TurtleBot3 Autonomy (No-Sim) — Architecture Overview

1) Exploration & SLAM
- Launch file starts slam_toolbox or static map server.
- Map (PGM+YAML) supplied from demo; ready to plug into Gazebo/RViz.
- Output: OccupancyGrid (/map).

2) Agentic Semantics (Mocked)
- semantic_mapper node provides ROS2 service /get_location (String → JSON pos
- Backed by semantic_mock.json; replace with VLM (CLIP/OpenAI) + TB3 camera la
- Output: Pose of label (x,y,theta).

3) Planner (RRT)
- rrt_planner node loads YAML+PGM, inflates obstacles, computes path (nav_msg
- Publishes /plan for RViz; offline script renders artifacts/rrt_path_example

Data Flow:
Map → RRT → Path
           ↑
Label → Semantics → Pose → Goal

Reviewer Run Order (Linux with ROS 2 Humble)
1) colcon build && source install/setup.bash
2) ros2 launch tb3_exploration slam_explore.launch.py map_yaml:=maps_demo/offi
3) ros2 run tb3_semantics semantic_mapper
4) ros2 run tb3_rrt_planner rrt_planner --ros-args -p map_yaml:=maps_demo/offi
5) RViz: add /map (OccupancyGrid), /plan (Path).

Offline Proof (Windows)
- python scripts/demo_rrt_local.py --map maps_demo/office.yaml --start 0.5 0.5
- python scripts/demo_semantic_query.py --label toilet
Artifacts:
- artifacts/occupancy_grid_preview.png
- artifacts/rrt_path_example.png
- artifacts/semantic_queries_demo.txt