Robots have always been big, strong, robust devices that work on specific tasks which were designed for them. They've been kept in cages and surrounded by guards for safety purposes. Their bright color was used to warn surrounding workers about the danger they represented. And it took a lot of programming skills just to set up these robots.

Cobots or collaborative robots, unlike industrial robots are designed to work alongside humans. Cobots are integrated with safety features such as passive compliance, or overcurrent detection, force and tactile sensors, due to which, on contact with human or some other object, it comes to a stop. If an external force acts on a joint, this joint will submit itself to this force. So, in the case of a collision, the joint will move in the opposite direction or stop completely to avoid causing injury. We have the collaborative robot UR5e from Universal Robots. Some key specifications of the UR5 Cobot are:

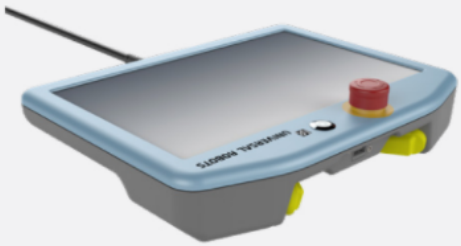
UR5

Degrees of freedom	6
Payload	5 kg
Weight	18.4 kg
Repeatability	+/- 0.1 mm
Reach	850 mm
Safety	TUV approved
Price	+/- 35,000 USD
Ease of programming	8/10

TARGETED APPLICATION

Machine tending, assembly, packaging, pick-and-place

The power to automate is in your hands.



e-Series 3PE Teach Pendant

All e-Series cobots include the standard e-Series Teach Pendant, offering an intuitive user interface for easy programming with UR's powerful PolyScope software.
A 3-position enabling teach pendant is also available as a variant for all payloads of e-Series robots, and as a UR+ component. The 3PE device is mechanically and functionally integrated with the e-Series Teach Pendant – just Plug & Produce with any e-Series control box. Additionally, it is fully integrated into the PolyScope user interface to enable all robot motion, including Freedrive, in manual mode.

Key Benefits

- Full mechanical 3PE device integration
- Full software integration - the 3PE Teach Pendant is natively supported in PolyScope
- Connects to the control box with the same connector as the standard e-Series teach pendant
- Can be mounted to any existing e-Series teach pendant brackets
- Includes two 3PE devices, allowing comfortable use with left or right hand
- Included in TÜV NORD certifications ISO 10218-1:2011 and ISO 13849-1:2015

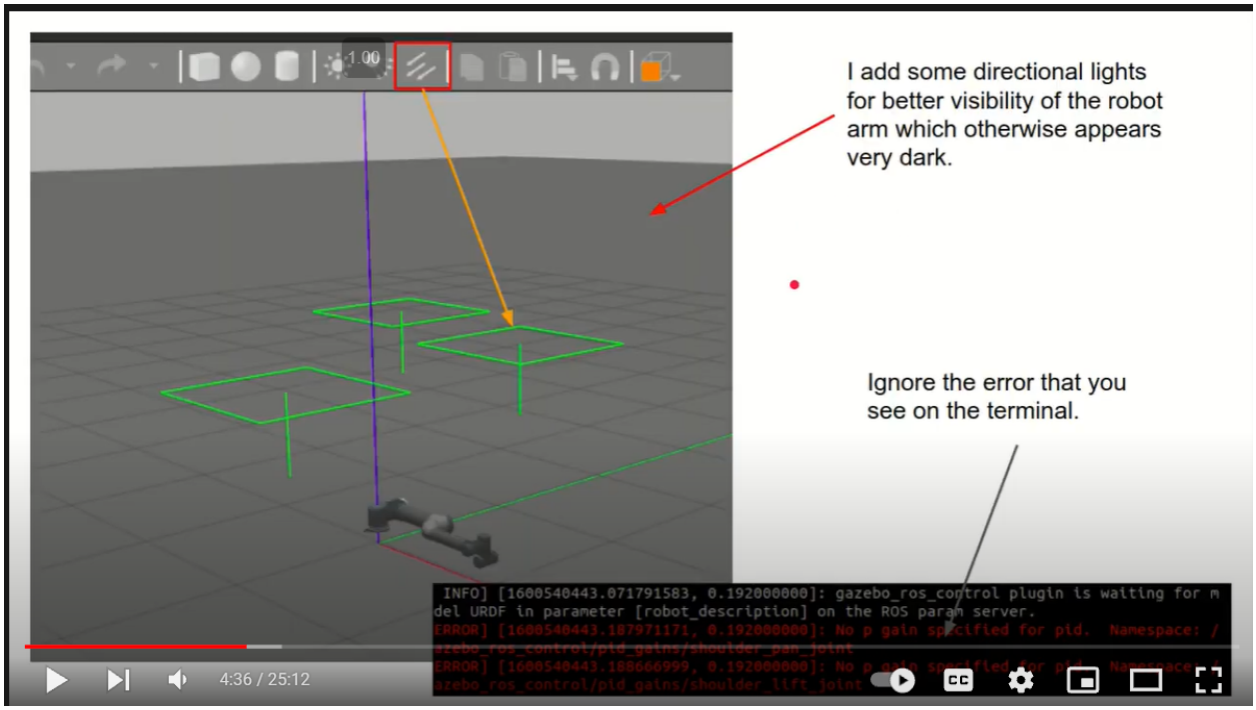
Hardware Specifications

Width	300 mm (11.81 in)
Height	231 mm (9.09 in)
Thickness	50 mm (1.97 in)
Weight, including 1 meter of cable	1.8 kg (3.961 lbs)
IP Classification	IP54

Step 1: Install Universal Robot Repository

```
$ cd catkin_ws/src
$ git clone -b melodic-devel
https://github.com/ros-industrial/universal_robot.git
$ catkin_make
$ source devel/setup.bash
$ roslaunch ur_gazebo ur5.launch
```

It shows some error on the terminal regarding something “No p gain specified for pid” which is to be ignored.



