The Journal of Open Source Software

ros_control: A generic and simple control framework for ROS

Sachin Chitta[9, 11], **Eitan Marder-Eppstein**[1], **Wim Meeussen**[1], **Vijay Pradeep**[1], **Adolfo Rodríguez Tsouroukdissian**[12, 2], **Jonathan Bohren**[8, 10], **David Coleman**[3, 11], **Bence Magyar**[4, 2], **Gennaro Raiola**[5, 2], **Mathias Lüdtke**[6], **and Enrique Fernandez Perdomo**[7, 2]

1 hiDOF, Inc. (at the time of this work) **2** PAL Robotics (at the time of this work) **3** PickNik Consulting **4** Heriot-Watt University, Edinburgh Centre for Robotics **5** Department of Advanced Robotics, Istituto Italiano di Tecnologia (IIT) **6** Fraunhofer IPA **7** Clearpath Robotics **8** Honeybee Robotics **9** Kinema Systems Inc. **10** John Hopkins University **11** Willow Garage Inc. (at the time of this work) **12** Pick-it NV

DOI: 10.21105/joss.00456

Software

- Review
- Repository
- Archive

Submitted: 09 November 2017
Published: 05 December 2017

Summary

In recent years the Robot Operating System (Quigley et al. 2009) (ROS) has become the 'de facto' standard framework for robotics software development. The `ros_control` framework provides the capability to implement and manage robot controllers with a focus on both *real-time performance* and *sharing of controllers* in a robot-agnostic way. The primary motivation for a sepate robot-control framework is the lack of realtime-safe communication layer in ROS. Furthermore, the framework implements solutions for controller-lifecycle and hardware resource management as well as abstractions on hardware interfaces with minimal assumptions on hardware or operating system. The clear, modular design of `ros_control` makes it ideal for both research and industrial use and has indeed seen many such applications to date. The idea of `ros_control` originates from the `pr2_controller_manager` framework specific to the PR2 robot but `ros_control` is fully robot-agnostic. Controllers expose standard ROS interfaces for out-of-the box 3rd party solutions to robotics problems like manipulation path planning (`MoveIt!` (Chitta, Sucan, and Cousins 2012)) and autonomous navigation (the `ROS navigation stack`). Hence, a robot made up of a mobile base and an arm that support `ros_control` doesn't need any additional code to be written, only a few controller configuration files and it is ready to navigate autonomously and do path planning for the arm. `ros_control` also provides several libraries to support writing custom controllers.

