Robot Operating System

Jingyi Li

Introduction:

Developing a robot with a computer brain requires a bunch of software tools on the computer side—like software drivers, third party tools for computer vision and simulation tools. ROS framework gathers all these tools and manages how you develop a code for your robot. Robot Operating System is a flexible framework that designed for robot software development. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. Its usefulness is not limited to robots, but most tools provided are focused on working with peripheral hardware.

Typically, ROS consists of code and tools that can help users' project code run and do what it needs, including the infrastructure that runs it, such as messages that are passed between processes. ROS is designed as a loosely coupled system where one process is called a node and each node is responsible for a task. Messages are communicated between nodes using messages that are passed through a logical channel called a topic. Each node can send or retrieve data from other nodes using the publish/subscribe model. We will see its practical application later. The main goal of ROS is to support code reuse in robotics research and development so you can find a built-in package system. Again,

keep in mind that ROS is not an OS, library, or RTOS. It is a framework for using operating system concepts.

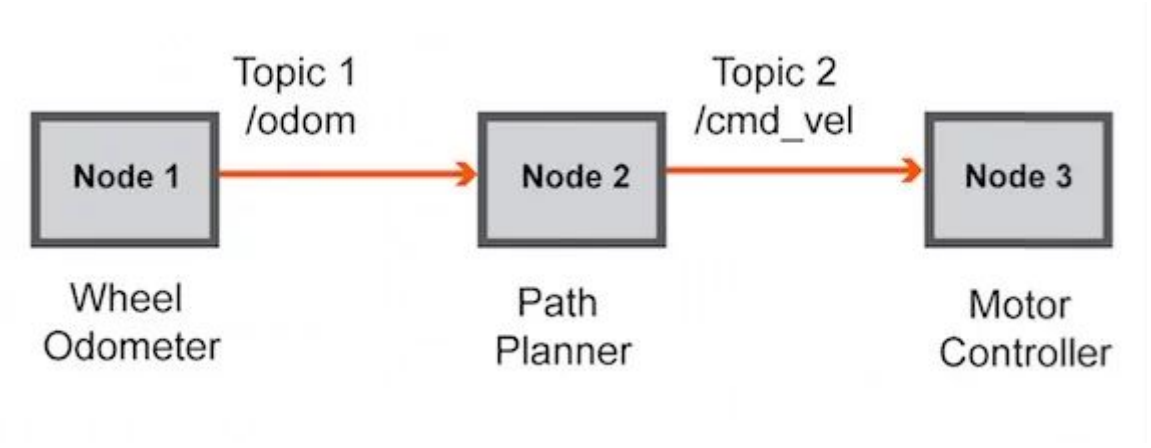
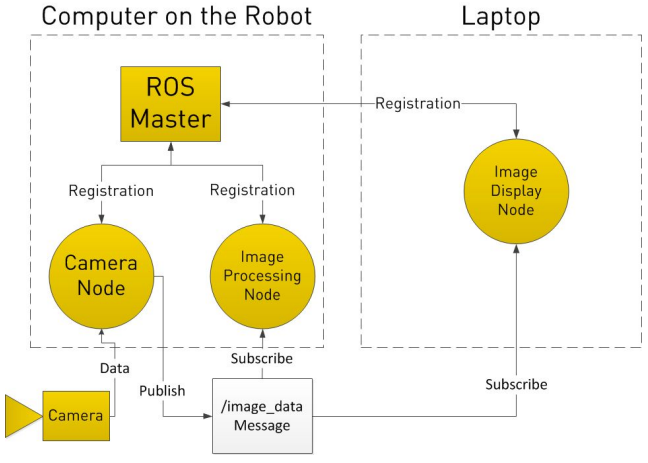
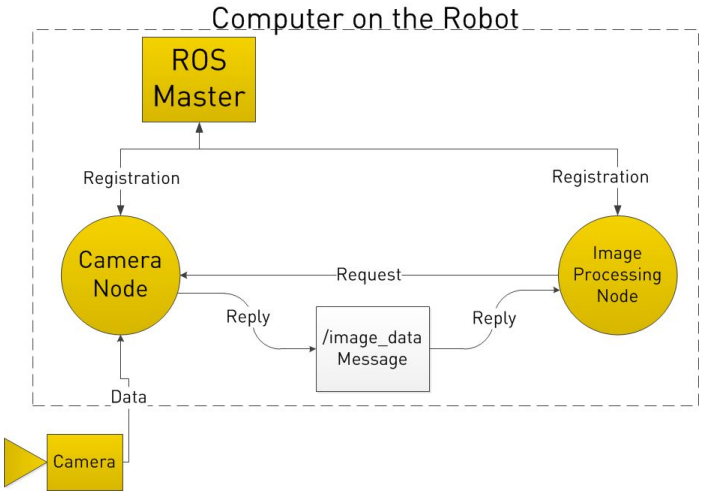


Figure 1. Nodes and topics.





Summary of references:

1. Clearpath, R. (2014, January 29). ROS 101: Intro to the Robot Operating System. Retrieved from https://robohub.org/ros-101-intro-to-the-robot-operating-system/.
2. Ademovic, A. (2016, March 3). An Introduction to Robot Operating System: The Ultimate Robot Application Framework. Retrieved from https://www.toptal.com/robotics/introduction-to-robot-operating-system.
3. Ademovic, A. (2016, March 3). An Introduction to Robot Operating System: The Ultimate Robot Application Framework. Retrieved from https://www.toptal.com/robotics/introduction-to-robot-operating-system.