Step 2: Build your own “Robot Description”

This is where you can decide to mount your UR5 robot on a pedestal or table or attach a gripper or a camera etc.

First, we will only mount the robot arm on a pedestal. Later we will include other components into our world.

So create a catkin package called “`myur5_description`” inside your `catkin_ws/src` folder.

```
$ cd catkin_ws/src
$ mkdir myur5sim
$ catkin_create_pkg myur5_description
$ cd myur5_description
$ mkdir urdf
```

- Create `myur5.urdf.xacro` file inside 'urdf' folder that provides description of your pedestal with necessary joint and link information.

- Look into the following files for getting a better understanding of what to include in your URDF file:

```
~/catkin_ws/src/universal_robot/ur_gazebo/launch/ur5.launch
~/catkin_ws/src/universal_robot/ur_description/launch/ur5_upload.launch
~/catkin_ws/src/universal_robot/ur_description/urdf/ur5_robot.urdf.xacro
~/catkin_ws/src/universal_robot/ur_description/urdf/ur5.urdf.xacro
```

- Basically, we need to include UR5 URDF model and Gazebo related files in our own urdf file.

```
1  <?xml version="1.0"?>
2  <robot xmlns:xacro="http://wiki.ros.org/xacro" name="myur5">
3    <link name="world"/>
4    <link name="pedestal">
5      <inertial>
6        <origin xyz="0 0 0.5" rpy="0 0 0"/>
7        <mass value="20"/>
8        <inertia ixx="200" ixy="200" ixz="200" iyy="200" iyz="200" izz="200"/>
9      </inertial>
10     <visual>
11       <origin xyz="0 0 0.5" rpy="0 0 0"/>
12       <geometry>
13         <cylinder radius="0.1" length="1"/>
14       </geometry>
15       <material name="Gray">
16         <color rgba="0.5 0.5 0.5 0" />
17       </material>
18     </visual>
19     <collision>
20       <origin xyz="0 0 0.5" rpy="0 0 0"/>
21       <geometry>
22         <cylinder radius="0.1" length="1"/>
23       </geometry>
24     </collision>
25   </link>
```

Pedestal is just a cylinder of radius 0.1m and height 1m.

See the XML URDF link page on ROS wiki for more information on how to create such link.

File: ~/catkin_ws/src/myur5sim/myur5_description/urdf/myur5.urdf.xacro

File: ~/catkin_ws/src/myur5sim/myur5_description/urdf/myur5.urdf.xacro

```
26   <gazebo reference="pedestal">
27     <mu1>0.2</mu1>
28     <mu2>0.2</mu2>
29     <material>Gazebo/Orange</material>
30   </gazebo>
31   <joint name="world_joint" type="fixed">
32     <parent link="world" />
33     <child link="pedestal" />
34     <origin xyz="0 0 0" rpy="0.0 0.0 0.0"/>
35   </joint>
36   <xacro:arg name="transmission_hw_interface" default="hardware_interface/PositionJointInterface"/>
37   <!-- common stuff -->
38   <xacro:include filename="$(find ur_description)/urdf/common.gazebo.xacro" />
39   <!-- ur5 -->
40   <xacro:include filename="$(find ur_description)/urdf/ur5.urdf.xacro" />
41   <!-- arm -->
42   <xacro:ur5_robot prefix="" joint_limited="true"
43     transmission_hw_interface="$(arg transmission_hw_interface)"  />
44
45   <joint name="base_joint" type="fixed">
46     <parent link="pedestal" />
47     <child link="base_link" />
48     <origin xyz="0 0 1" rpy="0.0 0.0 0.0"/>
49   </joint>
50 </robot>
```

Gazebo related Information

Define joint between world and the pedestal.

These informations are required for loading a functional UR5 model into your environment.

Define joint between pedestal and robot base_link

Step 3: Create a Launch File to load the robot into Gazebo

- Create a file called "`myur5.launch`" inside the folder `myur5_description/launch` package.
- Please see the following files to have a better understanding of what to include in this file:
  `~/catkin_ws/src/universal_robot/ur_gazebo/launch/ur5.launch`
- Basically, we load an *empty world* file. Spawn the robot model provided through the variable "*robot_description*" and load necessary *controller files* to get the gazebo model working.
- Now you can load the gazebo model using the following command:
  `$ roslaunch myur5_description myur5.launch`

