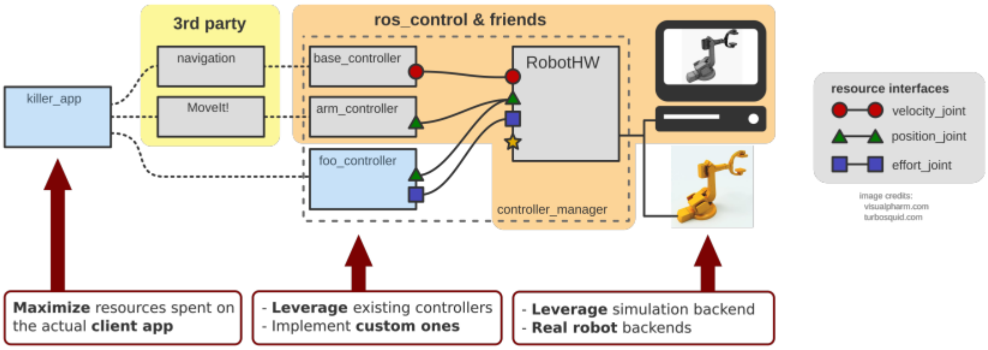
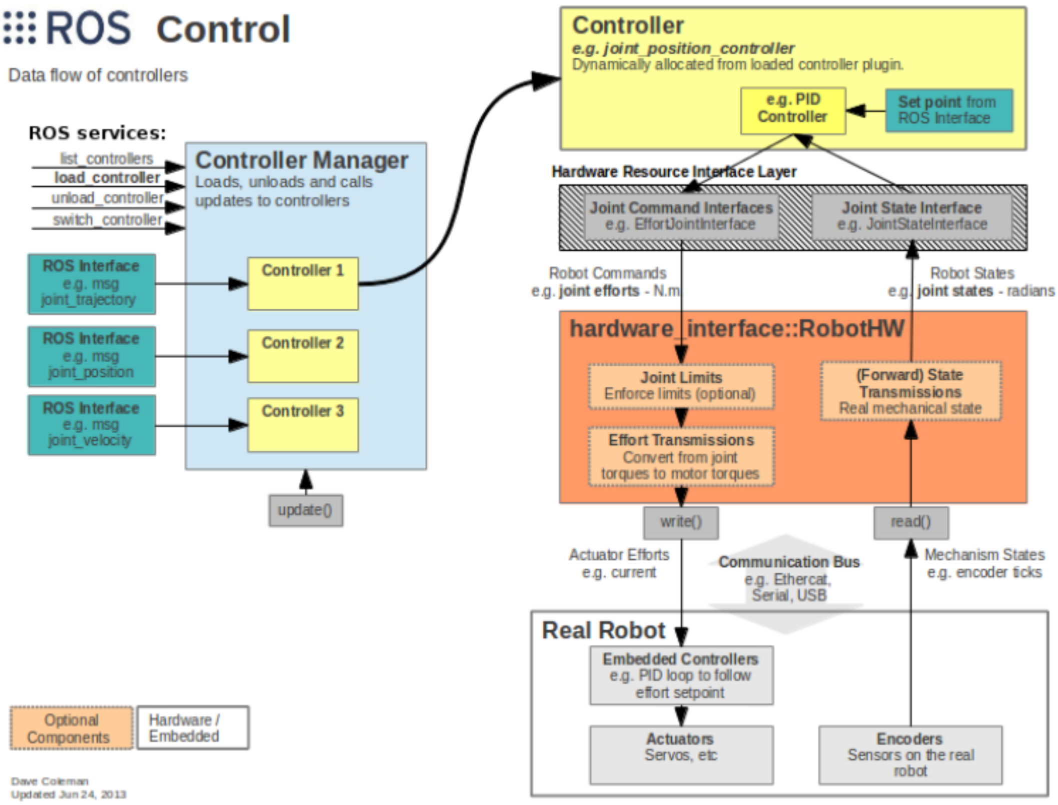
ros_control
A generic and simple control framework for ROS

⇒ 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.





★ 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.

⇒ 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.

⇒ The controller_manager is responsible for managing the lifecycle of controllers, and hardware resources through the interfaces and handling resource conflicts between controllers.

⇒ 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.

⇒ The joint_limits_interface package contains data structures for representing joint 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.

↳ Example: joint_trajectory_controller

⇒ Finally, control_msgs provides ROS messages used in most controllers offered in ros_controllers.