ABB-IRB120 TUTORIAL

REQUIREMENTS
Install the following softwares

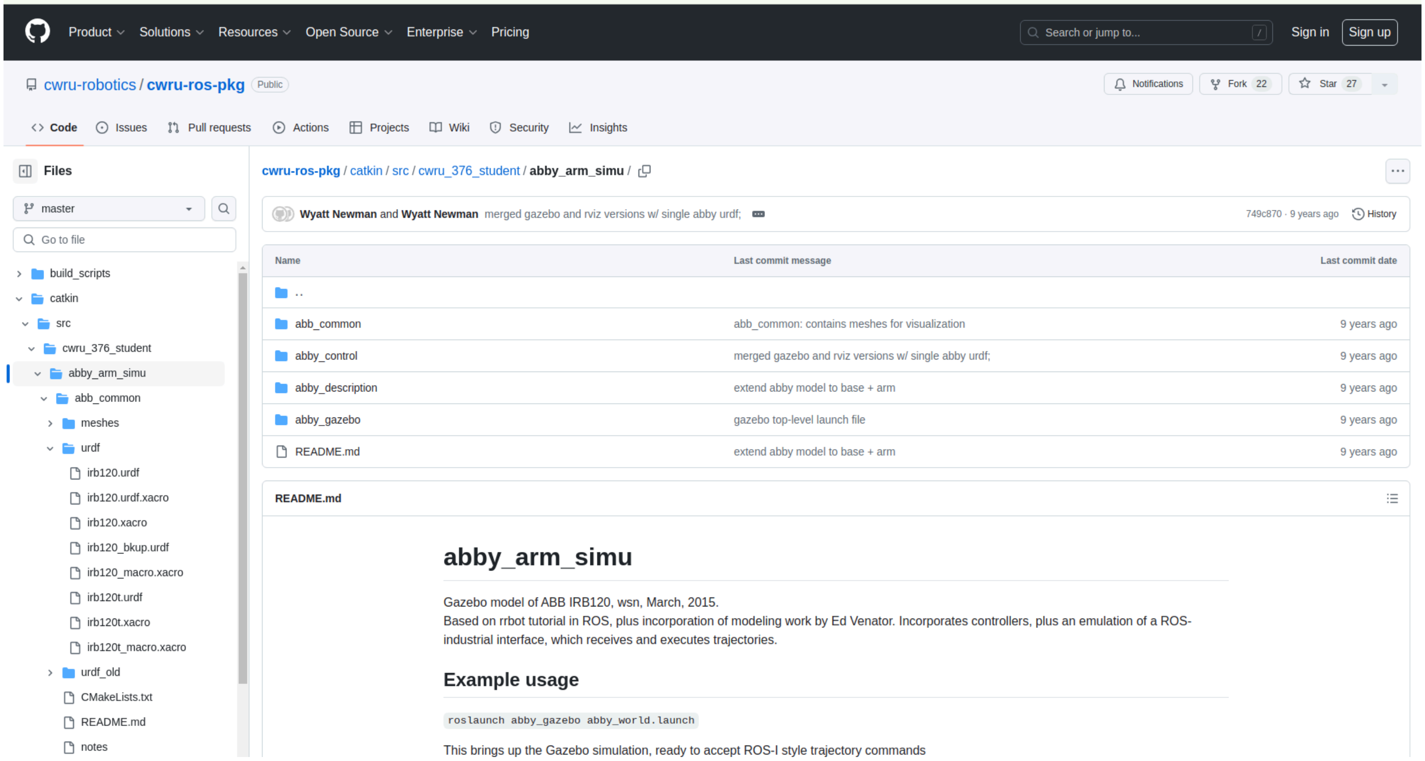
1. Ubuntu 20.04 LTS
2. ROS-noetic
3. Move-it
4. Gazebo