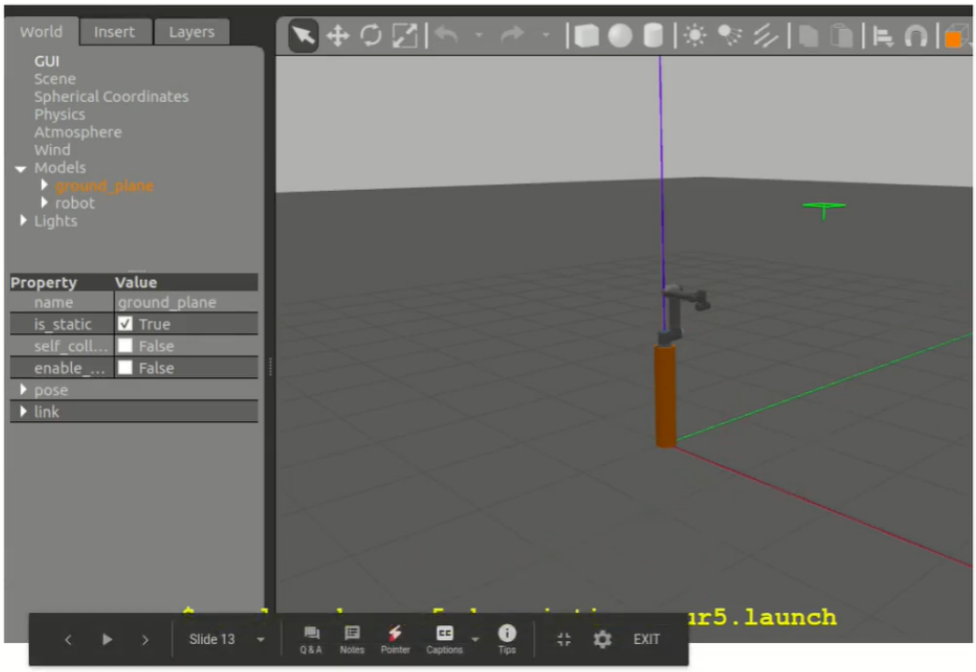
File: ~/catkin_ws/src/myur5sim/myur5_description/launch/myur5.launch

```
1  <?xml version="1.0"?>
2  <launch>
3    <arg name="limited" default="true"  doc="If true, limits joint range [-PI, PI] on all joints." />
4    <arg name="paused" default="false" doc="Starts gazebo in paused mode" />
5    <arg name="gui" default="true" doc="Starts gazebo gui" />
6    <arg name="transmission_hw_interface" default="hardware_interface/PositionJointInterface" />
7
8    <!-- startup simulated world -->
9    <include file="$(find gazebo_ros)/launch/empty_world.launch">
10     <arg name="world_name" default="worlds/empty.world"/>
11     <arg name="paused" value="$(arg paused)"/>
12     <arg name="gui" value="$(arg gui)"/>
13   </include>
14
15   <param name="robot_description" command="$(find xacro)/xacro '$(find myur5_description)/urdf/myur5.urdf.xacro'
   transmission_hw_interface:=$(arg transmission_hw_interface)" />
16
17   <!-- push robot_description to factory and spawn robot in gazebo -->
18   <node name="spawn_gazebo_model" pkg="gazebo_ros" type="spawn_model" args="-urdf -param robot_description -model robot -z 0.0"
   respawn="false" output="screen" />
19
20   <include file="$(find ur_gazebo)/launch/controller_utils.launch"/>
21
22   <!-- start this controller -->
23   <rosparam file="$(find ur_gazebo)/controller/arm_controller_ur5.yaml" command="load"/>
24   <node name="arm_controller_spawner" pkg="controller_manager" type="controller_manager" args="spawn arm_controller"
   respawn="false" output="screen"/>
25
26 </launch>
```

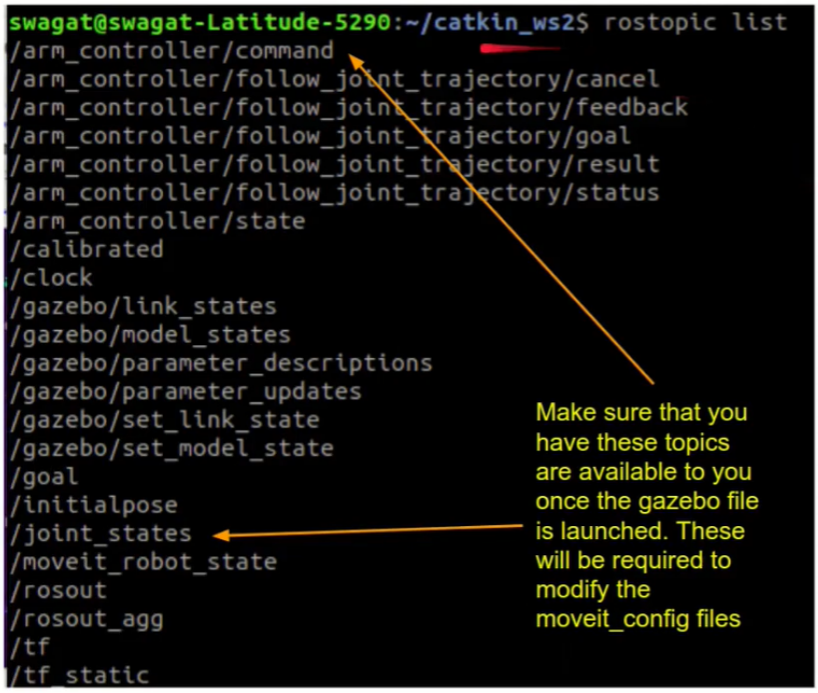
Load gazebo with empty world

Load the robot model into Gazebo

Load controllers



It still gives error saying “no p gain specified for pid” on the terminal. You can safely ignore this error.



