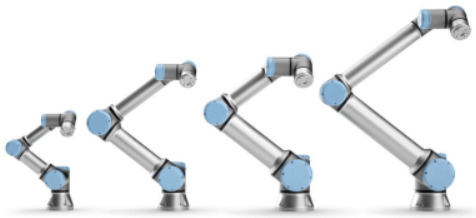
ROS 2 Control

Introduction à ROS-2 Control

Olivier STASSE,
DR-2, CNRS,
Gepetto,
LAAS CNRS

Mars 2023



ROS industrial

ROS

Open Source Robotics Foundation

Table des matières

1 Introduction

2 Hardware

3 Controllers

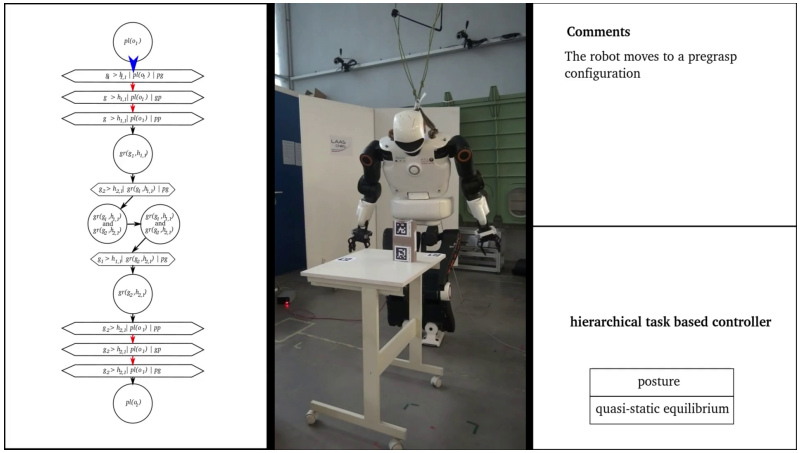
Table des matières

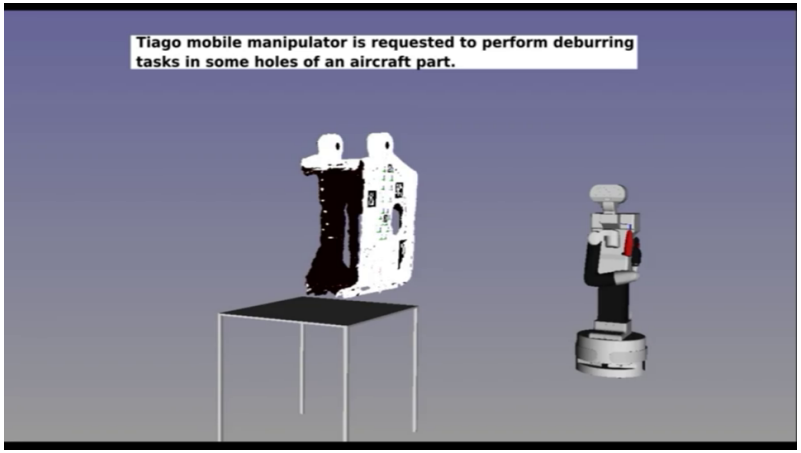
1 Introduction

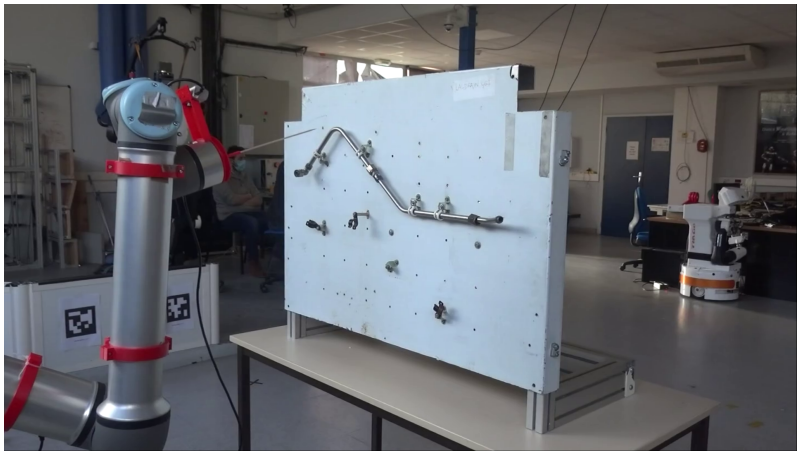
2 Hardware

3 Controllers

Succès au sein de Gepetto avec **ros_control**





