실물 RMF 교육자료

Auto Mobile Robot

Exported on 05/21/2024

Table of Contents

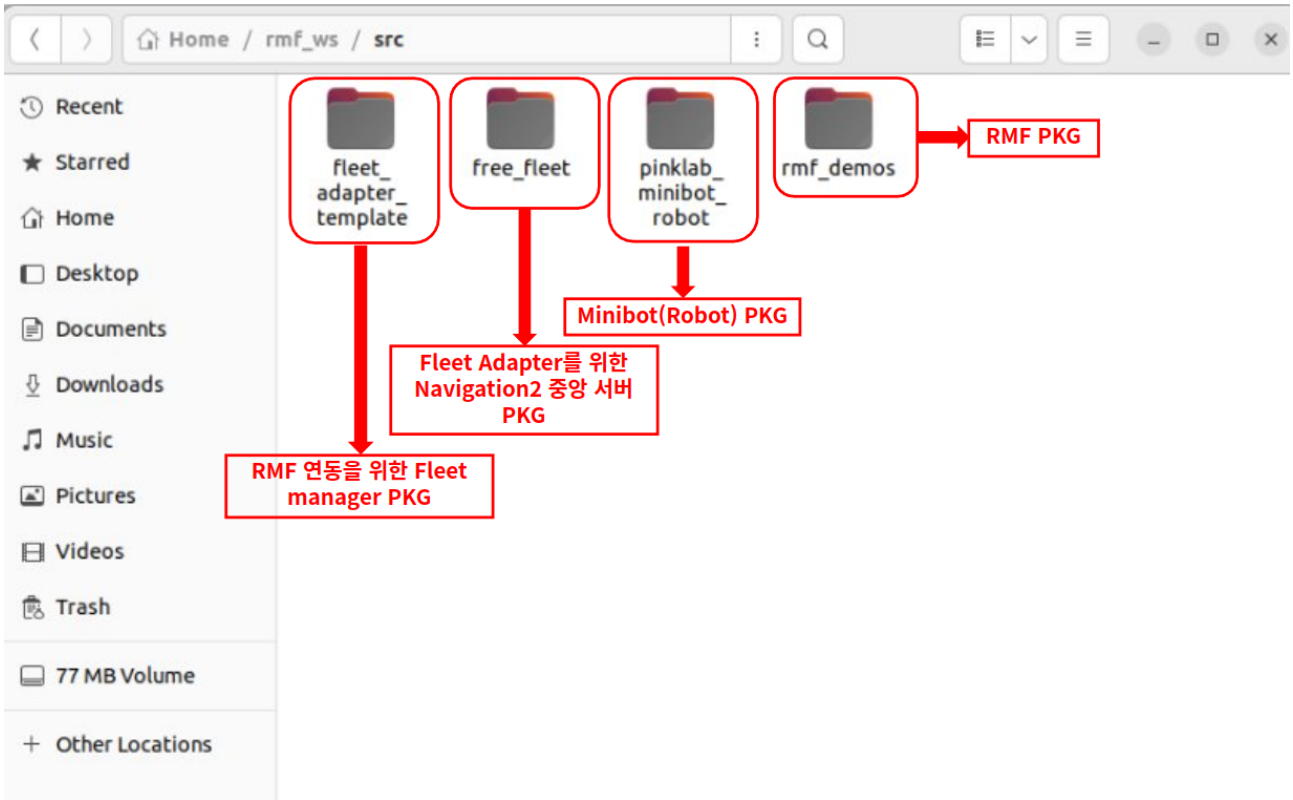
1 1. SLAM을 통한 RMF 시스템 구축 5
1.1 현재 RMF시스템의 ROS2패키지 구성은 이렇게 되어있다. 5
1.2 만약 패키지 구성이 없다면 5
1.3 rmf_demos_maps에 새로운 폴더를 추가하자 6
1.4 Navigation2이 문제 없는 Mobile로봇을 준비하자. 7
1.5 RMF시스템의 기반이 될 바닥도면 혹은 Map데이터를 준비해야한다. Minibot을 이용해서 Map을 그리자. 7
1.6 돌아다니면서 맵을 그리자. 8
1.7 Map을 저장할때 다음과 같은 명령어로 png로 저장한다. 링크1 링크2 8
1.8 전부 종료하고, Traffic editor를 열어서 New File만들기. 이름은 구분짓기 위해서 pinklab_real로 만들자. 9
1.9 이전에 Minibot(Mobile robot)으로 그린 png파일로 levels을 구성. 링크3 10
1.10 그 다음 현실의 장애물의 길이를 측정값으로 넣어준다. 링크4 10
1.11 저장 및 종료 후 다시 Traffic-editor를 실행하면 Scale이 맞춰진다. 11
1.12 이제 로봇의 이동 지점 및 경유지를 Graph 0으로 이어주고, 링크5 11
1.13 로봇의 시작 지점에 Properties를 설정하자. 링크6 12
1.14 마지막으로 나머지 vertex에 이름을 지어주고 저장 및 종료. 링크7 12
1.15 이제 RMF 시스템을 시작하는 rmf_demos 패키지에서 launch파일을 새로 만들자. 13
1.16 다음 코드 붙혀넣기. 링크8 13
1.17 rmf_demos에서 launch/include안에 pinklab_real폴더를 만들고 rviz2설정파일을 하나 만들자. 시뮬레이션에서 사용하던 pinklab.rviz를 복사 붙혀넣기. 링크9 14
1.18 rmf_demos_dashboard_resources 패키지에 새로운 폴더를 만들고 dashboard_config.json파일 만들기. 15
1.19 이전에 Traffic-editor에 있던 vertex와 허용하는 Task를 적용한다. 링크10 15
1.20 그 다음 Cmakelist.txt에서 pinklab_real이 빌드 될 수 있도록 고치자. 링크11 16
1.21 그 다음 빌드 후 실행하자 16
1.22 다른 터미널을 열어서 RMF가 작동중인지 확인해보자. 17

2 2. Fleet adapter 구성 18
2.1 Fleet adapter를 위한 Config파일을 생성 18
2.2 RMF시스템 내 Minibot Fleet의 설정 작성. 링크1 19
2.3 minibot_config.yaml에 추가로 로봇의 개수, 업데이트 주기, 시작지점, 충전지점을 설정 링크2 19
2.4 Traffic-editor를 실행해서, 이미지의 4점의 좌표를 minibot_config.yaml에 작성하기. 20
2.5 그 다음 모바일 로봇의 Navigation2을 실행하자. 20
2.6 터미널을 한개 더 열어서 /clicked_point를 구독한 뒤에, Rviz2에서 네 점을 찍어 나온 좌표를 minibot_config에 기입하자. 링크3 21
2.7 기존 RMF실행파일에 Fleet adapter를 실행하는 코드를 추가. 링크4 21
2.8 그리고 Navigation2 정보를 가져오는 Free Fleet Server코드를 추가. 링크5 22
2.9 이제 Navigation2과 Free Fleet client를 같이 실행하도록 코드를 작성. 링크6 22
2.10 빌드 후 Navigation2부터 실행. 23
2.11 그 다음에 RMF를 실행하면 23
2.12 RMF-Panel에서 이동 시키기. 23
2.13 로봇이 이동하는 것을 볼 수 있다. 24
2.14 만약에 로봇 Navigation2의 inital pose가 안맞는다면 navigation2패키지의 nav2_params패키지를 수정하자. 링크7 25
3 3. 2대 이상의 RMF 시스템 구축 26
3.1 RMF시스템에 같은 Fleet인 minibot을 한개 더 추가하는 방법 26
3.1.1 Minibot을 하나 더 다뤄보자. 26
3.1.2 Traffic editor를 열어서, 다른 minibot이 시작할 지점을 만들자.링크1 27
3.1.3 fleet_config에서 방금 만든 vertex로 minibot_2의 시작점을 만들어준다. 링크2 27
3.1.4 Navigation2시작시 Free Fleet client설정을 바꿔서 이름을 입력인자로 받을 수 있게 변경하자. 링크3 28
3.1.5 빌드 후 실행해보자 29
3.1.6 minibot_navigation2/param에서 nav2_params.yaml을 복사한 뒤, Rviz2에서 minibot_2의 spawn 지점을 initial_pose로 지정하자. 링크4 30
3.1.7 minibot_2는 다른 ROS_DOMAIN에서 실행하자. 30

3.1.8 그 다음 RMF를 실행하면.....31
3.1.9 Panel로 Task를 주면 입찰받은 minibot이 움직이게 된다. 31
3.1.10 같은 Fleet만 사용할 경우 RMF시스템의 소프트웨어 아키텍쳐 (예시).....33
3.2 다른 Fleet를 추가하기. 35
3.2.1 Minibot과 다른 로봇인 Turtlebot3를 RMF시스템에 추가해보자. 35
3.2.2 Turtlebot3 ROS2 PKG 다운로드 35
3.2.3 Traffic editor를 열고… 36
3.2.4 Graph 1번으로 Minibot Fleet와는 다른 경로를 지정하자. 36
3.2.5 Spawn 지점과 관련된 설정을 하고, 저장 및 종료. 링크5.....37
3.2.6 그 다음 Fleet adapter를 구축해야한다.....37
3.2.7 Turtlebot3의 RMF정보를 작성할 yaml파일을 만들고… 38
3.2.8 Turtlebot3의 위치정보, 개체, 좌표변환을 작성. 링크6 39
3.2.9 RMF시스템 시작 시 turtlebot3 Fleet adapter를 같이 시작하도록 설정. 링크7...39
3.2.10 RMF시스템에 turtlebot3의 navigation2정보를 취합할 Free Fleet Server코드 작성. 링크8.....40
3.2.11 Navigation2을 시작하면서, Free fleet client를 같이 시작하기 위해 파일을 한 개 만들고… 40
3.2.12 Navigation2과 Free Fleet client 코드를 작성. 링크9.....41
3.2.13 빌드한 뒤, 먼저 Minibot을 다 실행하고… 42
3.2.14 Turtlebot3 실행.....42
3.2.15 그 다음 RMF를 실행하면 42
3.2.16 타 기종을 포함하는 RMF 시스템의 소프트웨어 아키텍쳐 예시는 다음과 같다....44

1 1. SLAM을 통한 RMF 시스템 구축

1.1 현재 RMF시스템의 ROS2패키지 구성은 이렇게 되어있다.

