développeurs quelques outils bien utiles pour déboguer leur système.

4.1 - Rqt / rqt_graph

Rqt et *Rqt_graph* [10] sont des outils ROS permettant de visualiser en temps réel les nodes, topics, services et actions en cours d'exécution. *Rqt_graph* offre une représentation de la topologie logicielle sous forme de graphe. Il permet de visualiser les relations entre les nodes et les topics, ainsi que les messages qui sont échangés entre eux.

Lancée dans un terminal, la commande `rqt_graph` fournit un schéma de communication entre les nodes en cours d'exécution.

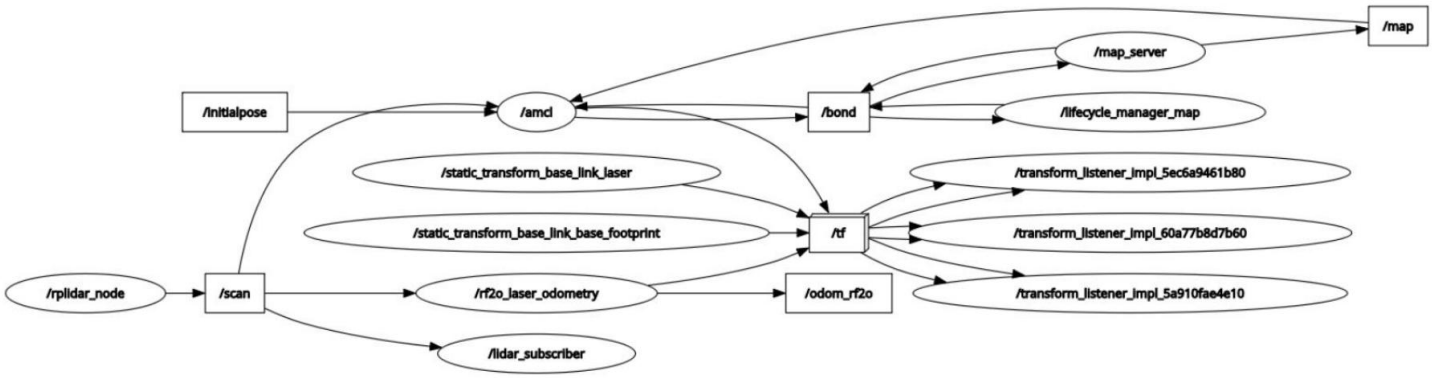

Figure 12 : Diagramme rqt_graph d'une voiture effectuant du SLAM (Simultaneous Localization and Mapping)

Le package *rqt* contient d'autres affichages graphiques pour visualiser les données de ROS, notamment *rqt_plot* qui permet de visualiser les données des topics.

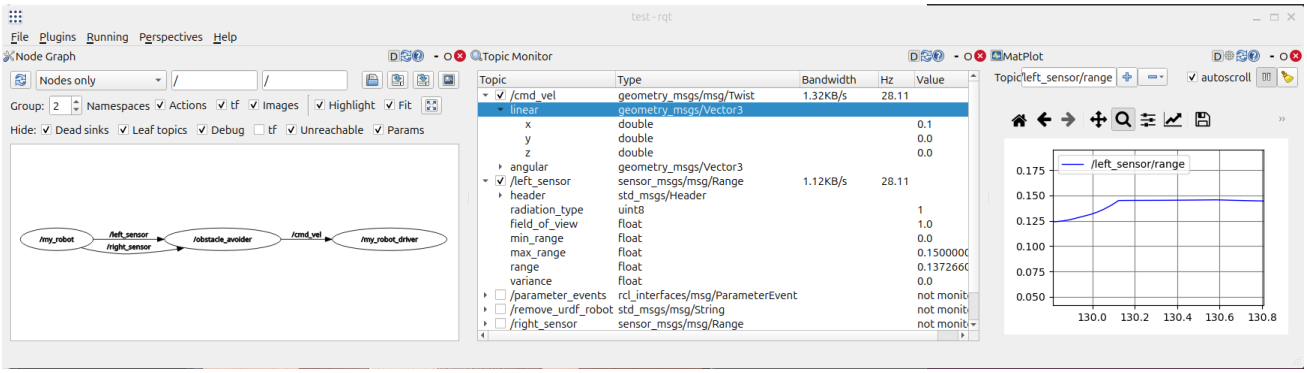

Figure 13 : Outil rqt utilisé sur un l'exemple ROS2 pour webots pour l'affichage du diagramme des nodes, des valeurs de 2 topics et de l'évolution du champ d'un des topics

