ROS 2

Introduction à ROS-2

Olivier STASSE,
DR-1, CNRS,
Gepetto,
LAAS CNRS

ANF CNRS ROS 2,
14 - 17 Novembre 2023

ROS industrial

ROS

Open Source Robotics Foundation

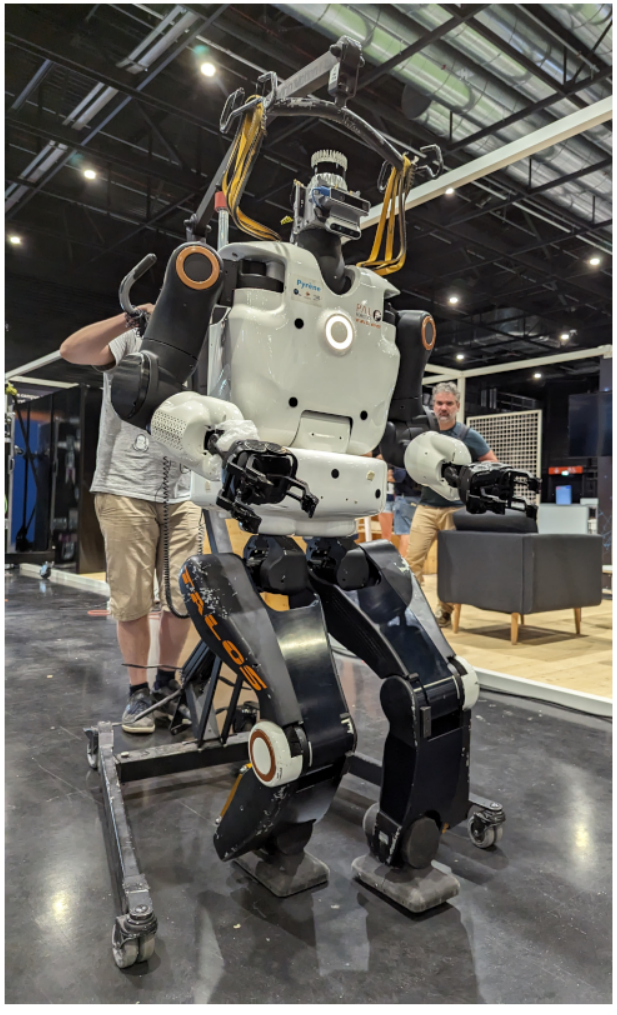
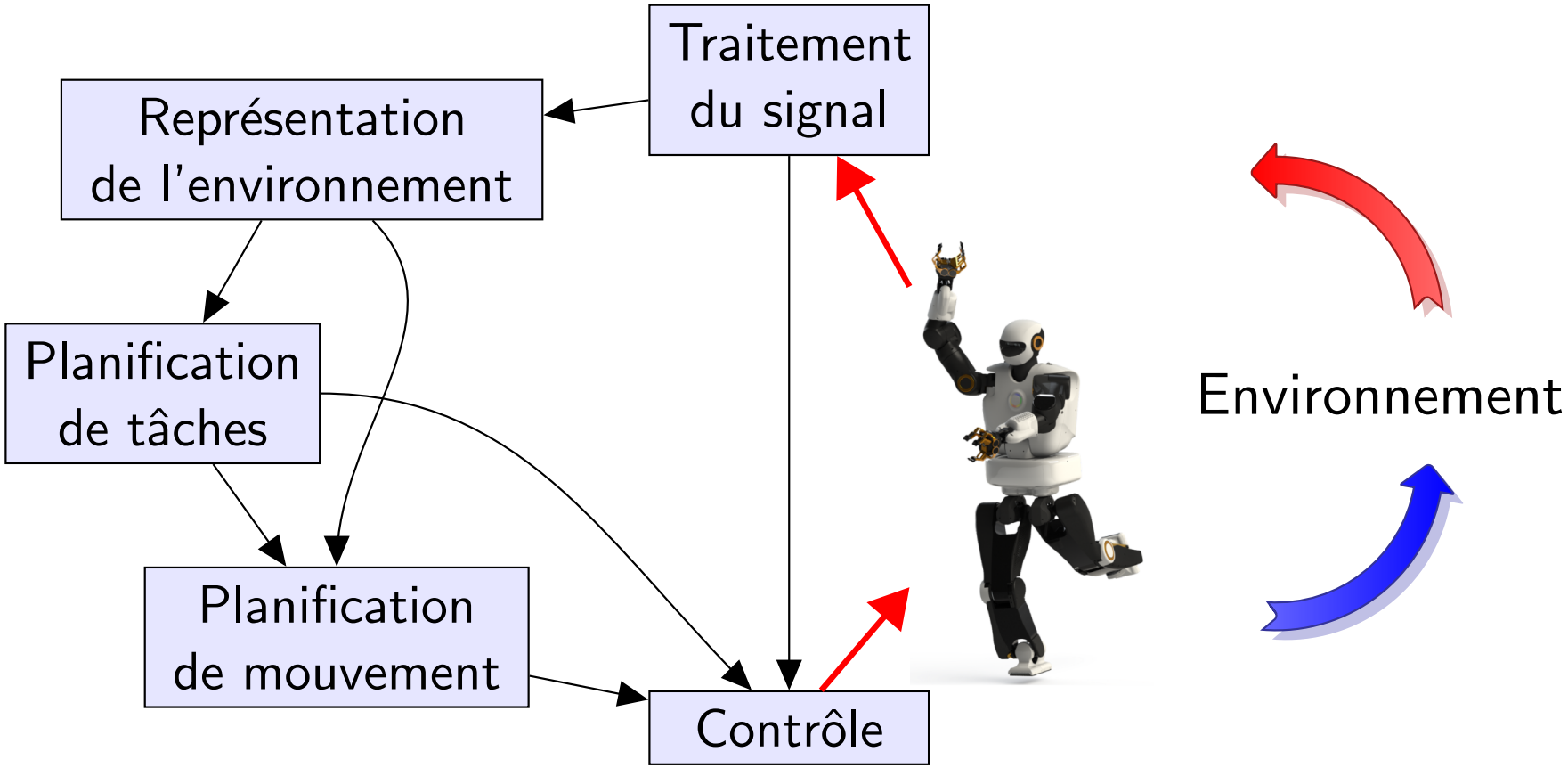