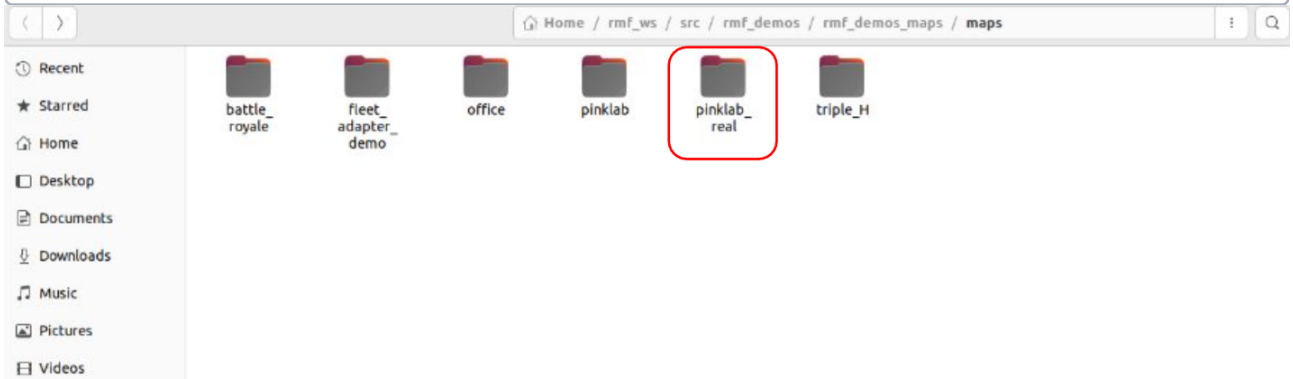


1.2 만약 패키지 구성이 없다면 ...

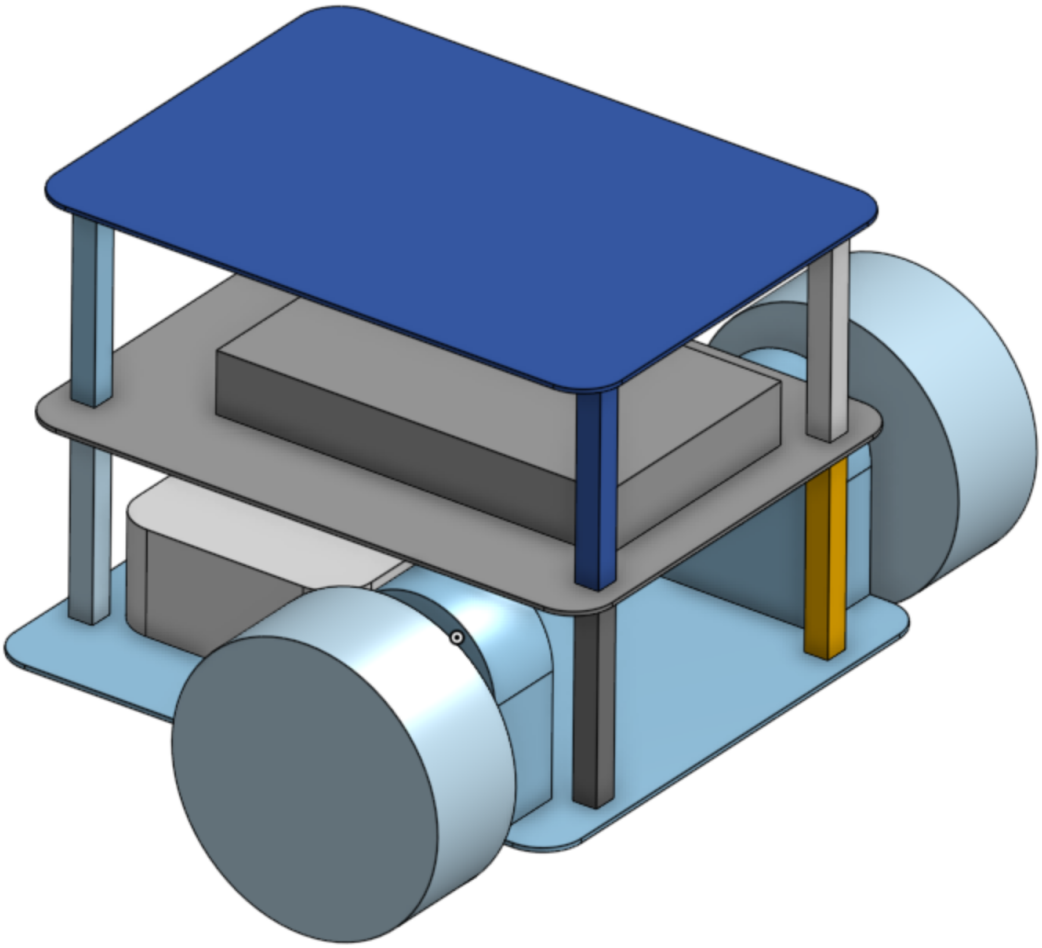
```
cd ~/rmf_ws/src
git clone https://github.com/pinklabstudio/rmf_demos.git -b real_startpack
git clone https://github.com/pinklabstudio/fleet_adapter_template.git -b
real_startpack
git clone https://github.com/open-rmf/free_fleet -b main
git clone https://github.com/pinklabstudio/pinklab_minibot_robot.git -b main
```

1.3 rmf_demos_maps에 새로운 폴더를 추가하자.

```
mkdir ~/rmf_ws/src/rmf_demos/rmf_demos_maps/maps/pinklab_real
```



1.4 Navigation2이 문제 없는 Mobile로봇을 준비하자.



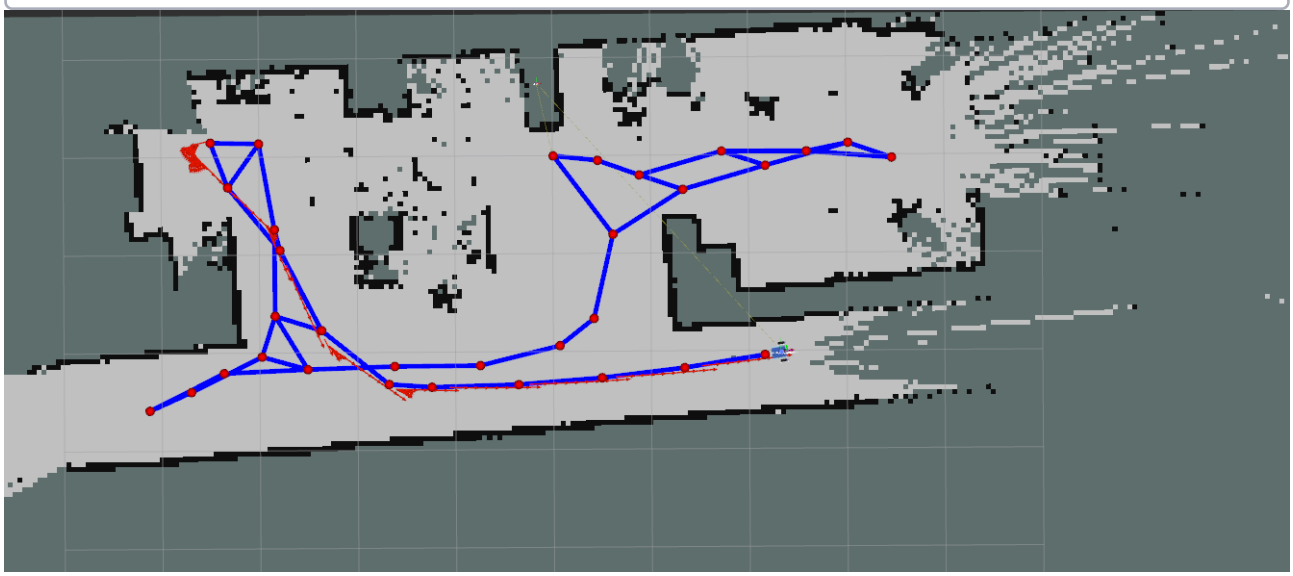
1.5 RMF시스템의 기반이 될 바닥도면 혹은 Map데이터를 준비해야한다. Minibot을 이용해서 Map을 그리자.

```
ros2 launch minibot_bringup bringup_robot.launch.xml # SSH로 로봇 안에서 실행
ros2 launch minibot_navigation2 map_building.launch.xml # PC에서 실행
rviz2 -d `ros2 pkg prefix minibot_navigation2`/share/minibot_navigation2/rviz/
map_building.rviz # PC
```

- 맵을 그리기 전에 측정 가능한 장애물을 한개 이상 배치하자!(30cm이상)

1.6 돌아다니면서 맵을 그리자.