Analysis of results including pros and cons:

Pros:

Robot Operating System has many advantages. First, it was designed very portability, users do not need exchange main frame and system, because the code that is written by ROS can be used in other software framework of robots. Language independent is also an advantage of ROS. The ROS framework is easy to implement in any programming language, such as Python, C++. Third is simple testing – ROS has a built-in unit/assembly testing framework called "rostest". This makes integration and decomposition debugging easy. Fourth is scalability – ROS is suitable for large real-time systems and large system development projects.

Cons:

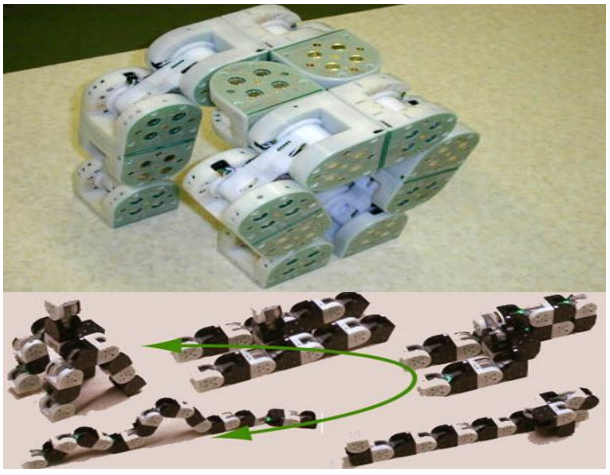
1. takes too much time to prepare and perform the same high level movements (like welding) that a robot can perform while being easily program with its own vendor programming interface.

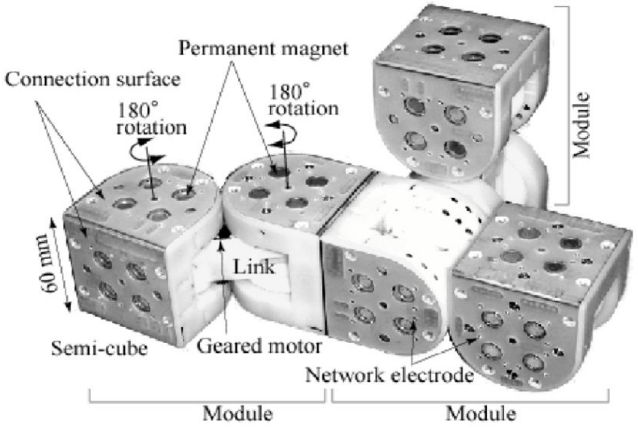
2. Not such reliable satisfied with precise industrial requirement.

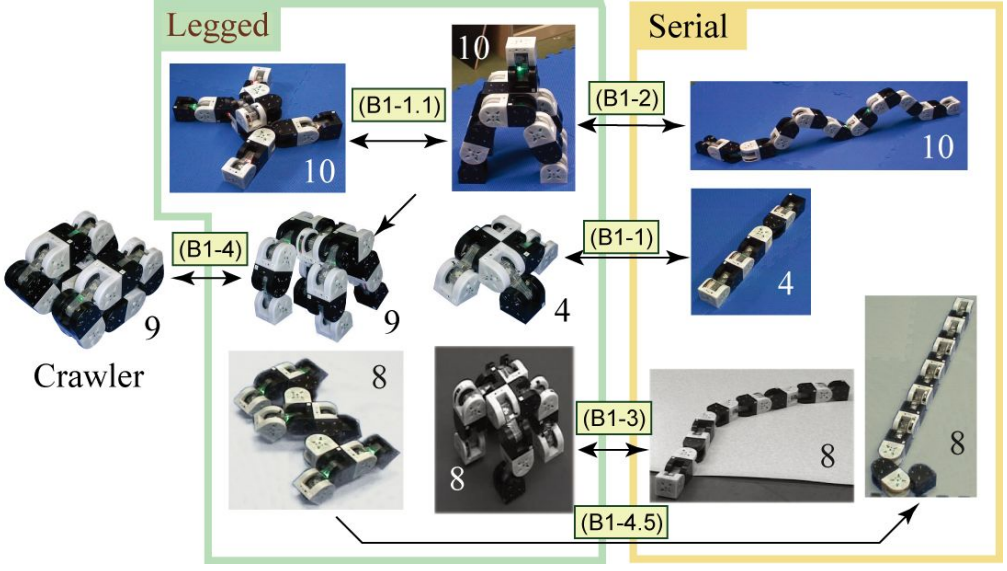
3. Not suitable for **M-TRAN robotic system** because ROS only do simulation for one cup.

M-TRAN Robotic System:

M-TRAN is a self-reconfigurable modular robot belonging to a class of robotic system. The modules can form a 3-D structure which changes its own configuration. Compared with homogenous robotic system, self-configurable can change its shape and functionality without external help. This kind of flexibility is highly desirable for robotic system used in unstructured and unpredictable environment. Such as space, deep-sea exploration, or rescue operations in earthquake areas. Since M-TRAN system can reshape and consist of many small units, robotic can self-repair by replacing damaged modules by spare modules.







From those pictures, obviously, M-TRAN consist of many small components. And each component has its CPU. they communicate with each other and cooperating with each other.

Recommendations:

It provides functionality for hardware abstraction, device drivers, communication between processes over multiple machines, tools for testing and visualization, and much more. Nodes in ROS do not have to be on the same system (multiple computers) or even of the same architecture. Users could have a Arduino publishing messages, a laptop subscribing to them, and an Android phone driving motors. This makes ROS really flexible and adaptable to the needs of the user. ROS is also open source, maintained by many people. That's the reasons why people should use ROS.

Conclusions:

In conclusion, ROS works well for homogenous robotic system, which is controlled by a single CPU other than M-TRAN robotic system, controlled by variable CUPs. In addition, it is not very reliable, if the robotic designed for precise and complex work. However, if your design for general purpose, ROS should be fine and many libraries provide. In addition, beginners can learn from other's software design shared by owners.