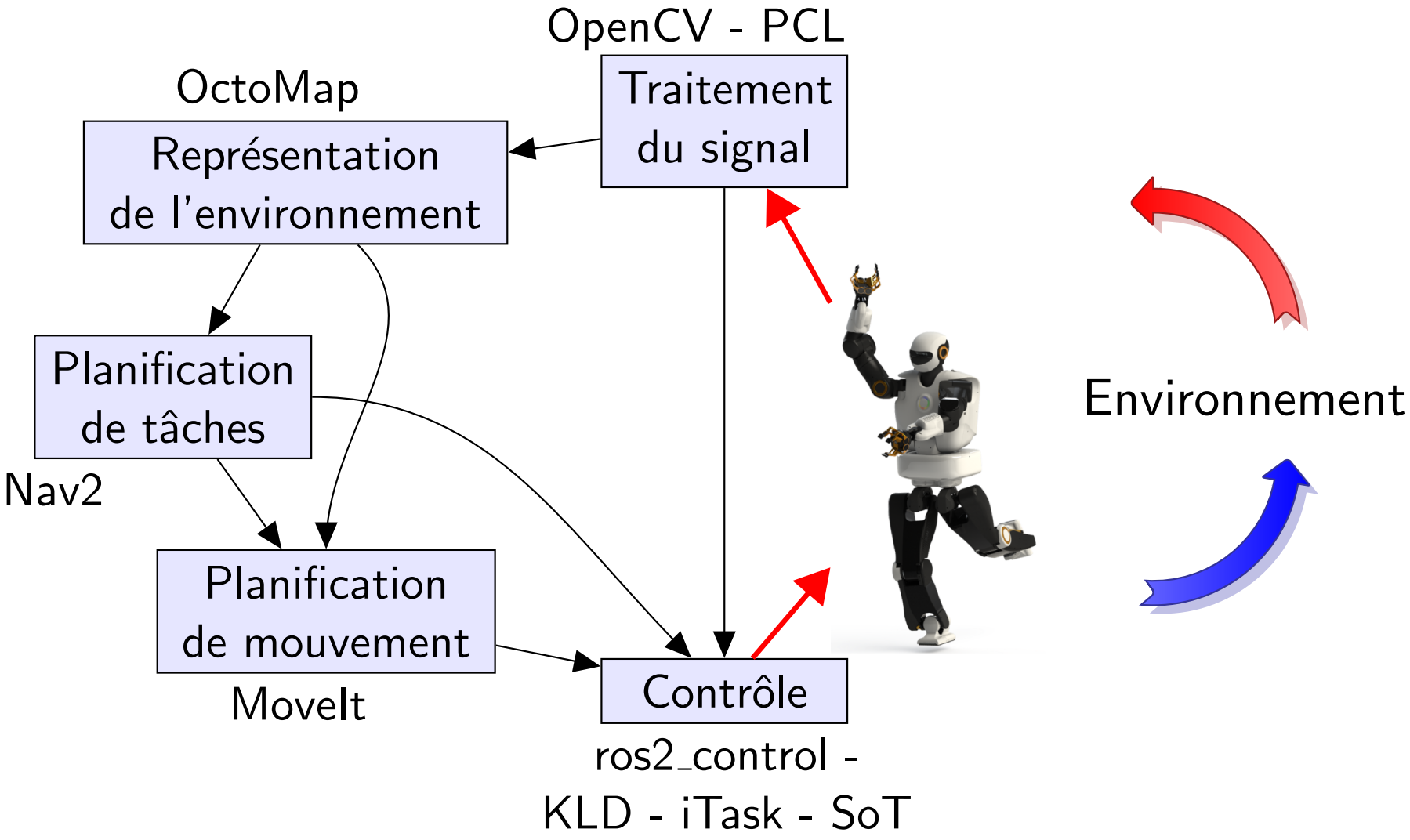
Table des matières

Table des matières

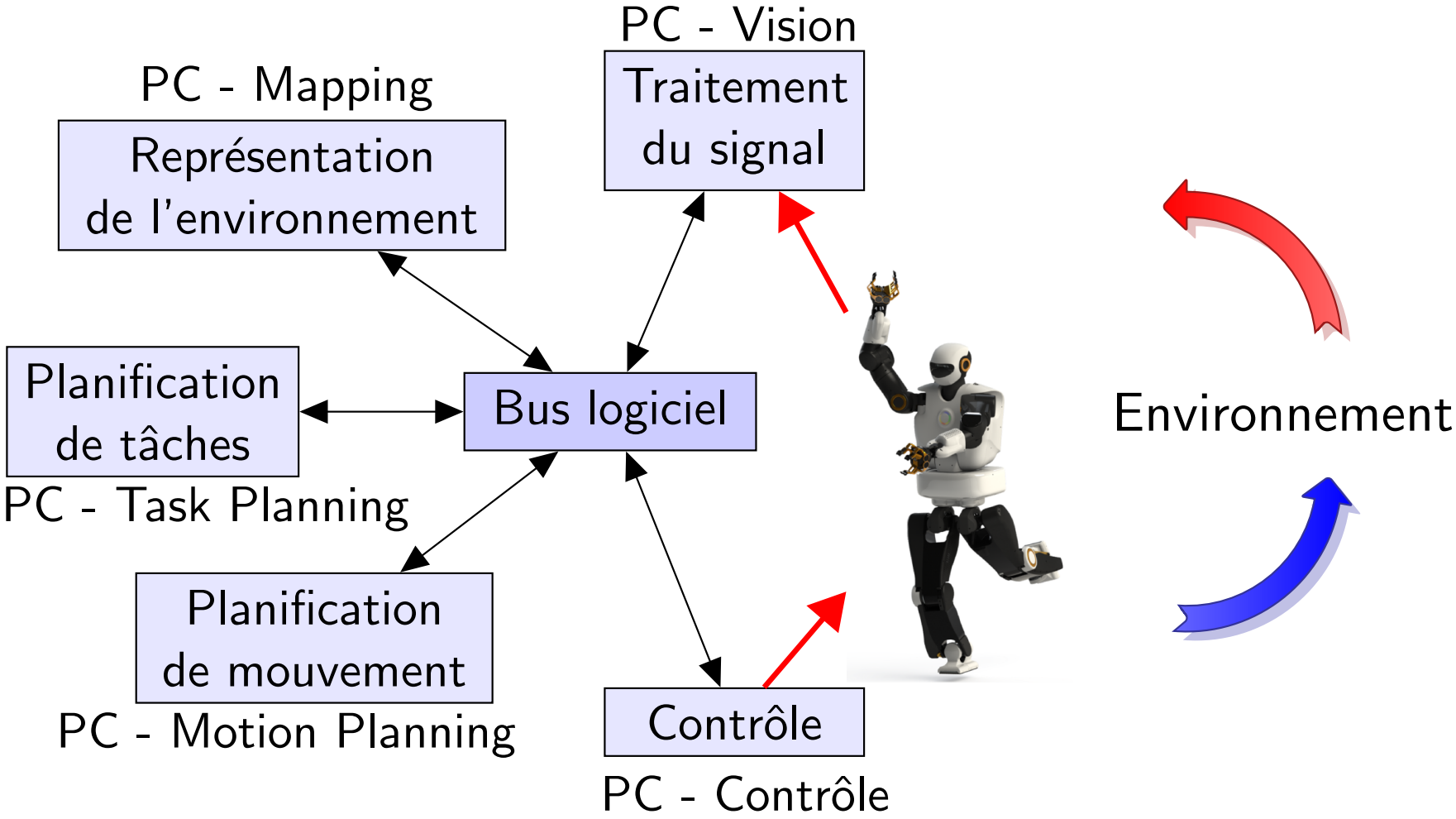
Motivations - Application robotique



Motivations - Application robotique - Outils



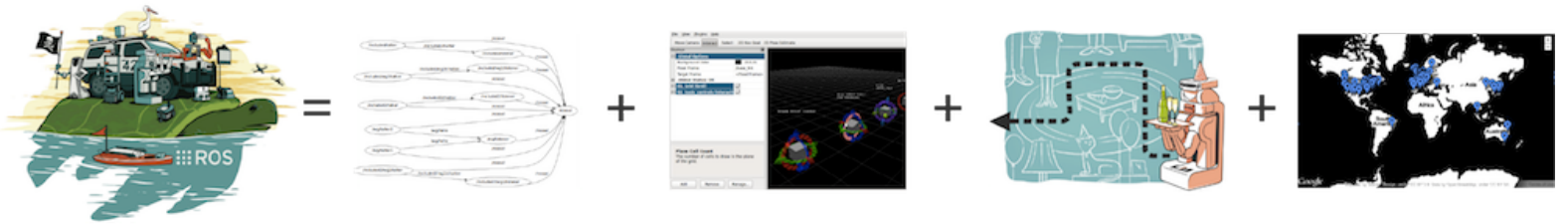
Motivations - Application robotique distribuée



Motivations

Passage à l'échelle

La communauté robotique doit *collaborer* pour créer des systèmes robotiques de qualité à l'exemple de l'informatique et la physique. ROS est l'outil qui a permis ce passage.



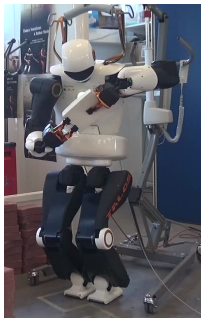
Plomberie Outils Capacités Ecosystème

Buts de ROS 2

Support des systèmes multi-robots impliquant des réseaux non fiables

Supprimer le saut entre le prototypage et les produits finaux

Support de multiple OS

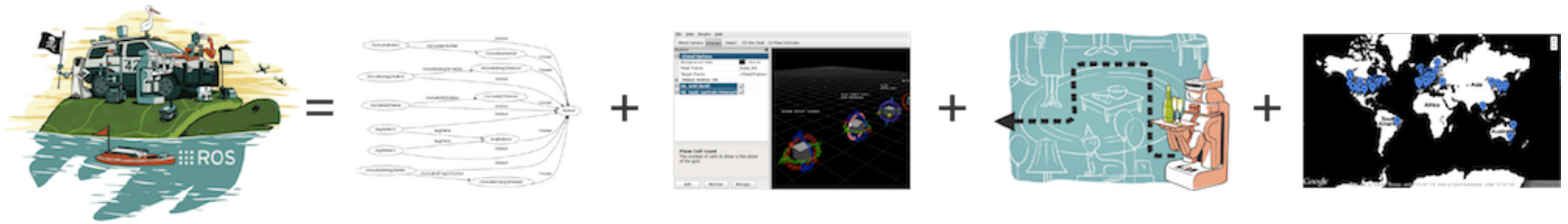


Support pour le contrôle temps réel

Micro controller

ROS equation

DDS



Plomberie Outils Capacités Ecosystème

Motivations - Plomberie

Middleware

- Publication/souscription à la transmission de messages anonymes (Message Passing)
- Enregistrer et rejouer des messages
- Requêtes/réponses à des remote procedure calls
- Système de paramètres distribués

Motivations - Outils

rviz Visualisateur graphique 3D.

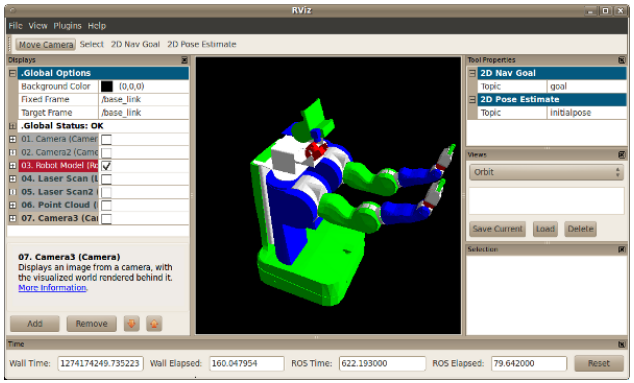
rosbag Enregistrement et visualisation des données.

rxplot Affichage en ligne.

rxgraph Visualisation du système.

rqt Interface graphique de contrôle incrémentale

colcon Système de compilation et de gestion de paquets.



Motivations - Outils

rviz Visualisateur graphique 3D.

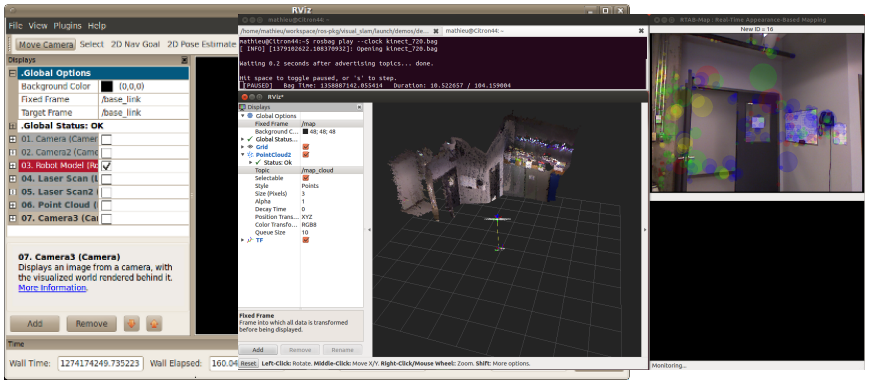
rosbag Enregistrement et visualisation des données.

rxplot Affichage en ligne.

rxgraph Visualisation du système.

rqt Interface graphique de contrôle incrémentale

colcon Système de compilation et de gestion de paquets.



Motivations - Outils

rviz Visualisateur graphique 3D.