Step 01
Create Directory

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws
catkin_make
source ~/catkin_ws/devel/setup.bash
```

Step 02
Download and extract the directories from github

https://github.com/cwru-robotics/cwru-ros-pkg/tree/master/catkin/src/cwru_376_student/abby_arm_simu



cwru-robotics / cwru-ros-pkg Public

- build_scripts
- catkin
  - src
    - cwru_376_student
      - abby_arm_simu
        - abb_common
          - meshes
          - urdf
            - irb120.urdf
            - irb120.urdf.xacro
            - irb120.xacro
            - irb120_bkup.urdf
            - irb120_macro.xacro
            - irb120t.urdf
            - irb120t.xacro
            - irb120t_macro.xacro
          - urdf_old
          - CMakeLists.txt
          - README.md
          - notes

cwru-ros-pkg / catkin / src / cwru_376_student / abby_arm_simu /

Wyatt Newman and Wyatt Newman merged gazebo and rviz versions w/ single abby urdf; 749c870 · 9 years ago

Name	Last commit message	Last commit date
..		
abb_common	abb_common: contains meshes for visualization	9 years ago
abby_control	merged gazebo and rviz versions w/ single abby urdf;	9 years ago
abby_description	extend abby model to base + arm	9 years ago
abby_gazebo	gazebo top-level launch file	9 years ago
README.md	extend abby model to base + arm	9 years ago

README.md

abby_arm_simu

Gazebo model of ABB IRB120, wsn, March, 2015.
Based on rrbot tutorial in ROS, plus incorporation of modeling work by Ed Venator. Incorporates controllers, plus an emulation of a ROS-industrial interface, which receives and executes trajectories.

Example usage

```
roslaunch abby_gazebo abby_world.launch
```

This brings up the Gazebo simulation, ready to accept ROS-I style trajectory commands

Step 03

Copy/cut the following directories from the downloaded directory to your src directory then

```
cd ~/catkin_ws
catkin_make
source ~/catkin_ws/devel/setup.bash
```

Step 04

Create a new directory in your src directory irb120_moveit_config then open a new terminal where you will launch moveit_setup_assistant.

1. Open moveit_setup_assistant

