# Développement sous ROS d’une tâche de contrôle d’un robot yaskawa avec intégration dans une cellule robotisée multi-constructeur

Présenté par: Andrea PEREZ

AIP PRIMECA

Tuteurs Professionnels: Michel Taix & Thierry Germa

# Plan

Introduction

ROS

Architecture ROS-I pour Yaskawa

Etude de la problématique & le travail effectué

Conclusions

# Introduction

AIP PRIMECA

*Atelier Inter-établissements de Productique et Pôle de Ressources Informatiques pour la MÉCAnique*

Smart

Systems.Manufacturing.Academics.Resources.Technologies

**Pôle Nord-Pas de Calais**
UPHF, Arts et Métiers campus Lille, Centrale Lille, Univ. Lille

**Pôle Ile de France**
UTC, CentraleSupélec, Arts et Métiers campus Paris, Supméca, UTT, Univ. Paris 13

**Pôle Lorraine**
Univ. Lorraine, Arts et Métiers campus Metz

**Pôle Franche-Comté**
ENSMM, Univ. Franche-Comté, UTBM, Univ. Bourgogne

**Pôle Pays de la Loire**
Univ. Nantes, Centrale Nantes, Arts et Métiers campus Angers

**Pôle Rhône-Alpes Ouest**
INSA Lyon, ECAM, Centrale Lyon, Univ. Lyon 1, Univ. Lyon 2, Lyon 3, ISTP

**Pôle Grenoble-Alpes**
Grenoble INP, Univ. Grenoble-Alpes, Univ. Savoie-Mont Blanc

**Pôle Occitanie**
Univ. Toulouse 3 - Paul Sabatier, INSA Toulouse, Toulouse INP

**Pôle Auvergne**
SIGMA Clermont, Univ. Clermont-Auvergne

**Pôle PACA**
Univ. Toulon, Arts et Métiers campus Aix

# Introduction

AIP PRIMECA

*Atelier Inter-établissements de Productique et Pôle de Ressources Informatiques pour la MÉCAnique*

RIO

*Ros In Occitanie*

# Introduction

AIP PRIMECA

*Atelier Inter-établissements de Productique et Pôle de Ressources Informatiques pour la MÉCAnique*

RIO

*Ros In Occitanie*

## Cobot Yaskawa HC10

Nombre d'axes: 6

Poids: 47Kg

Plage de travail: 1.2m

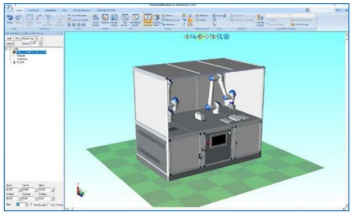
Charge admissible: 10 Kg

Options de montage: Sol, Mur, Plafond


Contrôleur Yrc1000

Boîtier de commande

Env. de simulation



Des outils

# Introduction

AIP PRIMECA

*Atelier Inter-établissements de Productique et Pôle de Ressources Informatiques pour la MÉCAnique*

RIO

*Ros In Occitanie*

ROS

Robot Operating System

## Yaskawa HC10


## Cellule robotisée

# Plan

Introduction

ROS

Architecture ROS-I  pour Yaskawa

Etude de la problématique & le travail effectué

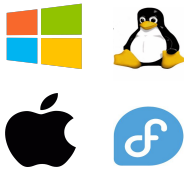
Conclusions

# ROS

*Robot Operating System*

Considéré comme un méta-système d'exploitation

**Système d'exploitation** & **Middleware**



- Abstraction du matériel
- Contrôle des périphériques de bas niveau
- Gestion de la concurrence

- Transmission de messages synchrones ou asynchrones dans une architecture de communication inter-processus et inter-machine


# ROS-I

*Robot Operating System* *Industrial*

ROS-Industrial est un projet open-source qui étend les capacités avancées des logiciels ROS au matériel et aux applications industrielles pertinentes.

- Favorise l'interopérabilité
- Il fournit des bibliothèques modulaires encapsulées
- Il assure la supervision et la structure du développement libre de logiciels d'automatisation de la fabrication.