Step 4: Create a Moveit_Config Package

- Follow the instructions available at this link (watch the video) to create a moveit package for the robot.
- This basically involves using `moveit_setup_assistant` to create a package called "`myur5_moveit_config`" inside the `~/catkin_ws/src/myur5sim/` folder.
- While defining the planning group, create a kinematic chain from pedestal to tool0.
- Create / Modify the following files as per the instruction provided in the above video:

```
~/catkin_ws/src/myur5sim/myur5_moveit_config/config/controllers.yaml
~/catkin_ws/src/myur5sim/myur5_moveit_config/config/joint_names.yaml
~/catkin_ws/src/myur5sim/myur5_moveit_config/launch/myur5_moveit_controller_manager.launch.xml
~/catkin_ws/src/myur5sim/myur5_moveit_config/launch/myur5_planning_execution.launch
```

https://www.theconstructsim.com/control-gazebo-simulated-robot-moveit-video-answer/

Define Virtual Joints

Create a virtual joint between a robot link and an external frame of reference (considered fixed with respect to the robot).

	Virtual Joint Name	Child Link	Parent Frame	Type
1	FixedBase	pedestal	world	fixed
2	RobotBase	base_link	pedestal	fixed

Main settings on moveit_setup_assistant manager

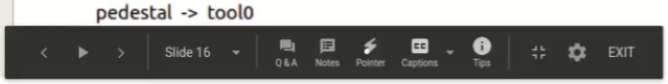
```
$ roslaunch moveit_setup_assistant setup_assistant.launch
```

Define Planning Groups

Create and edit 'joint model' groups for your robot based on joint collections, link collections, kinematic chains or subgroups. A planning group defines the set of (joint, link) pairs considered for planning and collision checking. Define individual groups for each subset of the robot you want to plan for.Note: when adding a link to the group, its parent joint is added too and vice versa.

Current Groups

- **manipulator**
  - *Joints*
  - *Links*
  - *Chain*
    - pedestal -> tool0



Define Robot Poses

Create poses for the robot. Poses are defined as planning groups. This is useful for things like *hor* each robot will be its initial pose in simulation.

	Pose Name	Group Name
1	AllZeros	manipulator
2	HomePose	manipulator

File:
~/catkin_ws/src/myur5sim/myur5_moveit_config/config/controllers.yaml

These are taken from “rostopic list” output while Gazebo is running.

This information is available in ur5.urdf.xacro file present in the universal_robot package.

```yaml
controller_list:
  - name: arm_controller
    action_ns: "follow_joint_trajectory"
    type: FollowJointTrajectory
    joints:
      - shoulder_pan_joint
      - shoulder_lift_joint
      - elbow_joint
      - wrist_1_joint
      - wrist_2_joint
      - wrist_3_joint
```

The joint names could also be provided in the form of a list as mentioned in the video.

File:
~/catkin_ws/src/myur5sim/myur5_moveit_config/config/joint_names.yaml

```yaml
controller_joint_names: [shoulder_pan_joint,
                         shoulder_lift_joint,
                         elbow_joint,
                         wrist_1_joint,
                         wrist_2_joint,
                         wrist_3_joint]
```

Modify the following File:

~/catkin_ws/src/myur5sim/myur5_moveit_config/launch/myur5_moveit_controller_manager.launch.xml

```
<launch>

  <!-- loads moveit_controller_manager on the parameter server which is taken as argument
    if no argument is passed, moveit_simple_controller_manager will be set -->
  <arg name="moveit_controller_manager" default="moveit_simple_controller_manager/MoveItSimpleControllerManager" />
  <param name="moveit_controller_manager" value="$(arg moveit_controller_manager)"/>
  <param name="use_controller_manager" value="false"/>
  <param name="trajectory_execution/execution_duration_monitoring" value="false"/>

  <!-- loads ros_controllers to the param server -->
  <!--rosparam file="$(find myur5_moveit_config)/config/ros_controllers.yaml"/-->
  <rosparam file="$(find myur5_moveit_config)/config/controllers.yaml"/>
</launch>
```

We include the “controllers.yaml” in this file. Probably, it is also possible to modify the ros_controllers.yaml file as well.

Create the following File:
~/catkin_ws/src/myur5sim/myur5_moveit_config/launch/myur5_planning_execution.launch

```
<?xml version="1.0"?>
<launch>
  <rosparam command="load" file="$(find myur5_moveit_config)/config/joint_names.yaml" />
  <include file="$(find myur5_moveit_config)/launch/planning_context.launch" >
    <arg name="load_robot_description" value="true" />
  </include>

  <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher">
    <param name="/use_gui" value="false"/>
    <rosparam param="/source_list">[/joint_states]</rosparam>
  </node>

  <include file="$(find myur5_moveit_config)/launch/move_group.launch">
    <arg name="publish_monitored_planning_scene" value="true"/>
  </include>

  <include file="$(find myur5_moveit_config)/launch/moveit_rviz.launch">
    <!--arg name="config" value="true"/--> <!-- this gives error-->
  </include-->
</launch>
```