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard #PC에서 실행
or
ros2 run teleop_twist_keyboard teleop_twist_keyboard \
  --ros-args -r cmd_vel:=base_controller/cmd_vel_unstamped  #PC에서 실행
```

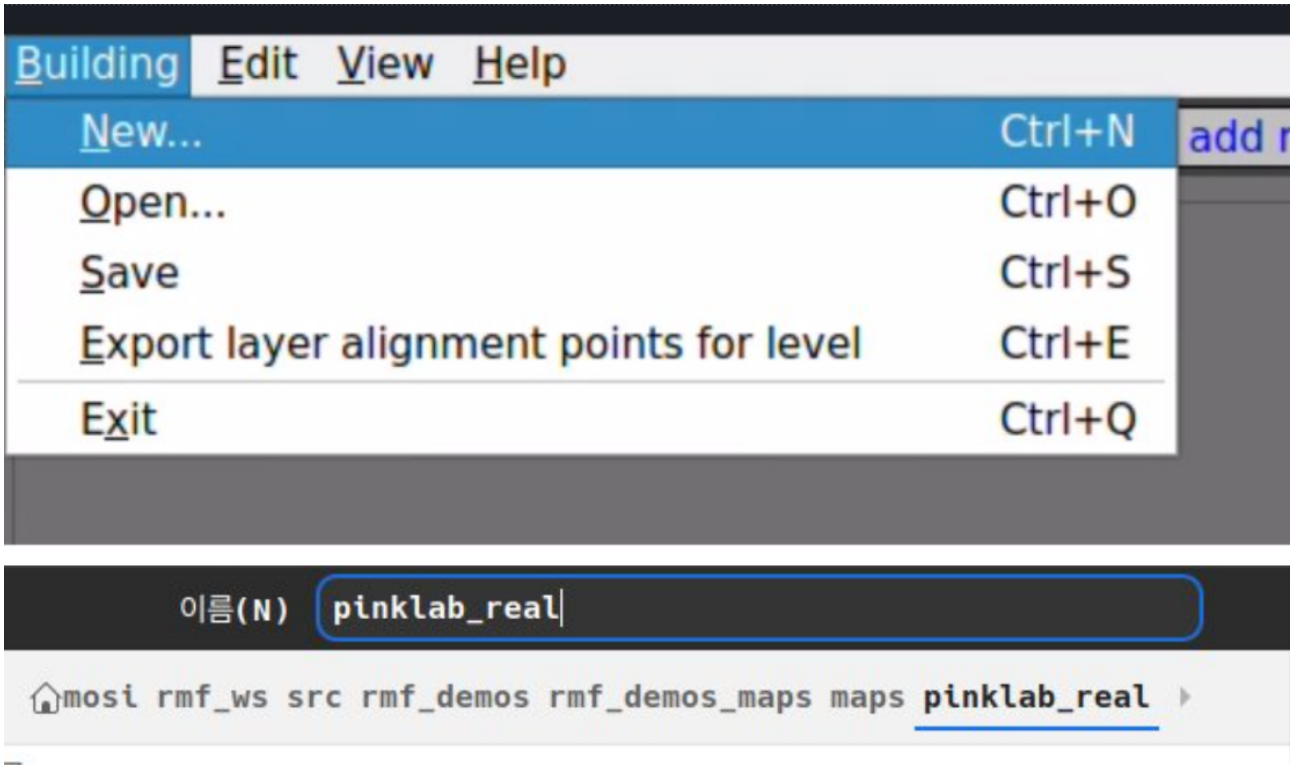


1.7 Map을 저장할때 다음과 같은 명령어로 png로 저장한다. 링크1[1] 링크2[2]

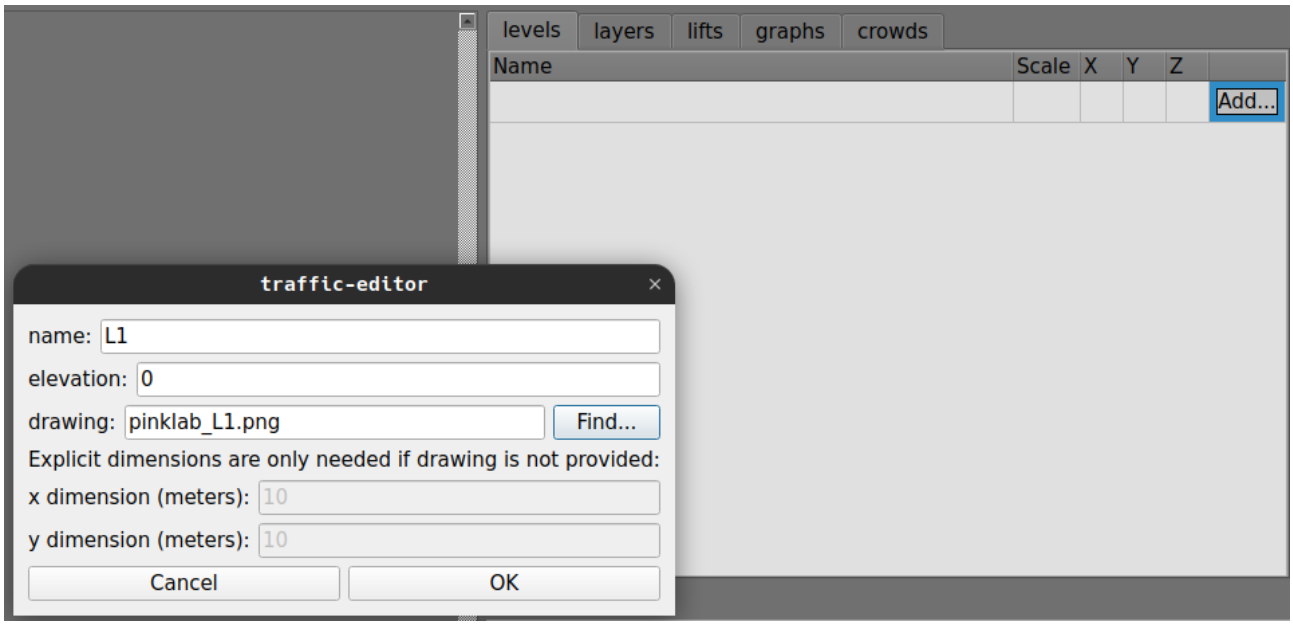
```
ros2 run nav2_map_server map_saver_cli --fmt png \
  -f ~/rmf_ws/src/rmf_demos/rmf_demos_maps/maps/pinklab_real/pinklab_L1
```

1 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/14/14-1pinklab_L1.png
2 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/14/14-2pinklab_L1.yaml

1.8 전부 종료하고, Traffic editor를 열어서 New File만들기. 이름은 구분짓기 위해서 pinklab_real로 만들자.

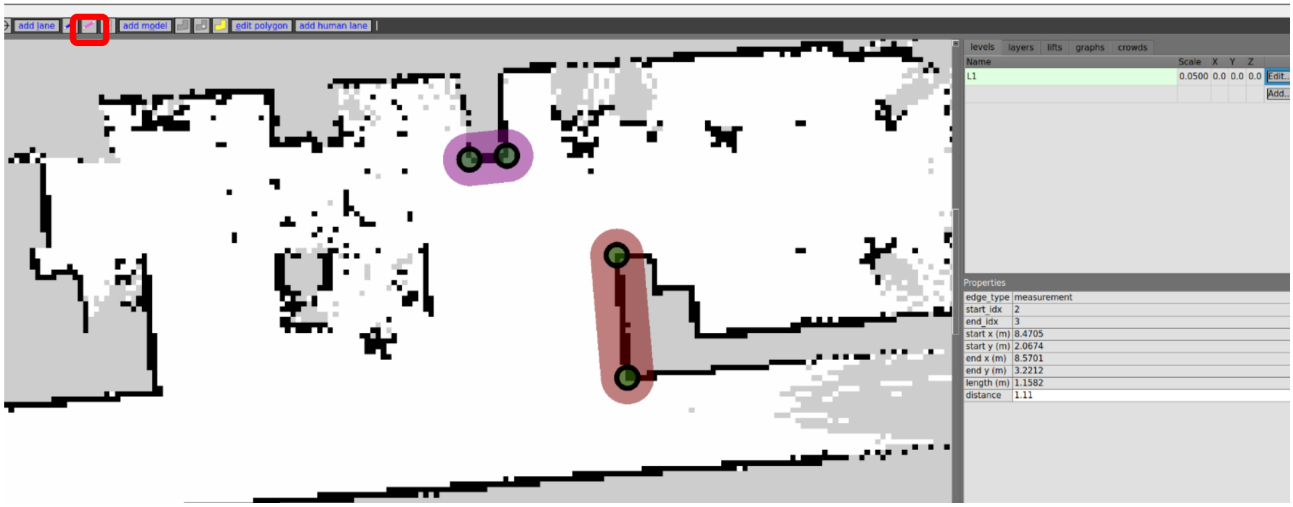


1.9 이전에 Minibot(Mobile robot)으로 그린 png파일로 levels을 구성. 링크3[3]



1.10 그 다음 현실의 장애물의 길이를 측정값으로 넣어준다. 링크4[4]

- 한 개만 해도 상관없으며 여러개 할 수록 Scale이 정확해진다.



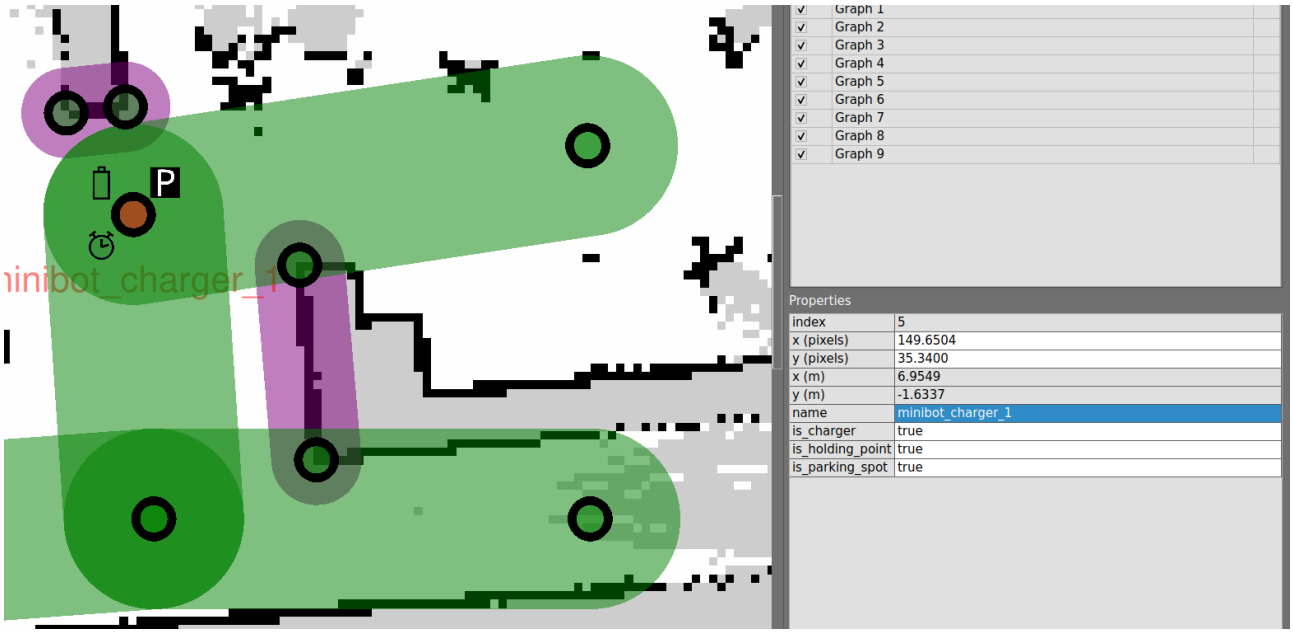
3 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/14/14-3pinklab_real.building.yaml
4 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/14/14-4pinklab_real.building.yaml

1.11 저장 및 종료 후 다시 Traffic-editor를 실행하면 Scale이 맞춰진다.

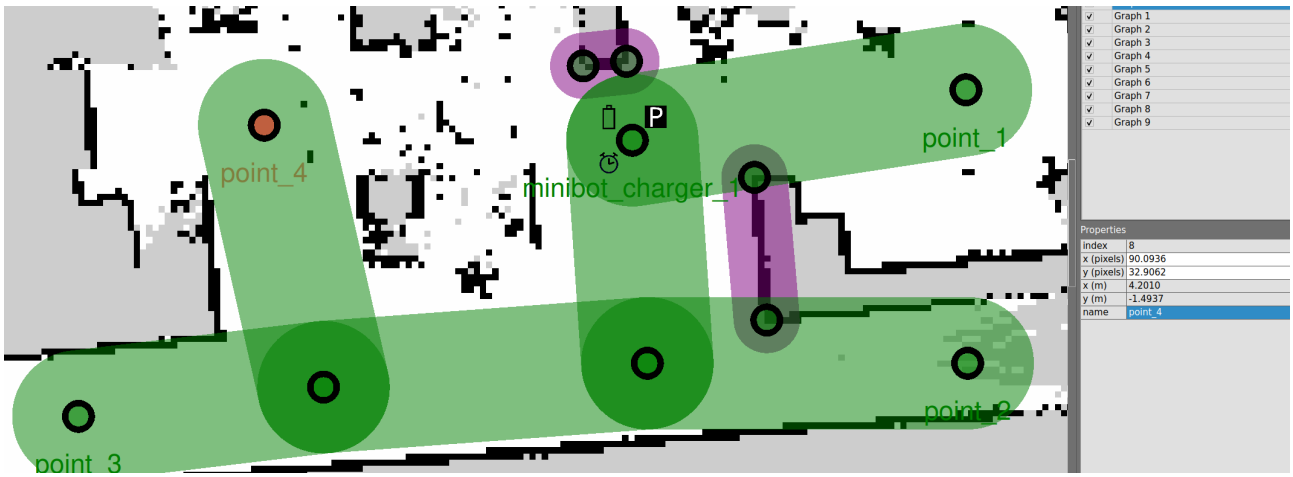
1.12 이제 로봇의 이동 지점 및 경유지를 Graph 0으로 이어주고, 링크5[5]

5 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/14/14-5pinklab_real.building.yaml

1.13 로봇의 시작 지점에 Properties를 설정하자. 링크6[6]

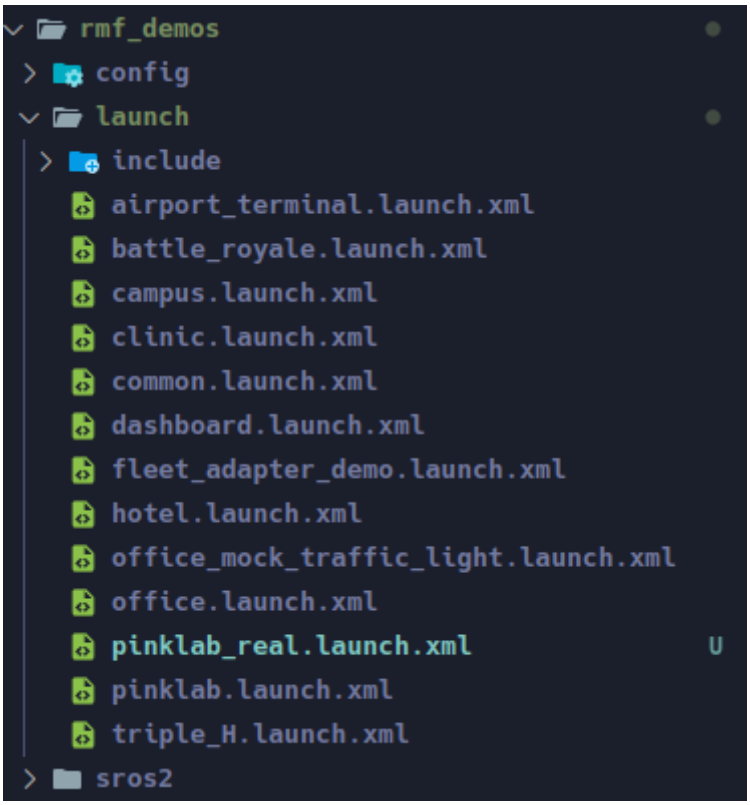


1.14 마지막으로 나머지 vertex에 이름을 지어주고 저장 및 종료. 링크7[7]



6 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/14/14-6pinklab_real.building.yaml
7 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/14/14-7pinklab_real.building.yaml

1.15 이제 RMF 시스템을 시작하는 rmf_demos 패키지에서 launch파일을 새로 만들자.

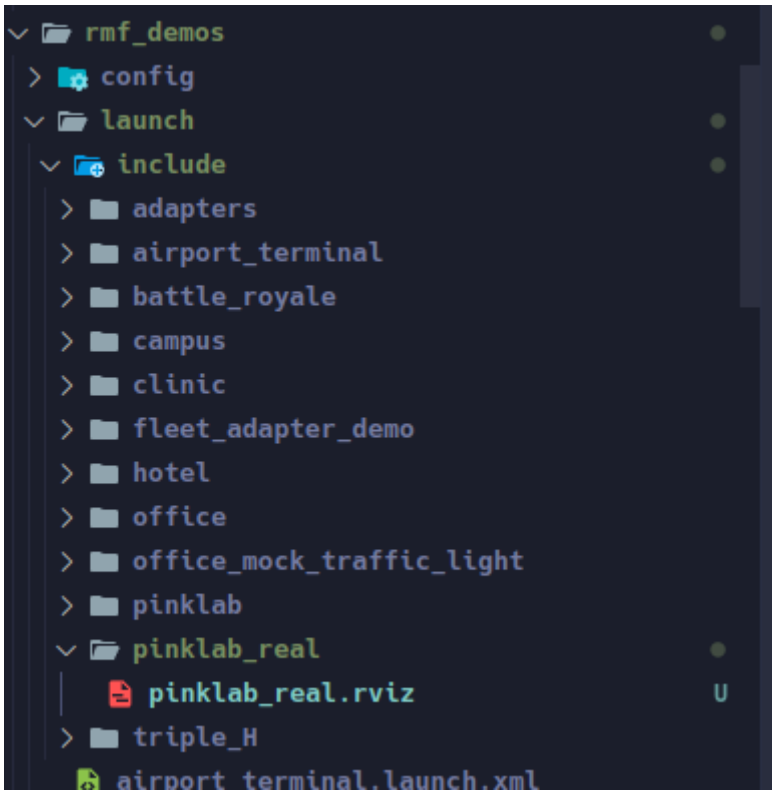


1.16 다음 코드 붙혀넣기. 링크8[8]