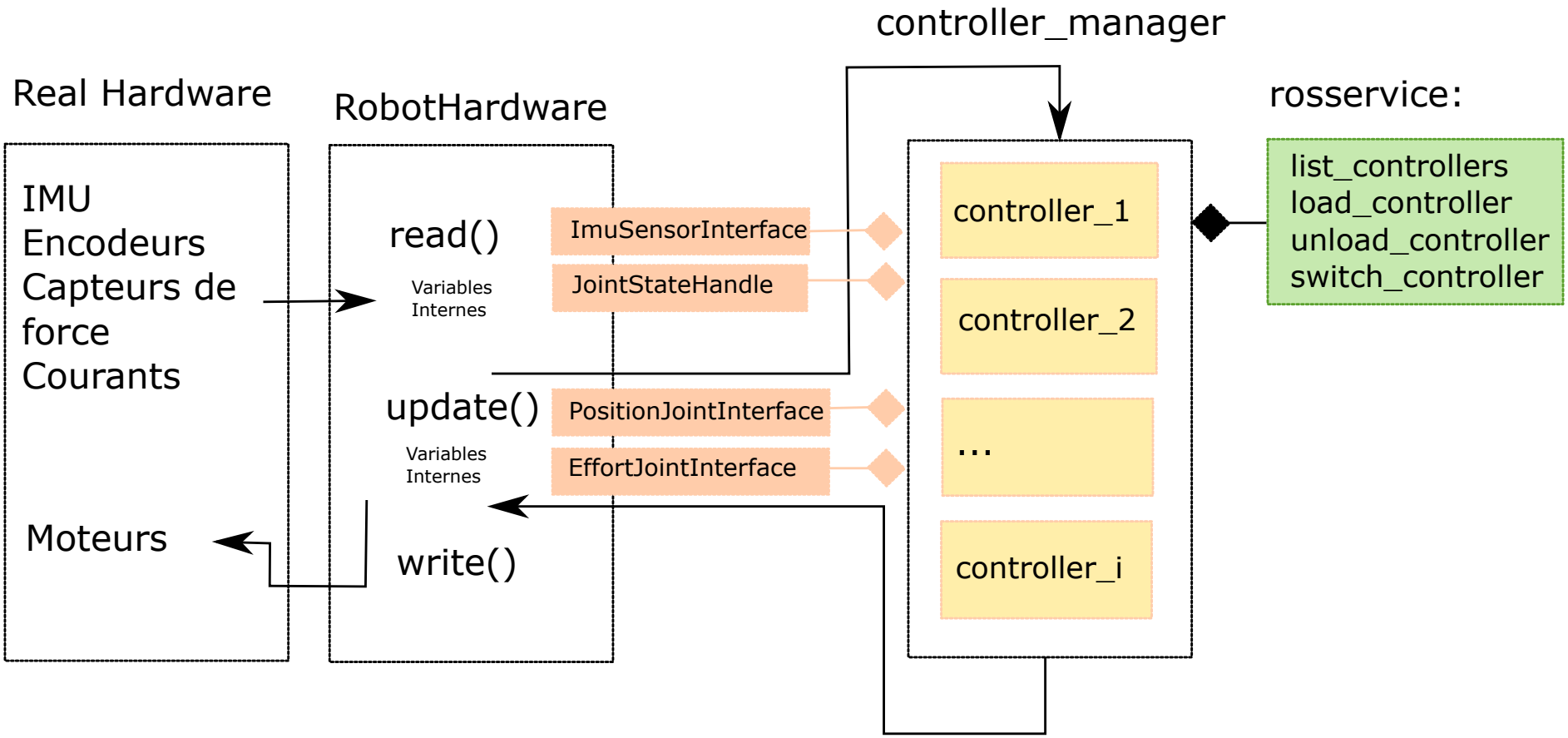


Portage du Tiago à l'UR-10 en une semaine

Les points clefs de ros(2)_control

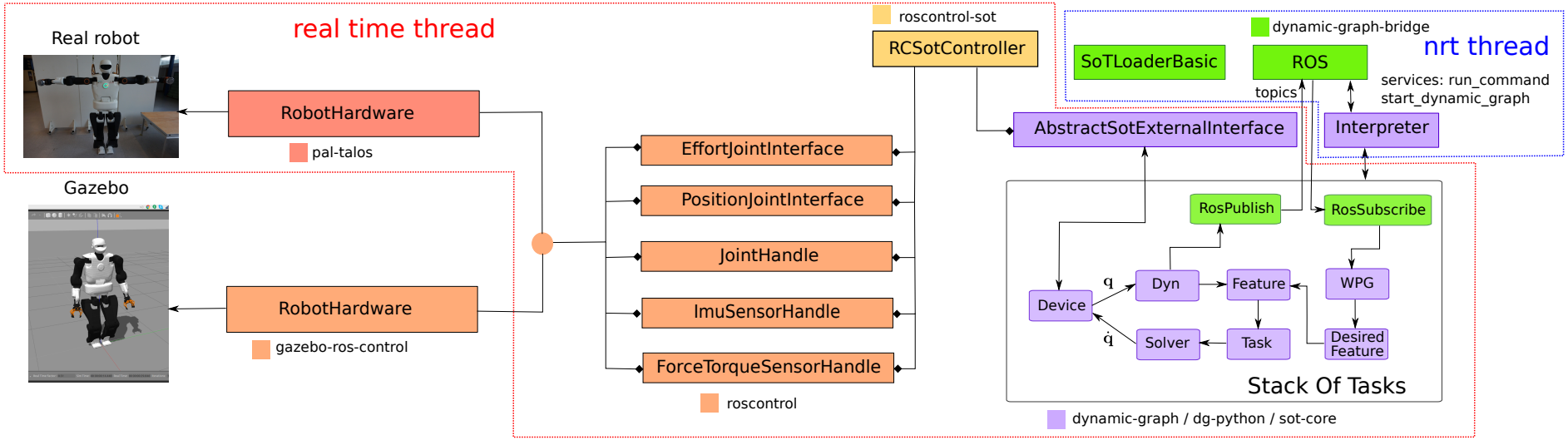
- Temps réel sur RT_PREEMPT (200 Hz – 1 KHz – 2 KHz).
- Hardware Abstract Layer - Standardisation du matériel.
- Controller Abstract Layer - Standardisation des contrôleurs.
- Réutilisation des composants matériels.
- Réutilisation des contrôleurs.
- C++ Librairie.
- Le middleware n'est utilisé que pour le monitoring et la gestion de haut niveau.

