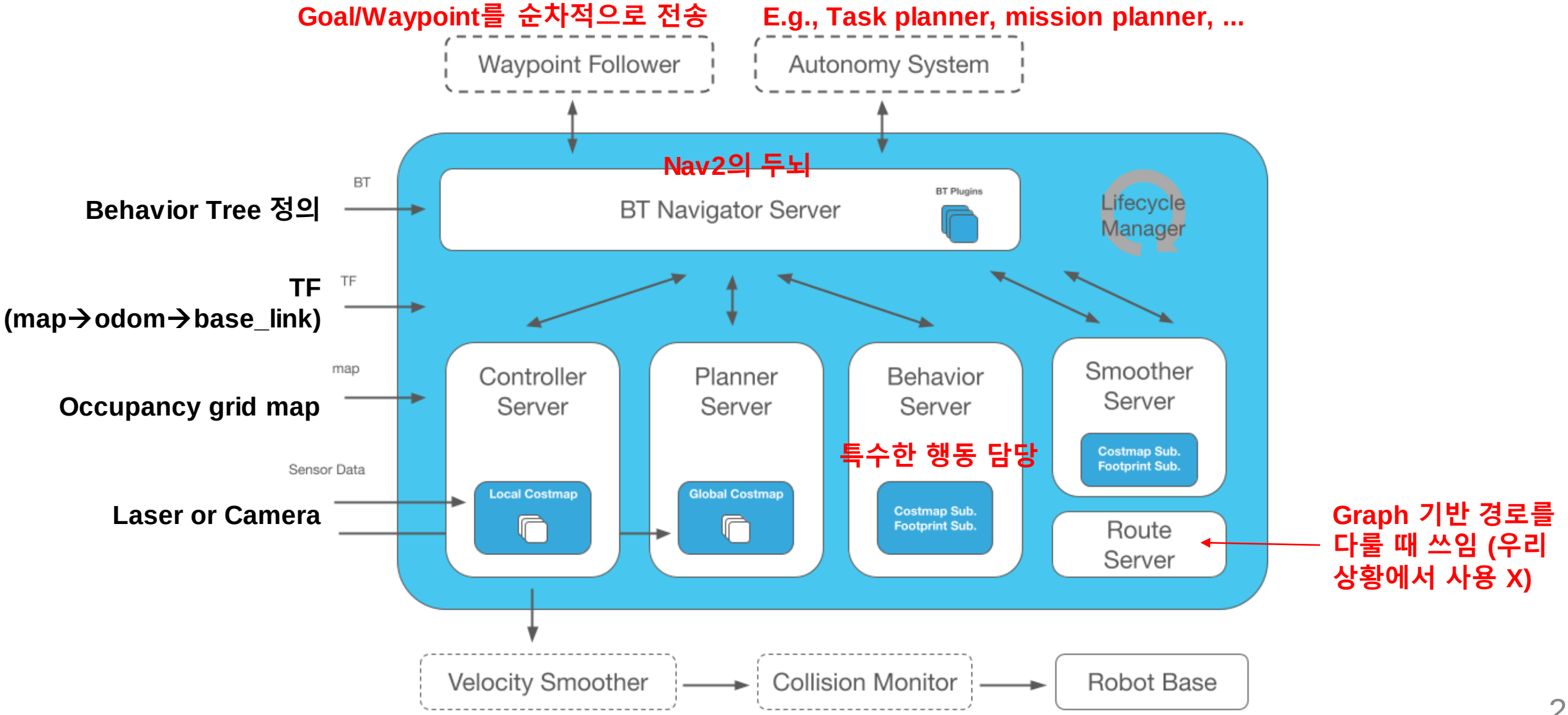
Turtlebot4 Nav2 API

운영체제의 실제

안인규 (Inkyu An)