```
<?xml version='1.0' ?>
<launch>
  <arg name="use_sim_time" default="false"/>
  <!-- Common launch -->
  <include file="$(find-pkg-share rmf_demos)/common.launch.xml">
    <arg name="use_sim_time" value="$(var use_sim_time)"/>
    <arg name="viz_config_file" value ="$(find-pkg-share rmf_demos)/include/
pinklab_real/pinklab_real.rviz"/>
    <arg name="config_file" value="$(find-pkg-share rmf_demos_maps)/pinklab_real/
pinklab_real.building.yaml"/>
    <arg name="dashboard_config_file" value="$(find-pkg-share
rmf_demos_dashboard_resources)/pinklab_real/dashboard_config.json"/>
  </include>
</launch>
```

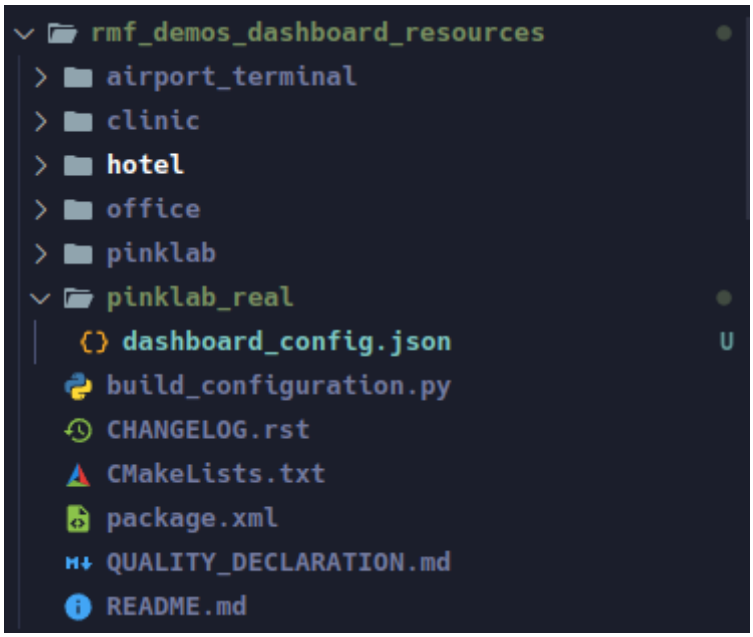
8 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/14/14-8pinklab_real.launch.xml

1.17 rmf_demos에서 launch/include안에 pinklab_real폴더를 만들고 rviz2설정파일을 하나 만들자. 시뮬레이션에서 사용하던 pinklab.rviz를 복사 붙혀넣기. 링크9[9]



9 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/14/14-9pinklab_real.rviz

1.18 rmf_demos_dashboard_resources 패키지에 새로운 폴더를 만들고 dashboard_config.json파일 만들기.



1.19 이전에 Traffic-editor에 있던 vertex와 허용하는 Task를 적용한다. 링크10[10]