# Plan

Introduction

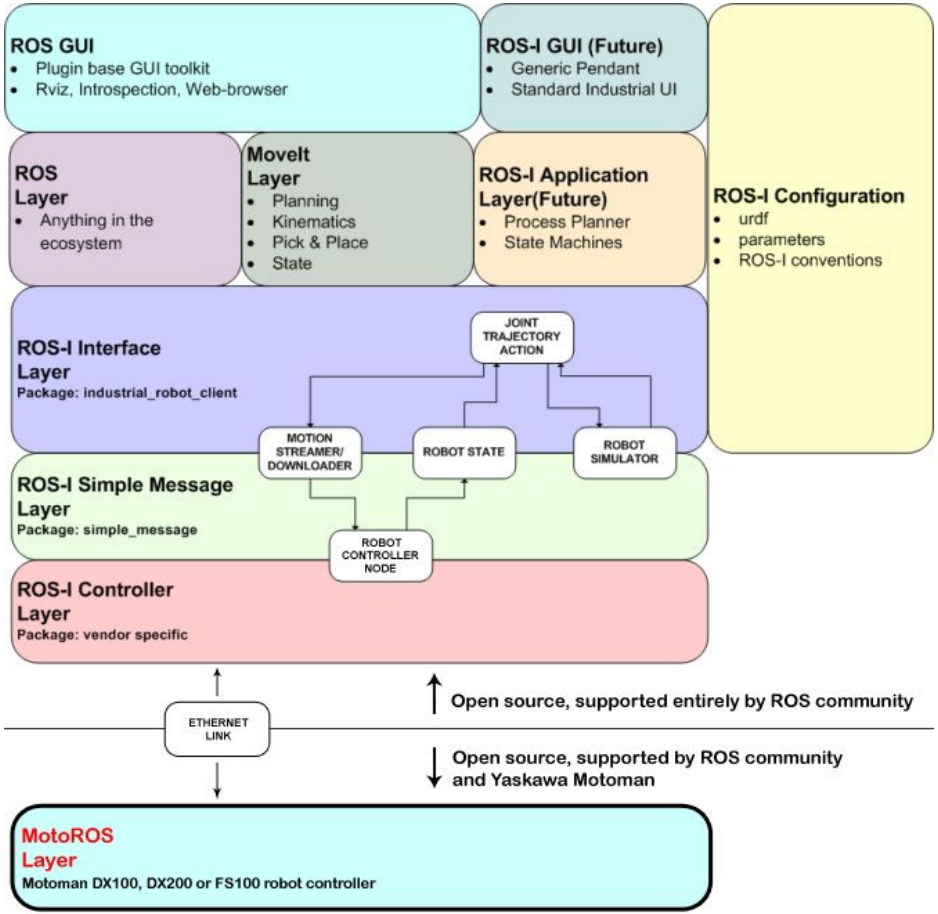
ROS

Architecture ROS-I pour Yaskawa

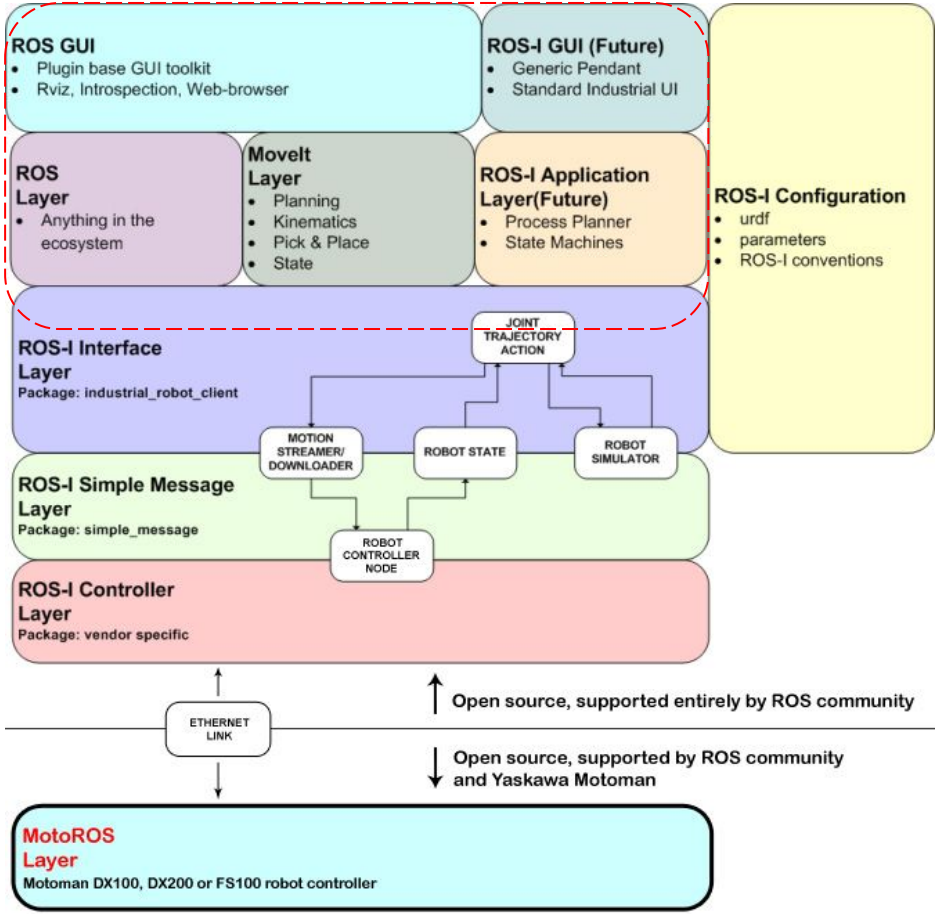
Etude de la problématique & le travail effectué