4.2 - Ros_bag

Ros_bag [11] est un outil de ROS permettant d'enregistrer les données publiées par les nodes dans des topics. Ces données peuvent être des images, des données de capteurs, des données de navigation, etc. Elles peuvent ensuite être relues pour analyse ou rejouées afin de tester différents algorithmes qui les exploitent.

Pour enregistrer des données dans un bag, on utilise la commande suivante sur des topics :

```
ros2 bag record <nom_du_topic> <nom_du_topic>
```

Pour utiliser ses données, on utilise la commande suivante :

```
ros2 bag play <nom_du_bag>
```

On peut ajouter -loop pour lire en boucle :

```
ros2 bag play <nom_du_bag> --loop
```

4.3 - Rviz2

Rviz2 [12] est un outil de ROS2 qui permet de visualiser des données en 3D. Il peut afficher graphiquement des données de capteurs, LiDAR ou caméras. Il se lance comme un node qui s'abonne à des topics pour visualiser en temps réel les données transmises :

```
ros2 run rviz2 rviz2
```

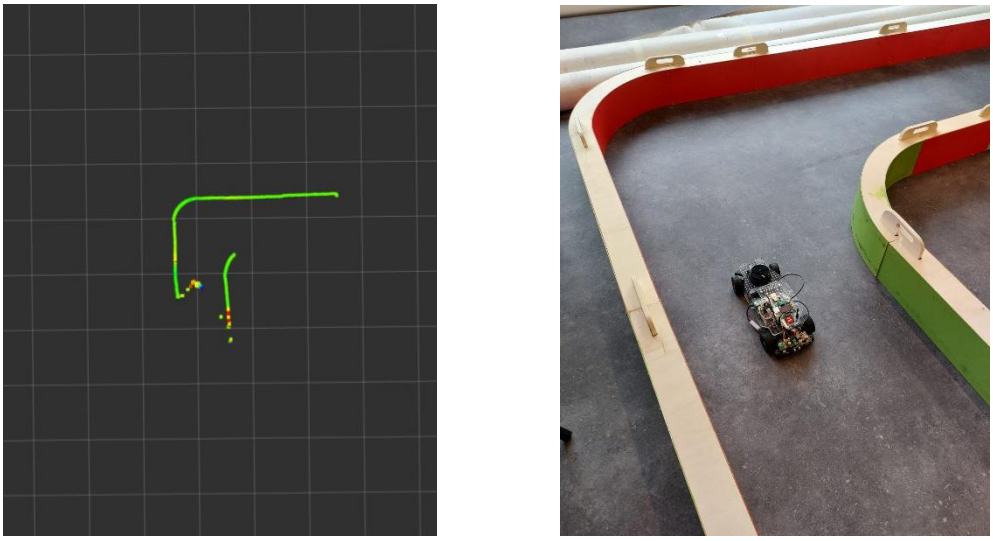

Figure 14 : (a) Visualisation des données du LiDAR dans Rviz, (b) Voiture et son lidar au moment de l'acquisition

4.4 - Les transformées et repères

La position de capteurs dans le repère du robot ou dans le repère terrestre sont des éléments essentiels pour exploiter leurs données. ROS propose un système puissant de transformée des repères de coordonnées nommé tf, qui permet de gérer la transformation des repères de coordonnées en temps réel. La documentation [14] propose son propre tutoriel pour bien comprendre cet outil puissant.