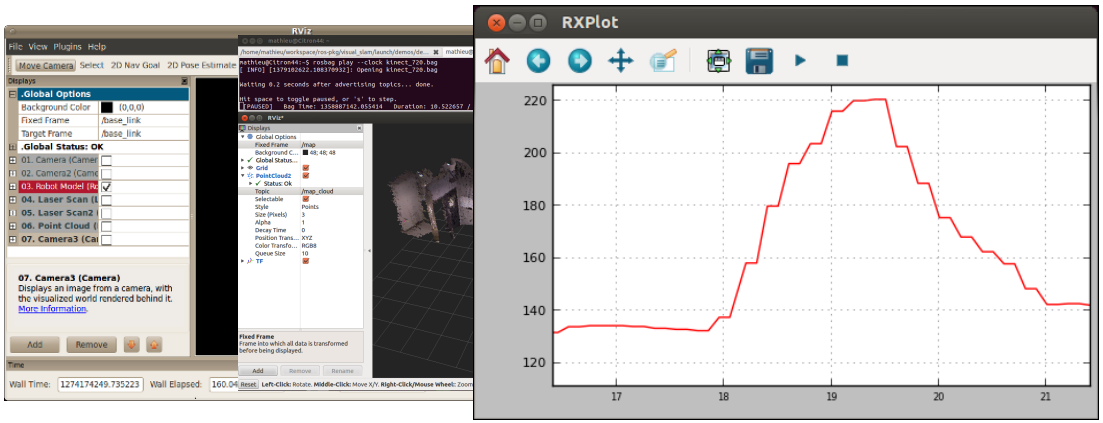
rosbag Enregistrement et visualisation des données.

rxplot Affichage en ligne.

rxgraph Visualisation du système.

rqt Interface graphique de contrôle incrémentale

colcon Système de compilation et de gestion de paquets.



Motivations - Outils

rviz Visualisateur graphique 3D.

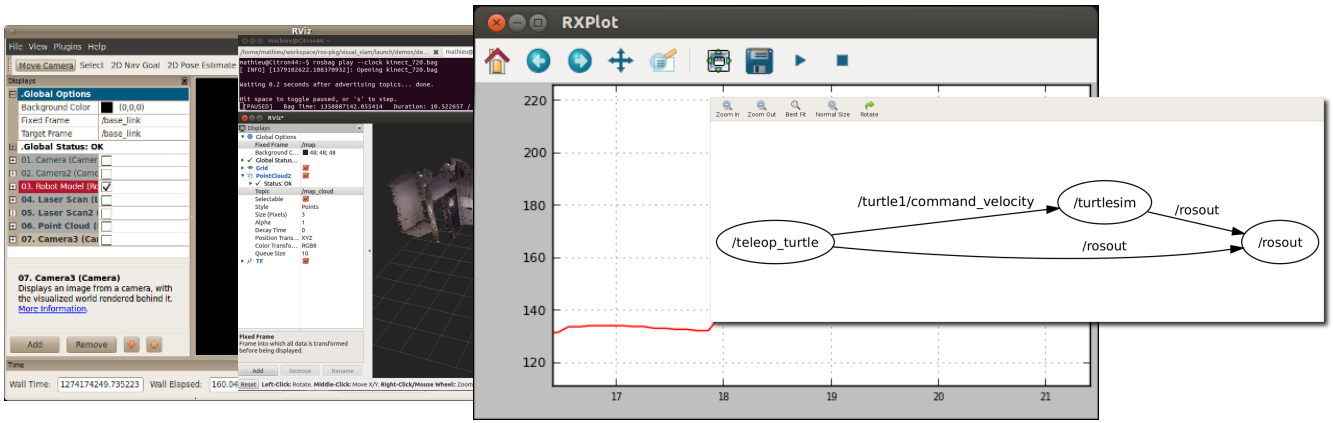
rosbag Enregistrement et visualisation des données.

rxplot Affichage en ligne.

rxgraph Visualisation du système.

rqt Interface graphique de contrôle incrémentale

colcon Système de compilation et de gestion de paquets.



Motivations - Outils

rviz Visualisateur graphique 3D.

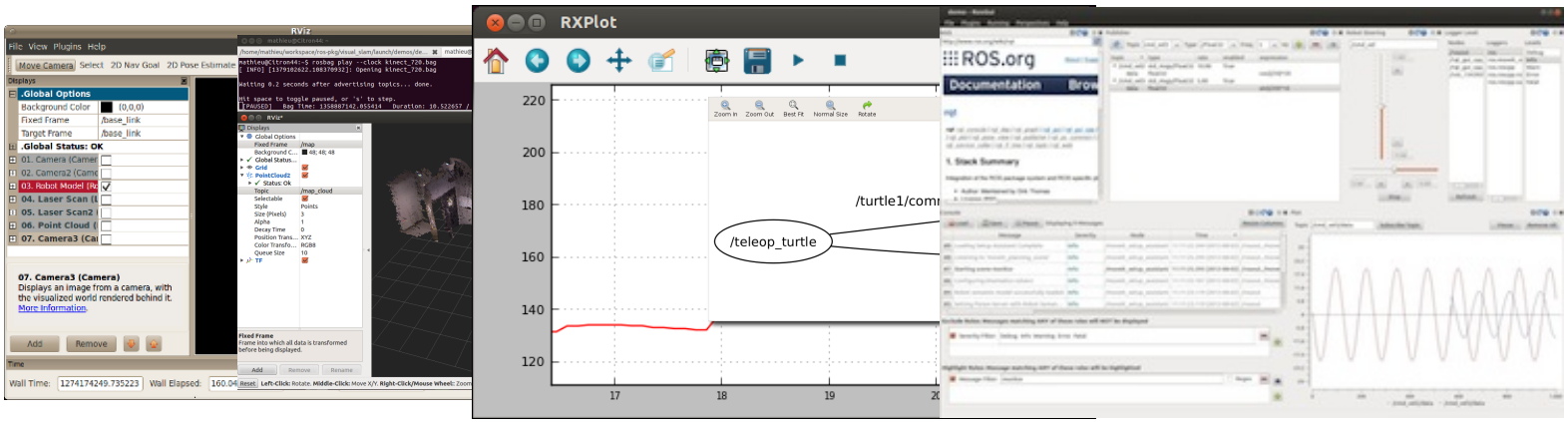
rosbag Enregistrement et visualisation des données.

rxplot Affichage en ligne.

rxgraph Visualisation du système.

rqt Interface graphique de contrôle incrémentale

colcon Système de compilation et de gestion de paquets.



Motivations - Ecosystème

- Définitions de messages standards pour les robots
- Librairie pour la géométrie des robots
- Language de description des robots
- Remote Procedure Calls préemptable
- Diagnostiques
- Estimation de pose
- Localisation
- Cartographie
- Navigation

Motivations - Ecosystème

- Systèmes d'exploitation
- Robots
- Packages

Supporté :

Expérimentaux :

Motivations - Ecosystème

- Systèmes d'exploitation
- Robots
- Packages

01/05/2014 : 111 Robots supportés sur `http://wiki.ros.org/Robots`
02/11/2020 : 63 Robots supportés sur `http://wiki.ros.org/Robots`
14/11/2023 : 194 Robots supportés sur `https://github.com/ros-infrastructure/robots.r`
01/05/2014 : 2048 Packages supportés sur `http://wiki.ros.org/Packages`
25/04/2017 : 1400 Packages supportés sur `http://wiki.ros.org/Packages`
02/11/2020 : 2700 Packages supportés sur `http://wiki.ros.org/Packages`

ROS : Bref historique

2008	ROS started with Willow Garage
2010 - Jan	ROS 1.0
2010 - Mar	Box Turtle
2010 - Aug	C Turtle
2011 - Mar	Diamondback
2011 - Aug	Electric Emys
2012 - Mar	Fuerte Turtle
2012 - Dec	Groovy Galapagos
2013 - Feb	Open Source Robotics Fundation gére ROS
2013 - Aug	Willow Garage est absorbé par Suitable Technologies
2013 - Aug	Hydro Medusa
2014 - Jul	Indigo Igloo (EOL - Apr 2019)
2015 - May	Jade Turtle (EOL - May 2017)
2016 - May	Kinetic Kame (EOL - May 2021)
2017 - May	Lunar Loggerhead (EOL - May 2019)
2018 - May	Melodic Morenia (EOL - May 2023)
2020 - May	Noetic Ninjemys (EOL - May 2025)

ROS-2 : Bref historique

2017 - Dec	Ardent Apalone (EOL - Dec 2018)
2018 - Jun	Bouncy Bolson (EOL - June 2019)
2018 - Dec	Crystal Clemmys (EOL - Dec 2019)
2019 - May	Dashing Diamenta (EOL - May 2021)
2019 - Nov	Eloquent Elusor (EOL - Nov 2020)
2020 - May	Foxy Fitzroy (EOL - May 2023)
2021 - May	Galactic Geochelone (EOL - Nov 2022)
2022 - May	Humble Hawksbill (EOL - May 2027)
2023 - May	Iron Irwin (EOL - May 2027)
2020 - Jun	Rolling Ridley (Ongoing)

ROS : Normalisation des releases

- Humble (Mai 2022 - Mai 2027)

Tier 1 Ubuntu Jammy (22.04/das) Windows 10 (VS 2019 - as)
Tier 2 RHEL 8 (das)
Tier 3 macOS (s) - Debian Bullseye (s)