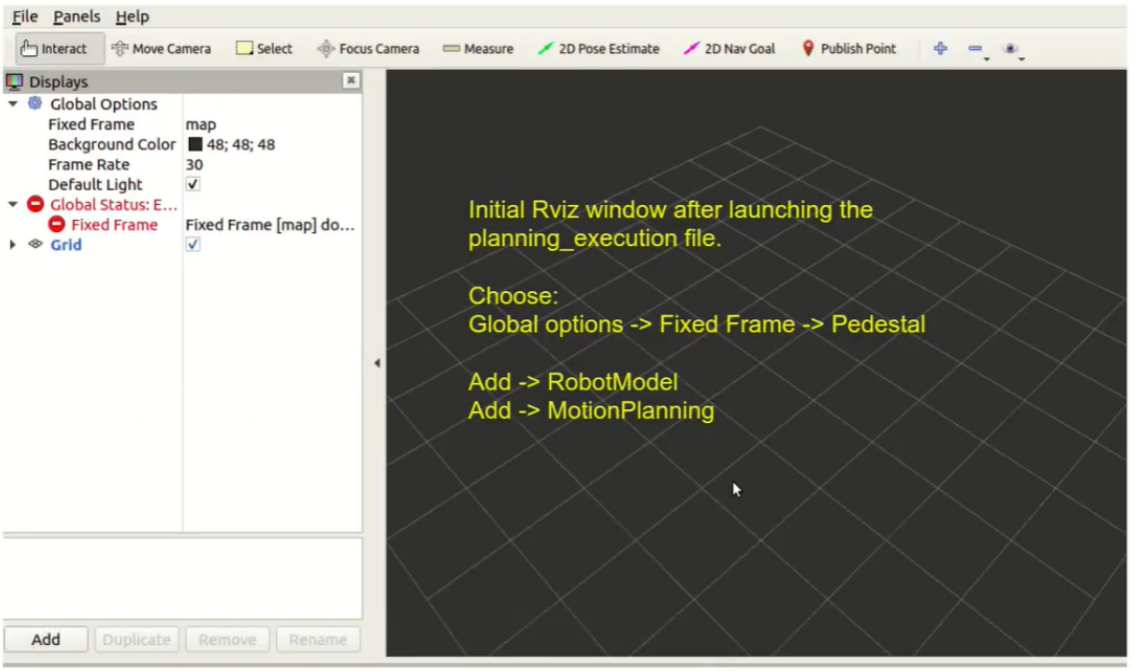
Notice these components.

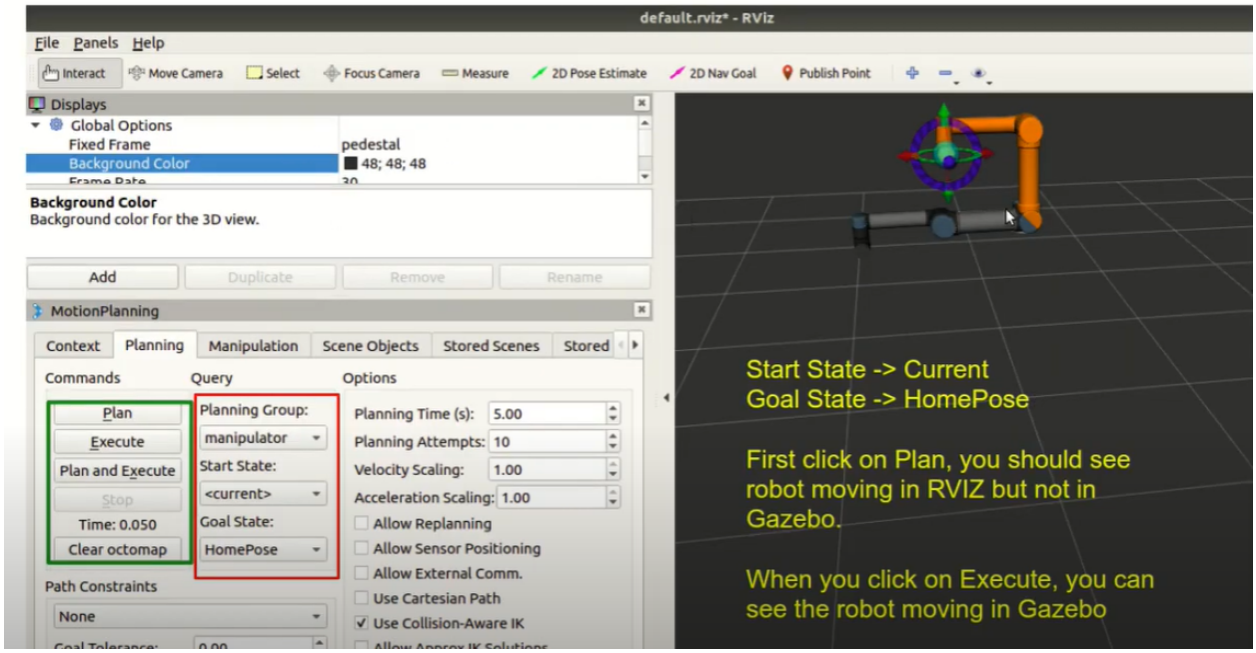
- Run the following command on a separate terminal while the Gazebo is running.

