PES UNIVERSITY, Bangalore

(Established under Karnataka Act No. 16 of 2013)

Department of Computer Science & Engineering

Mobile and Autonomous Robots (UE22CS343BB7)

6th Semester

Mini-Project

Project Title: Automated Object Handling and Assembly using an Industrial Robotic Arm

Team Details:

1.Harshita gujjar	**PES1UG23CS810**
2.Hemavathi VM	**PES1UG23CS811**
3.Bindu S	**PES1UG23CS840**
4.Diganth SD	**PES1UG23CS841**

Professor Name: Dr. Ashok Kumar Patil

Project Description:

This project involves simulating an automated pick-and-place system within a virtual environment using ROS , Gazebo, and RViz. The system comprises a robotic arm, a moving conveyor belt, various objects, and designated bins. The robotic arm is programmed to identify objects on the conveyor belt, pick them up, and place them into the appropriate bins based on their characteristics.

The simulation leverages Gazebo for realistic physics and environment modeling, while RViz provides visualization capabilities. ROS serves as the middleware facilitating communication between different components of the system.

Project Objectives:

- Simulate a Conveyor Belt System
- Object Placement and Identification
- Robotic Arm Manipulation
- Integration with ROS
- Visualization and Monitoring

Methods and Materials:

1. System Design :

- Architecture Overview: A ROS system integrating Gazebo for physics simulation and RViz for visualization. Components communicate via ROS topics/services.

- Components:

- **Conveyor Belt Module**: Gazebo model + plugin to generate continuous belt motion.
- **Object Spawner:** Python node that periodically spawns object types at the upstream end of the belt.
- **Perception & Detection:** Simplified "color-based" detection: each spawned model carries a unique ROS topic or frame name; the pick-and-place node subscribes to the model states topic and infers type from model name.
- **Robot Controller:** MoveIt –enabled Python node that plans and executes pick-and-place trajectories for a UR5 arm.
- **Sorting Bins:** Bin models positioned around the arm; each bin's location is hard-coded in the controller.

2. Algorithm/Model Development

- **Object Identification Algorithm:**

1. Subscribe to /gazebo/model_states.
2. Filter for models matching our objects.
3. Select the nearest object whose x-position crosses a pick threshold.
4. Map model name → bin index.

- **Pick-and-Place Workflow:**

1. **Approach:** Generate approach pose above the object.
2. **Grasp:** Descend, close gripper (simulated), ascend.
3. **Transport:** Plan via MoveIt to above target bin.
4. **Release:** Open gripper (simulated), retract to safe pose.

- **Collision Avoidance:** Use MoveIt's built-in planning scene to avoid collisions with belt and bins.

3. Implementation Steps

1. **Workspace & Package Setup:**

- ros_ws/src/pick_and_place_sim with package.xml, setup.py, etc.

2. Modeling:

- Write urdf/robot.urdf.xacro for UR5.
- Define object meshes/primitives in urdf/objects.urdf.xacro.
- Build models/conveyor_belt with a Gazebo plugin for belt motion.
- Create worlds/pick_and_place.world.

3. Node Development (Python/rclpy):

- **object_spawner.py**: spawns objects at intervals.
- **conveyor_control.py**: toggles belt plugin via ROS service.
- **robot_controller.py**: subscribes to model states, calls MoveIt 2 APIs for motion.

4. Launch & Configuration:

- **launch/simulation.launch.py**: starts Gazebo world, belt plugin, spawner, controller.
- **config/rviz_config.rviz**: pre-configured RViz with robot, TF, and planning markers.

5. Build & Test:

- colcon build --symlink-install → source install/setup.bash → ros launch pick_and_place_sim simulation.launch.py.
- Monitor in Gazebo/RViz and iterate fixes.

4. Software Tools

- **ROS** (rclpy, ament_python, ros_control)
- **Gazebo** (sdf 1.6, gazebo_ros plugin)
- **MoveIt** (for motion planning and collision avoidance)
- **RViz** (visualization of robot, planning scene, TF frames)
- **Python 3** (scripts and nodes)
- **colcon** (build system)
- **Git** (version control)

Project Outcome:

1. Output results
    - Belt runs continuously
    - Robot correctly sorts all object types into respective bins without collision.

2. Simulation video link (drive link)

   https://drive.google.com/file/d/1naRYwHTipeixgiPiQ-pdGLZuBcX1TCQh/view

3. GitHub link (Source code)

   https://github.com/Diganthshiri/pick_and_place_Industrial-Robotic-Arm.git

References:

- MoveIt 2 Tutorials, ROS 2 Documentation: https://moveit.ros.org
- ROS Installation Guide: https://docs.ros.org/en/humble/Installation.html
- IFRA Conveyor Belt Plugin: https://github.com/IFRACranfield/IFRA_ConveyorBelt
- Automatic Addison, “Pick-and-Place Task using MoveIt 2 and Perception in ROS 2 Jazzy,” 2023.