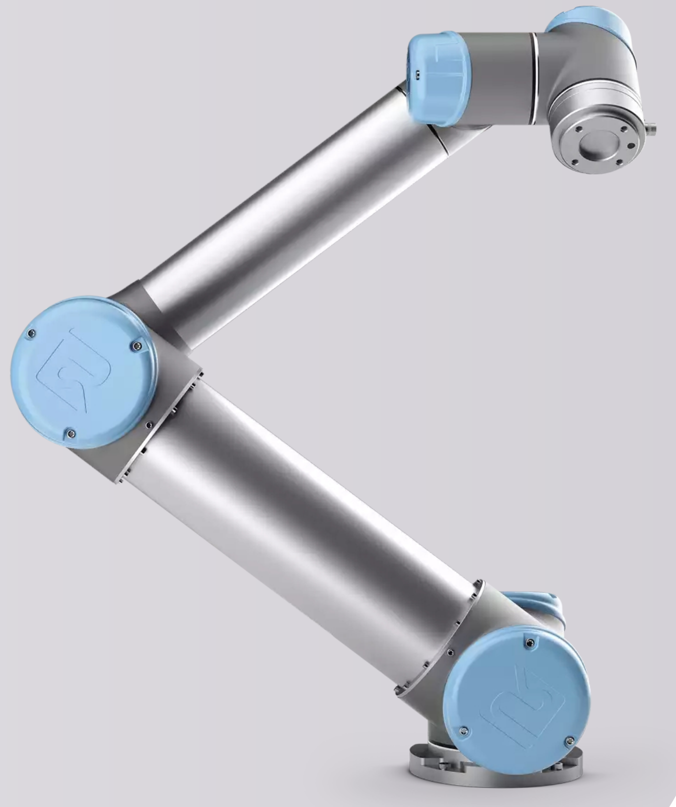


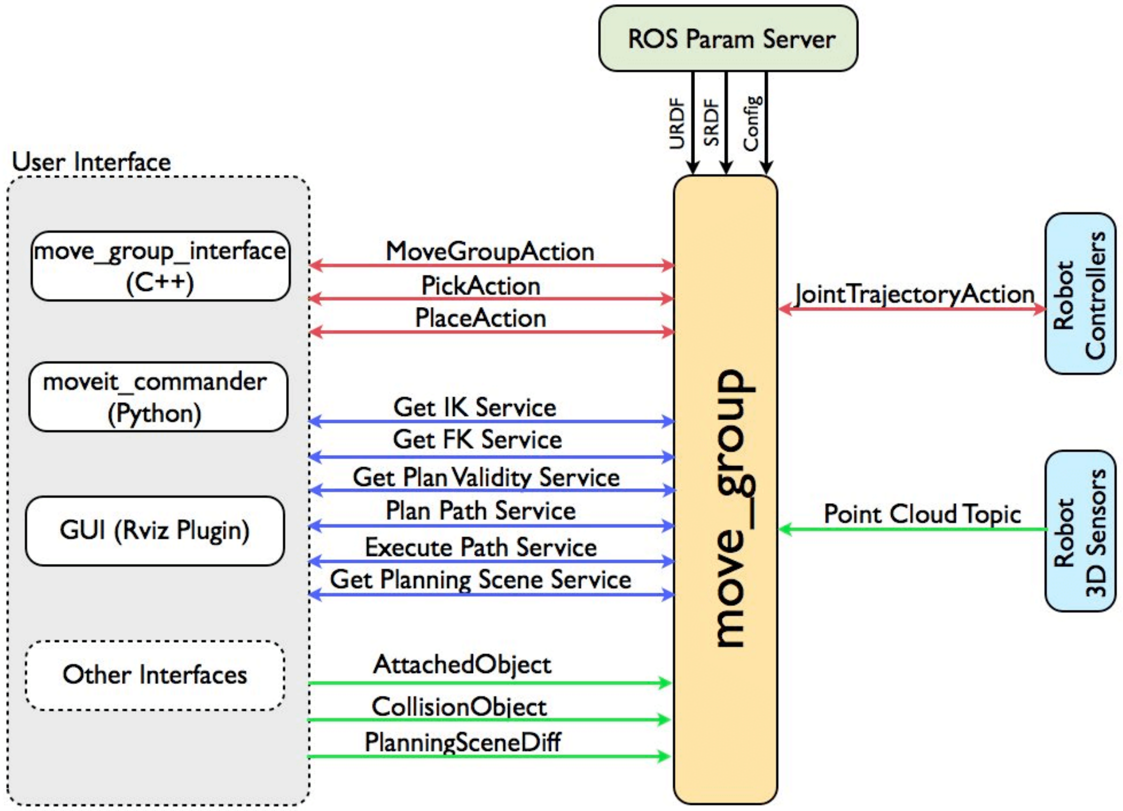
آموزش کاربردی

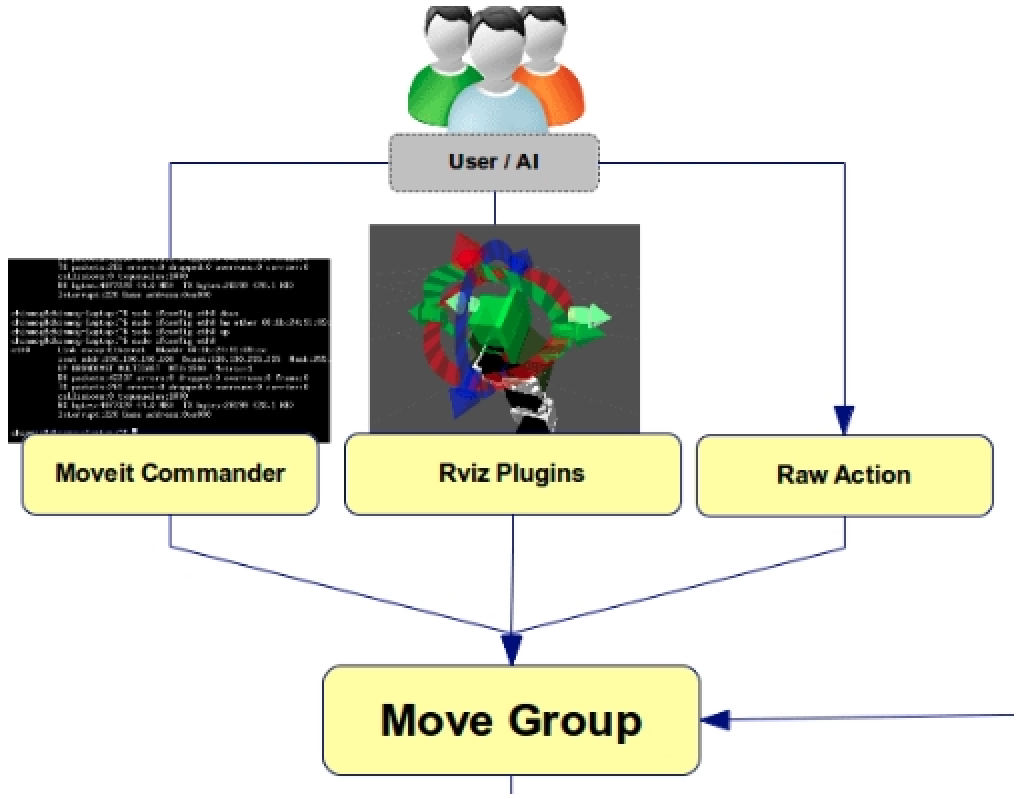
MoveIt

مبانی مهندسی مکاترونیک

اسکریپ پایتون و MoveIt

کنترل و هدایت بازوی رباتیک از طریق اسکریپ پاتون با کمک moveit_commander انجام می‌شود.





moveit_commander نصب

```
$ sudo apt install ros-noetic-moveit-commander
```

شروع برنامه نویسی!

کتابخونه هایی که نیاز داریم

```
import sys
import rospy
import moveit_commander
import moveit_msgs.msg
import geometry_msgs.msg
import copy
from math import tau
```

تعریف مقادیر اولیه

```
if __name__ == '__main__':

    # initialize moveit_commander and ros node
    moveit_commander.roscpp_initialize(sys.argv)
    rospy.init_node("move_group_python_interface", anonymous=True)


    # define robot
    robot = moveit_commander.RobotCommander()

    # define scene
    scene = moveit_commander.PlanningSceneInterface()

    # define move_group
    group_name = 'arm'
    move_group = moveit_commander.MoveGroupCommander(group_name)
```

نمایش اطلاعات پایه

```
# Getting basic informations
planning_frame = move_group.get_planning_frame()
print(f'============ Planning frame : {planning_frame}')

group_names = robot.get_group_names()
print(f'============ Group Names : {group_names}')

print("============ Robot's Current State ==========")
print(robot.get_current_state())
```