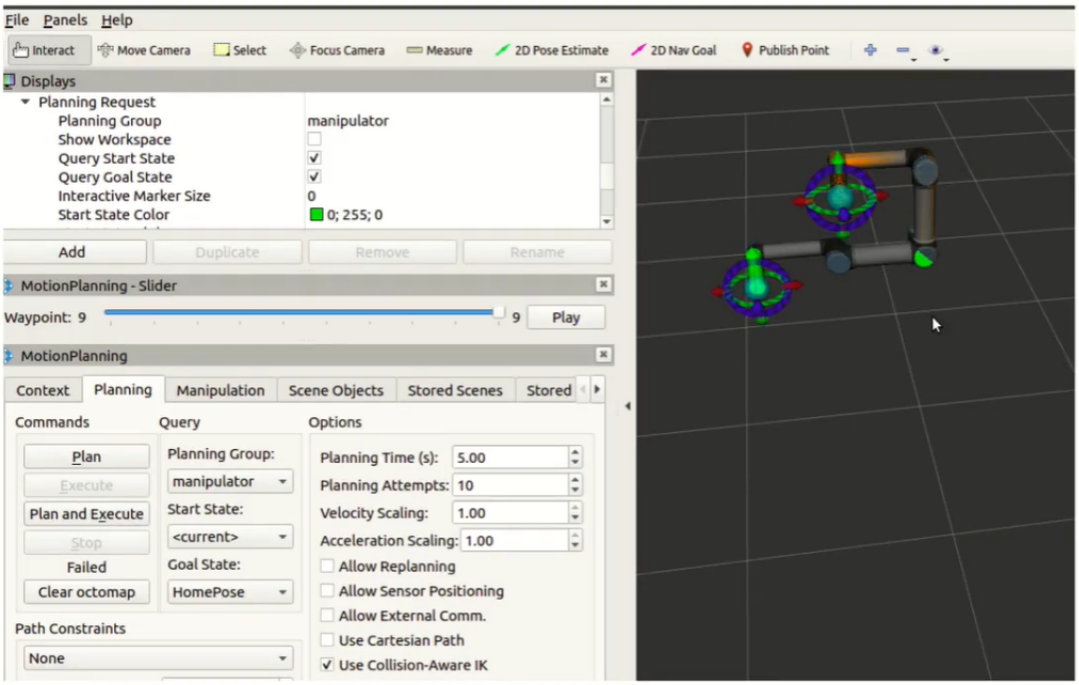
  ```
  $ roslaunch myur5_moveit_config myur5_planning_execution.launch
  ```

  It might give an error as follows, which can be ignored:
  ```
  [ERROR] [1600601614.145766164, 3039.711000000]: Could not find the planner configuration 'None' on the param server
  ```

- Now you should be able to plan motion in Rviz using *MotionPlanner* tool and execute them on the Gazebo Robot.
- Go through the Moveit RVIZ tutorial to know more about this topic.
- You can save the modified RVIZ environment (after including robot and tools) as a separate file “moveit.rviz” file.