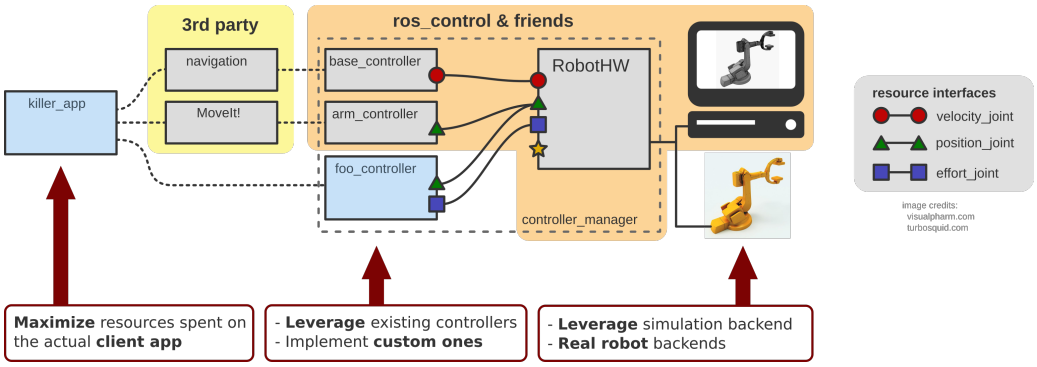

Figure 1: Overview

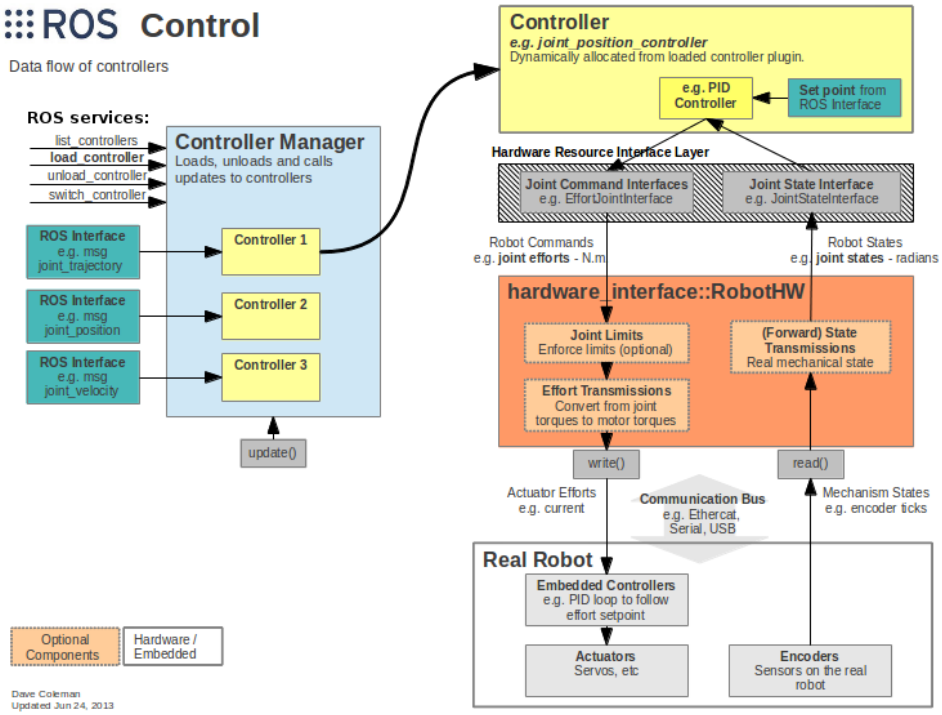

Figure 2: ROS Control overview

Packages and functionalities

The backbone of the framework is the Hardware Abstraction Layer, which serves as a bridge to different simulated and real robots. This abstraction is provided by the `hardware_interface::RobotHW` class; specific robot implementations have to inherit from this class. Instances of this class model hardware resources provided by the robot such as electric and hydraulic actuators and low-level sensors such as encoders and force/torque sensors. It also allows for integrating heterogeneous hardware or swapping out components transparently whether it is a real or simulated robot.

There is a possibility for composing already implemented `RobotHW` instances which is ideal for constructing control systems for robots where parts come from different suppliers, each supplying their own specific `RobotHW` instance. The rest of the `hardware_interface` package defines read-only or read-write typed joint and actuator interfaces for abstracting hardware away, e.g. state, position, velocity and effort interfaces. Through these typed interfaces this abstraction enables easy introspection, increased maintainability and controllers to be hardware-agnostic.

The `controller_manager` is responsible for managing the lifecycle of controllers, and hardware resources through the interfaces and handling resource conflicts between controllers. The lifecycle of controllers is not static. It can be queried and modified at runtime through standard `ROS services` provided by the `controller_manager`. Such services allow to start, stop and configure controllers at runtime.

Furthermore, `ros_control` ships software libraries addressing real-time ROS communication, transmissions and joint limits. The `realtime_tools` library adds utility classes handling ROS communications in a realtime-safe way. The `transmission_interface`

package supplies classes implementing joint- and actuator-space conversions such as: simple reducer, four-bar linkage and differential transmissions. A declarative definition of transmissions is supported directly with the kinematics and dynamics description in the robot's Universal Robot Description Format (URDF) (Willow Garage 2009) file. The `joint_limits_interface` package contains data structures for representing joint limits, methods to populate them through URDF or yaml files and methods to enforce these limits. `control_toolbox` offers components useful when writing controllers: a PID controller class, smoothers, sine-wave and noise generators.

The repository `ros_controllers` holds several ready-made controllers supporting the most common use-cases for manipulators, mobile and humanoid robots, e.g. the `joint_trajectory_controller` is heavily used with position-controlled robots to interface with MoveIt!. Finally, `control_msgs` provides ROS messages used in most controllers offered in `ros_controllers`.

`ros_control` was conceptualized by Sachin Chitta at Willow Garage Inc. and initial design and implementation was done by Sachin Chitta (then at Willow Garage), Wim Meussen, Vijay Pradeep and Eitan Marder-Epstein (then at HiDOF) before being released open-source.

`ros_control` is released as binary packages with each new version of ROS, source code is hosted at the ros-controls Github organization. Documentation on behaviour, interfaces, doxygen-generated pages and tutorials can be found at ros_control and ros_controllers. For a thorough presentation we invite the interested reader to watch the talk given at ROSCon2014 (Adolfo Rodríguez Tsouroukdissian, n.d.).