Schéma d'interactions des objets roscontrol



Objets pour la commande !

Exemple de contrôleur avancé



Les points clefs de ros2_control

- Hardware Abstract Layer - Standardisation du matériel
- Controller Abstract Layer - Standardisation des contrôleurs
- Le Controller Manager permet de gérer les deux.
- Réutilisation des composants matériels à travers le Ressource Manager
- Réutilisation des contrôleurs (Joint Trajectory Controller, Diff Driver Controller)
- C++ Librairie
- Le middleware n'est utilisé que pour le monitoring et la gestion de haut niveau

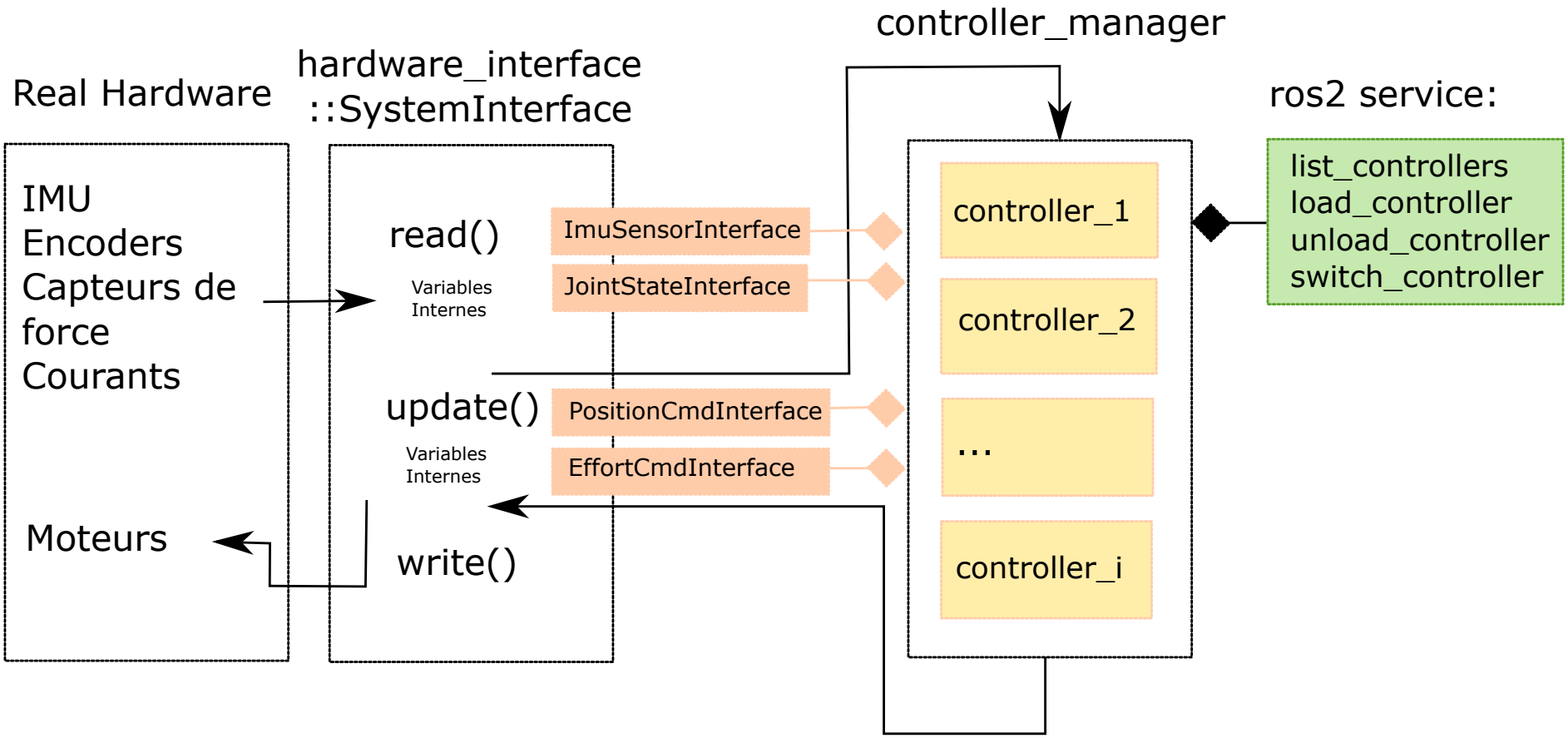
L'éco-système de ros2_control

- Denis Stogl et Bence Magyar sont les mainteneurs principaux. Le premier a fondé sa société et ros_control y joue un rôle central. Le deuxième rapporte au Technical Committee de ROS les avancées de ros_control.
- Andy Zelenak and Tyler Weaver pour PickNick Robotics.
- ros2_control est important pour MoveIt_2 !.
- Documentation : https://control.ros.org Il y a du travail !
- Tous les quinze jours les développeurs se réunissent pour des discuter des développements et des problèmes.
  - Google meet.
  - L'ordre du jour est publique et disponible.
  - Inscription via le groupe google ros2-control-working-group.
  - Pas de condition pour y entrer.