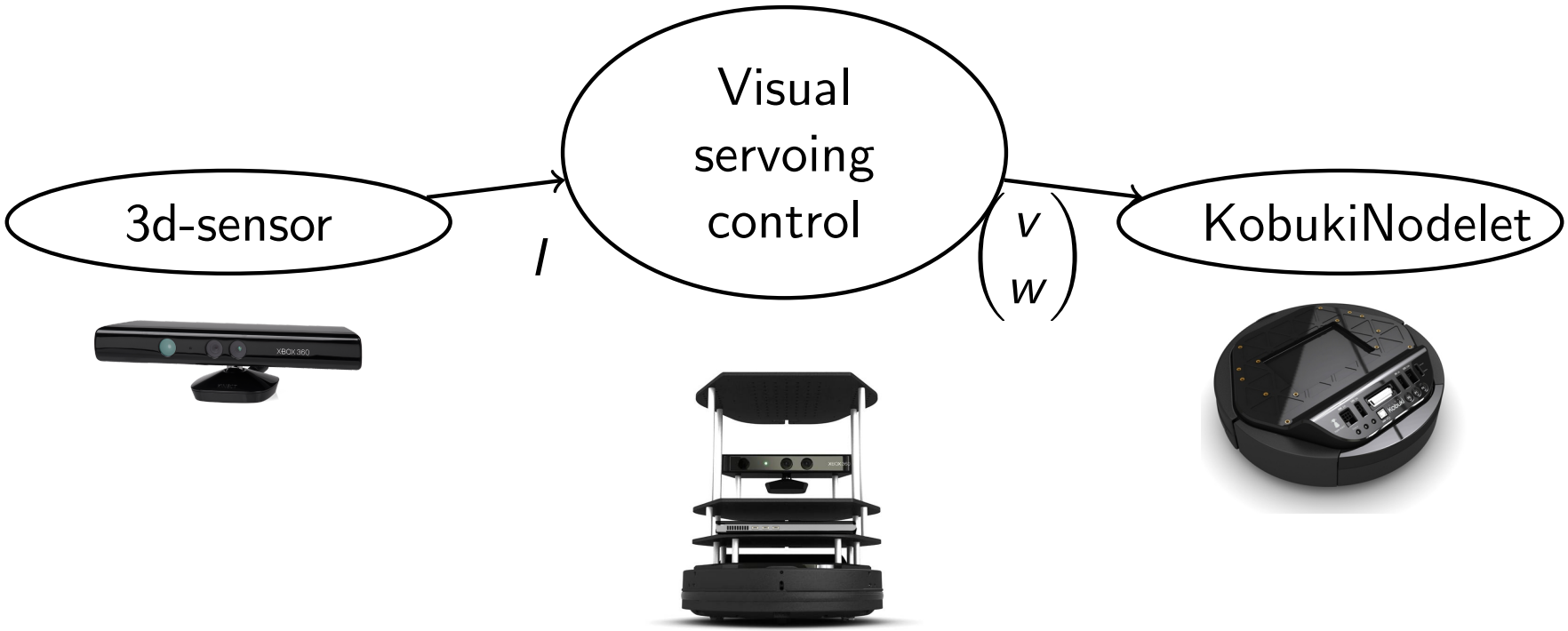
  - C++17, Python 3.6, CMake 3.22.1
  - Ogre3D 1.12.1, Gazebo 11, PCL 1.12.1, OpenCV 4.5.4, Qt 5.15.3

- Rolling Ridley (Juin 2020 - En cours)
  - Depuis Mars 2022 vise les mêmes plateformes que Humble
  - Utilisé pour préparer les nouvelles versions de ROS-2.
  - Empaquetage continue avec le système sémantique des versions (Majeur.Mineur.Patch)
- Liens vers les ROS Enhancement Proposal (REPs) :

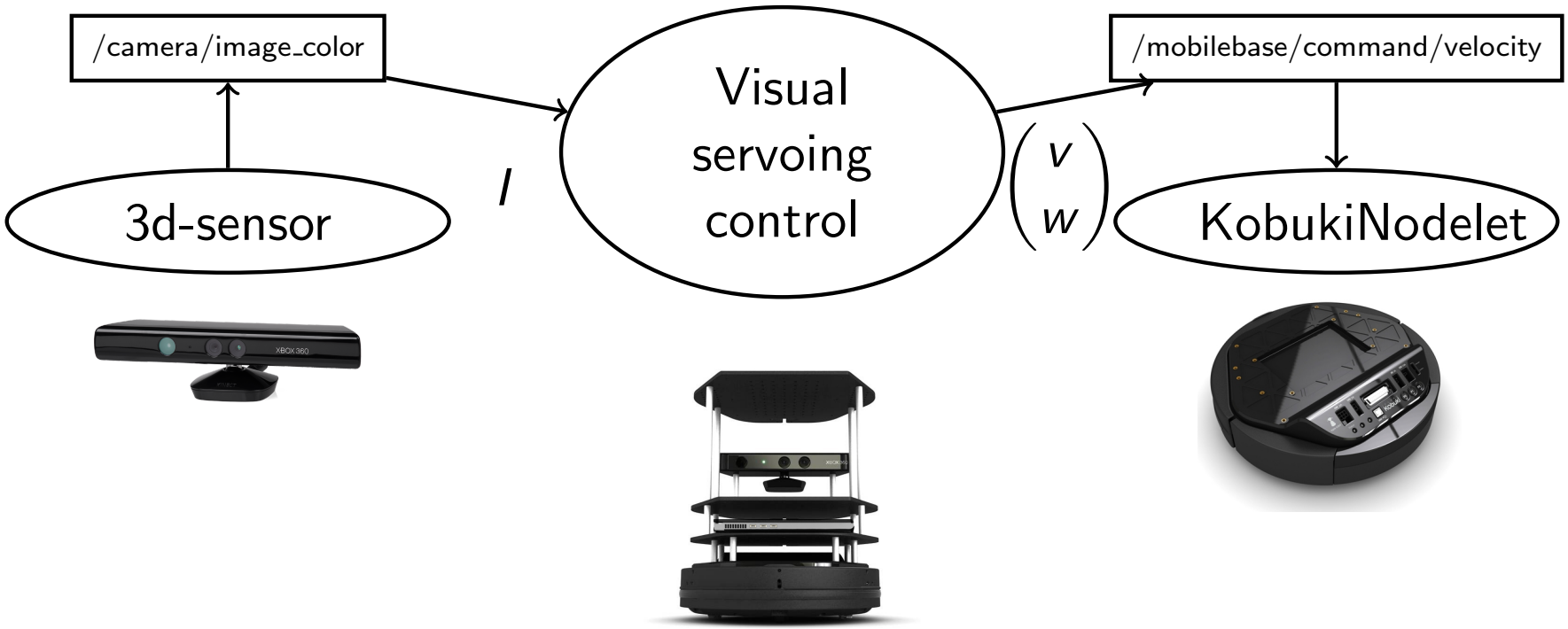
  Pour ROS-1 `http://www.ros.org/reps/rep-0003.html`