Conclusions

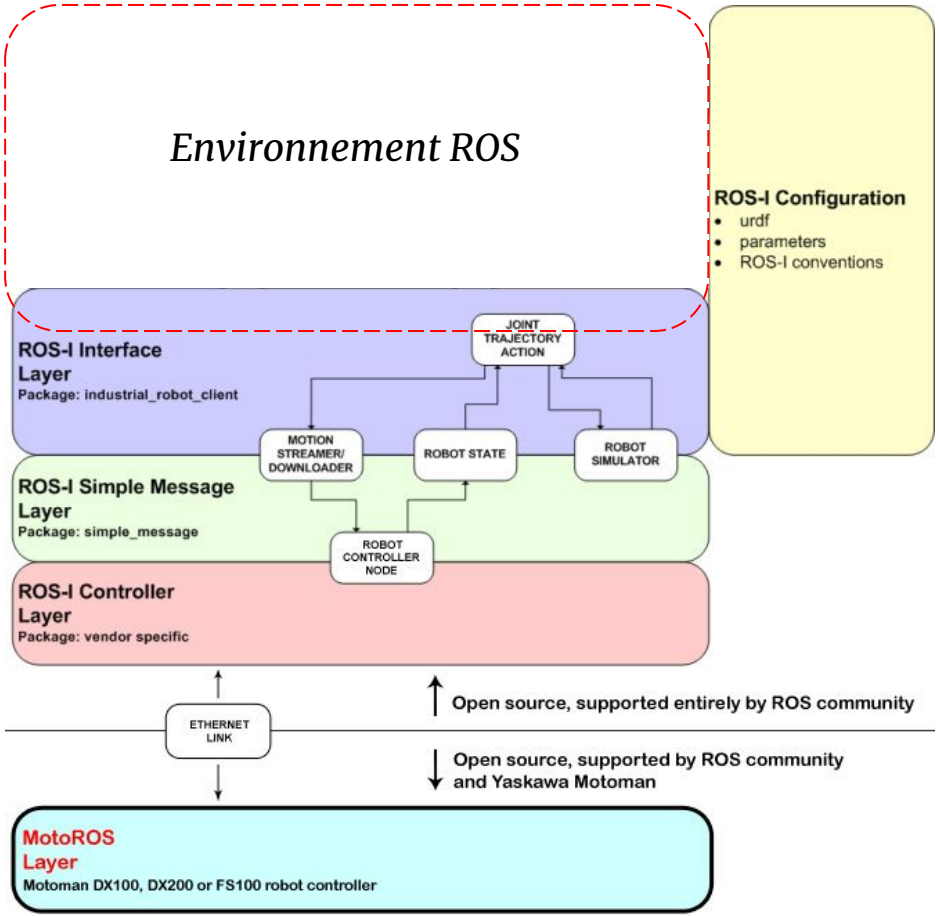
# Architecture ROS-I pour Yaskawa



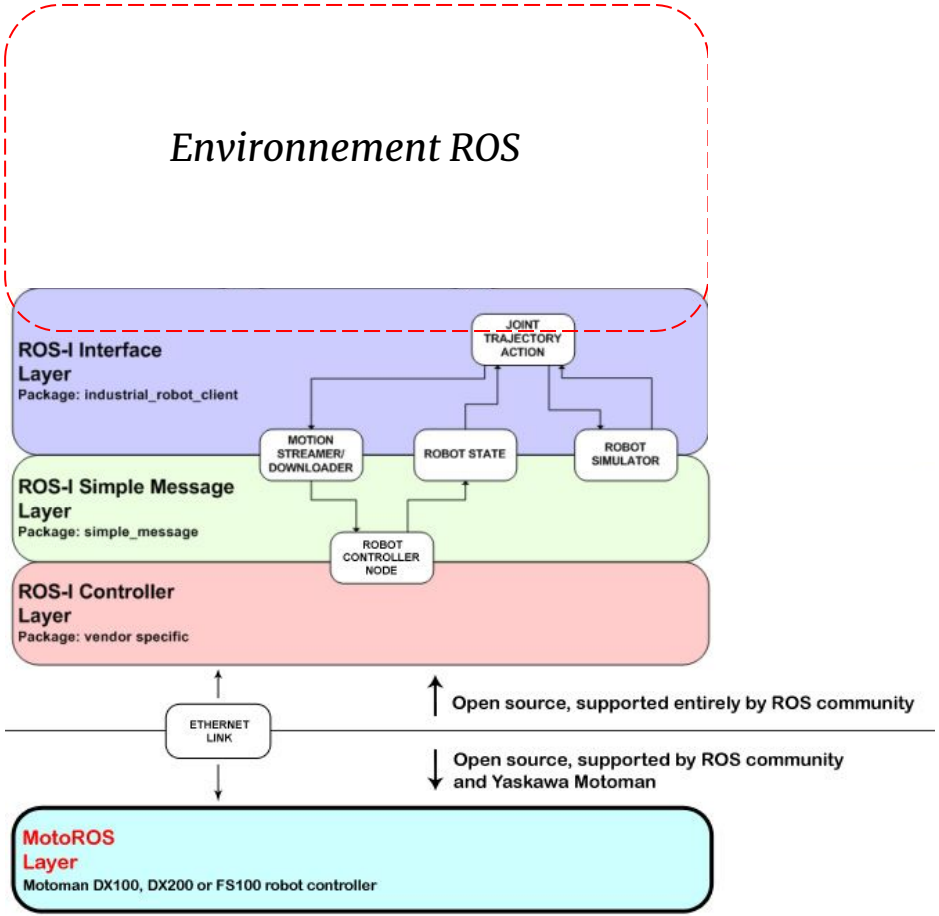
# Architecture ROS-I pour Yaskawa



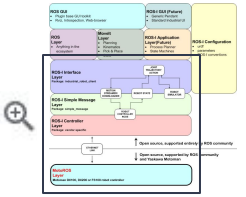
# Architecture ROS-I pour Yaskawa



# Architecture ROS-I pour Yaskawa

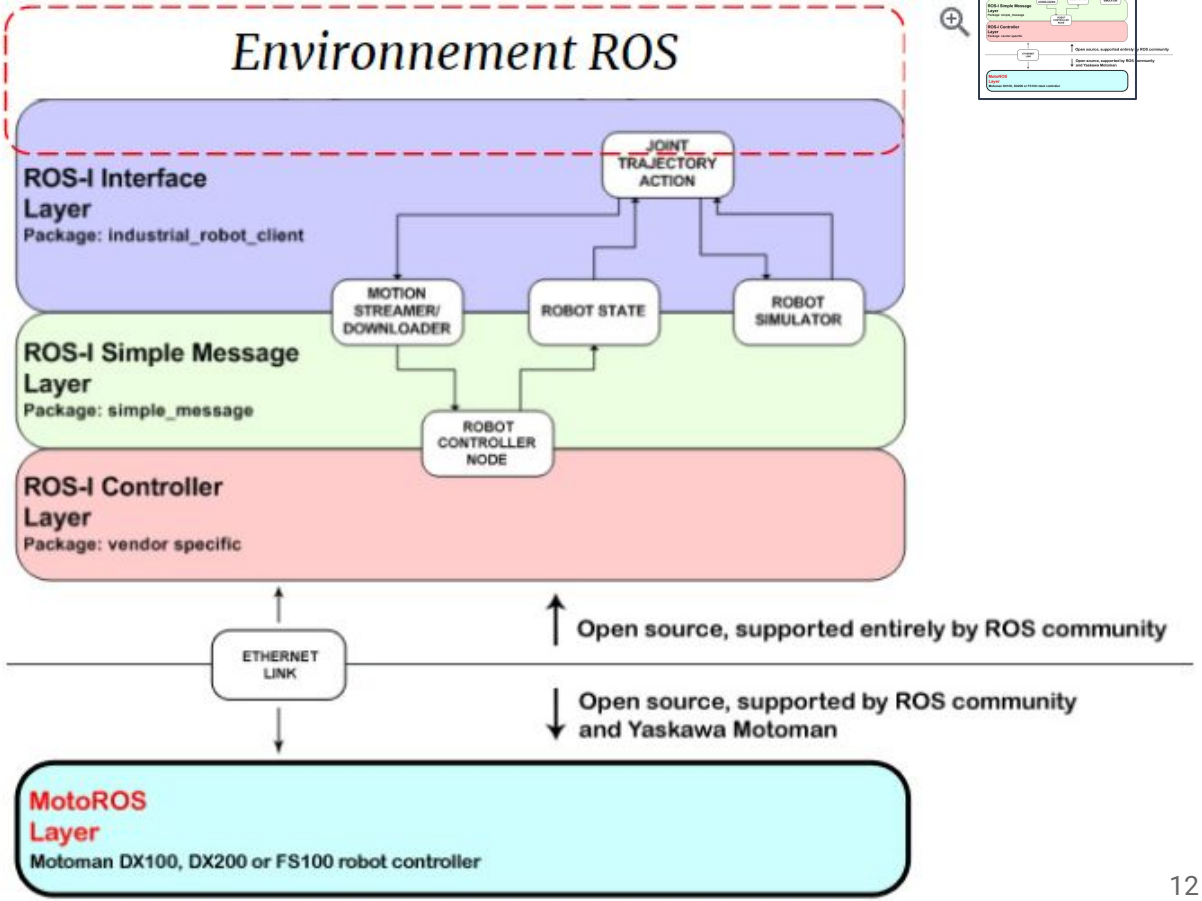