Les (dés-)avantages par rapport à ros_control

- Avantages
  - Possibilité d'étendre les interfaces d'état (state) et les interfaces de contrôle (command)
  - Controlleurs chaînables
  - Gestion de l'état des contrôleurs à la ROS-2.
- Inconvénients
  - Un peu plus compliqué à cause des paramètres et des scripts de launch en python.
  - L'API bouge encore pour introduire les nouvelles fonctionnalités.

Schéma incomplet d'interactions des objets ros2_control

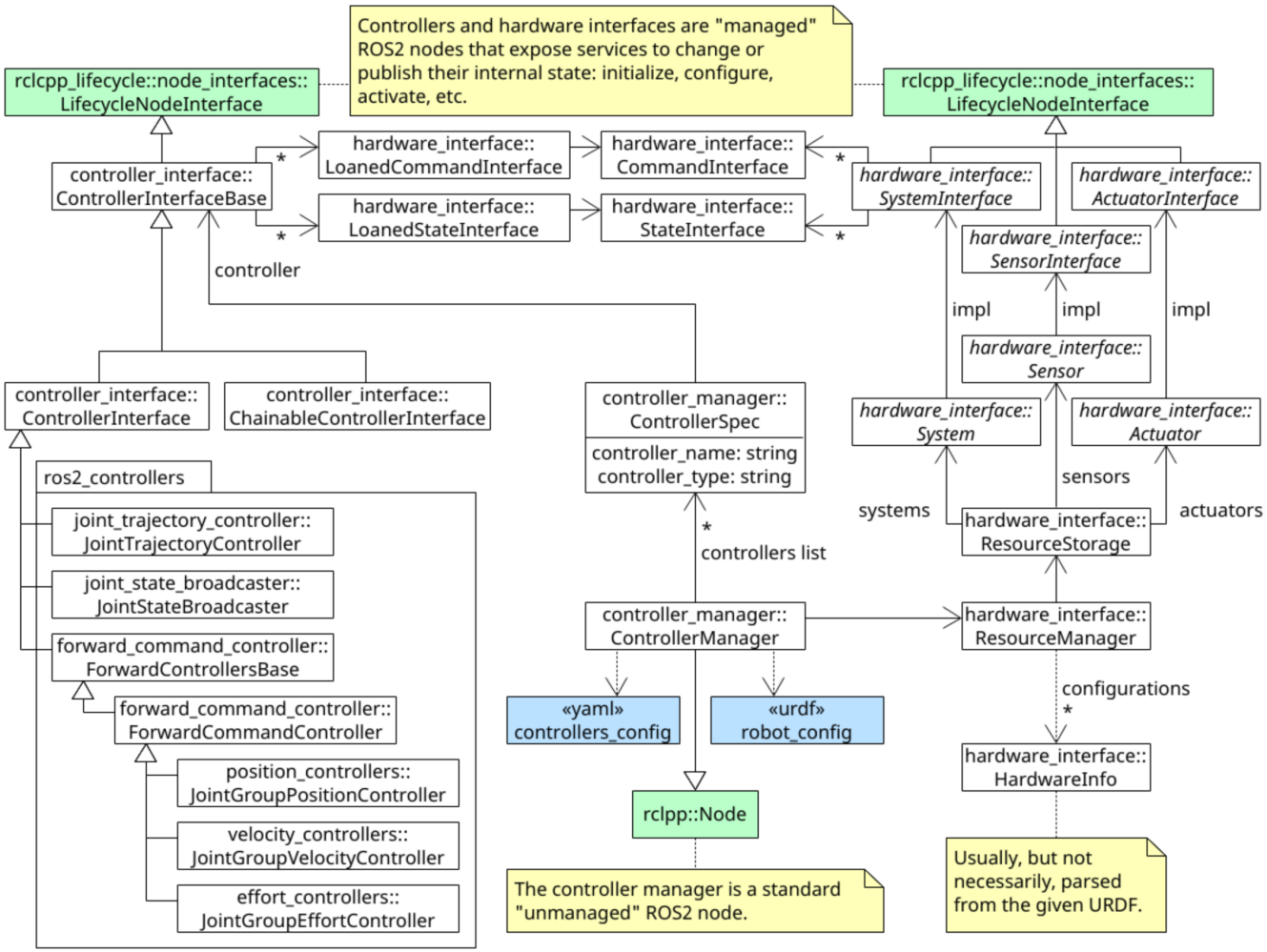


La boucle perception-contrôle-action

```
1   while (rclcpp::ok())
2     {
3       // calculate measured period
4       auto const current_time = cm->now();
5       auto const measured_period = current_time - previous_time;
6       previous_time = current_time;
7       // execute update loop
8       cm->read(cm->now(), measured_period);
9       cm->update(cm->now(), measured_period);
10      cm->write(cm->now(), measured_period);
11      // wait until we hit the end of the period
12      next_iteration_time += period;
13      std::this_thread::sleep_until(next_iteration_time);
14    }
```

from controller_manager/ros2_control_node.cpp

Organisation complète



La repository **ros2_control_demos**

- Exemple 1 : Revolute Revolute Bot (RRBot) / rviz et le joint_trajectory_controller. Contrôle en position.
- Exemple 2 : Exemple de robot différentiel.
- Exemple 3 : RRbot avec de multiples interfaces.
- Exemple 4 : RRbot avec un capteur intégré.
- Exemple 5 : RRbot avec un capteur externe.
- Exemple 6 : RRbot avec deux actionneurs indépendants.
- Exemple 7 : Multiples Robots (TBA)
- Exemple 8 : RRbot avec des transmissions.