  Pour ROS-2 `http://www.ros.org/reps/rep-2000.html`

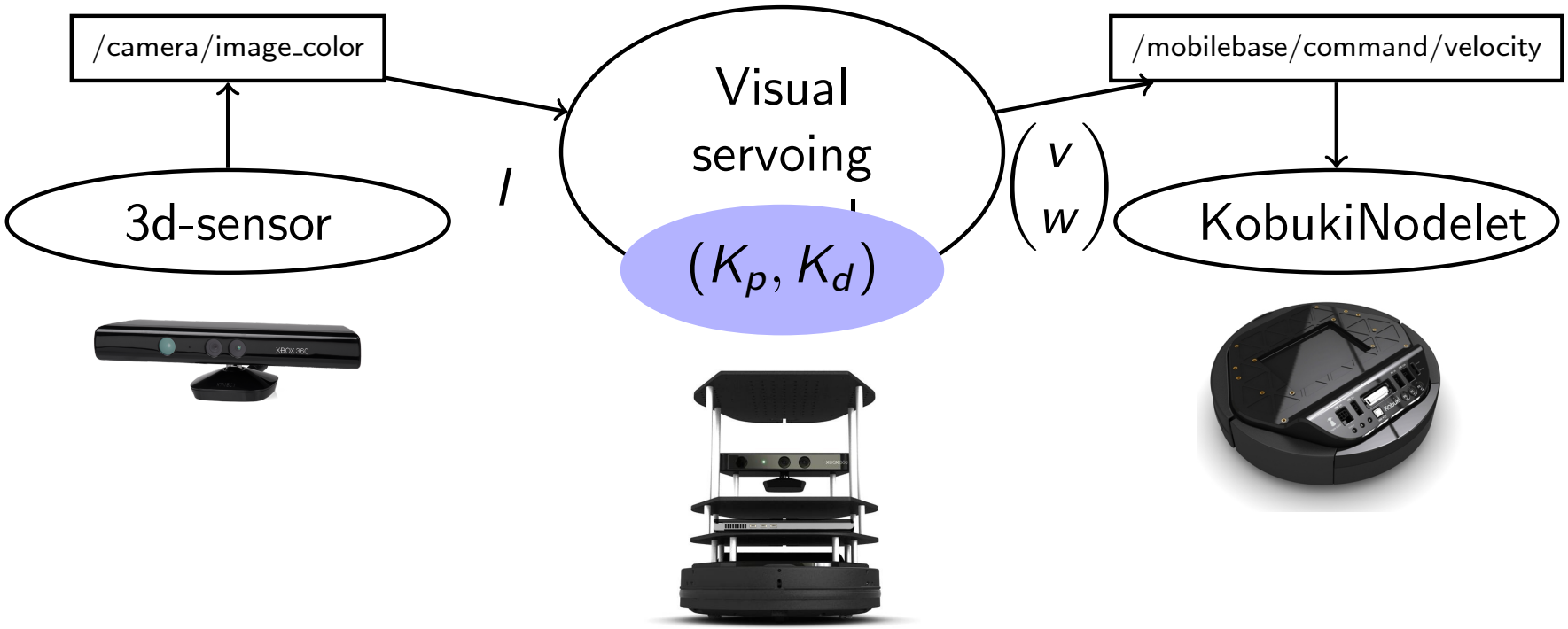
Exemple : Asservissement visuel



Exemple : Asservissement visuel - Topics



Exemple : Asservissement visuel - Topics - Params



Exemple : Asservissement visuel - Software bus

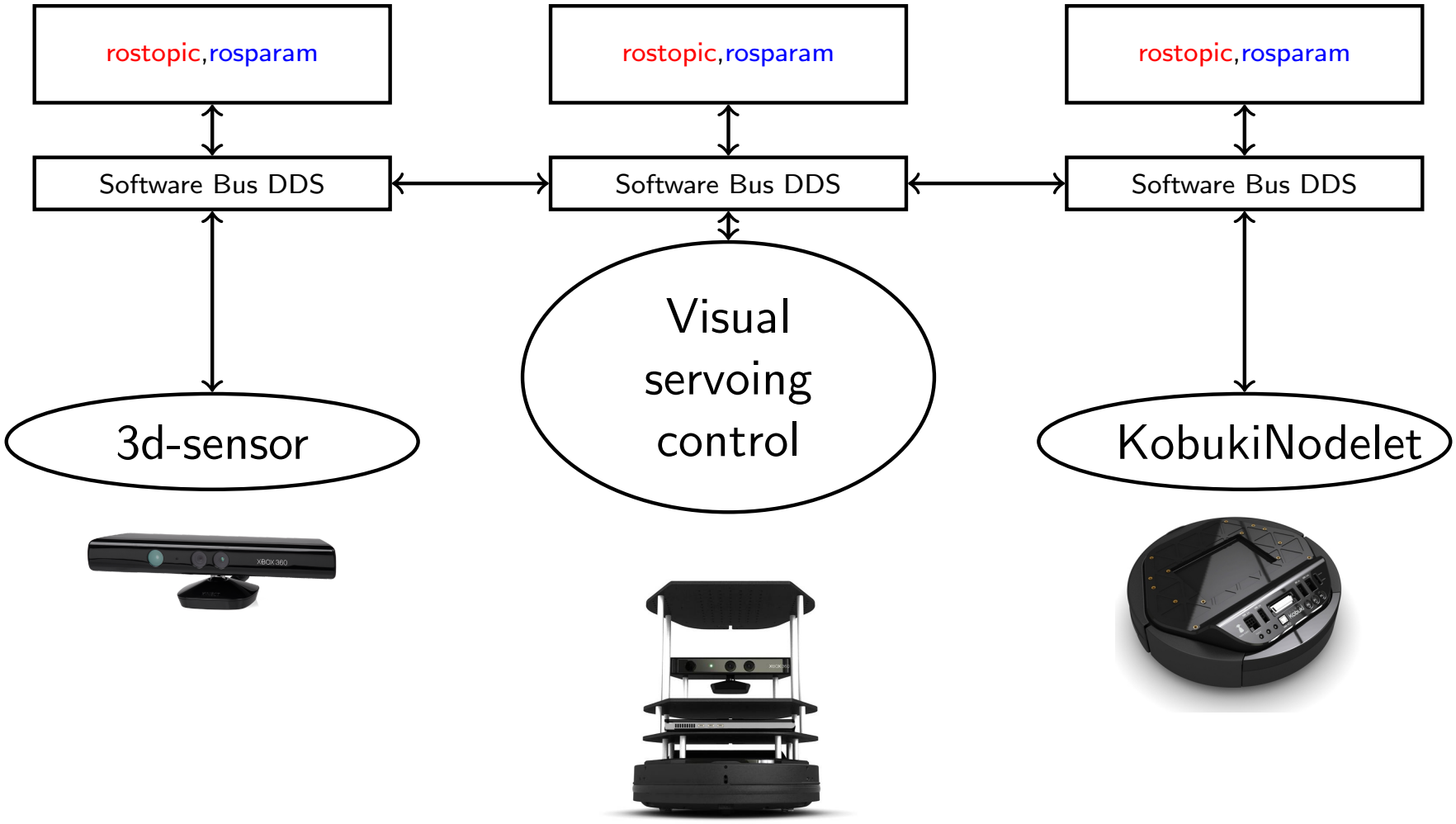
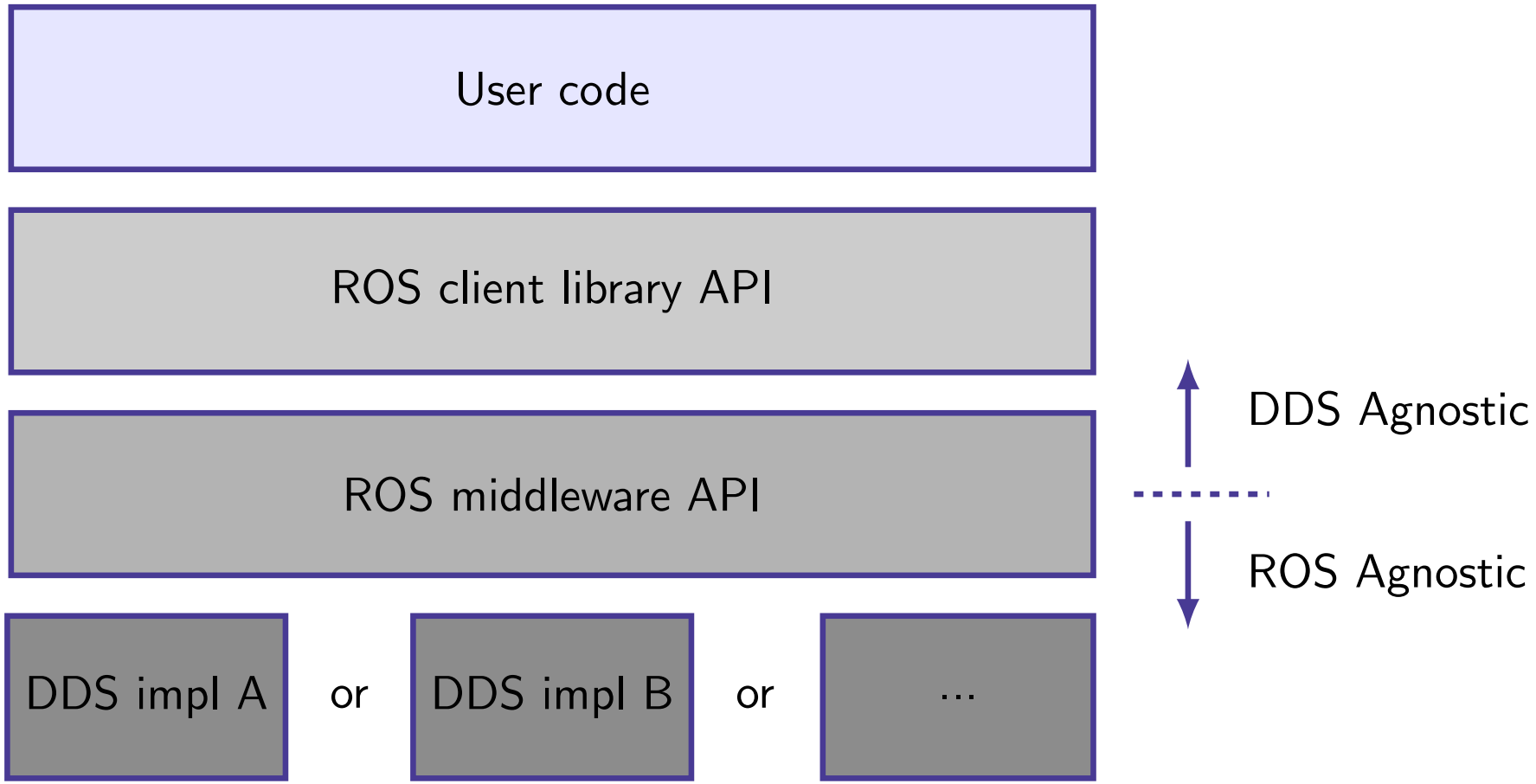


Table des matières

Main Features

- Multiple DDS support (Humble) :
  - **rmw_fastrtps_cpp** : eProsima Fast-DDS Tier 1
  - **rmw_cyclonedds_cpp** : Eclipse Cyclone DDS Tier 1 (aka OpenSplice -2021)
  - **rmw_connextdds** : RTI Connext Tier 1
  - Community supported : rmw_ecal, rmw_ecal (08/2023), rmw_gurumdds (03/2023), rmw_email (04/2023), rmw_dps (2020), rmw_opendds (2021), rmw_iceoryx (09/2023)
- Multiple Operating Systems support :
  Linux, OS, Windows
- Multiple programming languages
  Ada,C++, Go, Python, Java, Node.js, Obj. C, C, Rust, .Net
- Co-existence with ROS-1 systems (ROS 1 bridge)

Architectural overview



DDS : Data Distribution Service

- OMG Standard - https ://www.dds-foundation.org/
- Pas d'annuaire d'objets
- Première version 1.0 en Décembre 2004 (Librement accessible)
- Dernière version 1.4 en Avril 2015 (Librement accessible)
- Centré sur les applications par flux temps-réel de données (real-time data flow)
- Modèle Data-Centric Publish-Subscribe (DCPS)
- 4 vendeurs (parmi lesquels on trouve Eclipse et eProsima qui fournissent une implémtentation libre et open-source)