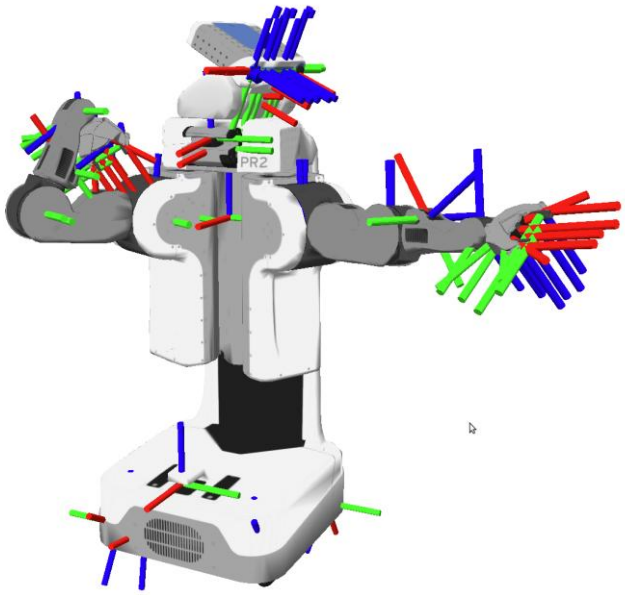

Figure 15 : Illustration ROS associée au paquet tf2

5 - Conclusion

Cette ressource a présenté la structure d'un système reposant sur ROS2. L'association des nodes fournis par des laboratoires ou des industriels avec ses propres nodes, ainsi que l'utilisation des transformées ROS, permet d'exploiter pleinement les capteurs et les algorithmes existants. Par ailleurs, les outils rqt_graph, rosbag et rviz2 seront des alliés essentiels pour la mise au point des programmes.

Références :

[1]: Open Source Robotics Foundation, Inc., “Robot Operating System (ROS).” , http://www.ros.org/

[2]: https://docs.ros.org/en/Jazzy/index.html#ros-community-resources

[3]: forum ROS, https://discourse.ros.org/

[4]: canal Discord, https://discord.com/servers/open-robotics-1077825543698927656

[5]: Site web de la ROScon, https://roscon.ros.org

[6]: Site web de la ROScon France, https://roscon.fr/

[7]: Documentation officielle ROS, https://docs.ros.org/

[8]: Wiki officiel ROS, https://wiki.ros.org/

[9]: Le middleware ROS2, https://design.ros2.org/articles/ros_on_dds.html

[10]: page de rqt et rqt_graph, https://wiki.ros.org/rqt et https://wiki.ros.org/rqt_graph

[11]: ROS Wiki Contributors, “rosbag - ROS Wiki.”, https://wiki.ros.org/rosbag/Commandline

[12]: Tutoriel et page ROS Wiki sur RVIZ2, https://docs.ros.org/en/Jazzy/Tutorials/Intermediate/RViz/RViz-Main.html et http://wiki.ros.org/rviz

[13]: https://github.com/ubuntu-robotics/ros2_cheats_sheet/blob/master/cli/cli_cheats_sheet.pdf

[14]: Tutoriel et page ROS Wiki sur les transformée tf, https://docs.ros.org/en/Jazzy/Tutorials/Intermediate/Tf2/Tf2-Main.html et https://wiki.ros.org/tf2

[15]: Mise en œuvre de ROS2 pour le contrôle d’une voiture autonome simulée sous Webots et réelle, J. Farnault, S. Rodriguez A. Juton, M. Goupillon, 2026, https://sti.eduscol.education.fr/si-ens-paris-saclay/ressources_pedagogiques/mise-en-oeuvre-ros2-pour-controle-voiture-autonome-1-10e

[16]: Annexes de ROS2 : bibliothèques et outils pour le développement logiciel en robotique, J. Farnault, S. Rodriguez A. Juton, 2026, https://sti.eduscol.education.fr/si-ens-paris-saclay/ressources_pedagogiques/ros2-bibliotheques-outils-pour-developpement-logiciel-en-robotique

- Annexe 1 : Installation de la machine virtuelle sous Linux
- Annexe 2 : Installation de la machine virtuelle sous Windows