```
{
    "world_name": "PinkLAB(REAL WORLD)",
    "valid_task": [
        "Loop"
    ],
    "task": {
        "Delivery": {
        },
        "Loop": {
            "places": [
                "minibot_charger_1",
                "point_1",
                "point_2",
                "point_3",
                "point_4"
            ]
        },
```

10 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/14/14-10dashboard_config.json

```
        "Clean": {
        },
        "Station": {},
        "Patrol": {},
        "Charging": {}
    }
}
```

1.20 그 다음 Cmakelist.txt에서 pinklab_real이 빌드 될 수 있도록 고치자. 링크11[11]

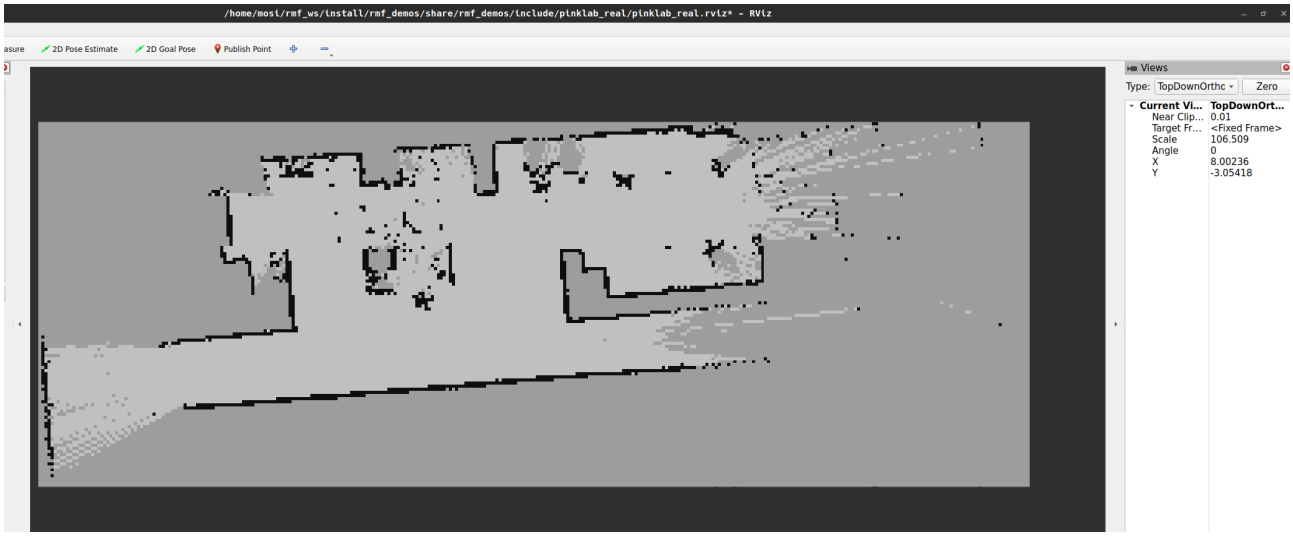
```
install(DIRECTORY
  airport_terminal
  office
  hotel
  clinic
  pinklab
  pinklab_real
  DESTINATION share/${PROJECT_NAME}
)
```

1.21 그 다음 빌드 후 실행하자.

```
ros2 launch rmf_demos pinklab_real.launch.xml #패키지 유의
```

- rmf_demos_gz_classic이 아닌 rmf_demos의 pinklab_real.launch.xml을 실행해야한다.

11 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/14/14-11CMakeLists.txt



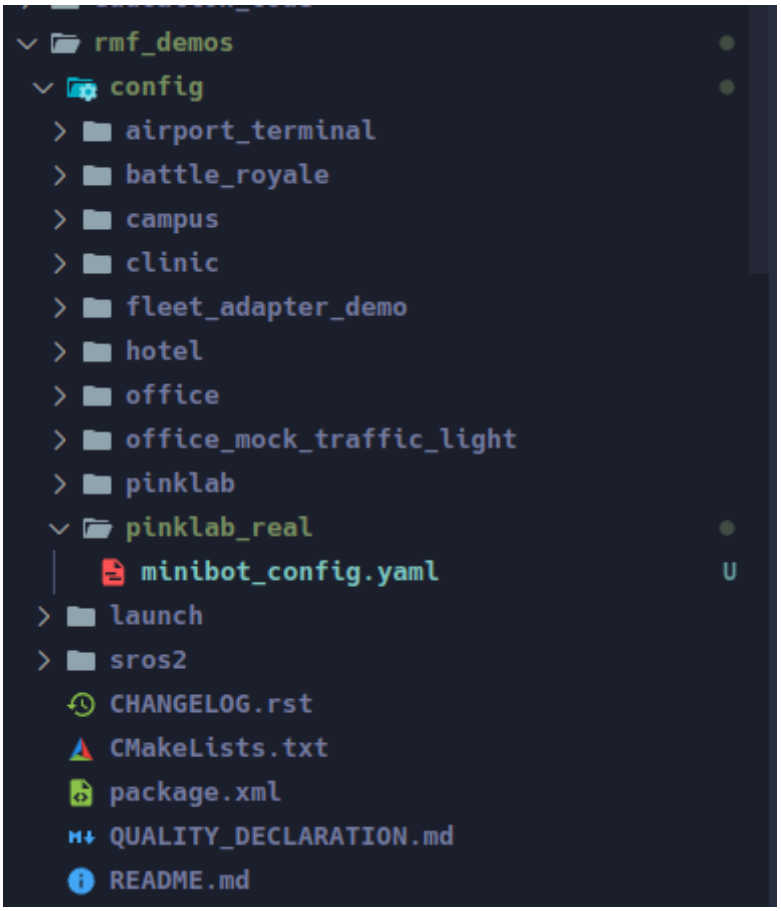
1.22 다른 터미널을 열어서 RMF가 작동중인지 확인해보자.

```
❯ ros2 topic list | grep rmf
/rmf_obstacles
/rmf_task/bid_notice
/rmf_task/bid_response
/rmf_task/dispatch_ack
/rmf_task/dispatch_request
/rmf_traffic/blockade_cancel
/rmf_traffic/blockade_heartbeat
/rmf_traffic/blockade_reached
/rmf_traffic/blockade_ready
/rmf_traffic/blockade_release
/rmf_traffic/blockade_set
/rmf_traffic/heartbeat
/rmf_traffic/itinerary_clear
/rmf_traffic/itinerary_delay
/rmf_traffic/itinerary_extend
/rmf_traffic/itinerary_reached
/rmf_traffic/itinerary_set
/rmf_traffic/negotiation_ack
/rmf_traffic/negotiation_conclusion
/rmf_traffic/negotiation_forfeit
/rmf_traffic/negotiation_notice
/rmf_traffic/negotiation_proposal
/rmf_traffic/negotiation_refusal
```

2 2. Fleet adapter 구성

2.1 Fleet adapter를 위한 Config파일을 생성

- rmf_demos/config에 pinklab_real폴더를 만들고 그 아래에 config파일 만들기.



2.2 RMF시스템 내 Minibot Fleet의 설정 작성. 링크1[12]

```yaml
rmf_fleet:
  name: "minibot"
  fleet_manager:
    prefix: "http://127.0.0.1:22016"
    ip: "127.0.0.1"
    port: 22016
    user: "some_user"
    password: "some_password"
  limits:
    linear: [0.3, 0.15] # 최대 속도 약 0.3 m/s, 가속도는 0.15 m/s^2
    angular: [1.5, 0.7] # 최대 각속도 약 1.5 rad/s, 가속도는  0.7 rad/s^2
  profile: # Robot profile is modelled as a circle
    footprint: 0.25 # minibot의 반지름 0.25 m
    vicinity: 0.5 # minibot의 안전 반경 0.5 m
  reversible: False # whether robots in this fleet can reverse
  battery_system:
    voltage: 12.0 # V
    capacity: 24.0 # Ahr
    charging_current: 5.0 # A
  mechanical_system:
    mass: 20.0 # kg
    moment_of_inertia: 10.0 #kgm^2
    friction_coefficient: 0.22
  ambient_system:
    power: 20.0 # W
  tool_system:
    power: 0.0 # W
  recharge_threshold: 0.10 # Battery level below which robots in this fleet will not operate
  recharge_soc: 1.0 # Battery level to which robots in this fleet should be charged up to during recharging tasks
  publish_fleet_state: 10.0 # Publish frequency for fleet state, ensure that it is same as robot_state_update_frequency
  account_for_battery_drain: False
  task_capabilities: # Specify the types of RMF Tasks that robots in this fleet are capable of performing
    loop: True
    delivery: False
    clean: False
    finishing_request: "park" # [park, charge, nothing]
  actions: ["teleop"]

