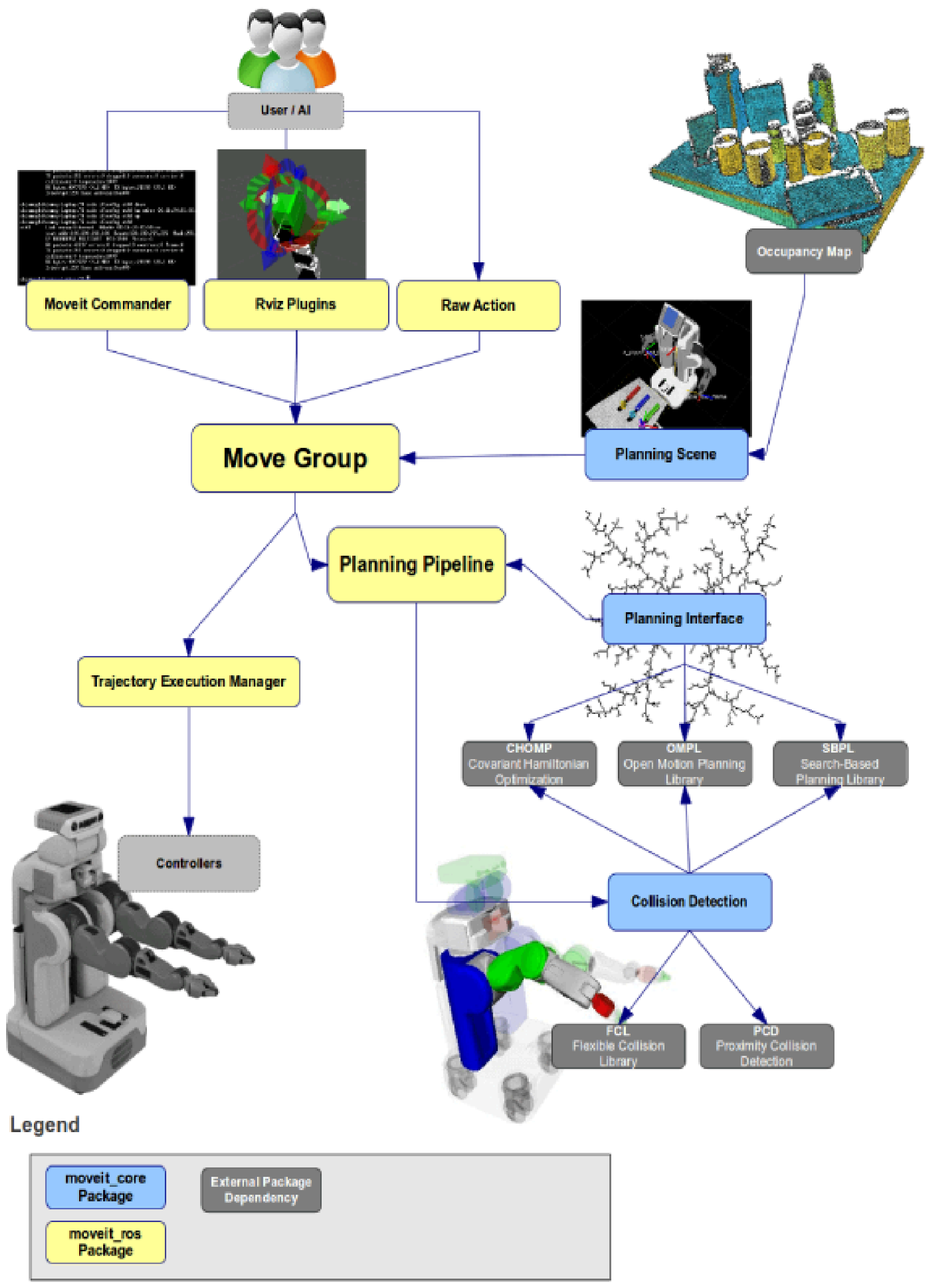
Moveit Config interface investigation

Reference: https://moveit.picknik.ai/main/index.html



User / AI
Moveit Commander
Rviz Plugins
Raw Action
Occupancy Map
Planning Scene
Move Group
Planning Pipeline
Planning Interface
Trajectory Execution Manager
CHOMP Covariant Hamiltonian Optimization
OMPL Open Motion Planning Library
SBPL Search-Based Planning Library
Controllers
Collision Detection
FCL Flexible Collision Library
PCD Proximity Collision Detection
Legend
moveit_core Package
moveit_ros Package
External Package Dependency

MOVEIT2 package gives more capability and functionality for serial manipulators
Based on Figure 1. We need to develop or outsource the grey rectangles
these are dependencies that need to be met: PCD, FCL, CHOMP, OMPL, SBPL And the controllers.
The solvers (: PCD, FCL, CHOMP, OMPL, SBPL) above are already existent as plugins.