```
roslaunch moveit_setup_assistant setup_assistant.launch
```

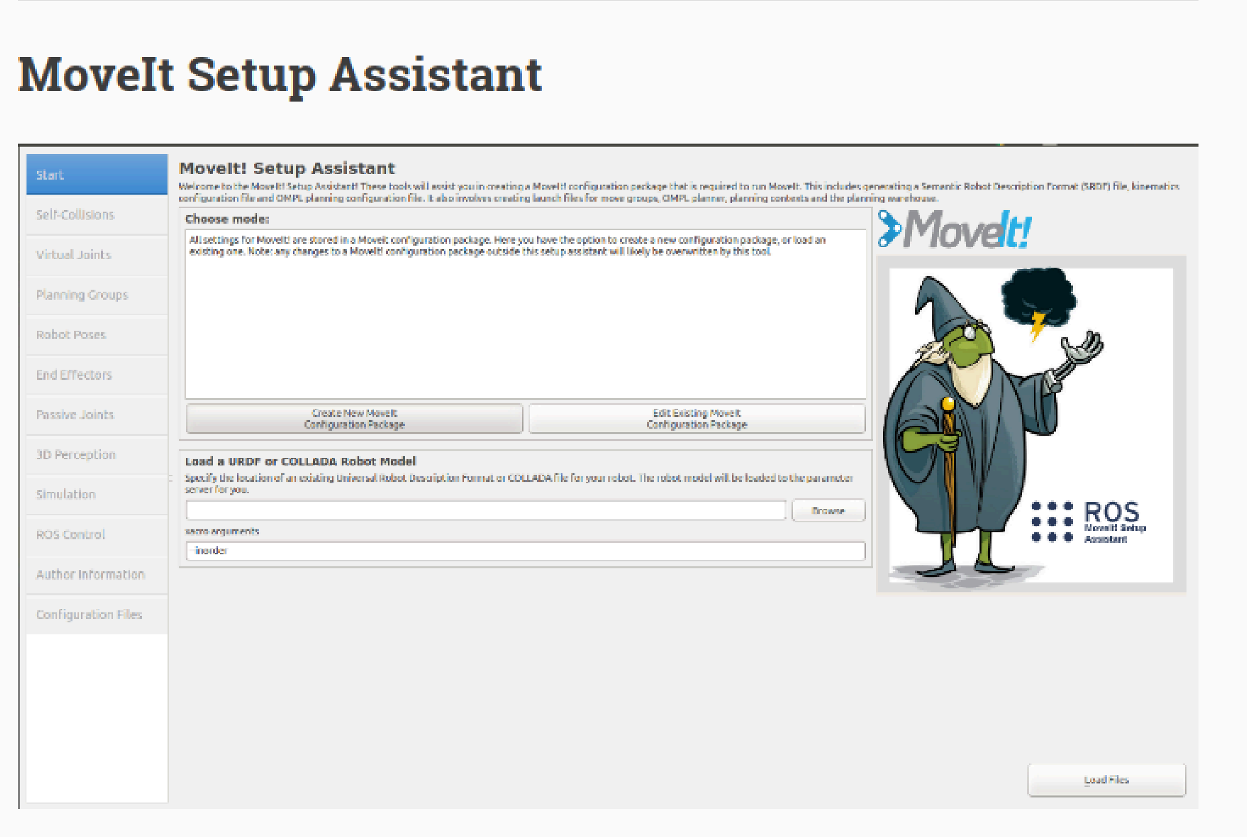
Then after follow the procedures as the following image shows

2. Click **Create New MoveIt Configuration Package**
   a. Load URDF file
   b. Click **Load Files** (Check whether proper robot or not)

irb120.urdf 8.6KB

3. Change to **Self-Collisions Tab**
   a. Click **Generate Collision Matrix**
4. Change to **Planning-Groups Tab**
   a. Click **Add Group**
   b. Fill in blanks
      a. Group Name: (ex. irb120)
      b. kinematic Solver: kdl_kinematics_plugin/KDLKinematicsPlugin
   c. Click **Add Joints**
      a. Click **world_joint, joint_1, joint_2, joint_3, joint_4, joint_5, joint_6**
      b. Click **>**
      c. Click **Save**
5. Change to **Robot Poses Tab**
   a. Click **Add Pose**
      i. Set desired joint angle
      ii. Fill in **Pose Name (ex. main)**
6. Change to **Author Information Tab**
   a. Fill in Name and Email
7. Change to **Configuration Files Tab**
   a. Set **Configuration package save Path**

   ex. /home/(username)/(catkin workspace name)/src/irb120_moveit_config
8. Click **Generate Package**
9. Click **Exit Setup Assistant**



MoveIt Setup Assistant

MoveIt! Setup Assistant

Welcome to the MoveIt! Setup Assistant! These tools will assist you in creating a MoveIt! configuration package that is required to run MoveIt. This includes generating a Semantic Robot Description Format (SRDF) file, kinematics configuration file and OMPL planning configuration file. It also involves creating launch files for move groups, OMPL planner, planning contexts and the planning warehouse.

Choose mode:

All settings for MoveIt! are stored in a MoveIt configuration package. Here you have the option to create a new configuration package, or load an existing one. Note: any changes to a MoveIt! configuration package outside this setup assistant will likely be overwritten by this tool.

Load a URDF or COLLADA Robot Model

Specify the location of an existing Universal Robot Description Format or COLLADA file for your robot. The robot model will be loaded to the parameter server for you.

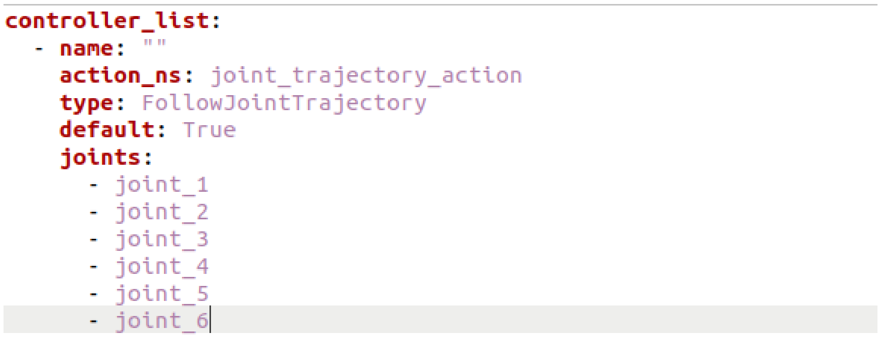
xacro arguments: --inorder

In **ROS Control** set auto

After finish all the step you can run demo.launch
roslaunch irb120_moveit_config demo.launch

To run a real robot, you need to Install ROS ABB Drivers, Installation command are available in notion

-keep follow notion procedure but you need to make some changes in config directory, yaml file simple_moveit_controllers.yaml it should look like this

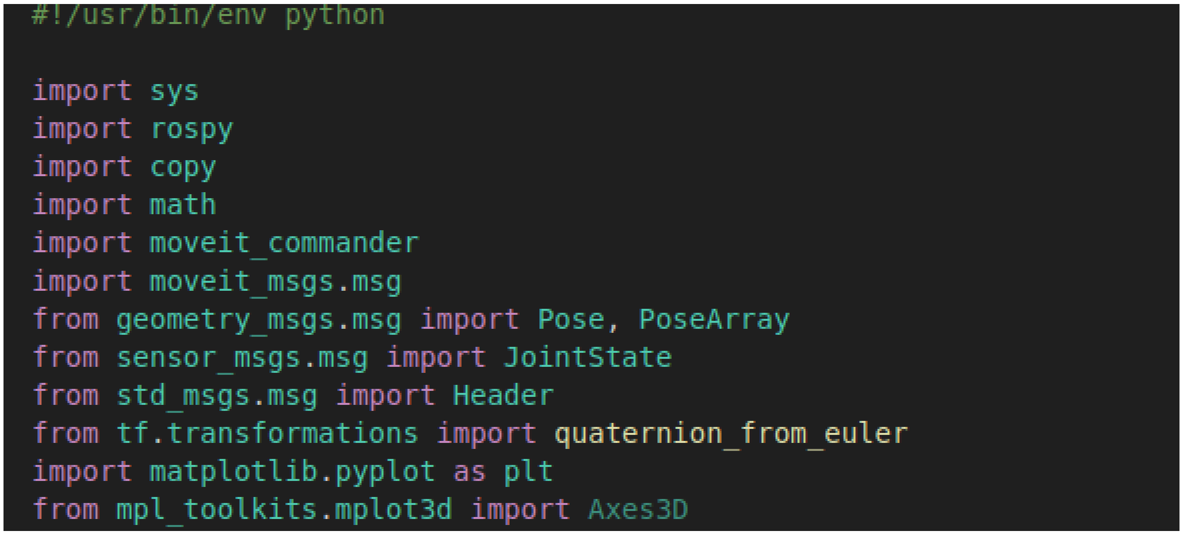