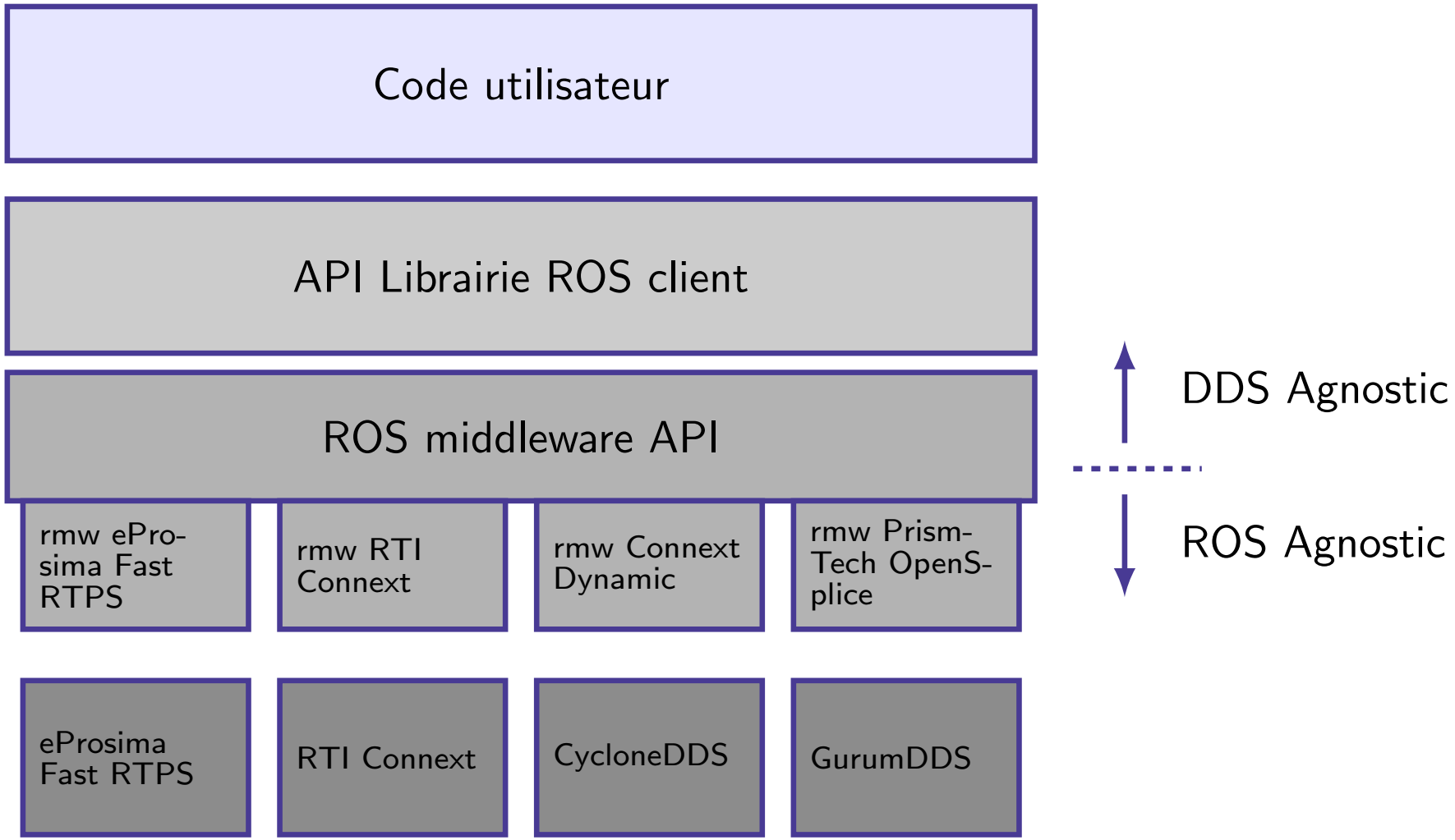
DDS a été utilisé dans

- les navires de guerres
- des installations d'utilité publique comme des barrages
- des systèmes financiers
- des systèmes spatiaux
- des systèmes de vols
- des systèmes d'aiguillages de trains

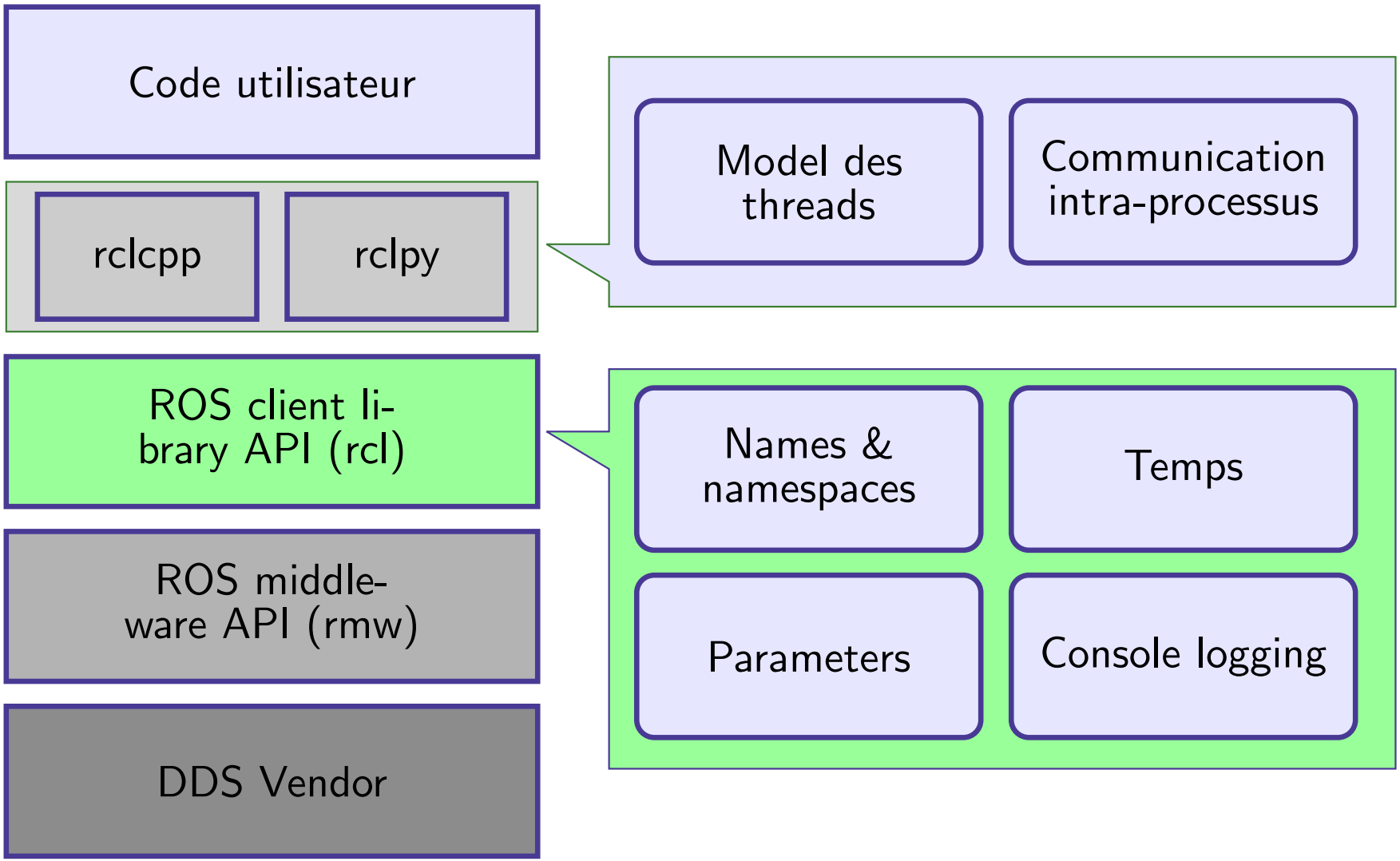
Caractéristiques de DDS

- Définition de la Qualité de Service des transmissions de données à travers les topics.
- La possibilité de définir des objets tels que **DataWriter** et **DataReader** permet une politique plus fine pour la publication et la souscription.
- Real-Time Publish Subscribe (RTPS)
- Plusieurs vendeurs DDS ont des implémentations spécifiques pour les systèmes embarqués. Elles sont optimisées vis à vis de la taille de la librairie qui se limite à quelques centaines de kilo-octets.
- Pas de procédure d'appels à des services distants.

Architecture conceptuelle



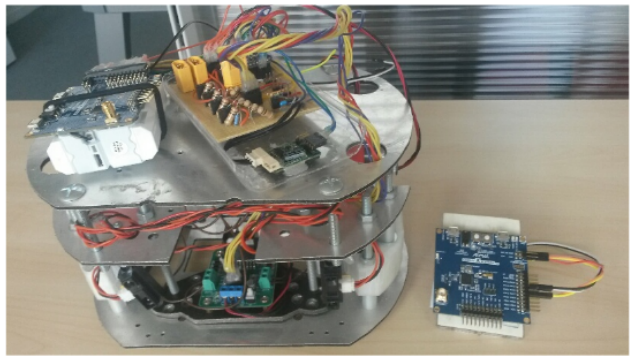
Libraries ROS client : rcl



Example of ROS 2 on ARM-M0

Loïc Dauphin, Team Interface, INRIA

Core :	ARM M0
Microprocessor :	ATSAMR21G18A 32Kb RAM 256Kb Flash
OS :	RIOT
2 others network protocols :	NDN - MQTT
rcl implementation :	C



[Dauphin,ACM, ICN, 2017]

Example on Turtlebot 2

- Joystick teleop
- Follower
- Cartographer
- ACML Localization

ROS-2 Evaluation

- Pas assez temps-réel : d'un bus de terrain à un autre bus de terrain le jitter est trop important.
- Comparaison avec des standards industriels :
  - Comparaison non favorable avec OPC UA [Profanter, ICIT,2018]
  - Comparaison non favorable avec PLCOpen [Fischer, ITASE, 2019]
- Oui mais avec ROS-2 on pourrait utiliser ce type de protocole, il est par contre possible que les mécanismes de synchronisation des topics soit un goulot d'étranglement.