Upcoming work:

- Investigating controllers
- Making Moveit Config. Using Moveit Setup Assistant

Moveit on low lever controllers

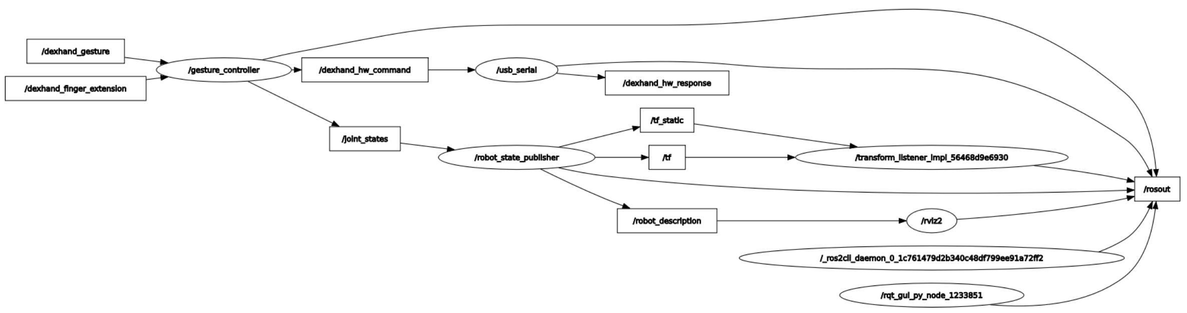
Investigating controllers (using ROS2_CONTROL)

Relevant links:

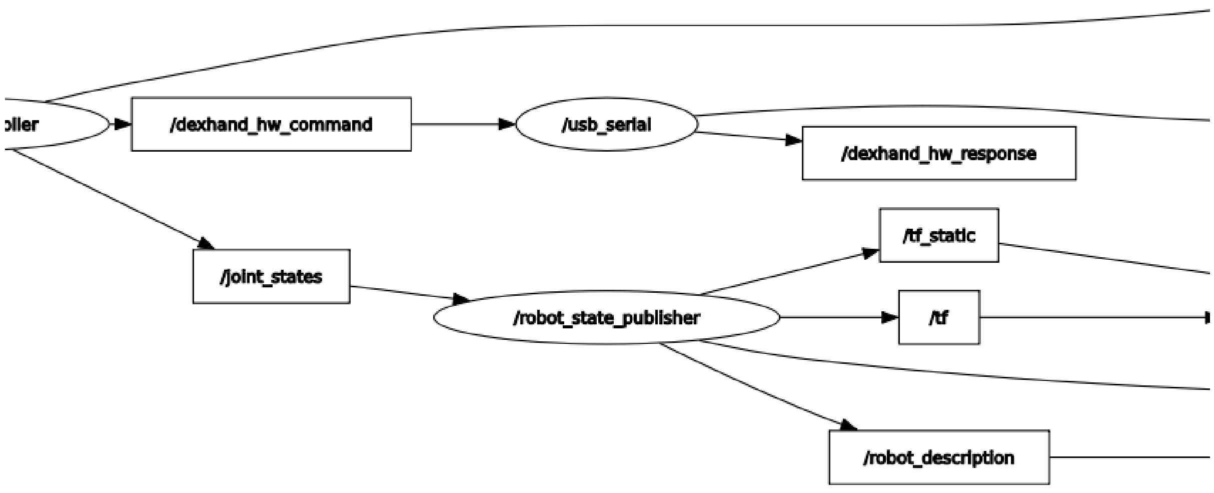
- Ros2 control home
- ROSCON2023 workshop ros2control on steroids
- Ros2_control demos and examples
- Simulator integration with ros_control