  responsive_wait: True # Should responsive wait be on/off for the whole fleet by default? False if not specified.
```

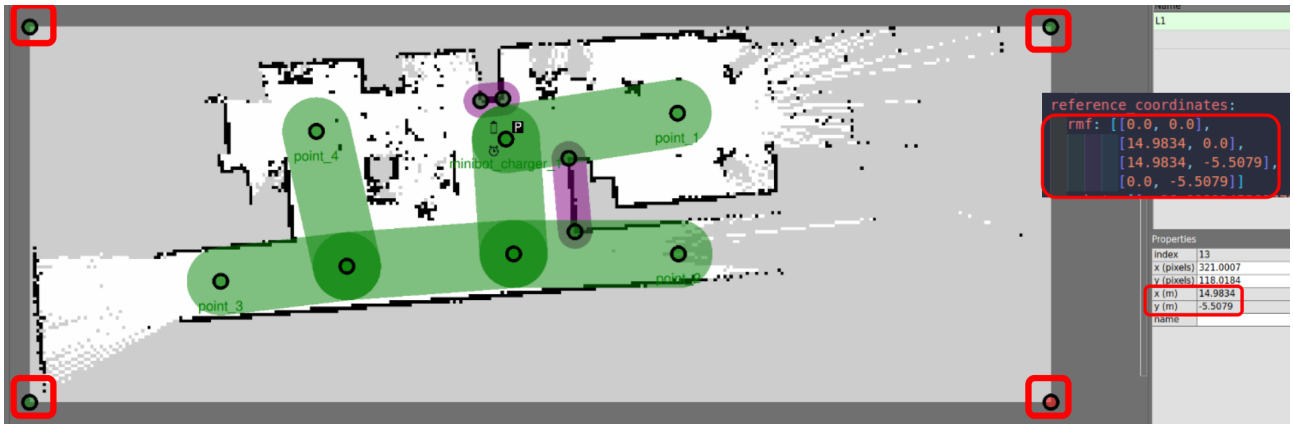
2.3 minibot_config.yaml에 추가로 로봇의 개수, 업데이트 주기, 시작지점, 충전지점을 설정 링크2[13]

```yaml
# minibot CONFIG ==============================================================
robots:
  minibot_1:
    robot_config:
      max_delay: 10.0 # allowed seconds of delay of the current itinerary before it gets interrupted and replanned
    rmf_config:
      robot_state_update_frequency: 1.0
      charger:
        waypoint: "minibot_charger_1"
      start:
        map_name: "L1"
        waypoint: "minibot_charger_1"
        orientation: 0.0 # radians
```

12 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/15/15-1minibot_config.yaml
13 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/15/15-2minibot_config.yaml

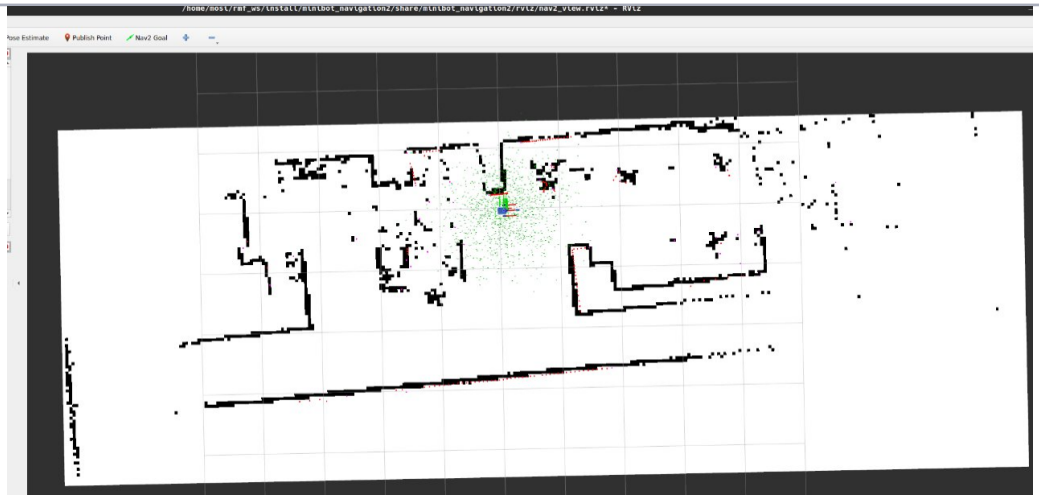
2.4 Traffic-editor를 실행해서, 이미지의 4점의 좌표를 minibot_config.yaml에 작성하기.

- 시계방향으로 기입하자.

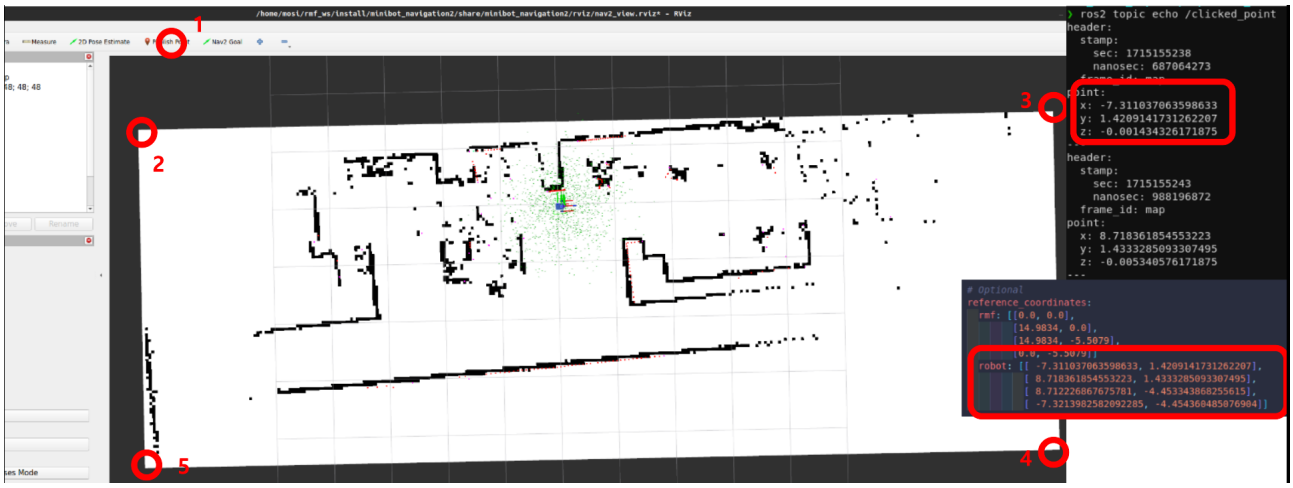


2.5 그 다음 모바일 로봇의 Navigation2을 실행하자.

```
ros2 launch minibot_bringup bringup_robot.launch.xml # Minibot에서
ros2 launch minibot_navigation2 bringup_launch.xml \
    map:=$HOME/rmf_ws/src/rmf_demos/rmf_demos_maps/maps/pinklab_real/pinklab_L1.yaml
#PC에서
rviz2 -d `ros2 pkg prefix minibot_navigation2`/share/minibot_navigation2/rviz/
nav2_view.rviz
```



2.6 터미널을 한개 더 열어서 /clicked_point를 구독한 뒤에, Rviz2에서 네 점을 찍어나온 좌표를 minibot_config에 기입하자. 링크3[14]



2.7 기존 RMF실행파일에 Fleet adapter를 실행하는 코드를 추가. 링크4[15]

- RMF를 실행하는 파일은 rmf_demos/rmf_demos/launch/pinklab_real.launch.xml이다.

```
    <!-- Minibot fleet adapter -->
    <include file="$(find-pkg-share fleet_adapter_template)/launch/
fleet_minibot_adapter.launch.xml">
        <arg name="use_sim_time" value="$(var use_sim_time)" />
        <arg name="nav_graph_file"
            value="$(find-pkg-share rmf_demos_maps)/maps/pinklab_real/nav_graphs/
0.yaml" />
        <arg name="config_file"
            value="$(find-pkg-share rmf_demos)/config/pinklab_real/
minibot_config.yaml" />
    </include>
```

14 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/15/15-3minibot_config.yaml
15 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/15/15-4pinklab_real.launch.xml

2.8 그리고 Navigation2 정보를 가져오는 Free Fleet Server코드를 추가. 링크5[16]

```
    <node pkg="free_fleet_server_ros2"
        exec="free_fleet_server_ros2"
        name="minibot_fleet_server_node"
        output="both">
        <param name="fleet_name" value="minibot" />
        <param name="fleet_state_topic" value="minibot/fleet_states" />
        <param name="mode_request_topic" value="minibot/robot_mode_requests" />
        <param name="path_request_topic" value="minibot/robot_path_requests" />
        <param name="destination_request_topic" value="minibot/
robot_destination_requests" />
        <param name="dds_domain" value="43" />
        <param name="dds_robot_state_topic" value="robot_state" />
        <param name="dds_mode_request_topic" value="mode_request" />
        <param name="dds_path_request_topic" value="path_request" />
        <param name="dds_destination_request_topic" value="destination_request" />
        <param name="update_state_frequency" value="20.0" />
        <param name="publish_state_frequency" value="2.0" />
        <param name="translation_x" value="0.0" />
        <param name="translation_y" value="0.0" />
        <param name="rotation" value="0.0" />
        <param name="scale" value="1.0" />
    </node>
```

2.9 이제 Navigation2과 Free Fleet client를 같이 실행하도록 코드를 작성. 링크6[17]

- Navigation2을 시작하는 코드인 pinklab_minibot_robot/minibot_navigation2/launch/bringup_launch.xml 에 Free Fleet client코드를 추가하자.

