We will continue on where we left! In this lecture, we will dive into some deeper details of the configuration of controllers. This is as important as the previous video for a proper understanding of MoveIt.

Important Note:
Because we are using a *new version of MoveIt*, there are some differences between the files you will get and those shown on the video. This is not a problem, as the components have the same behavior.

We have explained everything step-by-step on the description below.

Configuring controllers

0:00 / 0:00

1.0x

About controllers

We are focusing now on how to execute the motion, that's where *controllers* come in play!

- Trajectories are executed on simulated robots or real hardware, and they are controlled by *controllers.*
- Gazebo uses *controllers* to control the motion and to know the poses and status of the robot.
- MoveIt plans movements and send those movements to the *controllers* in Gazebo.
- Communication between all these components happens over ROS topics and action servers (studied on Week 1)

Components of Gazebo and MoveIt

Gazebo uses:

- `JointStateController` to publish current joint values (like a sensor)
- `JointTrajectoryController` to control the execution of trajectories sent by MoveIt

MoveIt uses:

- `Planning scene monitor` to update position of things it know (robot arms or other objects)
- `Simple Controller Manager` to send the desired movements to the trajectory controller.

Gazebo Simulation and MoveIt

- All the gazebo configuration is contained in the `hrwros_gazebo` package (part of the Week 4 content files)

- All MoveIt configuration is contained in the `hrwros_moveit_config` package (you just created it)

hrwros_gazebo

- In `hrwros_gazebo`, all controller files live under the `config` folder.
- Joint state controllers contain a name, a type, and a publishing frequency.
    - They are defined on the `rX_joint_state_controller.yaml` file
- Trajectory controllers contain a name, a type, a list of associated joints, the gains, their constraints, and some other information to do with tolerances.
    - They are defined on the `robotX_controller.yaml` file

Important Notes:

- **On the `robotX_controller.yaml` files, the video lecture, shows `position_controllers/JointTrajectoryController` as the type of controllers.**
- **On the files you will get this has changed to `effort_controllers/JointTrajectoryController`**
- **Also, the gains are not shown on the video, as they are only needed for effort_controllers.**

- Controllers are provided as ROS action servers: They need to be powered by nodes! We can launch them using launch files.
    - The launchers are found on the `launch` folder.
    - The `spawn_robots.launch` file contains these settings for the two controllers of each robot.

hrwros_moveit_config

- In `hrwros_moveit_config` the controller files are also under the `config` folder.
- You need to create a new file named `controllers.yaml` to list all the controller clients that MoveIt is going to use.

Fill in that file with the name of the controllers including their namespace, their action namespaces, their type, and the joints they control (from the gazebo controller), the contents should look like this:

```
controller_list:
 - name: robot1/robot1_controller
   action_ns: follow_joint_trajectory
   type: FollowJointTrajectory
   joints: ["robot1_elbow_joint",
            "robot1_shoulder_lift_joint",
            "robot1_shoulder_pan_joint",
            "robot1_wrist_1_joint",
            "robot1_wrist_2_joint",
            "robot1_wrist_3_joint"]
 - name: robot2/robot2_controller
   action_ns: follow_joint_trajectory
   type: FollowJointTrajectory
   joints: ["robot2_elbow_joint",
            "robot2_shoulder_lift_joint",
            "robot2_shoulder_pan_joint",
            "robot2_wrist_1_joint",
            "robot2_wrist_2_joint",
            "robot2_wrist_3_joint"]
```

- Finally, we need a launcher file for our MoveIt controller manager file, where we define a name, which controller manager system we want to use from MoveIt, and the file from the previous point where we defined the controller. So go to the launcher folder and open edit the `hrwros_moveit_controller_manager.launch.xml` file.

Important Note:
On the video,
the `hrwros_moveit_controller_manager.launch.xml` appears empty, whereas on the package you just created it's already pre-filled.

All you need to do is to edit it, so it reads the `controllers.yaml` file you just created and not the one it has by default (`ros_controllers.yaml`).

Now, we are all set to actually move our robot arms!

Question 1

1 point possible (ungraded)

Which of the following statements are true?

There are three correct statements.

- [ ] 1) The controllers.yaml file already exists in the generated config package by the setup assistant. We only have to update it.
- [ ] 2) The administrative information to be filled in to the controllers.yaml file can be found in the hrwros_gazebo package.
- [ ] 3) The joint state controller for each robot publishes the joint state information for each robot.
- [ ] 4) The simulated controllers to control the robot arms are provided by a Gazebo (via gazebo_ros_control ROS package).

Submit

Question 2

1 point possible (ungraded)

The line of xml code in hrwros_moveit_controller_manager.launch.xml that is within the rosparam tag is actually not necessary because the controllers.yaml file is already included in the config package.

True or False?

- ◯ True
- ◯ False

Submit