Existent nodes and topics

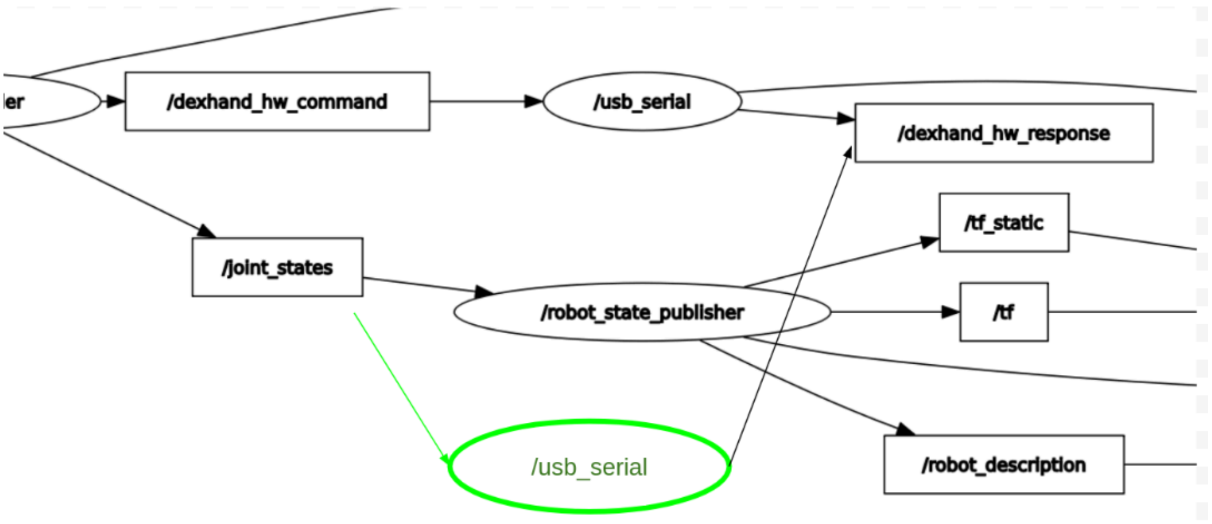
Using rqt-graph, all circles are nodes, and rectangles are topics
The current existent configuration in **DexHand:**
Joint state publisher with USB serial:



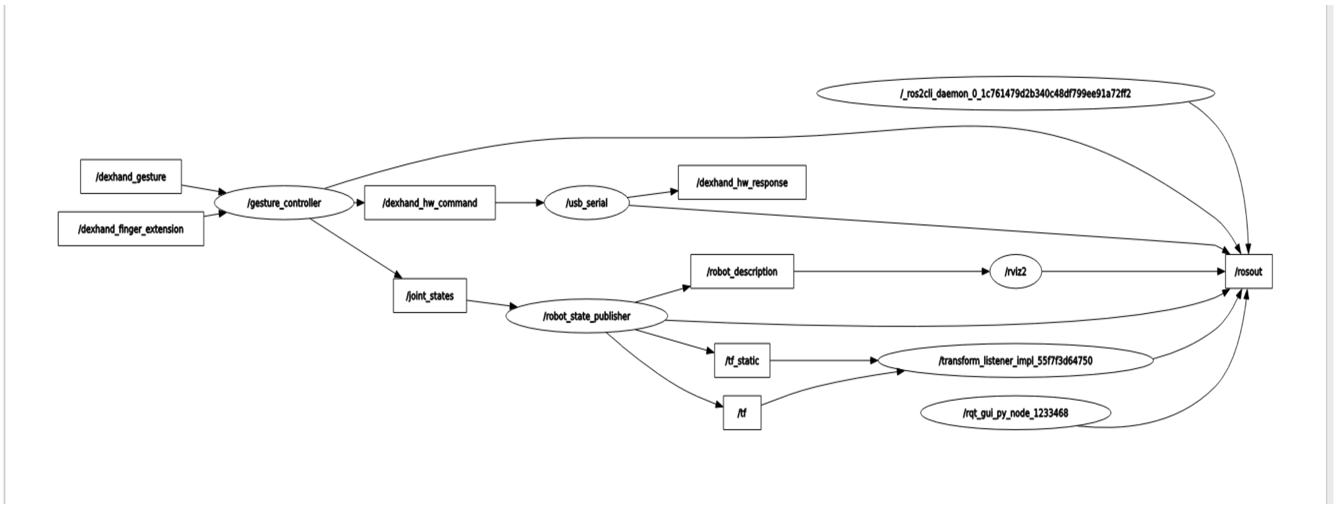
Zoomed view of the usb_serial, the current configuration



Desired change to usb_serial:



Gesture controller, with usb_serial:



Pretty interesting to read about ROS2 workshop

Prework to ROS2_control

- Setup the firmware through usb_serial to subscribe to joint angles from (joint_state type message) and act accordingly
-