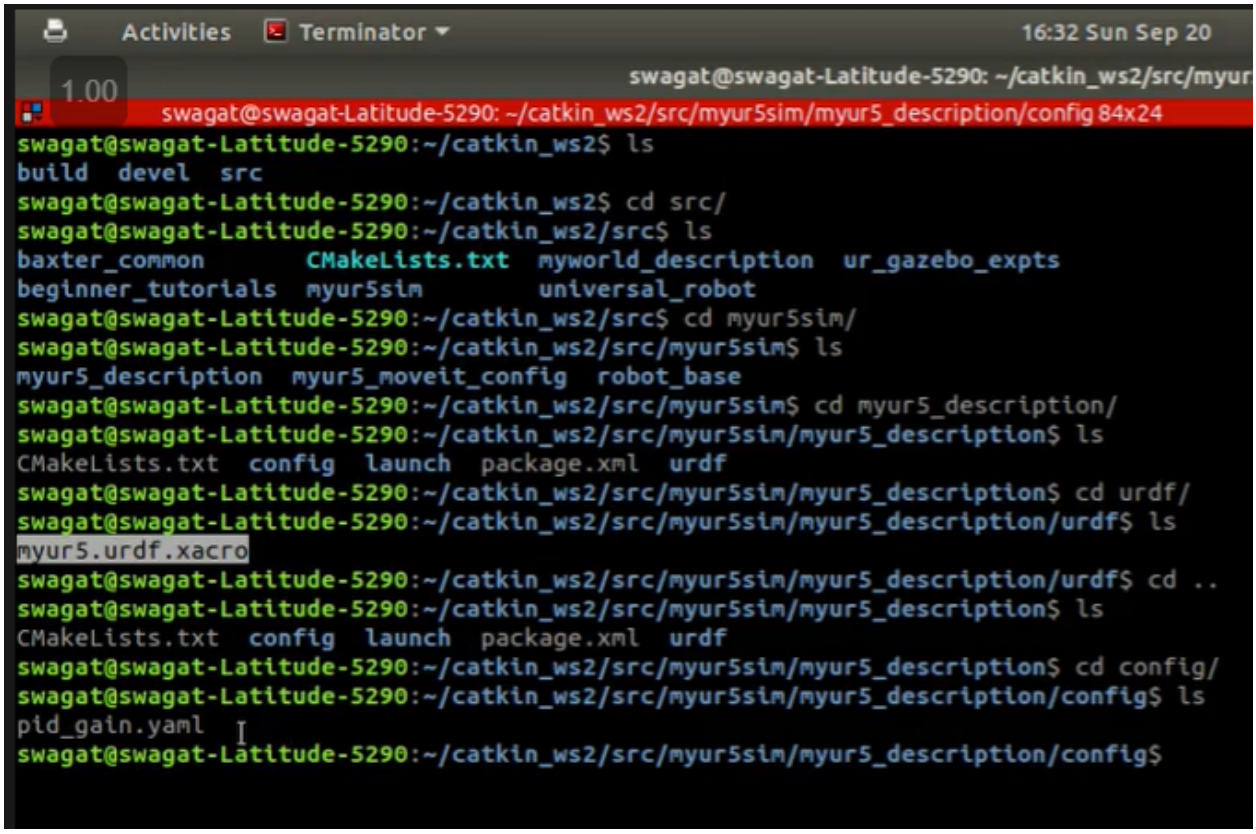


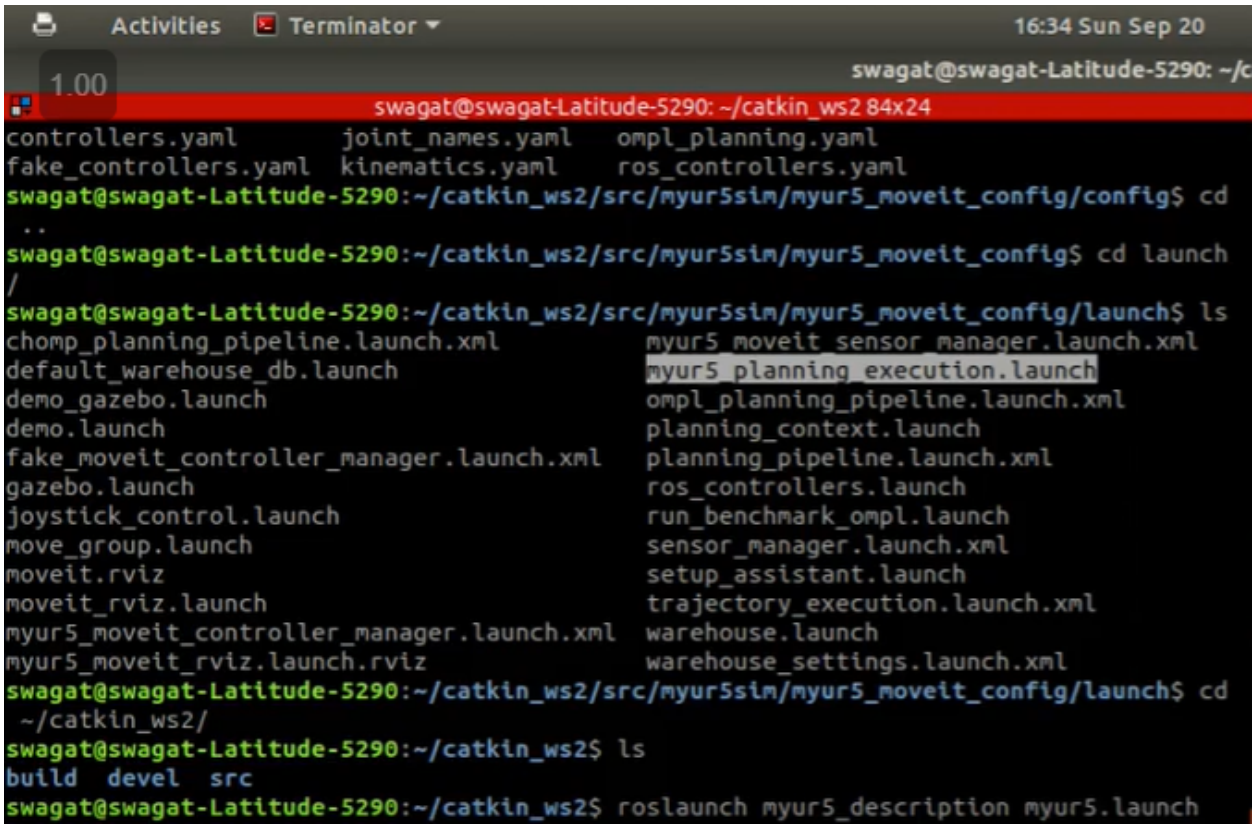

default.rviz* - RViz
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Global Options
Fixed Frame pedestal
Background Color 48; 48; 48
Background Color
Background color for the 3D view.
Add Duplicate Remove Rename
MotionPlanning
Context Planning Manipulation Scene Objects Stored Scenes Stored
Commands Query Options
Plan Execute Plan and Execute Stop Time: 0.050 Clear octomap
Planning Group: manipulator
Start State: <current>
Goal State: HomePose
Planning Time (s): 5.00
Planning Attempts: 10
Velocity Scaling: 1.00
Acceleration Scaling: 1.00
Allow Replanning
Allow Sensor Positioning
Allow External Comm.
Use Cartesian Path
Use Collision-Aware IK
Path Constraints
None
Start State -> Current
Goal State -> HomePose
First click on Plan, you should see robot moving in RVIZ but not in Gazebo.
When you click on Execute, you can see the robot moving in Gazebo

