YASKAWA
PARTNER SUPPORT
Shared Integration Experience

This document captures ideas, experiences, and informal recommendations from the Yaskawa Partner Support team. It is meant to augment – not supersede manuals or documentation from motoman.com. Please contact the Partner Support team at partnersupport@motoman.com for updates or clarification.

What is ROS?

Summary

Robot Operating System (ROS) is free software that allows programmers to develop industrial automation solutions that include Yaskawa Motoman (and other OEM) robots. The design of ROS allows the programmer to easily incorporate advanced capabilities (such as AI and vision) from any ROS-compatible product. The ROS application runs on a PC which is running Linux. Yaskawa Motoman develops and supports ROS drivers that can be installed on our controllers.

ROS – Industrial (ROS-I) is an extension to core ROS functionality to make it better for use in industrial automation.

There is currently a steep learning curve associated with using ROS. However, the benefits of this approach and openness to incorporating capabilities from academia or business will likely lead to improvements in usability and availability of pre-packaged solutions based on ROS.

Introduction

You have probably heard of ROS (Robot Operating System). You may even have done some reading to try to understand how it might apply to your projects. There is a wealth of information available via the Internet, books, seminars, and presentations. Unfortunately, the explanations sometimes use terminology that needs more explanation! It can be hard to grasp what it all really means, how it applies to industrial automation and why people are excited about it.

The purpose of this document is to try to explain *in simple terms*, "What is ROS?" and help you answer the question "Why should I care?" We will quote some of the industry web sites and then unpack the language to try to make it more understandable. You should come away with a better understanding of why there is excitement around ROS and what it may mean to you.

Why ROS?

Before getting into what is ROS, it is helpful to set the stage by covering *why* it was created.

"In 2013, the sales of industrial robots in North America were strong, but the diversity of robot tasking remained utterly STAGNANT. For more than 20 years, industrial robots have been limited to roughly the same set of tasks – welding, material handling and dispensing – because robots are cost-effective for repetitive and high-volume tasks but are not cost-effective for lower volume mixed part production. Yet over the same 20+ years, governments and research institutions

have spent more than $1 billion on robotics research to enhance robot behaviors, such as those for acting on complex perception data, moving about on mobile platforms and collaborating with humans.

This research investment is largely languishing in labs in part because the limited software architectures of current industrial robots present a barrier to the transition of research products. As a result, it simply costs too much to apply these advanced capabilities to improve industrial productivity. Robots are flexible enough to accomplish a wide variety of automation tasks, but until the return on investment to integrate and program robots for agile processing becomes attractive, robots will continue to be deployed in only limited applications, and opportunities for greater productivity will be lost. Advances in robotics research require economical pathways to application to realize their full potential. ROS-Industrial seeks to provide this pathway." (The Challenge - ROS Industrial, n.d.)

OK, great. What does that mean? It means there are literally billions of dollars being spent to create better software for use with robots, but it cannot easily be used. It is too hard to make it work with the diverse software used by different robot manufacturers and all the peripheral equipment we want to work with. Without some sort of *middleman* to translate how the new software wants to talk to the myriad of old, new, and changing software designs, we are stuck. Enter ROS - Robot Operating System.

What is ROS?

According to Wikipedia, *"Robot Operating System (ROS or ros) is robotics middleware (i.e., collection of software frameworks for robot software development). Although ROS is not an operating system, it provides services designed for a heterogeneous computer cluster such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management."* (Robot Operating System, n.d.)

Here is the middleman we need! A little unpacking of the language used above may help.

Middleware – In software architectures, this is the code that bridges the gap between other pieces of code. In other words, a middleman to translate and manage information being sent between different parts of a system.

Heterogeneous computer cluster – This is just a bunch of different computing devices networked together. "Heterogeneous" leads you to believe they are from different vendors and / or not normally meant to work together.

Hardware abstraction – a technique to hide the differences between the same kind of hardware that works differently because it is from different vendors or is a different hardware model. Hardware abstraction is what allows you to click "print" on a Windows PC and have your document print correctly on different kinds of printers. Windows talks to a print driver. Printer manufacturers write drivers for their specific printers and Windows apps don't have to care whether they are printing to an HP or a Brother printer. Windows and the driver create hardware abstraction for the app.

Message passing – a method for different pieces of software or hardware to exchange information without having to know and manage all the details. It can actually be a big deal to do this, as different messages may have different priority, error checking requirements, error recovery rules, etc. If every component in the system does it differently, we get nowhere. ROS offers a standard way for software from different vendors to talk with each other.

Package Management – a set of tools to automate the process of installing, upgrading, configuring, and removing ROS software components. So, as ROS improves it is easy to adopt the improved code into our installation(s).

What does it cost?

Nothing.

ROS is "open source", meaning the source code is available to anyone to do with as they please. Anyone can embed it in a product for free, or even re-brand it and sell it. Updates are developed and maintained by volunteers.

