**Lab -3**

**Working with Python-ROS API and a Simple Pick-and-Place Task**

**Introduction :**

We have executed a simple pick-and-place task by manipulating individual joint positions and end-effector trajectories by using the ROS interface and Python -Ros interface

**Required software’s**

* **Hardware:** Interbotix PX100 robot arm
* **Software:** ROS2, Python, and Visual Studio Code
* **Simulation Environment:** RViz

**Procedure**

1. **Launch Robot Arm:**
   * Open a terminal and launch the robot arm in simulation using the ros2 launch interbotix\_xsarm\_control xsarm\_control.launch.py robot\_model:=px100 use\_sim:=true command.

A computer screen with a blue square

Description automatically generated

1. **Control Joint Positions:**
   * Create a Python script (joint\_control\_px100.py) to control individual joint positions.
   * Use the InterbotixManipulatorXS class to interact with the robot arm.
   * Set desired joint positions in radians using the set\_joint\_positions() method.

A computer screen with text and images

Description automatically generated

1. **Perform Pick-and-Place Task:**
   * Create a Python script (pick\_and\_place.py) to execute a pick-and-place task.
   * Utilize the set\_ee\_cartesian\_trajectory(), grasp(), and release() methods to control the end-effector's position and the gripper's actions.
   * Test the script in simulation and then on the physical robot.A computer screen with text on it

     Description automatically generated

**Results**

* **Joint Position Control:** The robot successfully moved its joints to specified positions, demonstrating accurate control over individual joints.
* **Pick-and-Place Task:** The robot successfully completed the pick-and-place task, demonstrating its ability to grasp, lift, and move objects.

**Discussion**

* **Python-ROS API:** The Python-ROS API provides a user-friendly interface for controlling robotic systems. It simplifies tasks such as joint position control and end-effector manipulation.
* **Task Planning:** The pick-and-place task involved careful planning of joint positions and end-effector trajectories to ensure accurate object manipulation.
* **Simulation:** Using simulation was crucial for testing and debugging the code without risking damage to the physical robot.

**Conclusion**

We did the lab successfully demonstrated the capabilities of the Python-ROS API for controlling a robotic arm. By controlling individual joint positions and executing a pick-and-place task, we gained valuable insights into the potential applications of robotics in various domains.

Note : we have faced a small issue of installing all python packages