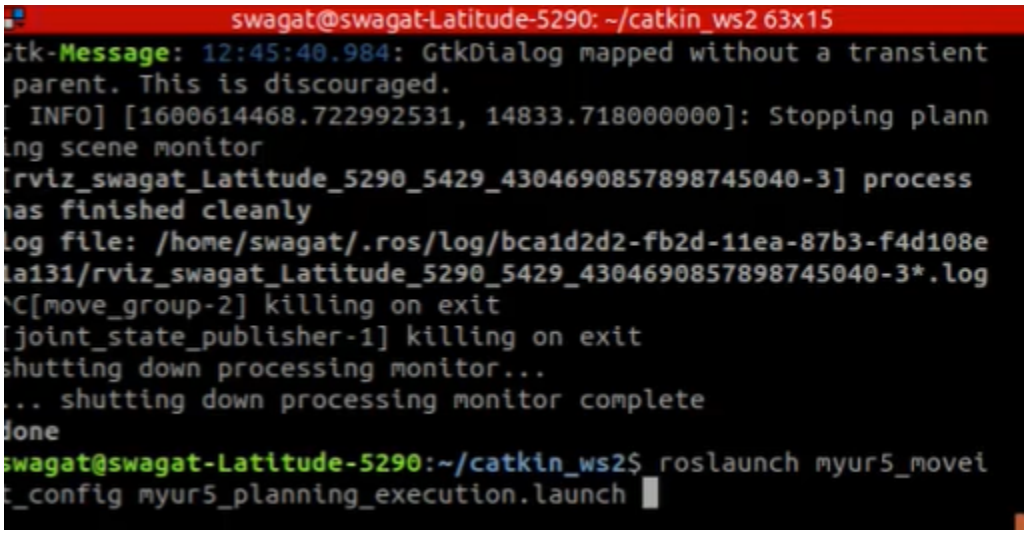

File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Planning Request
Planning Group manipulator
Show Workspace
Query Start State
Query Goal State
Interactive Marker Size 0
Start State Color 0; 255; 0
Add Duplicate Remove Rename
MotionPlanning - Slider
Waypoint: 9 9 Play
MotionPlanning
Context Planning Manipulation Scene Objects Stored Scenes Stored
Commands Query Options
Plan Execute Plan and Execute Stop Failed Clear octomap
Planning Group: manipulator
Start State: <current>
Goal State: HomePose
Planning Time (s): 5.00
Planning Attempts: 10
Velocity Scaling: 1.00
Acceleration Scaling: 1.00
Allow Replanning
Allow Sensor Positioning
Allow External Comm.
Use Cartesian Path
Use Collision-Aware IK
Path Constraints
None

Summary for Part 1

- We saw how to add a “pedestal” to our UR5 robot. This paves the way to include other components into our Gazebo simulation which will be demonstrated in the next part.
- Use Moveit to control robot motion in the Gazebo Simulation.
- To do list:
  - How to use Gazebo’s model editor to build models and include them in the simulation?
  - Include a mesh file for the robot pedestal which will be visible in RVIZ.




Activities
Terminator
16:34 Sun Sep 20
swagat@swagat-Latitude-5290: ~/c
1.00
swagat@swagat-Latitude-5290: ~/catkin_ws2 84x24
controllers.yaml joint_names.yaml ompl_planning.yaml
fake_controllers.yaml kinematics.yaml ros_controllers.yaml
swagat@swagat-Latitude-5290:~/catkin_ws2/src/myur5sim/myur5_moveit_config/config$ cd
..
swagat@swagat-Latitude-5290:~/catkin_ws2/src/myur5sim/myur5_moveit_config$ cd launch
/
swagat@swagat-Latitude-5290:~/catkin_ws2/src/myur5sim/myur5_moveit_config/launch$ ls
chomp_planning_pipeline.launch.xml myur5_moveit_sensor_manager.launch.xml
default_warehouse_db.launch myur5_planning_execution.launch
demo_gazebo.launch ompl_planning_pipeline.launch.xml
demo.launch planning_context.launch
fake_moveit_controller_manager.launch.xml planning_pipeline.launch.xml
gazebo.launch ros_controllers.launch
joystick_control.launch run_benchmark_ompl.launch
move_group.launch sensor_manager.launch.xml
moveit.rviz setup_assistant.launch
moveit_rviz.launch trajectory_execution.launch.xml
myur5_moveit_controller_manager.launch.xml warehouse.launch
myur5_moveit_rviz.launch.rviz warehouse_settings.launch.xml
swagat@swagat-Latitude-5290:~/catkin_ws2/src/myur5sim/myur5_moveit_config/launch$ cd
~/catkin_ws2/
swagat@swagat-Latitude-5290:~/catkin_ws2$ ls
build devel src
swagat@swagat-Latitude-5290:~/catkin_ws2$ roslaunch myur5_description myur5.launch


swagat@swagat-Latitude-5290: ~/catkin_ws2 63x15
Gtk-Message: 12:45:40.984: GtkDialog mapped without a transient
parent. This is discouraged.
[INFO] [1600614468.722992531, 14833.718000000]: Stopping plann
ing scene monitor
[rviz_swagat_Latitude_5290_5429_4304690857898745040-3] process
has finished cleanly
log file: /home/swagat/.ros/log/bca1d2d2-fb2d-11ea-87b3-f4d108e
a131/rviz_swagat_Latitude_5290_5429_4304690857898745040-3*.log
^C[move_group-2] killing on exit
[joint_state_publisher-1] killing on exit
shutting down processing monitor...
... shutting down processing monitor complete
done
swagat@swagat-Latitude-5290:~/catkin_ws2$ roslaunch myur5_movei
t_config myur5_planning_execution.launch