Navigation2 Overview



Nav2 API

- The goal of the Nav2 Simple (Python3) Commander is to provide a “navigation as a library” capability to Python3 users
- Nav2 provides an API that handles all the ROS 2 and Action Server tasks for us such that we can focus on building an application leveraging the capabilities of Nav2
- Nav2 also provides use with demos and examples of API usage to build common basic capabilities in autonomous mobile robots
- **We can use three types of APIs**
  - Commander API: 기본적인 navigation API 제공
  - Costmap API: costmap 2d를 위한 API 제공
  - Footprint Collision Checker API: 충돌 탐지 관련 API 제공

Nav2 – Commander API

Functions	Description
setInitialPose(initial_pose: PoseStamped)	- Localization(AMCL/SLAM)에 로봇의 초기 자세를 알려줌. 보통 frame_id='map'인 PoseStamped를 넣음. - 만약 외부 SLAM (예: SLAM_Toolbox)를 활용한다면, 초기화 필요 없음.
waitUntilNav2Active(navigator='bt_navigator', localizer='amcl')	- Nav2가 완전히 onlin이 되고, lifecycle nodes가 모두 active state가 될 때까지 기다림. - 만약 외부 localizer (예: SLAM_Toolbox)를 사용한다면, localizer를 설정해야 함.
getPath(start: PoseStamped, goal: PoseStamped, planner_id: str = '', use_start: bool = False)	- start → goal 사이의 nav_msgs/Path를 한 번 계산해서 반환. - planner_id로 특정 플래너를 선택할 수 있음.
smoothPath(path: Path, smoother_id: str = '', max_duration: float = 2.0, check_for_collision: bool = False)	- 주어진 Path를 smoother plugin으로 부드럽게 보정. - 필요시 충돌 체크도 가능.
goToPose(pose: PoseStamped, behavior_tree: str = '')	- 하나의 목표 pose로 이동하라고 Nav2에 요청. - 비동기 함수라 바로 return 되고, 진행 상황은 아래 함수들로 확인.
isTaskComplete(task: RunningTask = RunningTask.NONE) -> bool	- 현재 navigation task가 끝났으면 True, 진행 중이면 False 반환. - 메인 루프 조건으로 많이 사용.
getFeedback(task: RunningTask = RunningTask.NONE)	- 진행 중인 task의 action feedback을 반환. - navigation_duration, 남은 거리 등으로 타임아웃/상태 체크할 때 사용.
cancelTask(task: RunningTask = RunningTask.NONE)	- 현재 진행 중인 navigation task를 취소. (특정 task ID를 넘길 수도 있음.)
getResult(task: RunningTask = RunningTask.NONE)	- task 종료 후 최종 result를 반환 (보통 TaskResult). - SUCCEEDED / CANCELED / FAILED와 비교해서 분기 처리.

https://docs.nav2.org/commander_api/index.html#id1

Nav2 – Commander API

```python
from nav2_simple_commander.robot_navigator import BasicNavigator
import rclpy

rclpy.init()
nav = BasicNavigator()
# ...
nav.setInitialPose(init_pose)
navigator.waitUntilNav2Active(localizer='controller_server')
# ...
path = nav.getPath(init_pose, goal_pose)
smoothed_path = nav.smoothPath(path)
# ...
nav.goToPose(goal_pose)
while not nav.isTaskComplete():
  feedback = nav.getFeedback()
  if feedback.navigation_duration > 600:
    nav.cancelTask()
# ...
result = nav.getResult()
if result == TaskResult.SUCCEEDED:
    print('Goal succeeded!')
elif result == TaskResult.CANCELED:
    print('Goal was canceled!')
elif result == TaskResult.FAILED:
    print('Goal failed!')