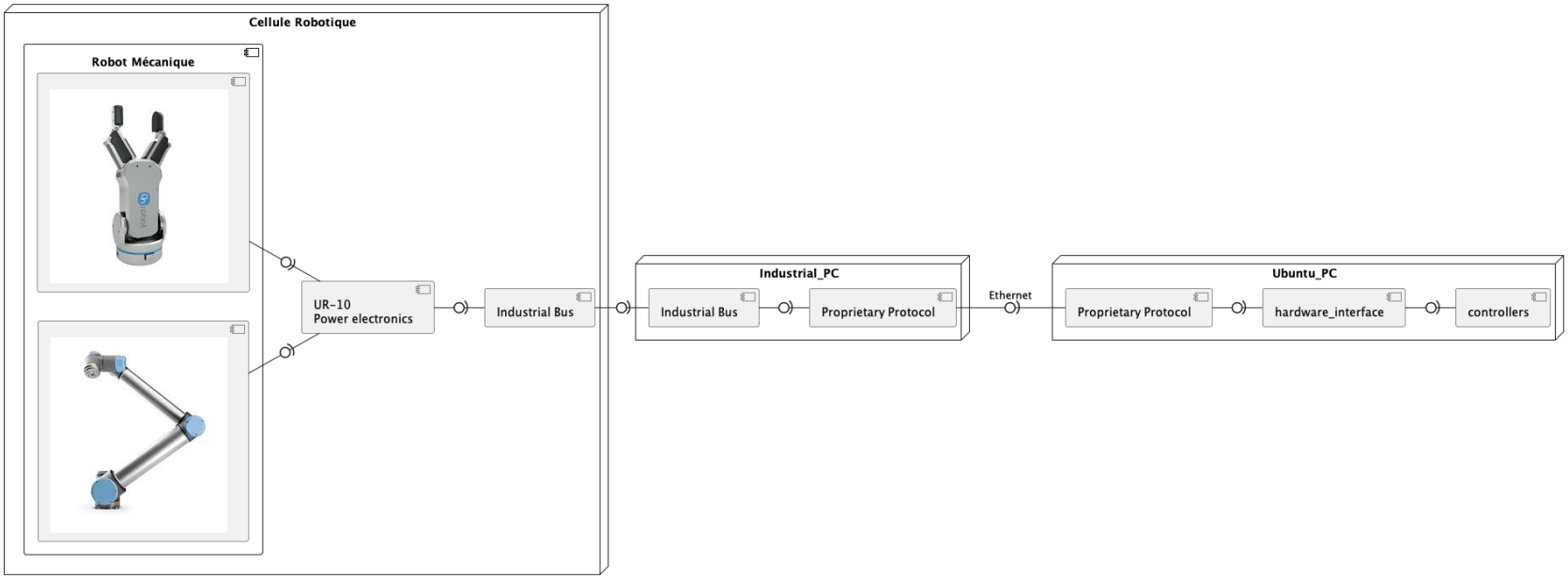
Table des matières

1 Introduction

2 Hardware
- Exemples
- Robot prototype
- Resource Manager
- Hardware components
- Exemples

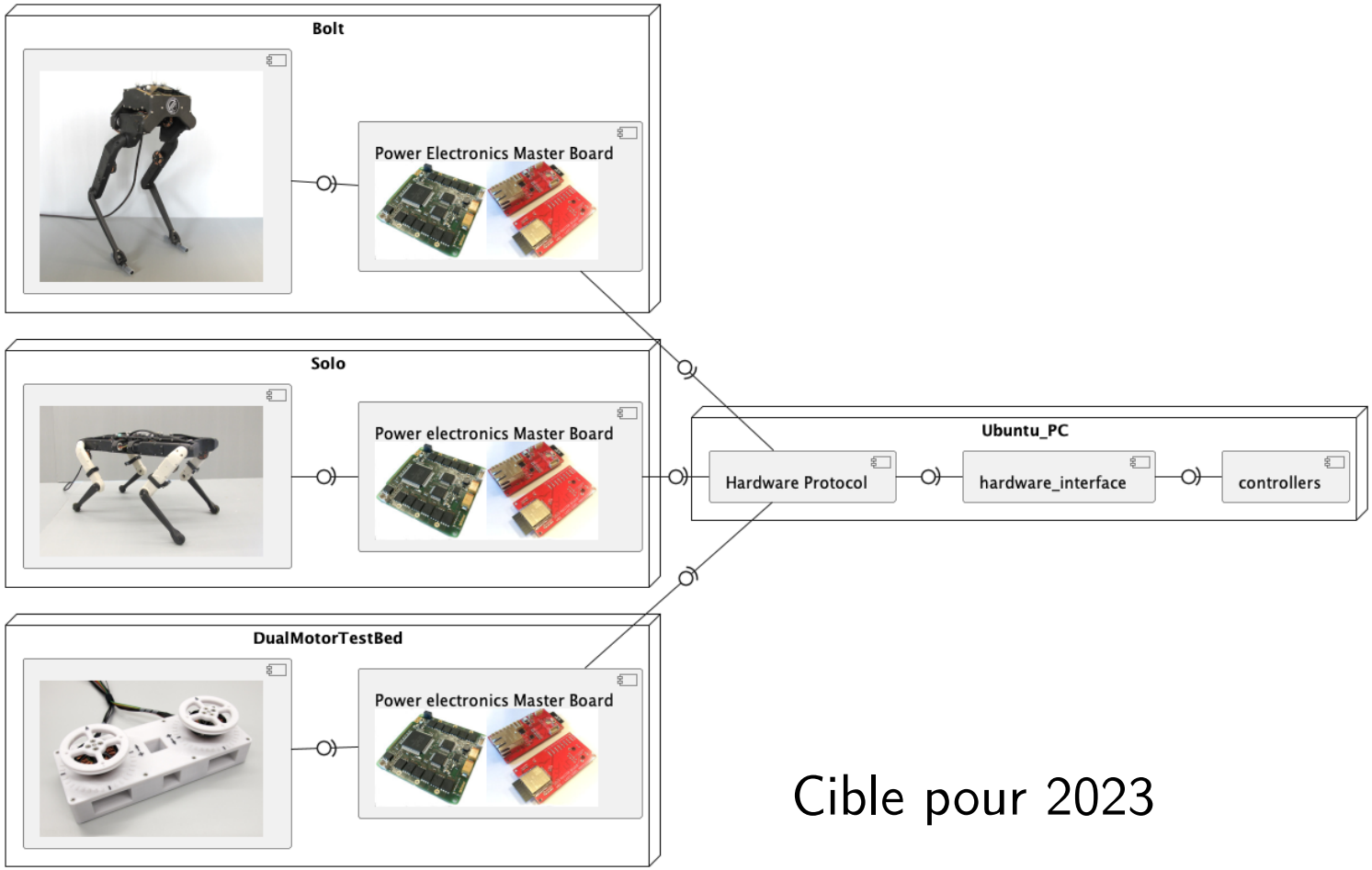
3 Controllers

Exemples (1/2) : Robot industriel



Probablement le premier cas d'utilisation du système (Panda, UR-e-Series)

Exemples (2/2) : Prototypes de Robot dans Gepetto



Cible pour 2023

Resource Manager (RM)

Définition

- Abstrait le matériel physique et ses pilotes (appelés composants matériels)
- RM charge les plugins en utilisant la librairie pluginlib, gére leurs états, leurs cycles et les commandes interfaces.
- RM permet de réutiliser les plugins du matériel
- RM permet l'indépendence du matériel et du logiciel