```
controller_list:
  - name: ""
    action_ns: joint_trajectory_action
    type: FollowJointTrajectory
    default: True
    joints:
      - joint_1
      - joint_2
      - joint_3
      - joint_4
      - joint_5
      - joint_6
```

Run an ABB-irb120 robot in gazebo simulation or real robot by Python script

Step 01

In your Python script, import the necessary files that you want to use and make sure you have launched your robot, it's important to test on demo.launch before launching the real robot. Example of import code.



```
#!/usr/bin/env python

import sys
import rospy
import copy
import math
import moveit_commander
import moveit_msgs.msg
from geometry_msgs.msg import Pose, PoseArray
from sensor_msgs.msg import JointState
from std_msgs.msg import Header
from tf.transformations import quaternion_from_euler
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

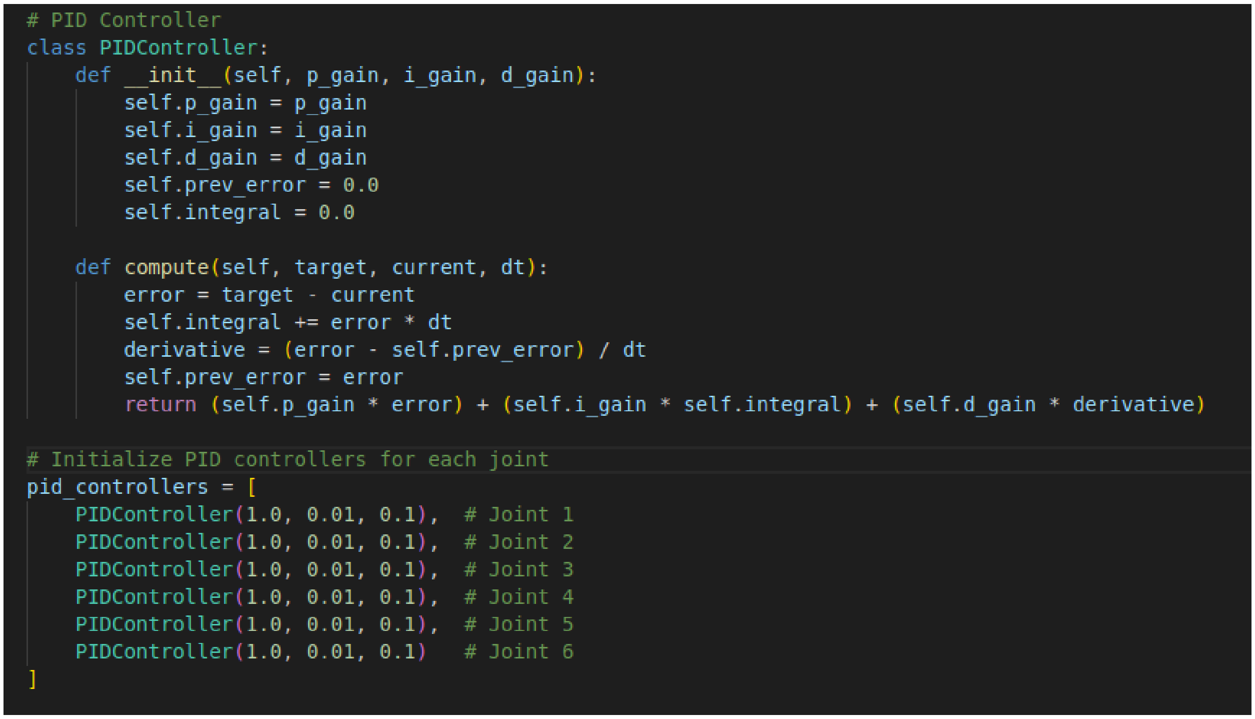
Step 02

Initialize moveit commander.