```

Localization의 초기 위치 지정

Nav2가 online이 되길 기다림

경로 반환

부드러운 경로 생성

목적지까지 이동 명령

Navigation task가 완료될 때 까지 기다림

결과 확인

Nav2 – Costmap API

Functions	Description
PyCostmap2D(occupancy_map: nav_msgs.msg.OccupancyGrid)	OccupancyGrid 메시지로부터 PyCostmap2D 객체 생성. 내부에 costmap 데이터, 해상도, origin, frame_id, timestamp 등을 저장.
getSizeInCellsX() -> int	맵의 X 방향 셀 개수(폭, cell 단위)를 반환.
getSizeInCellsY() -> int	맵의 Y 방향 셀 개수(높이, cell 단위)를 반환.
getSizeInMetersX() -> float	맵의 X 방향 크기를 미터 단위로 반환. (size_x * resolution)
getSizeInMetersY() -> float	맵의 Y 방향 크기를 미터 단위로 반환.
getOriginX() -> float	월드 좌표계에서 맵의 origin X [m]. (OccupancyGrid origin.x)
getOriginY() -> float	월드 좌표계에서 맵의 origin Y [m].
getResolution() -> float	맵 해상도 [m/cell].
getGlobalFrameID() -> str	이 costmap이 표현되는 전역 frame ID (예: "map", "odom").
getCostmapTimestamp() -> builtin_interfaces.msg.Time	costmap이 생성된 시각(타임스탬프) 반환.
getIndex(mx: int, my: int) -> int	map 좌표 (mx, my)를 1D 인덱스로 변환. (index = my * size_x + mx 형태)
getCostXY(mx: int, my: int) -> np.uint8	맵 좌표 (mx, my)에 해당하는 셀의 비용(cost)을 반환.
getCostIdx(index: int) -> np.uint8	1D 인덱스로 셀의 비용을 반환. getCostXY 내부에서도 사용.
setCost(mx: int, my: int, cost: np.uint8) -> None	맵 좌표 (mx, my)에 해당하는 셀의 비용을 설정. 커스텀 마킹/필터링에 사용 가능.
mapToWorld(mx: int, my: int) -> tuple[float, float]	맵 좌표 (mx, my)를 월드 좌표 (wx, wy) [m]로 변환. (origin + index * resolution)
worldToMap(wx: float, wy: float) -> tuple[int, int]	월드 좌표 (wx, wy)를 맵 좌표 (mx, my)로 변환 (맵 밖이면 예외/에러 조건 처리).

https://docs.nav2.org/commander_api/index.html#costmap-api

Nav2 – Costmap API

```python
from nav2_simple_commander.robot_navigator import BasicNavigator
from nav2_simple_commander.costmap_2d import PyCostmap2D

nav = BasicNavigator()
nav.waitUntilNav2Active()

global_costmap_msg = nav.getGlobalCostmap()

cm = PyCostmap2D(global_costmap_msg)

mx, my = cm.worldToMap(1.0, 2.0)
cost = cm.getCostXY(mx, my)


```

Get global costmap (msgs/msg/OccupancyGrid)

Wrapping with PyCostmap2D

Costmap 상 좌표 확인

해당 좌표의 cost 확인

https://docs.nav2.org/commander_api/index.html#costmap-api

Nav2 – Footprint Collision Checker API

Functions	Description
footprintCost(footprint: Polygon)	- 주어진 footprint(다각형)을 현재 costmap 상에서 충돌 여부를 검사. - footprint의 암묵적인 좌표계(이미 정의된 위치)에서 비용을 계산함.
lineCost(x0: float, x1: float, y0: float, y1: float, step_size: float = 0.5)	- 두 점 (x0, y0)–(x1, y1) 사이의 직선을 일정 간격(step_size)으로 샘플링하여 각 지점에서 충돌(cost)을 계산. - 중간에 LETHAL(254) 발견 시 즉시 충돌로 간주.
worldToMapValidated(wx: float, wy: float) -> tuple[int, int] or None	- 월드 좌표 (wx, wy)를 map 좌표 (mx, my)로 변환. 좌표가 - 유효하지 않거나 costmap이 아직 설정되지 않았다면 (None, None)을 반환.
pointCost(x: int, y: int) -> int	- 맵 좌표 (x, y) 위치의 costmap 셀 값을 반환 (예: 0=Free, 254=Lethal Obstacle 등).
setCostmap(costmap: PyCostmap2D)	- 충돌 검사를 수행할 기준 costmap(PyCostmap2D)을 지정. - 설정하지 않으면 다른 메서드 호출 시 오류 발생.
footprintCostAtPose(x: float, y: float, theta: float, footprint: Polygon)	- 특정 포즈 (x, y, θ)에 로봇 footprint를 회전 및 이동시켜 배치한 뒤, 그 포즈에서의 footprint 충돌 비용을 계산.