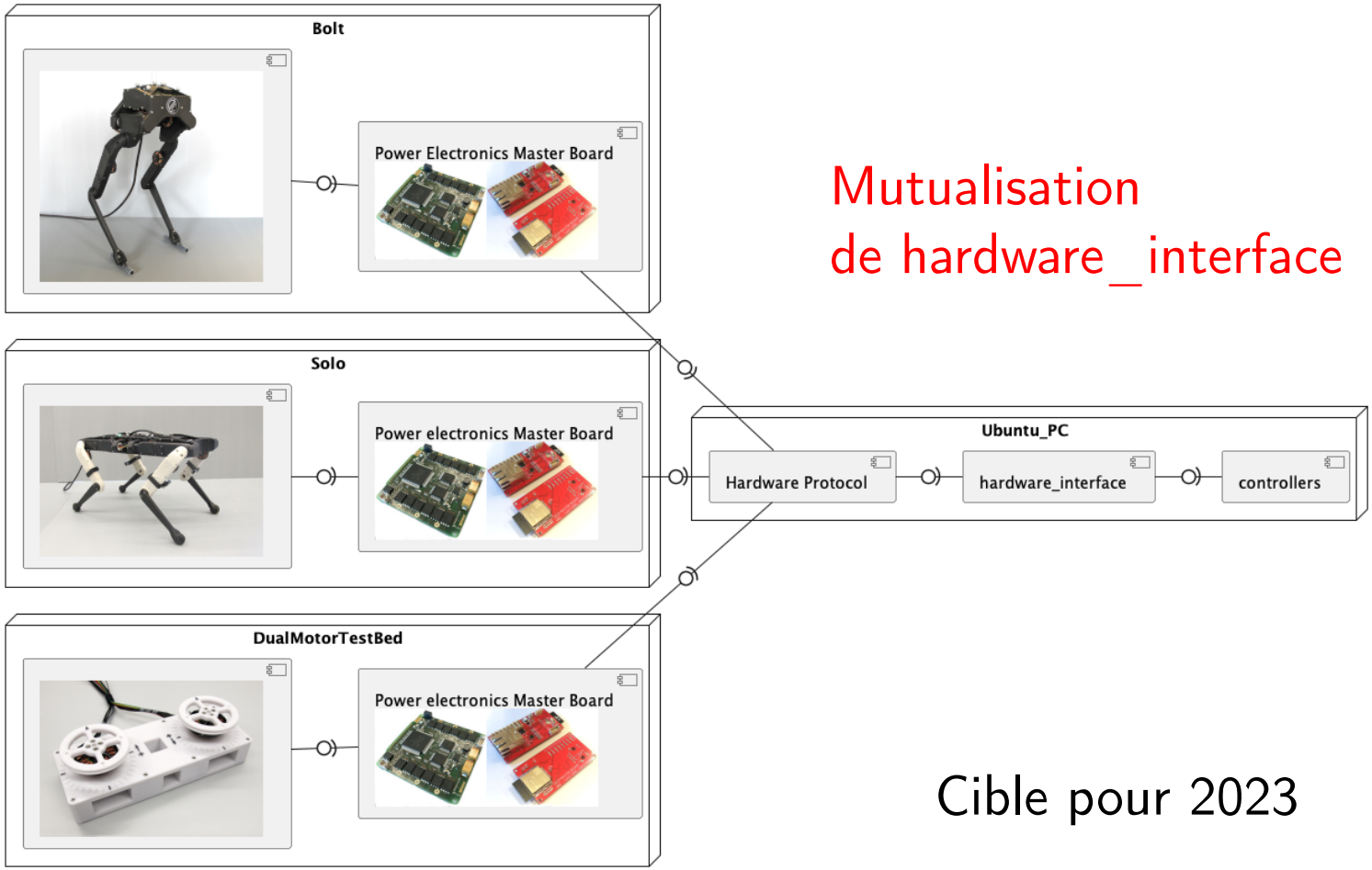
Composant Matériel

- System Matériel robotique complexe avec de multiples degrés de liberté comme les robots industriels.
  La principale différence avec le composant actuator est l'utilisation de transmissions complexes comme les mains des robots industriels. Le composant a des capacités de lecture et d'écriture.
  Il est utilisé lorsqu'il n'y a qu'un seul canal de communication au matériel.

Composant Matériel

- Sensor Matériel utilisé pour mesurer l'environnement.
  Un composant capteur est lié à un joint (i.e. encodeur) ou un corps (i.e. capteur de force-couple).
  Ce type de composant ne peut que lire.
- Actuator Matériel avec un seul degré de liberté comme les moteurs ou les valves. L'implémentation d'un actionneur n'est relié qu'à un seul joint.
  Ce type de composant peut lire et écrire.
  La lecture n'est pas obligatoire. Ce type de matériel peut-être utilisé sur un robot à plusieurs DOFs si le matériel permet une conception modulaire par exemple une CAN communication pour chaque moteur indépendamment.

Exemple : Un prototype de robot



Mutualisation
de hardware_interface

Cible pour 2023

Structure logicielle

1. L'électronique de puissance est gérée par deux librairies :
   - **master-board**,
   - **odri_control_interface**,
2. La couche ros2-control partagée
   ros2_hardware_interface_odri C'est le paquet mutualisé !
3. L'encapsulation dans le robot bolt **ros2_control_bolt**
4. L'encapsulation dans le banc de test
   odri_dual_motor_testbed_robot

Table des matières

1 Introduction

2 Hardware

3 Controllers
- Introduction
- Les observateurs
- Toolboxes
- Simulateurs
- Conclusion

Contrôleur

Définition

- Les contrôleurs comparent une valeur de référence et une valeur mesuré et ils en déduisant l'entrée du système.
- Les contrôleurs sont dérivés de ControllerInterface et exportés comme des plugins en utilisant la librairie pluginlib.