# Architecture ROS-I pour Yaskawa

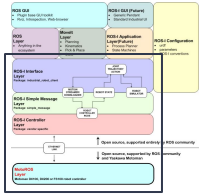


# Architecture ROS-I pour Yaskawa

Contrôleur YRC1000
avec MotoROS



# Architecture ROS-I pour Yaskawa



Protocole de Comm. ✉

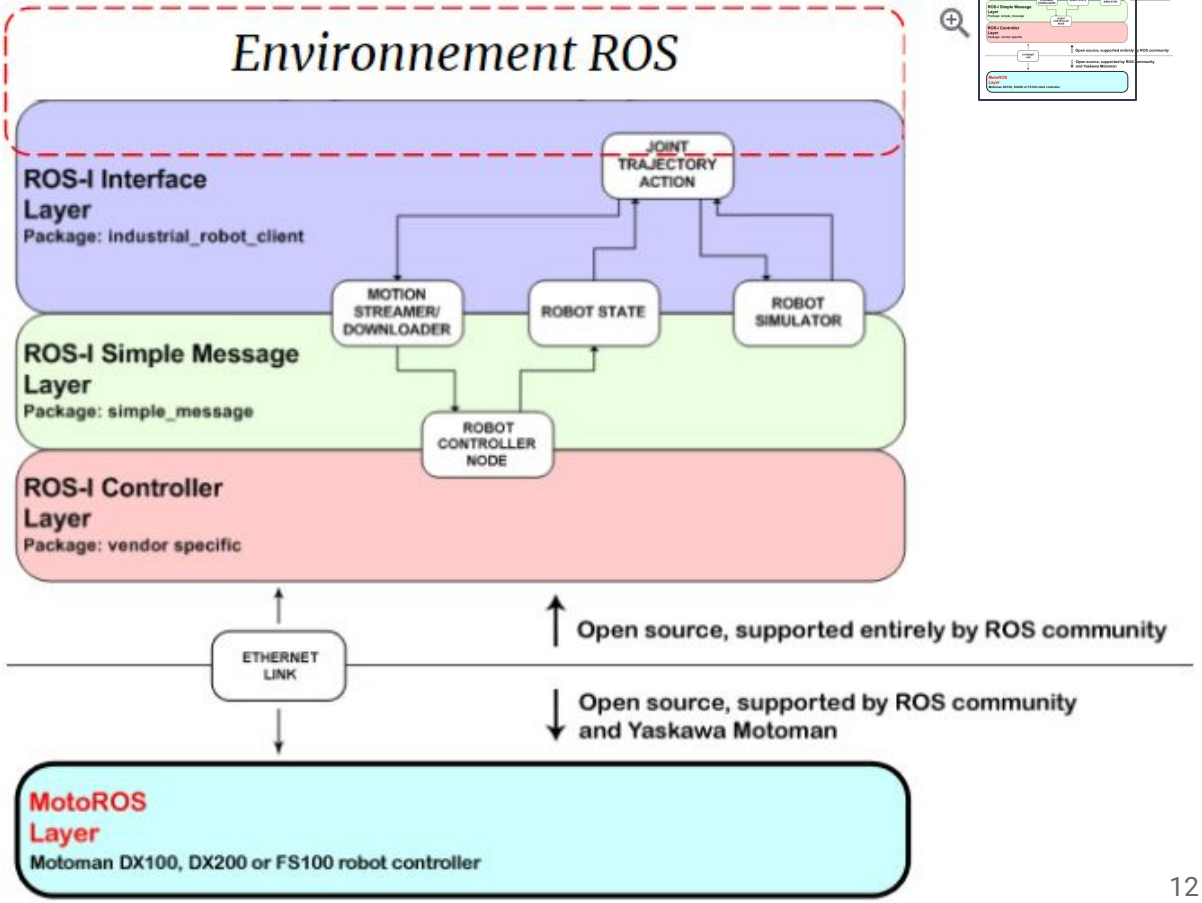
Contrôleur YRC1000 avec MotoROS

# Architecture ROS-I pour Yaskawa

Interface normalisée pour le contrôle des robots industriels

Protocole de Comm.