```
    <!-- launch the free fleet client first -->
    <node name="minibot_free_fleet_client_node" pkg="free_fleet_client_ros2"
        exec="free_fleet_client_ros2" output="both">
        <param name="fleet_name" value="minibot" />
        <param name="robot_name" value="minibot_1" />
        <param name="robot_model" value="minibot" />
        <param name="level_name" value="L1" />
        <param name="dds_domain" value="43" />
        <param name="max_dist_to_first_waypoint" value="10.0" />
```

16 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/15/15-5pinklab_real.launch.xml
17 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/15/15-6bringup_launch.xml

```
        <param name="map_frame" value="map" />
        <param name="robot_frame" value="base_footprint" />
        <param name="nav2_server_name" value="/navigate_to_pose" />
        <param name="use_sim_time" value="False" />
        <param name="dds_domain" value="43" />
    </node>
```

2.10 빌드 후 Navigation2부터 실행.

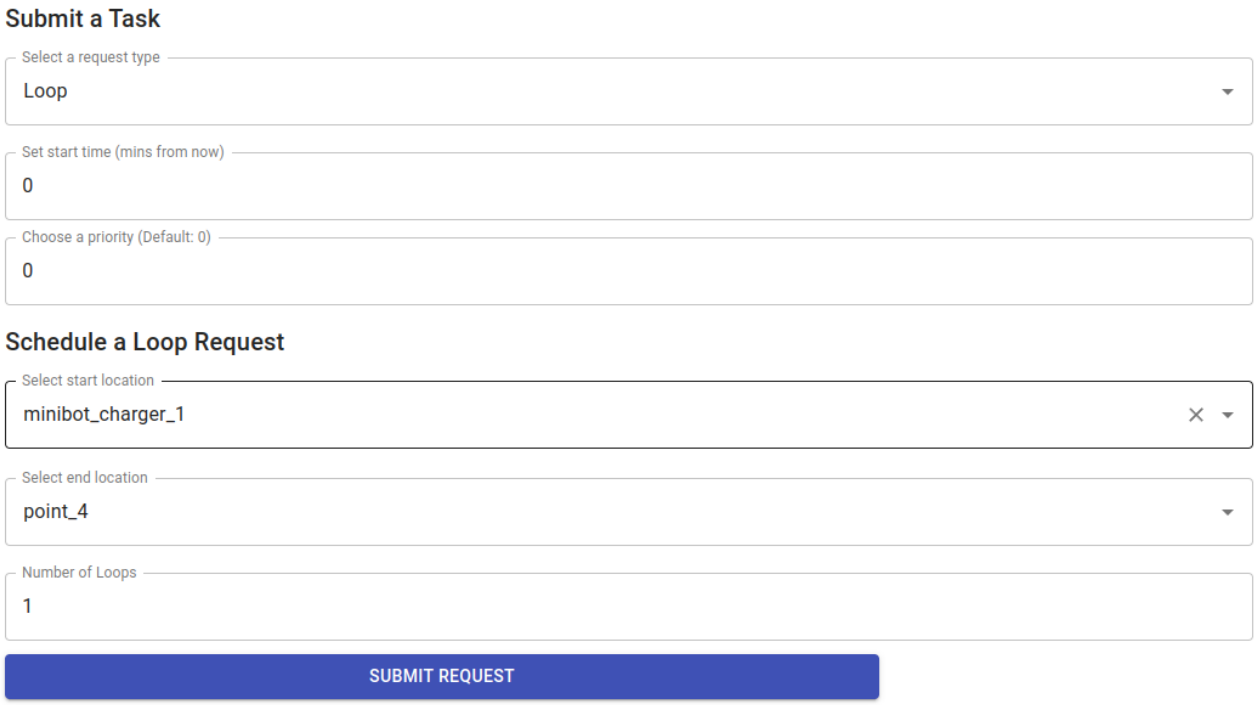
```
ros2 launch minibot_bringup bringup_robot.launch.xml # Minibot에서
ros2 launch minibot_navigation2 bringup_launch.xml \
    map:=$HOME/rmf_ws/src/rmf_demos/rmf_demos_maps/maps/pinklab_real/pinklab_L1.yaml
#PC에서 실행
```

2.11 그 다음에 RMF를 실행하면

```
ros2 launch rmf_demos pinklab_real.launch.xml
```

2.12 RMF-Panel에서 이동 시키기.

https://open-rmf.github.io/rmf-panel-js/



2.13 로봇이 이동하는 것을 볼 수 있다.

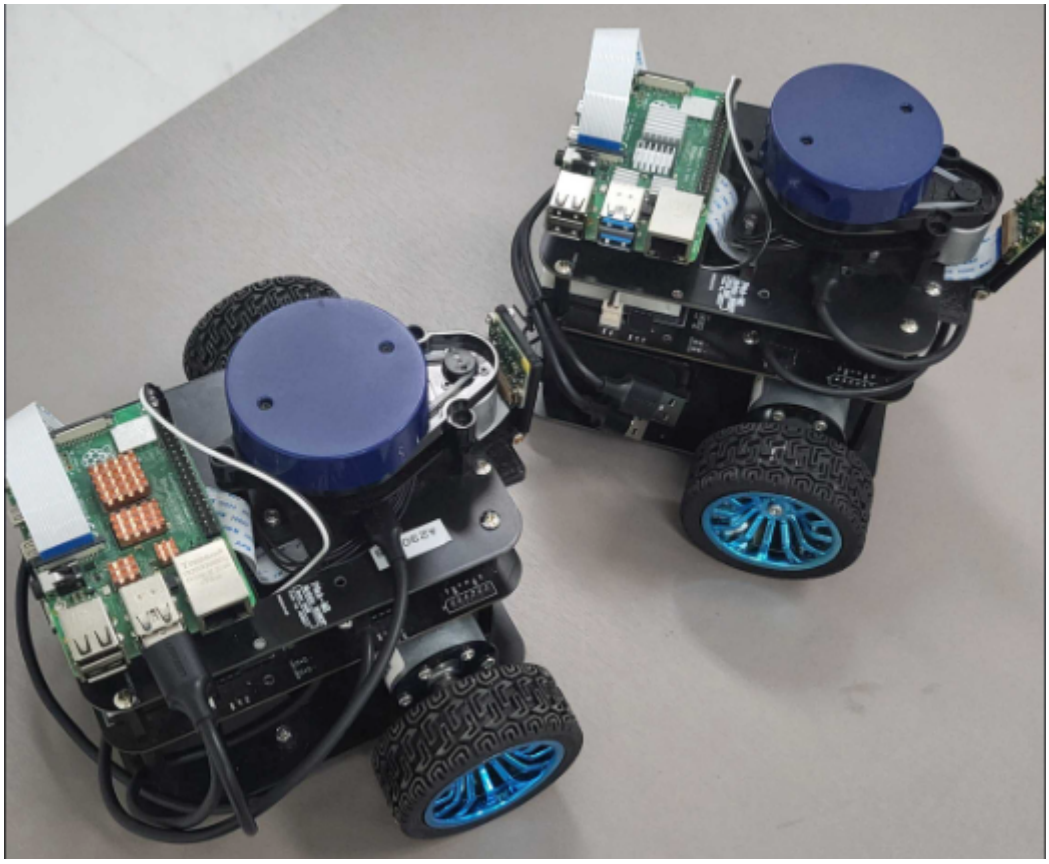
2.14 만약에 로봇 Navigation2의 inital pose가 안맞는다면 navigation2 패키지의 nav2_params패키지를 수정하자. 링크7[18]

18 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/15/15-7nav2_params.yaml

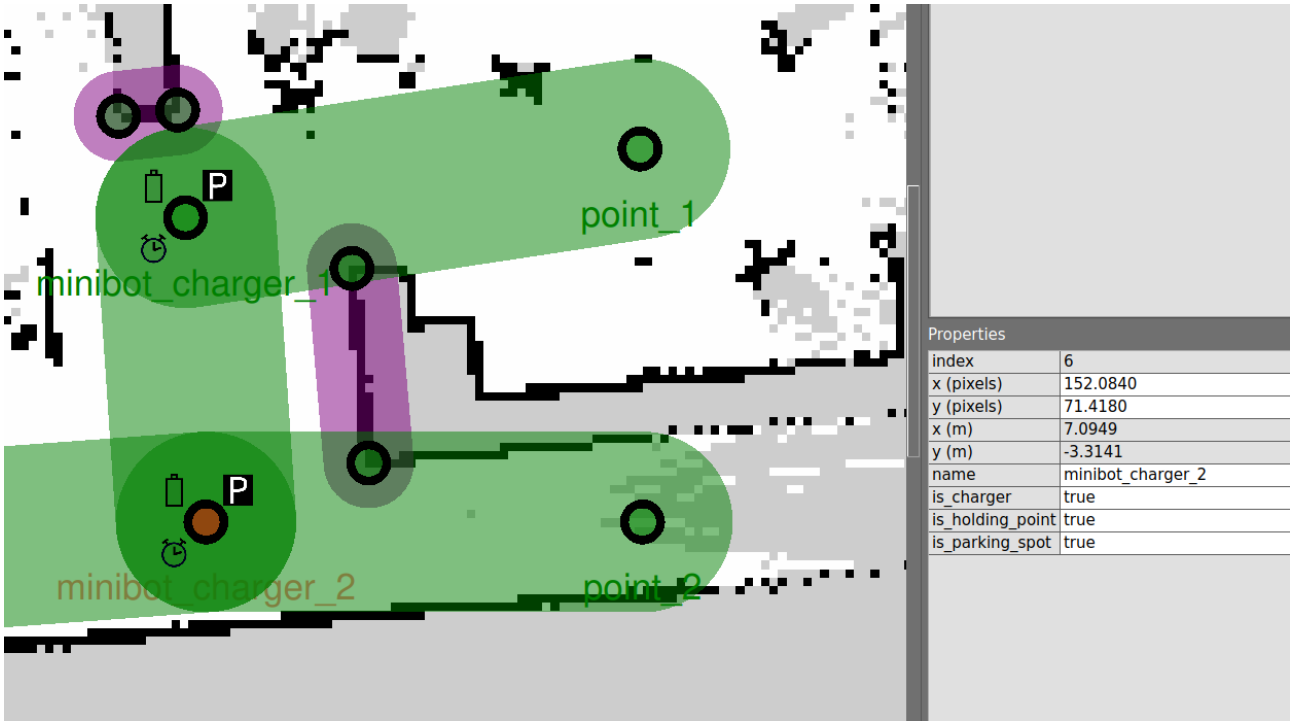
3 3. 2대 이상의 RMF 시스템 구축

3.1 RMF시스템에 같은 Fleet인 minibot을 한개 더 추가하는 방법

3.1.1 Minibot을 하나 더 다뤄보자.



3.1.2 Traffic editor를 열어서, 다른 minibot이 시작할 지점을 만들자.링크1[19]



3.1.3 fleet_config에서 방금 만든 vertex로 minibot_2의 시작점을 만들어준다. 링크2[20]

- rmf_demos/config/pinklab_real 에 minibot fleet의 config파일이 있다.

19 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/16/16-1pinklab_real.building.yaml
20 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/16/16-2minibot_config.yaml