- Le cycle de vie du contrôleur est basé sur la classe LifecycleNode et implémente la machine d'état comme décrite dans rclcpp : :

Liste de contrôleurs

- Contrôleur en admittance
- Contrôleur d'un entraînement différentiel
- Contrôleurs en effort
- Contrôleur par anticipation
- Contrôleurs des effecteurs
- Contrôleur de trajectoire
- Contrôleurs en position
- Contrôleur pour les tricycles
- Contrôleurs en vitesse

Les broadcasters

Définition

Les observateurs ou *broadcaster* ont pour but de transmettre via DDS les informations capteurs.

Exemple

Liste de broadcasters

- Le plus célébre est le JointStatePublisher.
- ForceTorqueSensorBroadcaster
- IMUSensorBroadcaster

Interface utilisateur

- Les utilisateurs interagissent avec l'architecture ros2_control en utilisant les services du **Controller Manager**.
  - ConfigureController **name** *bool*
  - ListControllerTypes *types base classes*
  - ListControllers *ControllerState*
  - ListHardwareComponents *HardwareComponentState*
  - ListHardwareInterfaces *command_interfaces state_interfaces*
  - LoadController **name** *bool*
  - ReloadControllerLibraries **force_kill** *bool*
  - SetHardwareComponentState **name target_state** *ok state*
  - SwitchController **activate_controllers** **deactivate_controllers** *bool*
  - UnloadController **name** *bool*

Les boîtes à outils (1/2)

- control_toolbox
  - **dither** : Classe C++ qui génére un bruit blanc.
  - **filters** : Filtre de lissage exponentiel.
  - **pid** : Classe C++ fournissant un contrôleur PID.
  - **pid_ros** : Même pid mais les paramètres des PIDs sont modifiables par des paramètres ROS.
  - **limited_proxy** : Simule la dynamique du second ordre d'un corps rigide.
  - **sine_sweep** : Signal sinusoidal dont la fréquence est augmentée exponentiellement de ω_1 à ω_2 en T secondes.
  - **sinusoid** : Cette classe calcule la sortie pour un signal sinusoïdal et ses dérivées, pour une amplitude, une phase, une fréquence et un décalage donnés.
- realtime_tools