Contrôleur YRC1000 avec MotoROS

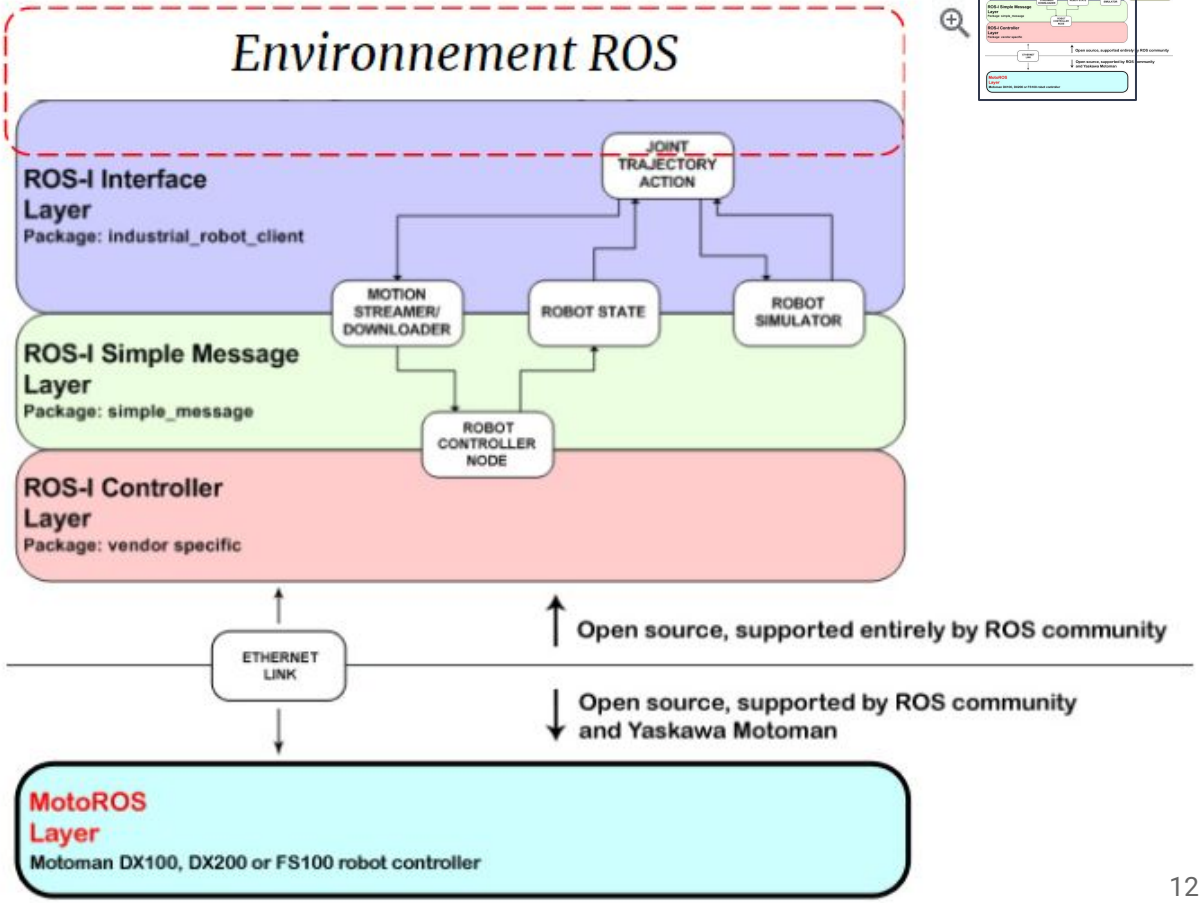


# Architecture ROS-I pour Yaskawa Simplifiée

Interface normalisée pour le contrôle des robots industriels

Protocole de Comm. ✉

Contrôleur YRC1000 avec MotoROS



# Plan

Introduction

ROS

Architecture ROS-I pour Yaskawa

Etude de la problématique & le travail effectué

Conclusions

# Etude de la problématique & travail effectué

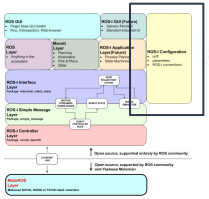
***Nous souhaitons contrôler le robot***

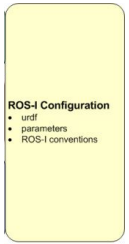
- Création de l'environnement de simulation
  *De façon à programmer et étudier le comportement du robot avant utilisation*

- Génération de trajectoire
  *Atteindre des points dans l'espace depuis une configuration quelconque*

- Retour en effort, position et vitesse
  *Pour avoir un aperçu global de l'état actuel du robot*