YASKAWA
PARTNER SUPPORT
Shared Integration Experience

This document captures ideas, experiences, and informal recommendations from the Yaskawa Partner Support team. It is meant to augment – not supersede manuals or documentation from motoman.com. Please contact the Partner Support team at partnersupport@motoman.com for updates or clarification.

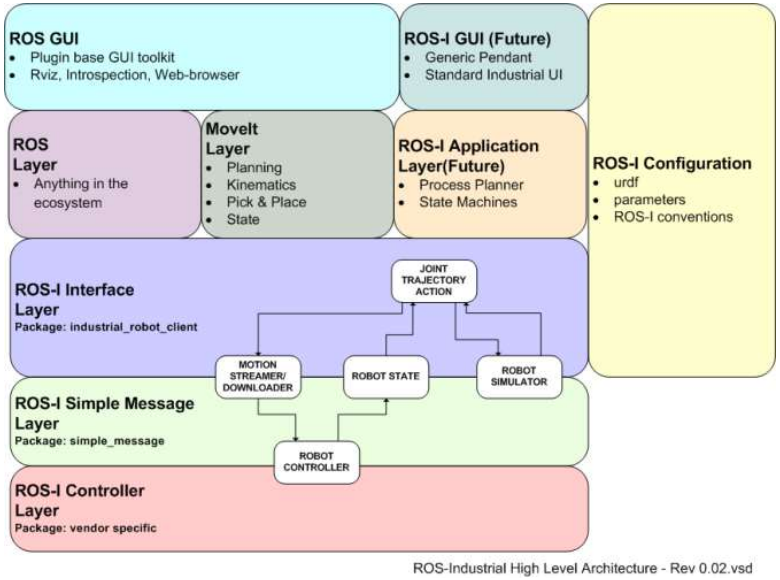
What is ROS-I?

ROS – Industrial is an extension to core ROS functionality to make it better for use in industrial automation.

From Wikipedia and rosindustrial.org: ROS-Industrial is an open-source project that extends the capabilities of ROS to manufacturing automation and robotics. The ROS-Industrial software repository includes interfaces for common industrial manipulators, grippers, sensors, and device networks. It also provides software libraries for automatic 2D/3D sensor calibration, process path/motion planning, applications like Scan-N-Plan, developer tools, and training curriculum that is specific to the needs of manufacturers. ROS-I is supported by an international Consortium of industry and research members.

The project began as a collaborative endeavor between Yaskawa Motoman Robotics, Southwest Research Institute, and Willow Garage to support the use of ROS for manufacturing automation. Currently, the Consortium is divided into three groups; the ROS-Industrial Consortium Americas (led by SwRI and located in San Antonio, Texas), the ROS-Industrial Consortium Europe (led by Fraunhofer IPA and located in Stuttgart, Germany) and the ROS-Industrial Consortium Asia Pacific (led by Advanced Remanufacturing and Technology Centre (ARTC) and Nanyang Technological University (NTU) and located in Singapore).

Below is a graphical representation of what ROS-I adds to the ROS core.



ROS-Industrial High Level Architecture - Rev 0.02.vsd

Note the standard components of ROS, surrounded by the ROS-I additions at the bottom and right side. Of special interest is the "urdf" bullet under ROS_I Configuration. urdf stands for "Unified Robot Description Format". It is a generic way to describe a robot (size, shape, number of axes, speed, range of motion, etc.) so that it can be used with ROS. Vendors (like Yaskawa Motoman) provide Controller software (bottom) and urdf files so that their equipment can get and send ROS messages. This provides some level of transparency between what your ROS-based application is designed to do and what equipment you choose to deploy. In theory, you could use ROS to develop a solution and mix and match the robots used later.

YASKAWA
PARTNER SUPPORT
Shared Integration Experience

This document captures ideas, experiences, and informal recommendations from the Yaskawa Partner Support team. It is meant to augment – not supersede manuals or documentation from motoman.com. Please contact the Partner Support team at partnersupport@motoman.com for updates or clarification.

What problem does ROS-I solve?

ROS-I solves many problems that will lead to growth of industrial robot use in new application (Industrial - ROS Wiki, n.d.) areas:

- It makes automation interoperable. Manipulators, end effectors, perception systems/sensors, mobility platforms, and peripherals can all speak one language (ROS messages) and interoperate regardless of OEM brands or communication bus.
- It provides advanced capabilities as encapsulated modular libraries for free (like ROS), so that the best technology can be disseminated without the impedance of proprietary vendor-lock, retraining, or price tag.
- It provides oversight and structure to the open-source development of manufacturing automation software. Code included in the project is reviewed and automatically assessed for quality metrics.
- It provides a conduit for academic research to be vetted and quickly implemented within industry.

Where can I learn more?

There is currently a steep learning curve associated with using ROS. Formalized, classroom training is strongly encouraged. Below are some links that provide additional information on ROS and ROS-I.

Top FREE tutorials to learn ROS – Robocademy

ROS for Beginners: How to Learn ROS | The Construct

ROS-Industrial

Motoman_driver - ROS Wiki

Motoman_driver/Tutorials/indigo/InstallServer - ROS Wiki