Conclusion - Messages

- Middleware basé sur DDS, Zenoh en local (09/2023)
- Multi platform
- Meilleure gestion de la Qualité de Service
- Nouveau système de construction du système catkin → ament (cament)
- Nouvelle librairie client
- Composabilité des nodes
- Complexité de la programmation beaucoup plus importante

WARNING !

colcon

Installation de ROS - Humble - Ubuntu 22.04.1 LTS

1 - Spécifications des clefs

2 - Spécifications des sources.list

3 - Mise à jour des listes de paquets

```
sudo apt-get update
```

4 - Installation des paquets

```
sudo apt-get install ros-humble-desktop
```

Plus de détails ici : `https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html`

Configuration - Humble - Ubuntu 22.04.1 LTS

1 - Spécifications de ROS_ROOT et ROS PACKAGE PATH

```
env | grep ROS
```

2 - Ligne à ajouter au fichier .bashrc

```
source /opt/ros/humble/setup.bash
```

3 - Création de l'espace colcon

```
mkdir -p ~/dev_ws/src
cd ~/dev_ws/src
```

Tutorial Installing and Configuring Your ROS Environment :

`https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Configuring-ROS2-Environment.html`

Configuration - Humble - Ubuntu 22.04.1 LTS

Votre version ROS est indiquée par ROS_VERSION

```
ROS_VERSION=2
```

La version de Python est indiquée par ROS_PYTHON

```
ROS_PYTHON=3
```

Si vous êtes plusieurs à travailler sous ROS-2 vous devez spécifier des domaines différents (0-101)

```
ROS_DOMAIN_ID=10
```

Il est possible de limiter les communications au localhost avec

```
ROS_LOCALHOST_ONLY=1
```

ROS Filesystem - Navigation - Humble

1 - ros2 pkg : Donne les informations sur les paquets (exemple rclcpp)

```
ros2 pkg prefix rclcpp
```

2 - ros2 pkg : Affiche les exécutables du paquet

```
ros2 pkg executables package_name
```

3 - ros2 pkg : Liste les paquets installés

```
ros2 pkg list
```

Création d'un paquet ROS - Définition - Humble

- Le paquet doit contenir un fichier package.xml qui suit le format approprié. Le fichier package.xml contient des meta informations sur le paquet.
- Il n'y a qu'un seul paquet par répertoire.
  Les paquets imbriqués, et les paquets dans un même répertoire sont interdits.

Le paquet le plus simple est :

```
mon_paquet/
CMakeLists.txt
package.xml
```

Outils de gestion des paquets pour ROS Humble

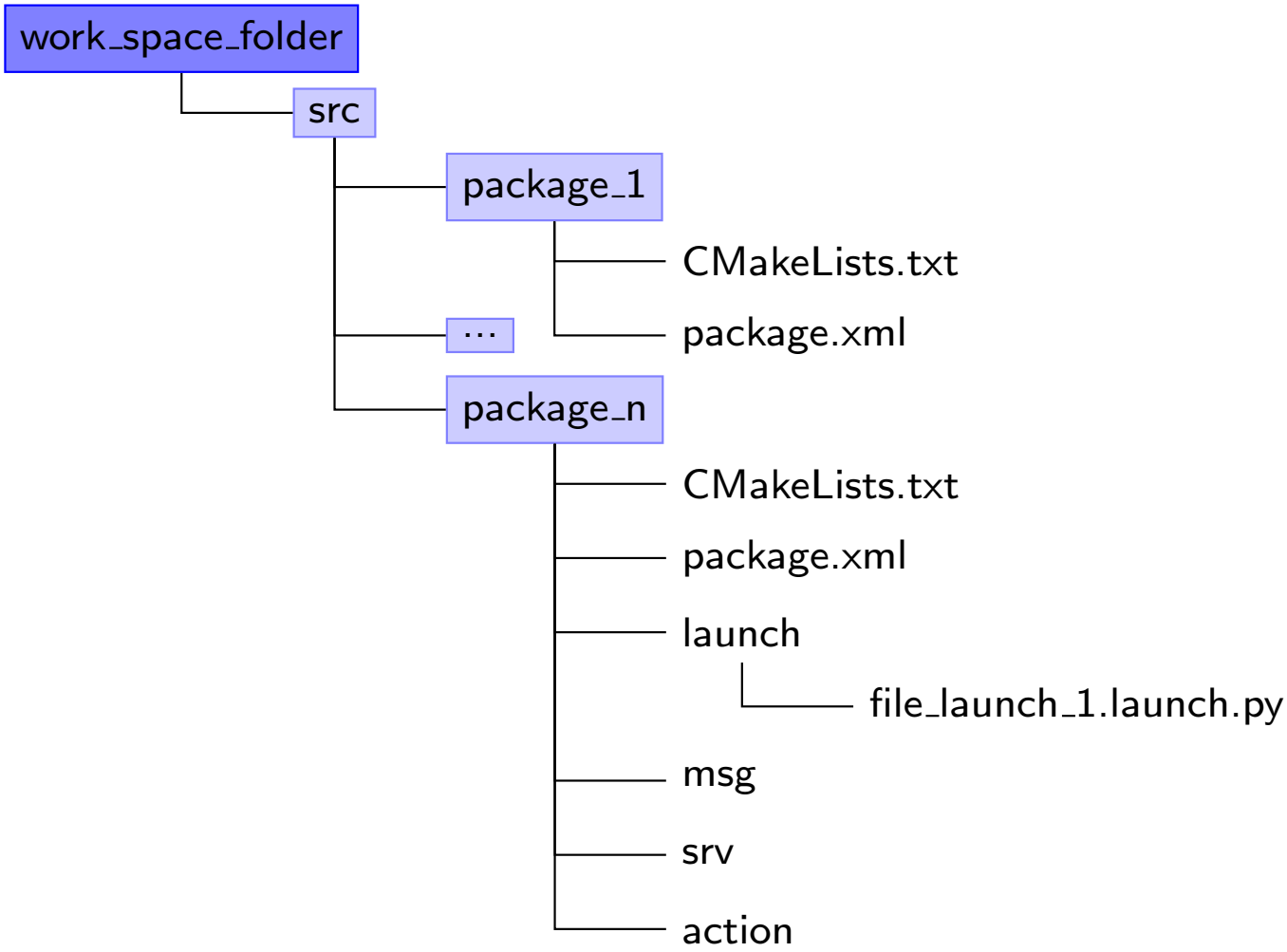
cmake : Un système de construction et de compilation du paquet. Indépendant de ROS. Il nécessite le fichier CMakeLists.txt

colcon : C'est un superbuild : Permet de gérer de multiples packages ensemble.
Autres outils : catkin tools

ament : Macros cmake pour permettre de prendre en compte la structure de ROS.

Table des matières

Paquets dans un workspace colcon - (1/9) - Humble



Paquets dans un workspace colcon - (2/9) - Humble

1 - Aller dans le répertoire src du paquet

```
cd ~/dev_ws/src
```

Pour créer un nouveau package, et ses dépendences :
ros2 pkg create

```
ros2 pkg create –build-type ament_cmake cpp_pubsub
```

Paquets dans un workspace ROS - (3/9) - Humble

3 - Les dépendences sont stockées dans le fichier package.xml

```
cd beginner_tutorials
cat package.xml
```

```
<package>
...
<buildtool_depend>ament_cmake</buildtool_depend>
<test_depend>ament_lint_auto</test_depend>
<test_depend>ament_lint_common</test_depend>
...
</package>
```

Paquets dans un workspace - (4/9) - Humble

4 - Dépendences indirectes d'un paquet

ros2 pkg xml beginner_tutorials

```
<package>
<package format="2">
...
<buildtool_depend>ament_cmake_ros</buildtool_depend>
...
</export>
</package>
```

Personnalisation du paquet - (5/9) - Humble

5 - Description du paquet

```
<description>The beginner_tutorials package</description>
```

```
<package>
<!- One maintainer tag required, multiple allowed, one person per tag
->
<!- Exemple : ->
<!- <maintainer email="jane.doe@example.com">Jane
Doe</maintainer> ->
<maintainer email="user@todo.todo">user</maintainer>
```

Personnalisation du paquet - (6/9) - Humble

7 - License du paquet

```
<!- One license tag required, multiple allowed, one license per tag ->
<!- Commonly used license strings : ->
<!- BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1,LGPLv3 ->
<license>TODO</license>
```

La license souvent utilisée sous ROS est la BSD

```
<license>BSD</license>
```

Personnalisation du paquet - (7/9) - Humble

8 - Spécification des dépendences

```
<!- The * depend tags are used to specify dependencies ->
<!- Dependencies can be other ROS packages or system dependencies ->
<!- Exemples : ->
<!- Use build_depend for packages you need at compile time : ->
<!- <build_depend>genmsg</build_depend> ->
<!- Use buildtool_depend for build tool packages : ->
<!- <buildtool_depend>ament_cmake</buildtool_depend> ->
<!- Use exec_depend for packages you need at runtime : ->
<!- <exec_depend>python-yaml</exec_depend> ->
<!- Use test depend for packages you need only for testing : ->
<!- <test depend>gtest</test depend> ->
<buildtool_depend>ament_cmake</buildtool_depend>
<build_depend>roscpp</build_depend>
<build_depend>rospy</build_depend>

<build_depend>std_msgs</build_depend>
```

Personnalisation du paquet - (8/9) - Humble

9 - Ajout de la dépendence à l'exécution :

```
<buildtool_depend>ament_cmake</buildtool_depend>
<build_depend>roscpp</build_depend>
<build_depend>rospy</build_depend>
<build_depend>std_msgs</build_depend>
<run_depend>roscpp</run_depend>
<run_depend>rospy</run_depend>
<run_depend>std_msgs</run_depend>
```

Personnalisation du paquet - (9/9) - Humble

10 - Fichier package.xml final

```
<?xml version="1.0" ?>
<package>
<name>beginner_tutorials</name>
<version>0.1.0</version>
<description>The beginner tutorials package</description>
<maintainer email="you@yourdomain.tld">Your Name</maintainer>
<license>BSD</license>
<url type="website">http://wiki.ros.org/beginner_tutorials</url>
<author email="you@yourdomain.tld">Jane Doe</author>
<buildtool_depend>ament_cmake</buildtool_depend>
<build_depend>roscpp</build_depend>
<build_depend>rospy</build_depend>
<build_depend>std_msgs</build_depend>
<exec_depend>roscpp</exec_depend>
<exec_depend>rospy</exec_depend>
<exec_depend>std_msgs</exec_depend>

</package>
```

Compilation des paquets Humble

La compilation de tous les paquets s'effectue avec une seule commande.
Cette commande doit s'effectuer dans le répertoire du workspace.
Le répertoire **build** du workspace stocke les objets compilés intermédiaires.

```
colcon build
```

Graphe d'applications avec ROS

Nodes : Un node est un processus qui utilise ROS pour communiquer avec d'autres noeuds.

Messages : Types de données ROS utilisés pour souscrire ou publié sur un topic.