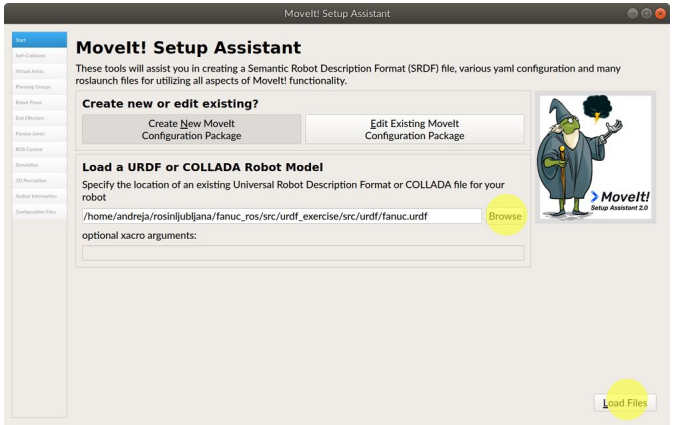
# Etude de la problématique & travail effectué

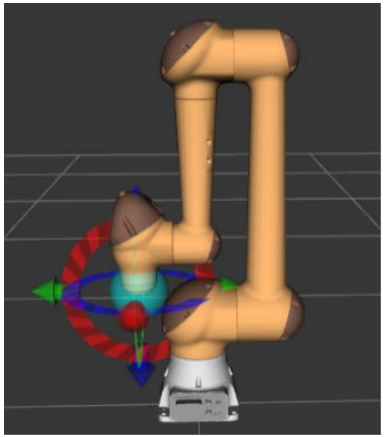


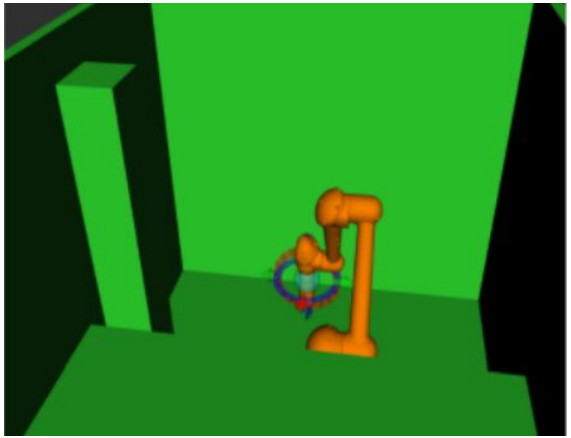



***Nous souhaitons contrôler le robot***

- Création de l'environnement de simulation
  *De façon à programmer et étudier le comportement du robot avant utilisation*





Avec l'axe X en rouge, l'axe Y en vert et l'axe Z en bleu.

# Etude de la problématique & travail effectué

***Nous souhaitons contrôler le robot***

- Génération de trajectoire

  *Atteindre des points dans l'espace depuis une configuration quelconque*

*A condition de ne pas avoir des limitations en vitesse déclarées du côté du contrôleur.*

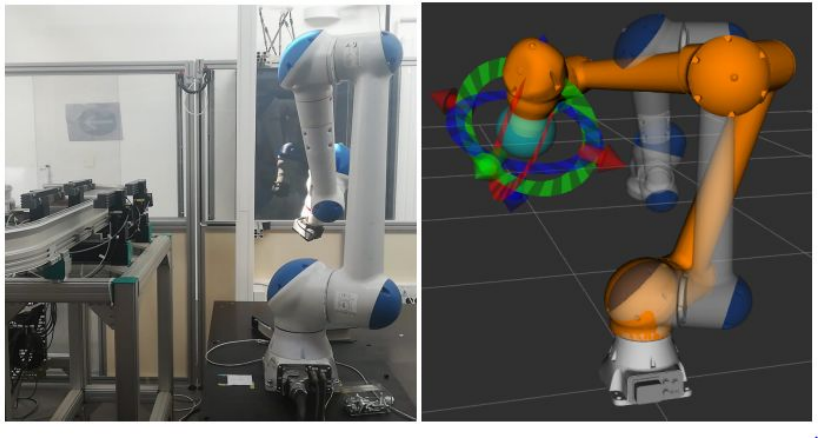
# Etude de la problématique & travail effectué