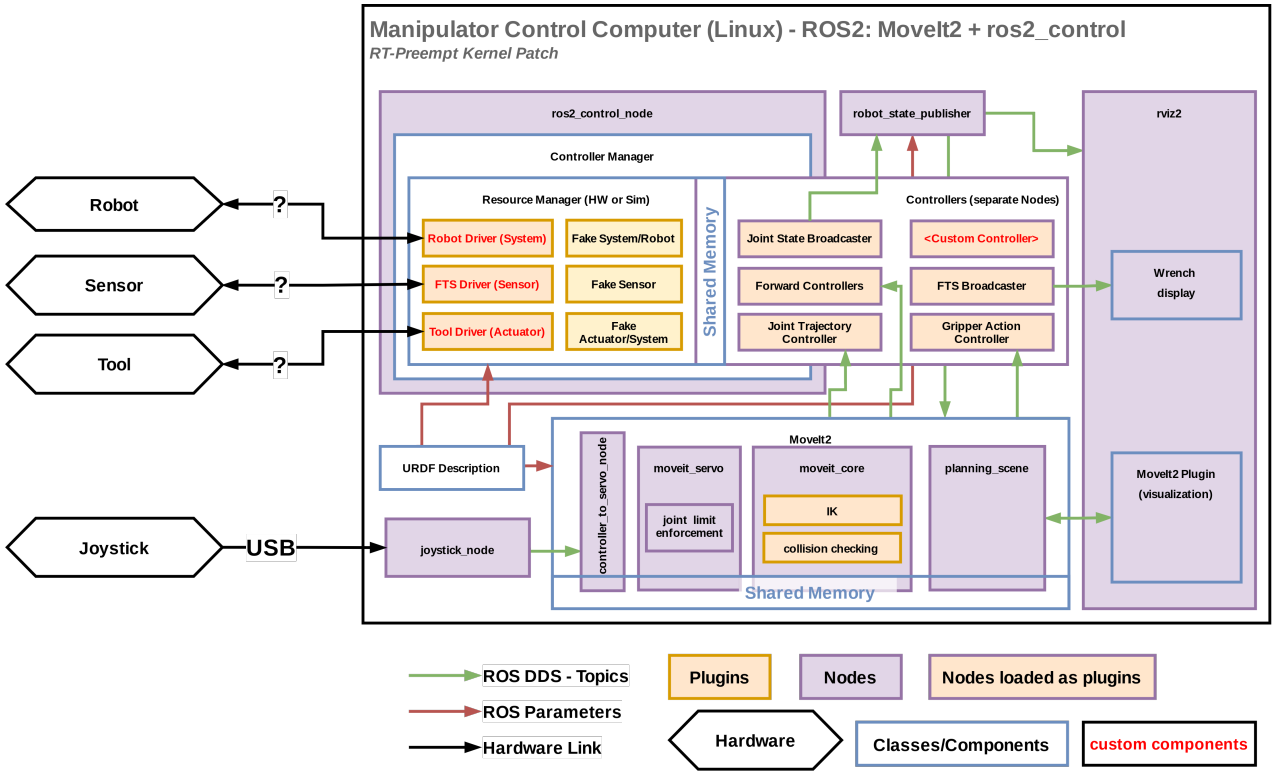
Les boîtes à outils (2/2)

- control_toolbox
- realtime_tools
  - **realtime_box** : Une boîte en C++ pour stocker et lire une donnée protégée par mutex.
  - **realtime_buffer** : Un buffer protéger par mutex pour une lecture temps réel ou non.
  - **realtime_clock** : Une horloge compatible temps réel.
  - **realtime_publisher** : Publication de données protégée par mutex pour être compatible avec le temps réel.
  - **realtime_server_goal_handle** : Gestion des actions protégée par mutex et le cycle du node.
  - **thread_priority** : Si l'OS le permet spécifie une priorité temps-réel pour le thread de la boucle perception action.

Gazebo

- Gazebo Classique
  - gazebo_ros2_control.
  - Nombre limité d'interfaces : position, velocity, effort.
  - Nécessite de faire un fork si on ajouter des informations supplémentaires.
- GZ (Ignition pendant un temps)
  - gz_ros2_control.
  - Expérience limitée sur l'utilisation de Gazebo Garden.
  - Même problème que la partie classique.

Integration with MoveIt 2



Conclusion - Messages à retenir

- Avantages
  - Si vous connaissez ros_control vous allez vous sentir chez vous.
  - Extension simple des interfaces d'état et de commandes.
  - ros2_control permet de faire du *temps-réel* parce qu'il n'y pas de middleware dans la boucle la plus rapide.
  - Une forte communauté (ANF ROS-2 pour les ingénieurs CNRS).
- Inconvénients
  - L'accumulation des détails peut être compliqué à appréhender.
  - La documentation pourrait être améliorée.
  - Le chaînage des contrôleurs doit être amélioré.
  - Les API sont encore appelées à évoluer d'où une instabilité du code.
  - Pas de binding python, nouveau paquet en cours d'écriture.