Steps to get ROS2_control working

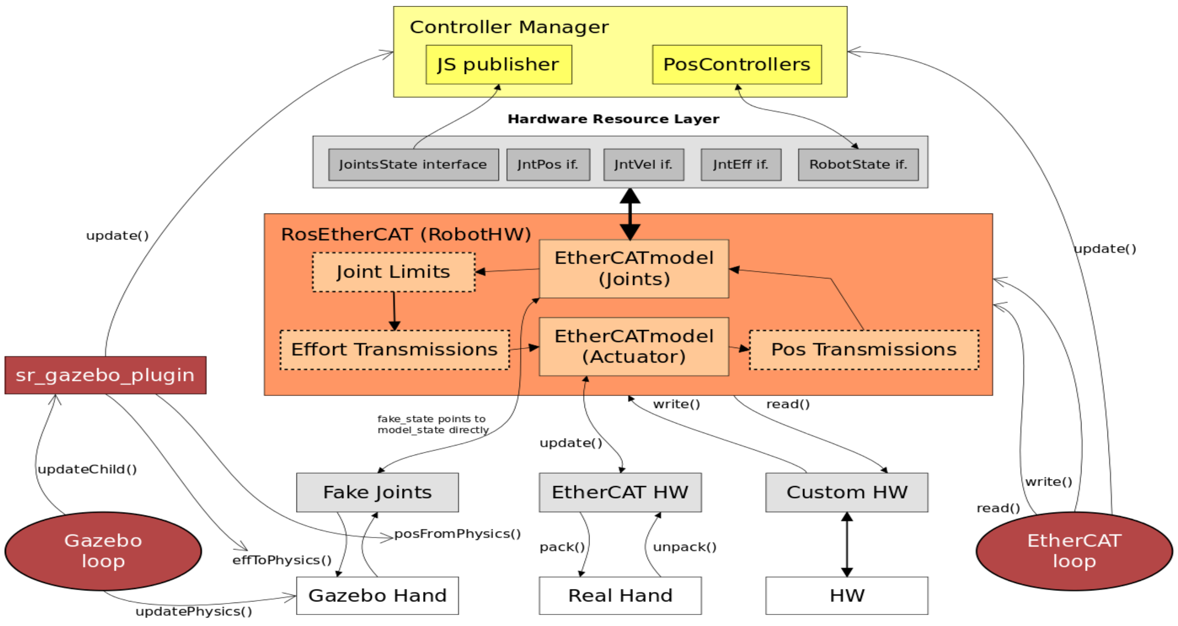
- Create a YAML file with the configuration of the controller manager and two controllers.
  - Textbook Example
  - Schunk ros driver example
  - You only need to worry about the structure of the yaml file, controller selection, its parameters …
  - : position controller, effort_controllers, and velocity_ controllers. Then a trajectory controller, which is a layer on top, sends (position or velocity) joint space commands over time to the driver.

    - **Explanation**: The ros2_control has a standard controller collection and configurable controller hierarchy, that can be specified based on a yaml file. Then the package controllers get (summoned) accordingly, and the YAML file includes parameters such as PID and motor limits etc for each controller.
- Extend the robot's URDF description with needed <ros2_control> tags. Using macro files (xacro) instead of pure URDF is recommended. (Example URDF for RRBot).
    - Text example
    - Chunk ros2 hand example (why not)
- "Create a launch file to start the node with Controller Manager. You can use a default ros2_control node (recommended) or integrate the controller manager in your software stack". (Example launch file for RRBot).

Appendix next page:

Appendix

- Other hand control stacks:
  Allegro, has two controllers, PD, and velocity saturation joint controller, Allegro seems to have developed their own 'custom' controller, written 10 years ago (ROS1)
  leapHand: custom controller, control it by publishing joint states. Leap hand currently does not have it moveit!, current control is directly through joint space.
  Shadowhand: used pr2 control mechanisms, which is the basis for ros_control (ros1), from the following image. They have many interfaces but mainly use the joint space publisher interface: (particularly in the Open AI Rubic cubecase).


Controller Manager
JS publisher
PosControllers
Hardware Resource Layer
JointsState interface
JntPos if.
JntVel if.
JntEff if.
RobotState if.
RosEtherCAT (RobotHW)
Joint Limits
EtherCATmodel (Joints)
Effort Transmissions
EtherCATmodel (Actuator)
Pos Transmissions
update()
update()
sr_gazebo_plugin
fake_state points to model_state directly
write()
read()
update()
updateChild()
Fake Joints
EtherCAT HW
Custom HW
write()
read()
posFromPhysics()
Gazebo loop
effToPhysics()
pack()
unpack()
EtherCAT loop
Gazebo Hand
Real Hand
HW
updatePhysics()