***Nous souhaitons contrôler le robot***

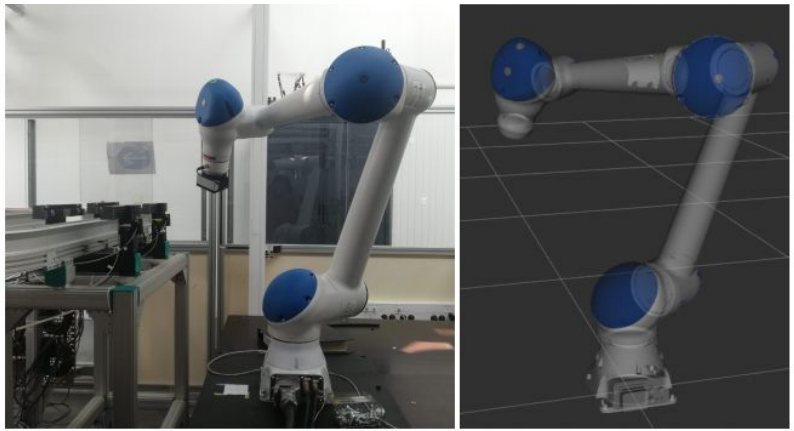
- Génération de trajectoire

  *Atteindre des points dans l'espace depuis une configuration quelconque*

Planification



Après l'exécution de la trajectoire

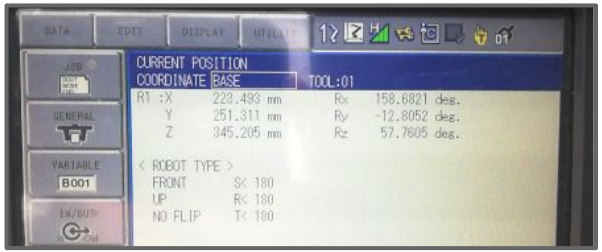


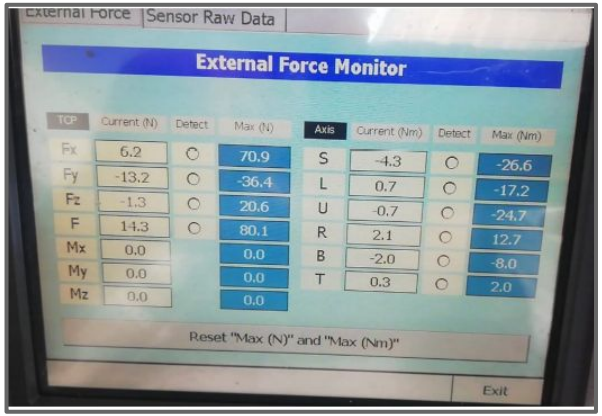
Retour en position et vitesse des 6 joints

# Etude de la problématique & travail effectué

***Nous souhaitons contrôler le robot***

- Retour en effort, position et vitesse
  *Pour avoir un aperçu global de l'état actuel du robot*

# Etude de la problématique & travail effectué

## *Nous souhaitons contrôler le robot*

- **Retour en effort**
  *Pour avoir un aperçu global de l'état actuel du robot*

Interface normalisée pour le contrôle des robots industriels

Protocole de Comm. ✉

Contrôleur YRC1000 avec MotoROS

**Message**

| Préfixe | En-tête | Corps |
|---|---|---|

- Préfixe = Longueur(En-tête + Corps)
- En-tête =

| msg_type | comm_type | reply_code |
|---|---|---|

- Corps = Données

*Le **préfixe**, somme en octets de l'en-tête et du corps. Agit comme une somme de contrôle qui vérifie l'intégrité du message envoyé.*

*L'**en-tête**, divisé en trois champs qui spécifient le type de message envoyé.*

- ***msg_type**: l'identifiant du message.*
- ***comm_type**: indique le type de flux des données.*
- ***reply_code**: avertit si l'information a été reçue.*

*Le **corps**, ce sont les données.*

# Etude de la problématique & travail effectué

## *Nous souhaitons contrôler le robot*

- Retour en effort

*Pour avoir un aperçu global de l'état actuel du robot*

Interface normalisée pour le contrôle des robots industriels

Protocole de Comm. ✉

Contrôleur YRC1000 avec MotoROS

Message

| Préfixe | En-tête | Corps |
|---|---|---|

- Préfixe = Longueur(En-tête + Corps)
- En-tête =

| msg_type | comm_type | reply_code |
|---|---|---|

- Corps = Données

*Le **préfixe**, somme en octets de l'en-tête et du corps. Agit comme une somme de contrôle qui vérifie l'intégrité du message envoyé.*

*L'**en-tête**, divisé en trois champs qui spécifient le type de message envoyé.*

- ***msg_type**: l'identifiant du message.* ✖
- ***comm_type**: indique le type de flux des données.*
- ***reply_code**: avertit si l'information a été reçue.*

*Le **corps**, ce sont les données.*

# Etude de la problématique & travail effectué

***Nous souhaitons contrôler le robot***

- Retour en effort, position et vitesse
  *Pour avoir un aperçu de l'état actuel du robot*