https://docs.nav2.org/commander_api/index.html#footprint-collision-checker-api

Nav2 – Footprint Collision Checker API

```python
from nav2_simple_commander.robot_navigator import BasicNavigator
from nav2_simple_commander.costmap_2d import PyCostmap2D
from nav2_simple_commander.footprint_collision_checker import FootprintCollisionChecker
from geometry_msgs.msg import Polygon, Point32

navigator = BasicNavigator()
navigator.waitUntilNav2Active()

global_cm_msg = navigator.getGlobalCostmap()
global_cm = PyCostmap2D(global_cm_msg)

checker = FootprintCollisionChecker()
checker.setCostmap(global_cm)

footprint = Polygon()
footprint.points = [
    Point32(x=0.20, y=0.15, z=0.0),
    Point32(x=0.20, y=-0.15, z=0.0),
    Point32(x=-0.20, y=-0.15, z=0.0),
    Point32(x=-0.20, y=0.15, z=0.0),
    Point32(x=0.20, y=0.15, z=0.0),
]

x, y, theta = 1.0, 2.0, 0.0
cost = checker.footprintCostAtPose(x, y, theta, footprint)

if cost >= 254:
    print("Collision! (lethal)")
else:
    print("Safe!, Max cost =", cost)

```

FootprintCollisionChecker에 costmap 설정

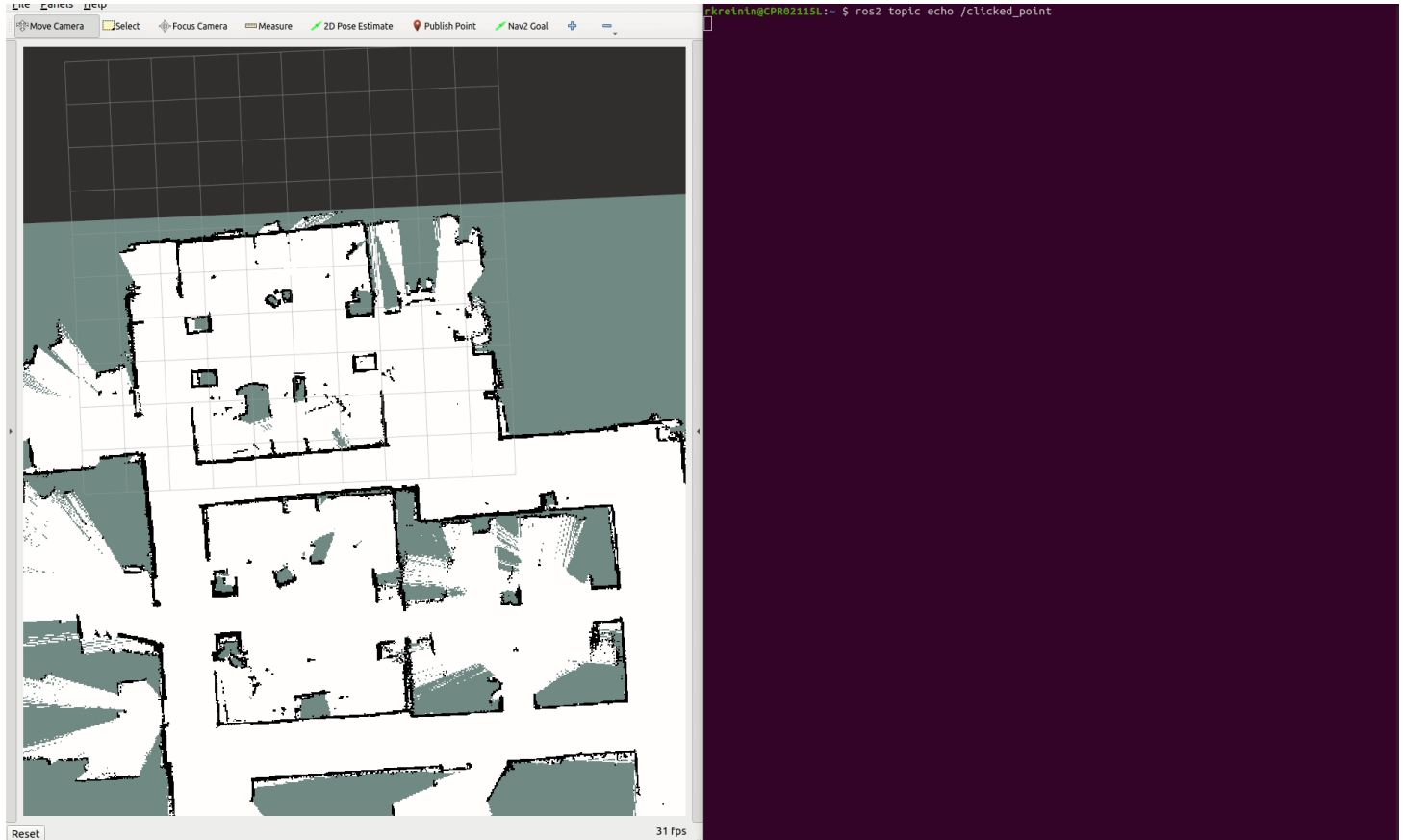
Robot footprint 정의 (예: 직사각형)