Step 03

Depending on the task that you want the robot to perform, it’s important to set a PID Controller in each joint to control the robot. Start by defining the class inside your python script or you can define it separately.

Example



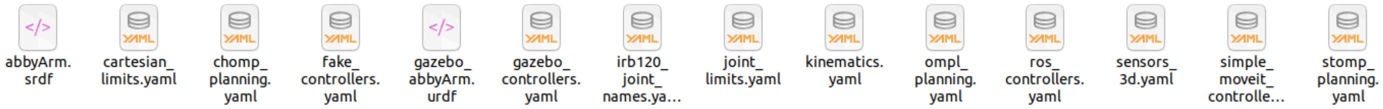
```
# PID Controller
class PIDController:
    def __init__(self, p_gain, i_gain, d_gain):
        self.p_gain = p_gain
        self.i_gain = i_gain
        self.d_gain = d_gain
        self.prev_error = 0.0
        self.integral = 0.0

    def compute(self, target, current, dt):
        error = target - current
        self.integral += error * dt
        derivative = (error - self.prev_error) / dt
        self.prev_error = error
        return (self.p_gain * error) + (self.i_gain * self.integral) + (self.d_gain * derivative)

# Initialize PID controllers for each joint
pid_controllers = [
    PIDController(1.0, 0.01, 0.1),  # Joint 1
    PIDController(1.0, 0.01, 0.1),  # Joint 2
    PIDController(1.0, 0.01, 0.1),  # Joint 3
    PIDController(1.0, 0.01, 0.1),  # Joint 4
    PIDController(1.0, 0.01, 0.1),  # Joint 5
    PIDController(1.0, 0.01, 0.1)   # Joint 6
]
```

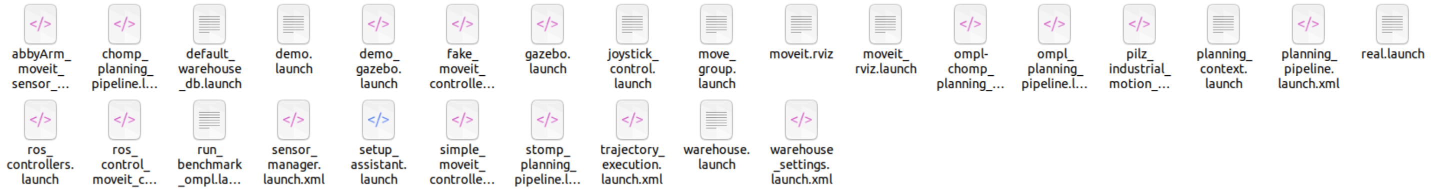
Note: Make sure your planning group name in python script is similar to the name that you set in the Moveit plan group.

Your irb120_moveit_config files should look like this



abbyArm.srdf
cartesian_limits.yaml
chomp_planning.yaml
fake_controllers.yaml
gazebo_abbyArm.urdf
gazebo_controllers.yaml
irb120_joint_names.ya…
joint_limits.yaml
kinematics.yaml
ompl_planning.yaml
ros_controllers.yaml
sensors_3d.yaml
simple_moveit_controlle…
stomp_planning.yaml

Your launch files should look like this


abbyArm_moveit_sensor_...
chomp_planning_pipeline.l...
default_warehouse_db.launch
demo.launch
demo_gazebo.launch
fake_moveit_controlle...
gazebo.launch
joystick_control.launch
move_group.launch
moveit.rviz
moveit_rviz.launch
ompl-chomp_planning_...
ompl_planning_pipeline.l...
pilz_industrial_motion_...
planning_context.launch
planning_pipeline.launch.xml
real.launch
ros_controllers.launch
ros_control_moveit_c...
run_benchmark_ompl.la...
sensor_manager.launch.xml
setup_assistant.launch
simple_moveit_controlle...
stomp_planning_pipeline.l...
trajectory_execution.launch.xml
warehouse.launch
warehouse_settings.launch.xml

To get all the directories and python script file
Clone the following repository
https://github.com/engjanelaurent/ABB-IRB120_Noetic.git