Serveur

Contrôleur YRC1000

Je veux lire les registres X

requête

réponse

Je renvoie l'information contenue dans le registres X

Client

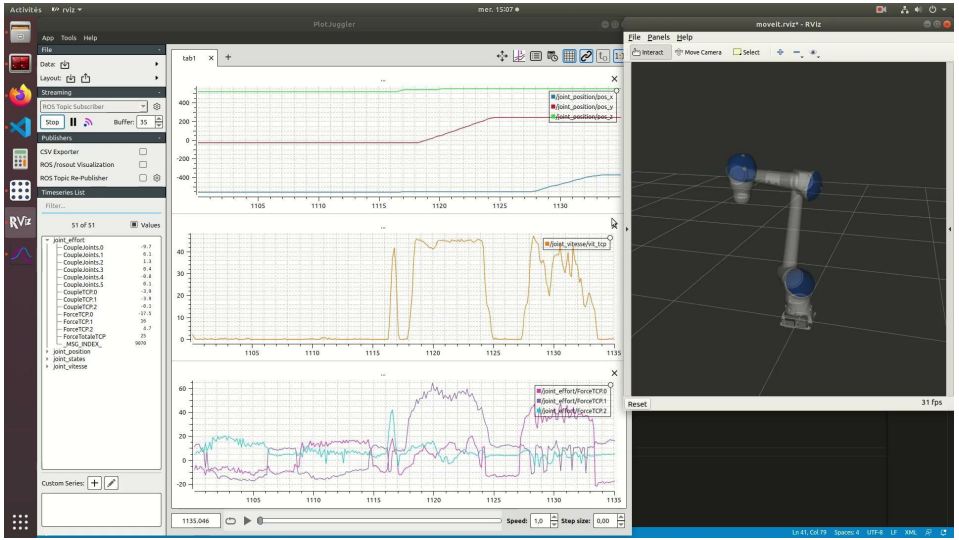
ROS
Robot Operating System

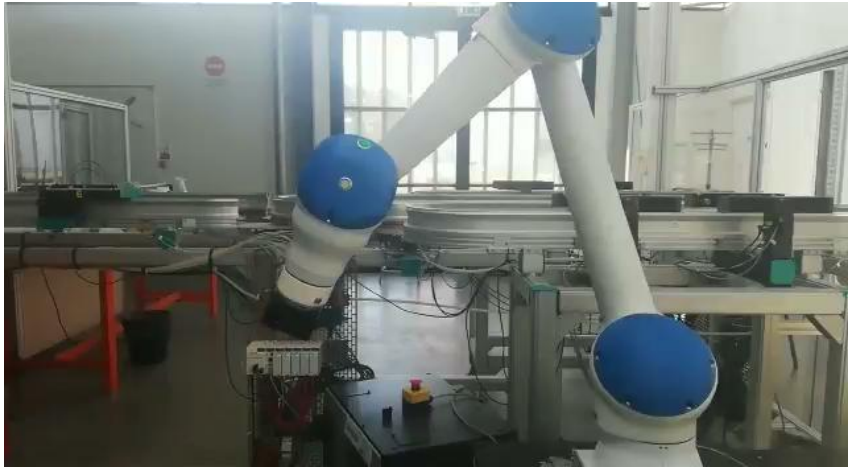
Ordinateur

# Etude de la problématique & travail effectué

***Nous souhaitons contrôler le robot***

- Retour en effort, position et vitesse
  *Pour avoir un aperçu de l'état actuel du robot*



# Etude de la problématique & travail effectué

***Nous souhaitons contrôler le robot***

- Création de l'environnement de simulation
  *De façon à programmer et étudier le comportement du robot avant utilisation*

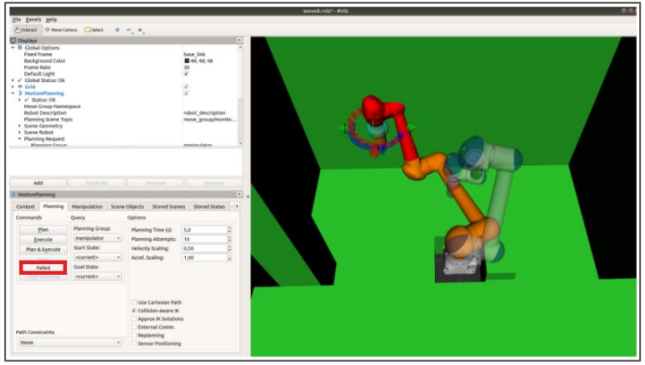
- Génération de trajectoire
  *Atteindre des points dans l'espace depuis une configuration quelconque*

- Retour en effort, position et vitesse
  *Pour avoir un aperçu global de l'état actuel du robot*

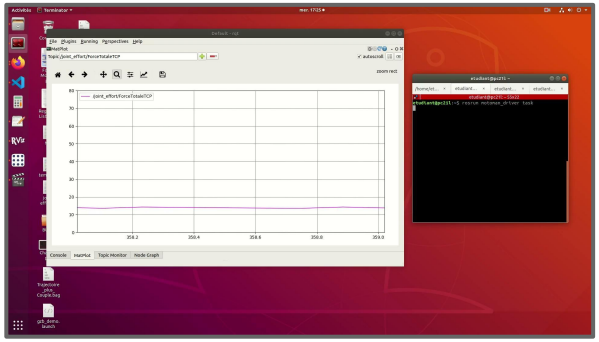
# Etude de la problématique & travail effectué

***Nous souhaitons contrôler le robot***

Évitement d'obstacles



Prise des points de passage

Chien de garde

# Plan

Introduction

ROS

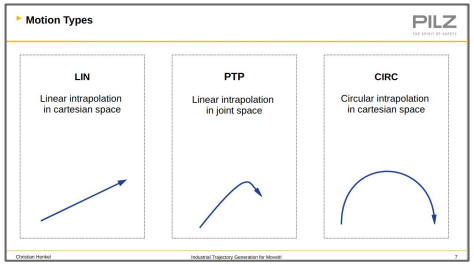
Architecture ROS-I pour Yaskawa

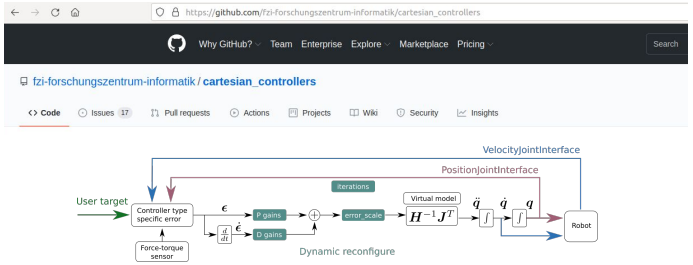
Etude de la problématique & le travail effectué

Conclusions

# Conclusions

- Nous sommes capables de contrôler un robot industriel sans avoir besoin de son environnement natif à l'aide de ROS
- L'utilisation des logiciels Open Source représente un véritable avantage pour le développement des nouvelles technologies.
- ROS comme plateforme pour le prototypage rapide d'applications robotiques avancées




- Cette expérience a été très enrichissante pour moi et correspond entièrement à mon projet professionnel

# Perspectives

- Développement d’une tâche industrielle
- Intégrer le paquet au répertoire Github de L’AIP-Primeca pour des futurs utilisateurs
- Intégration d’un capteur de précision 3D

# Merci

# Références


- *https://www.aip-primeca-occitanie.fr/*
- *https://s-mart.fr/*
- *https://www.yaskawa.fr/Global%20Assets/Downloads/Brochures_Catalogues/Robotics/Applications/Robot%20Assistants_Collaborative%20Applications_E_06.2021.pdf*
- *https://www.ros.org/*
- *https://rosindustrial.org/*
- *http://wiki.ros.org/motoman_driver*
- *PEREZ Andrea - Rapport de Stage: Développement sous ROS d'une tâche de contrôle d'un robot yaskawa avec intégration dans une cellule robotisée multi-constructeur.*