تعیین مقدار مشخص برای هر مفصل

```
# Planning to Joint Goal
joint_goal = move_group.get_current_joint_values()
joint_goal[0] = 0
joint_goal[1] = -tau / 8
joint_goal[2] = 0
joint_goal[3] = -tau / 4
joint_goal[4] = 0
joint_goal[5] = tau / 6

move_group.go(joint_goal, wait=True)
move_group.stop()
```

تعیین مختصات Goal

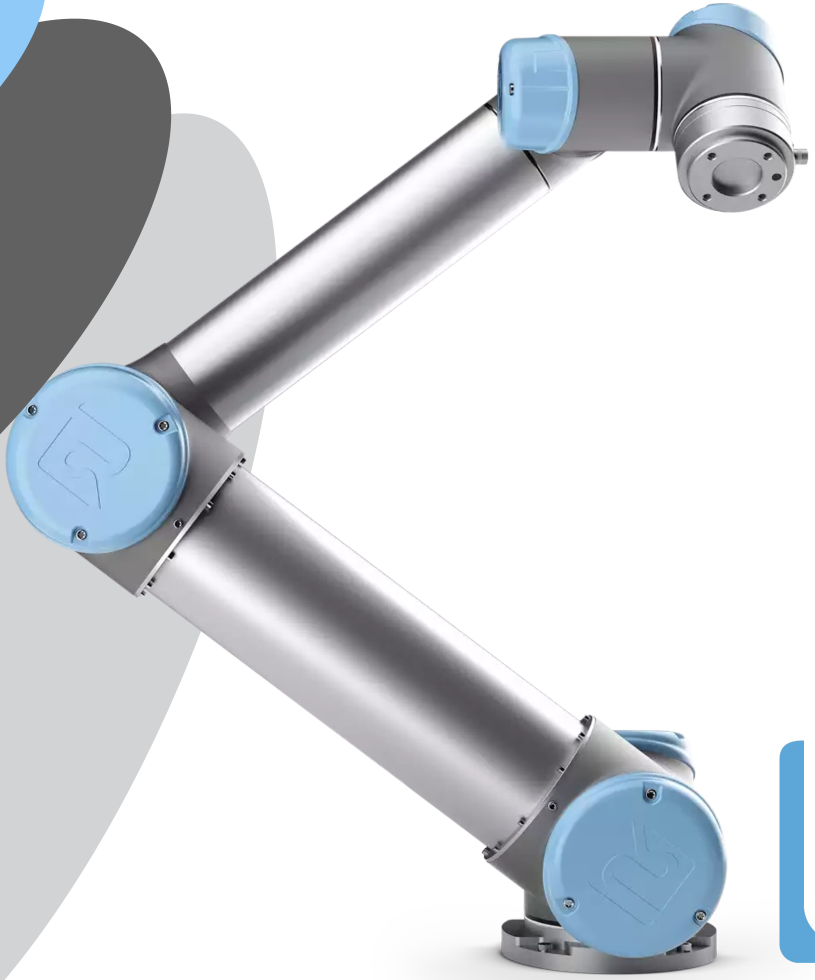
```
# Planning to Pose Goal
pose_goal = geometry_msgs.msg.Pose()
pose_goal.orientation.w = 1.0
pose_goal.position.x = -0.3
pose_goal.position.y = -0.3
pose_goal.position.z = 0.3

move_group.set_pose_target(pose_goal)

# call planner to compute the plan and execute it
plan = move_group.go(wait=True)
move_group.stop()
move_group.clear_pose_targets()
```

تعیین مسیر Carthesian

```
# carthesian path
waypoints = []
scale = 1.0
wpose = move_group.get_current_pose().pose
wpose.position.z -= scale * 0.1  # First move up (z)
wpose.position.y += scale * 0.2  # and sideways (y)
waypoints.append(copy.deepcopy(wpose))

wpose.position.x += scale * 0.1  # Second move forward/backwards in (x)
waypoints.append(copy.deepcopy(wpose))

wpose.position.y -= scale * 0.1  # Third move sideways (y)
waypoints.append(copy.deepcopy(wpose))

(plan, fraction) = move_group.compute_cartesian_path(
                waypoints, # waypoints to follow
                0.01,      # eef_step
                0.0)       # jump_threshold
move_group.execute(plan, wait=True)
```

مراجع و منابع مطالعاتی بیشتر

1. https://github.com/ros-industrial/universal_robot
2. https://moveit.ros.org/
3. http://docs.ros.org/en/noetic/api/moveit_commander/html/namespacemoveit__commander.html
4. https://ros-planning.github.io/moveit_tutorials/doc/getting_started/getting_started.html
5. https://github.com/ros-planning/moveit_tutorials



پایان