특정 위치에서 충돌이 발생하는지 확인

Navigation2 – Check World Coordinate

- The *Publish Point tool* allows you to click on a point on the map, and have the coordinates of that point published to the */clicked_point* topic:

```
$ ros2 topic echo /clicked_point
```



Navigation2 – nav_to_pose.py

```
import rclpy
from turtlebot4_navigation.turtlebot4_navigator import TurtleBot4Directions, TurtleBot4Navigator

def main():
    rclpy.init()

    navigator = TurtleBot4Navigator()

    # Set initial pose
    # initial_pose = navigator.getPoseStamped([0.0, 0.0], TurtleBot4Directions.NORTH)
    # navigator.setInitialPose(initial_pose)

    # Wait for Nav2
    navigator.waitUntilNav2Active(localizer='controller_server')

    # Set goal poses
    goal_pose = navigator.getPoseStamped([-13.0, 9.0], TurtleBot4Directions.EAST)

    # Go to each goal pose
    navigator.startToPose(goal_pose)

    rclpy.shutdown()

if __name__ == '__main__':
    main()
```

Navigation2 – nav_through_poses.py

```
import rclpy
from turtlebot4_navigation.turtlebot4_navigator import TurtleBot4Directions, TurtleBot4Navigator

def main():
    rclpy.init()

    navigator = TurtleBot4Navigator()

    # Wait for Nav2
    navigator.waitUntilNav2Active(localizer='controller_server')

    # Set goal poses
    goal_pose = []
    goal_pose.append(navigator.getPoseStamped([-3.0, 0.0], TurtleBot4Directions.EAST))
    goal_pose.append(navigator.getPoseStamped([-3.0, -3.0], TurtleBot4Directions.NORTH))
    goal_pose.append(navigator.getPoseStamped([3.0, -3.0], TurtleBot4Directions.NORTH_WEST))
    goal_pose.append(navigator.getPoseStamped([9.0, -1.0], TurtleBot4Directions.WEST))
    goal_pose.append(navigator.getPoseStamped([9.0, 1.0], TurtleBot4Directions.SOUTH))
    goal_pose.append(navigator.getPoseStamped([-1.0, 1.0], TurtleBot4Directions.EAST))

    # Navigate through poses
    navigator.startThroughPoses(goal_pose)

    rclpy.shutdown()

if __name__ == '__main__':
    main()
```