Topics : Les nodes peuvent *publier* des messages sur un topic aussi bien que *souscrire* à un topic pour recevoir des messages.

Paramètres : Informations très peu dynamiques qui doivent être partagés dans l'application.

Définitions - Node

- Un fichier éxecutable dans un paquet ROS.
- Un processus avec un nom.
- Les noeuds ROS utilisent une librairie client pour communiquer avec les autres noeuds.
- Les noeuds peuvent **publier** ou **souscrire** à des **topics**.
- Les noeuds peuvent fournir ou utiliser un **service**.
- Les librairies client sont :
  - *rclpy* pour python.
  - *rclcpp* pour C++.

Table des matières

Liste des nodes

Pour obtenir la liste des nodes actifs :

```
ros2 node list
```

Pour obtenir des informations sur un node :

```
ros2 node info /rosout
```

Tutorial Understanding ROS Nodes : `https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html`

Utilisation de ros2 run

Pour pouvoir lancer un fichier exécutable/node d'un paquet

```
ros2 run [package_name][node_name]
```

Par exemple pour lancer turtlesim :

```
ros2 run turtlesim turtlesim_node
```

Pour renommer des arguments (utiliser rosnode list pour vérifier) :

```
ros2 run turtlesim turtlesim_node __name :=my_turtle
```

Pour tester si le node est actif

```
ros2 node ping my_turtle
```

Tutorial Understanding ROS Nodes : `https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html`

Comprendre les topics - Préparation

Pour démarrer turtle_sim et turtle_teleop_key

```
ros2 run turtlesim turtlesim_node
ros2 run turtlesim turtle_teleop_key
```

Démarrer la visualization du graphe de l'application :

```
ros2 run rqt_graph rqt_graph
```

Démarrer le graphe de l'affichage des topics :

```
ros2 run rqt_plot rqt_plot
```

Tutorial Understanding ROS Topics : `https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding-ROS2-Topics.html`

Comprendre les topics - **rostopic**

Les topics sont des données **publiées** par des noeuds, et auquelles les noeuds **souscrivent**.
L'exécutable permettant d'avoir des informations sur les topics est **ros2 topic**.

ros2 topic bw display bandwith used by topic
ros2 topic echo print messages to screen
ros2 topic hz display publishing rate of topic
ros2 topic list print information about active topics
ros2 topic pub publish data to topic
ros2 topic type print topic type

Tutorial Understanding ROS Topics : `https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding-ROS2-Topics.html`

rqt - **rqt_console**

rqt est une interface d'affichage non 3D qui se peuple avec des plugins. Elle permet de construire une interface de contrôle incrémentalement. L'exécutable permettant d'afficher les messages des noeuds de façon centralisé est **rqt_console**.

```
ros2 run rqt_console rqt_console
```

Tutorial Using rqt console et ros2 launch `http://wiki.ros.org/ROS/Tutorials/UsingRqtconsoleRoslaunch`

Enregistrer des données - **rosbag**

ros2 bag permet d'enregistrer et de rejouer des données sur votre application.
Pour enregistrer un sac de données :

```
ros2 bag record -a
ros2 bag record -o subset /turtle1/cmd_vel /turtle1/pose
```

Le nom du fichier démarre avec l'année, la date et le temps et le suffixe .bag

Tutorial Recording and play back data `https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Recording-And-Playing-Back-Data/Recording-And-Playing-Back-Data.html`

Les nodes sont aussi des clients-serveurs - **rosservice**

rosservice permet de lister et d'appeller les services d'un noeud.

Pour enregistrer un sac de données :

```
ros2 service args print service arguments
ros2 service call call the service with the provided args
ros2 service find find services by service type
ros2 service info print information about service
ros2 service list list of services
```

Tutorial Understanding ROS Services and Parameters `http://wiki.ros.org/ROS/Tutorials/UnderstandingServicesParams`

La gestion de paramètres - **rosparam**

ros2 param permet de gérer des données de configuration.
Par exemple le modèle du robot.

```
ros2 param set set parameter
ros2 param get get parameter
ros2 param load load parameters from file
ros2 param dump dump parameters to file
ros2 param delete delete parameter
ros2 param list list parameter names
```

Tutorial Understanding ROS Services and Parameters `http://wiki.ros.org/ROS/Tutorials/UnderstandingServicesParams`

Les actions un design pattern clients-serveurs - **ros2 action**

ros2 action .

Pour enregistrer un sac de données :

ros2 action list Output a list of action names
ros2 action info Print information about an action
ros2 action send_goal Send an action goal

Tutorial Understanding ROS Services and Parameters `https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Actions/Understanding-ROS2-Actions.html`

Lancer plusieurs noeuds - **ros2 launch**

ros2 launch lit un fichier xml qui contient tous les paramètres pour lancer une application distribuée ROS.

```
ros2 launch [package] [filename.launch]
```

Exemple :

```
ros2 launch beginner_tutorials turtlemimic.launch
```

Tutorial Using rqt console et ros2 launch `http://wiki.ros.org/ROS/Tutorials/UsingRqtconsoleRoslaunch`

Table des matières

Tutorials couverts par cette partie

- Création de fichiers ROS msg and srv
- Ecrire un simple souscripteur et publieur (C++)
- Ecrire un simple souscripteur et publieur (Python)
- Ecrire un simple service et un client (C++)
- Ecrire un simple service et un client (Python)
- Faire fonctionner ensemble le client simple et le service simple.

Table des matières

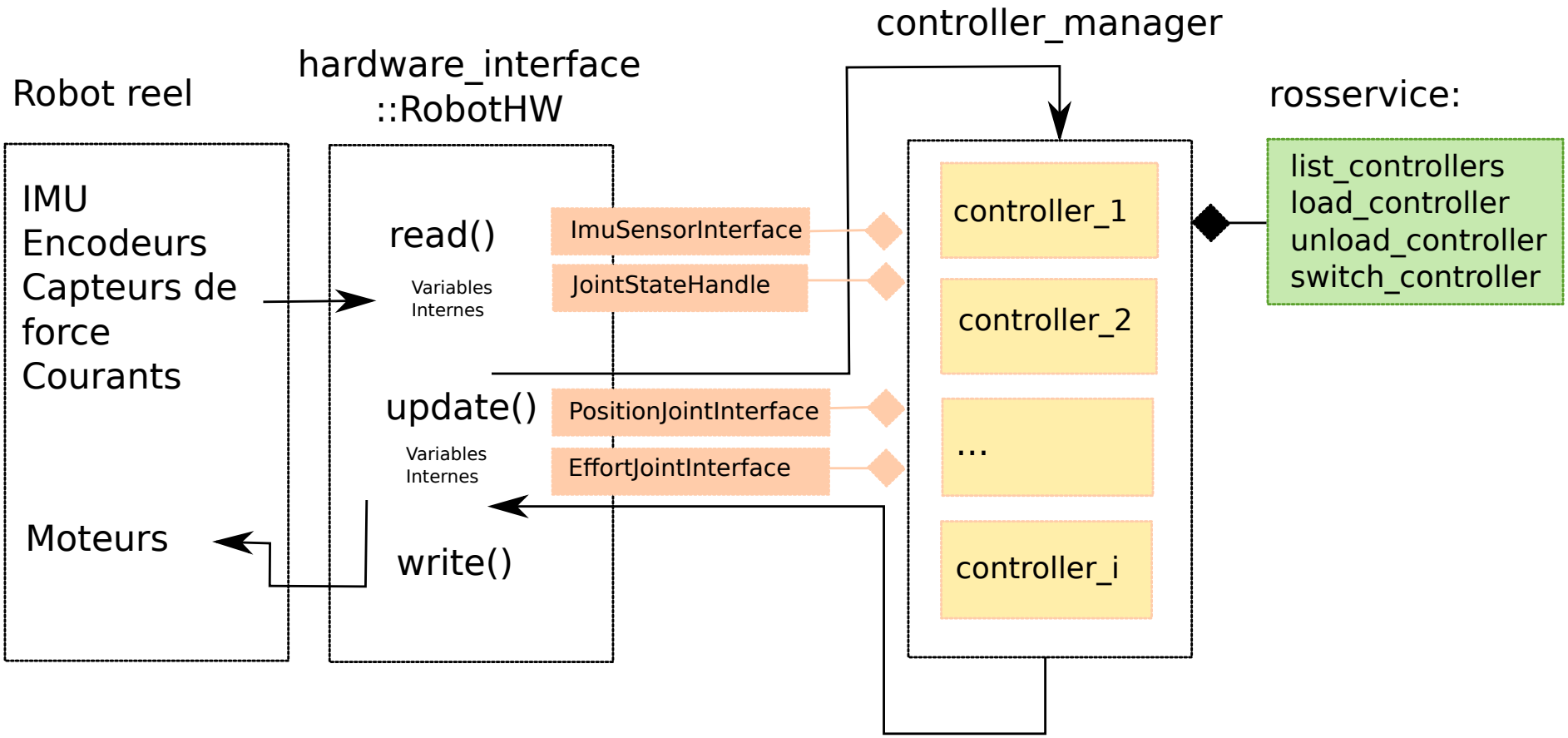
ROS-control

- Une abstraction du robot (Hardware Abstract Layer)
  - Joint Command Interface :
    - Effort Joint Interface
    - Velocity Joint Interface
    - Position Joint Interface
  - JointHandle
  - ForceTorqueSensorHandle
  - ImuSensorHandle
  - ForceTorqueSensorHandle
- Une abstraction des controlleurs :

ROS-control

- Une abstraction du robot (Hardware Abstract Layer)
- Une abstraction des controlleurs :
  - effort_controllers : Commande un couple ou une force désirée
  - joint_state_controller : Lit toutes les positions des joints
  - positions_controllers : Spécifie la position d'un ou plusieurs joints en même temps.
  - velocity_controllers : Spécifie la vitesse d'un ou plusieurs joints en même temps.
  - joint_trajectory_controllers : Suivi de trajectoire.

Schéma d'interactions des objets roscontrol



Exemple de contrôleur avancé