Robots using `ros_control`

Being a mature framework, `ros_control` is widely applied to both production and research platform robots. A few examples where the control system is implemented with `ros_control` are:

- Clearpath Robotics' outdoor mobile robots: Grizzly, Husky, Jackal ("New Universal Robots Driver Makes Manipulation Research Easier," n.d.), and OTTO Motors' industrial indoor mobile robots: OTTO 1500, OTTO 100
- The "Twil" robot at Federal University of Rio Grande do Sul (Lages 2017)
- The quadruped robots HyQ and HyQ2Max (Semini et al. 2011, Semini et al. (2017)) at Istituto Italiano di Tecnologia
- NASA's humanoid and biped robots: Valkyrie & Robonaut (N. A. Radford et al. 2015, Hart et al. (2014), Badger et al. (2016))
- PAL Robotics' humanoid, biped and mobile robots: REEM, REEM-C, PMB2, Tiago and Talos (Stasse et al. 2017)
- Shadow Robot's anthropomorphic, highly sensorized and precise Shadow Hand (Meier et al. 2016)
- Universal Robots' industrial arms: UR3, UR5 (Andersen 2015)

Robot names and credits in order of appearance: Valkyrie (Photo by NASA / Bill Stafford), Husky (Photo by Clearpath Robotics), OTTO (Photo by Otto Motors), TIAGo (Photo by PAL Robotics), Dexterous Hand (Photo by Shadow Robot), HyQ2Max (Photo by Istituto Italiano di Tecnologia), TALOS (Photo by PAL Robotics)

References

Adolfo Rodríguez Tsouroukdissian. n.d. "[ROSCon2014] Ros_control: An Overview." https://vimeo.com/107507546.

Andersen, Thomas Timm. 2015. "Optimizing the Universal Robots ROS Driver." Technical University of Denmark, Department of Electrical Engineering; http://orbit.dtu.dk/en/publications/optimizing-the-universal-robots-ros-driver(20dde139-7e87-4552-8658-dbf2cdaab24b).html.

Badger, Julia, Dustin Gooding, Kody Ensley, Kimberly Hambuchen, and Allison Thackston. 2016. "ROS in Space: A Case Study on Robonaut 2." In *Robot Operating System (ROS)*, 343–73. Springer. doi:10.1007/978-3-319-26054-9_13.

Chitta, Sachin, Ioan Sucan, and Steve Cousins. 2012. "Moveit![ROS Topics]." *IEEE Robotics & Automation Magazine* 19 (1). IEEE: 18–19.

Hart, Stephen, Paul Dinh, John D Yamokoski, Brian Wightman, and Nicolaus Radford. 2014. "Robot Task Commander: A Framework and IDE for Robot Application Development." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, 1547–54. IEEE. doi:10.1109/IROS.2014.6942761.

Lages, Walter Fetter. 2017. "Parametric Identification of the Dynamics of Mobile Robots and Its Application to the Tuning of Controllers in ROS." In *Robot Operating System (ROS)*, 191–229. Springer. doi:10.1007/978-3-319-54927-9_6.

Meier, Martin, Guillaume Walck, Robert Haschke, and Helge J Ritter. 2016. "Distinguishing Sliding from Slipping During Object Pushing." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016)*, 5579–84. IEEE. doi:10.1109/IROS.2016.7759820.

"New Universal Robots Driver Makes Manipulation Research Easier." n.d. https://www.clearpathrobotics.com/2016/02/new-universal-robots-driver-makes-manipulation-easier.

Quigley, Morgan, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob

Wheeler, and Andrew Y Ng. 2009. "ROS: An Open-Source Robot Operating System." In *ICRA Workshop on Open Source Software*, 3:5. 3.2. Kobe; http://www.willowgarage.com/papers/ros-open-source-robot-operating-system.

Radford, Nicolaus A., Philip Strawser, Kimberly Hambuchen, Joshua S. Mehling, William K. Verdeyen, A. Stuart Donnan, James Holley, et al. 2015. "Valkyrie: NASA's First Bipedal Humanoid Robot." *Journal of Field Robotics* 32 (3): 397–419. doi:10.1002/rob.21560.

Semini, Claudio, Victor Barasuol, Jake Goldsmith, Marco Frigerio, Michele Focchi, Yifu Gao, and Darwin G Caldwell. 2017. "Design of the Hydraulically Actuated, Torque-Controlled Quadruped Robot HyQ2Max." *IEEE/ASME Transactions on Mechatronics* 22 (2). IEEE: 635–46. doi:10.1109/TMECH.2016.2616284.

Semini, Claudio, Nikos G Tsagarakis, Emanuele Guglielmino, Michele Focchi, Ferdinando Cannella, and Darwin G Caldwell. 2011. "Design of HyQ–a Hydraulically and Electrically Actuated Quadruped Robot." *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 225 (6). SAGE Publications Sage UK: London, England: 831–49. doi:10.1109/IROS.2010.5651548.

Stasse, Olivier, Thomas Flayols, Rohan Budhiraja, Kevin Giraud-Esclasse, Justin Carpentier, Andrea Del Prete, Philippe Soueres, et al. 2017. "TALOS: A New Humanoid Research Platform Targeted for Industrial Applications." https://hal.archives-ouvertes.fr/hal-01485519.

Willow Garage. 2009. "Universal Robot Description Format (URDF)." http://wiki.ros.org/urdf/.