```yaml
# minibot CONFIG ==========================================================
robots:
  minibot_1:
    robot_config:
      max_delay: 10.0 # allowed seconds of delay of the current itinerary be
    rmf_config:
      robot_state_update_frequency: 1.0
      charger:
        waypoint: "minibot_charger_1"
      start:
        map_name: "L1"
        waypoint: "minibot_charger_1"
        orientation: 0.0 # radians
  minibot_2:
    robot_config:
      max_delay: 10.0 # allowed seconds of delay of the current itinerary be
    rmf_config:
      robot_state_update_frequency: 1.0
      charger:
        waypoint: "minibot_charger_2"
      start:
        map_name: "L1"
        waypoint: "minibot_charger_2"
        orientation: 0.0 # radians
```

3.1.4 Navigation2시작시 Free Fleet client설정을 바꿔서 이름을 입력인자로 받을 수 있게 변경하자. 링크3[21]

- Navigation2 시작파일은 minibot_navigation2/launch/bringup_launch.xml 에 있다.

21 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/16/16-3bringup_launch.xml

```
<launch>
    <set_env name="RCUTILS_LOGGING_BUFFERED_STREAM"

    <arg name="namespace" default="" />
    <arg name="map" default="$(find-pkg-share minibo
    <arg name="use_sim_time" default="False" />
    <arg name="params_file" default="$(find-pkg-shar
    <arg name="autostart" default="True" />
    <arg name="container_name" default="nav2_contain
    <arg name="use_composition" default="True" />
    <arg name="use_respawn" default="False" />
    <arg name="log_level" default="info" />
    <arg name="lifecycle_nodes_local" default="['map
    <arg name="lifecycle_nodes_nav"
        default="['controller_server',
                  'smoother_server',
                  'planner_server',
                  'behavior_server',
                  'bt_navigator',
                  'waypoint_follower',
                  'velocity_smoother']" />
    <arg name="robot_name" default="minibot_1" />
```

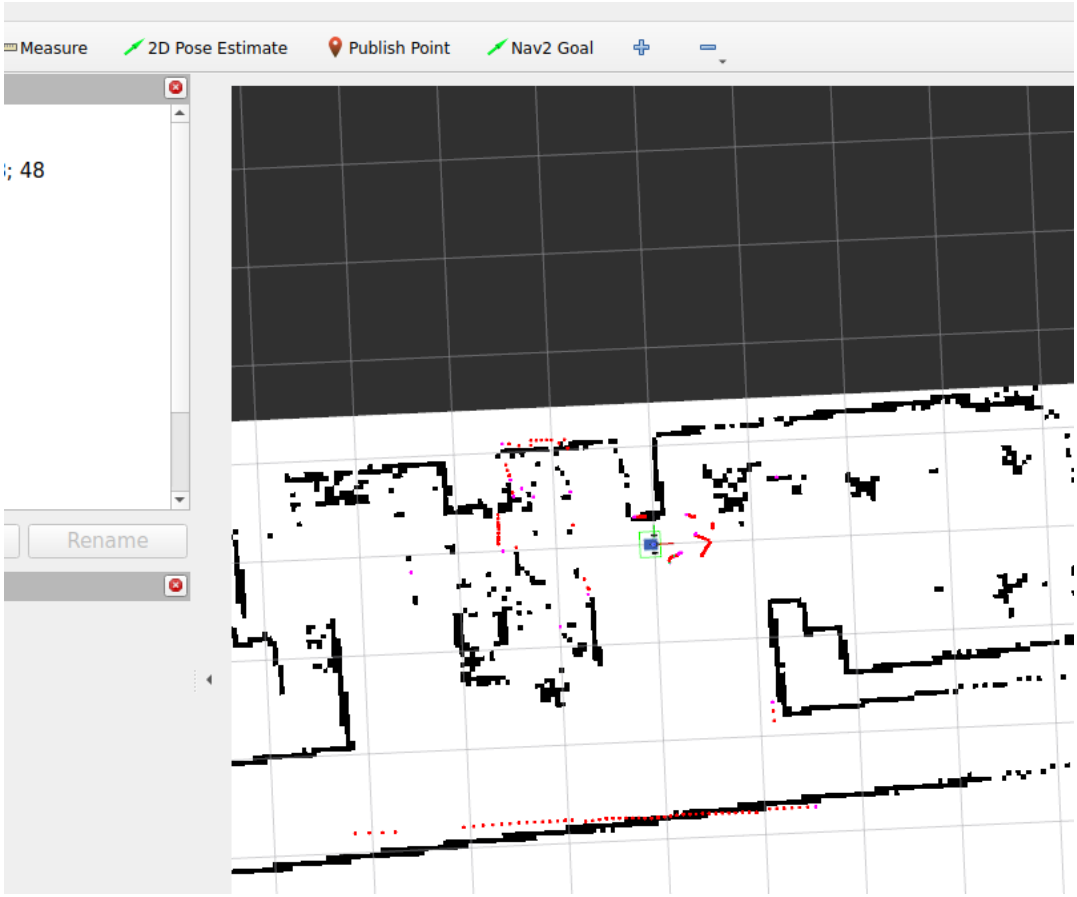
Free Fleet client parameter중 name을 argument로 받기.

```
<!-- launch the free fleet client first -->
<node name="minibot_free_fleet_client_node" pkg="free_fleet_client_ros2"
    exec="free_fleet_client_ros2" output="both">
    <param name="fleet_name" value="minibot" />
    <param name="robot_name" value="minibot_1" />
    <param name="robot_model" value="minibot" />
    <param name="level_name" value="L1" />
    <param name="dds_domain" value="43" />
    <param name="max_dist_to_first_waypoint" value="10.0" />
    <param name="map_frame" value="map" />
    <param name="robot_frame" value="base_footprint" />
    <param name="nav2_server_name" value="/navigate_to_pose" />
    <param name="use_sim_time" value="False" />
    <param name="dds_domain" value="43" />
</node>
```

```
<!-- launch the free fleet client first -->
<node name="minibot_free_fleet_client_node" pkg="free_fleet_client_ros2"
    exec="free_fleet_client_ros2" output="both">
    <param name="fleet_name" value="minibot" />
    <param name="robot_name" value="$(var robot_name)" />
    <param name="robot_model" value="minibot" />
    <param name="level_name" value="L1" />
    <param name="dds_domain" value="43" />
    <param name="max_dist_to_first_waypoint" value="10.0" />
    <param name="map_frame" value="map" />
    <param name="robot_frame" value="base_footprint" />
    <param name="nav2_server_name" value="/navigate_to_pose" />
    <param name="use_sim_time" value="False" />
    <param name="dds_domain" value="43" />
</node>
```

3.1.5 빌드 후 실행해보자

```
ros2 launch minibot_bringup bringup_robot.launch.xml # Minibot에서 (SSH로 minibot_1접속 후)
ros2 launch minibot_navigation2 bringup_launch.xml \
    map:=$HOME/rmf_ws/src/rmf_demos/rmf_demos_maps/maps/pinklab_real/pinklab_L1.yaml
#PC에서 실행
rviz2 -d `ros2 pkg prefix minibot_navigation2`/share/minibot_navigation2/rviz/nav2_view.rviz
```



3.1.6 minibot_navigation2/param에서 nav2_params.yaml을 복사한 뒤, Rviz2에서 minibot_2의 spawn 지점을 initial_pose로 지정하자. 링크4[22]

3.1.7 minibot_2는 다른 ROS_DOMAIN에서 실행하자.

- minibot_2의 ROS_DOMAIN_ID를 PC와 다르게 설정하고, 새 터미널에서 ROS_DOMAIN_ID를 맞춰준 후 Navigation2을 시작하자.
- params_files을 입력인자는 방금 만든 `nav2_params_minibot2.yaml` 를 입력하자.

22 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/16/16-4nav2_params_minibot2.yaml

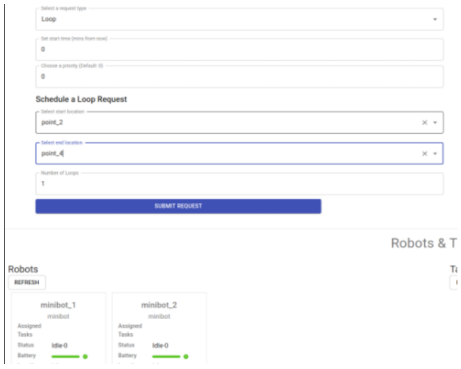
```
# minibot에서
export ROS_DOMAIN_ID=33 #Minibot에서 (SSH로 minibot_2접속 후)
ros2 launch minibot_bringup bringup_robot.launch.xml # Minibot에서 (SSH로 minibot_2접속
후)

# PC에서
export ROS_DOMAIN_ID=33 # PC에서 실행
ros2 launch minibot_navigation2 bringup_launch.xml \
    map:=$HOME/rmf_ws/src/rmf_demos/rmf_demos_maps/maps/pinklab_real/pinklab_L1.yaml
\
    robot_name:=minibot_2 \
    params_file:=$HOME/rmf_ws/src/pinklab_minibot_robot/minibot_navigation2/params/
nav2_params_minibot2.yaml
```

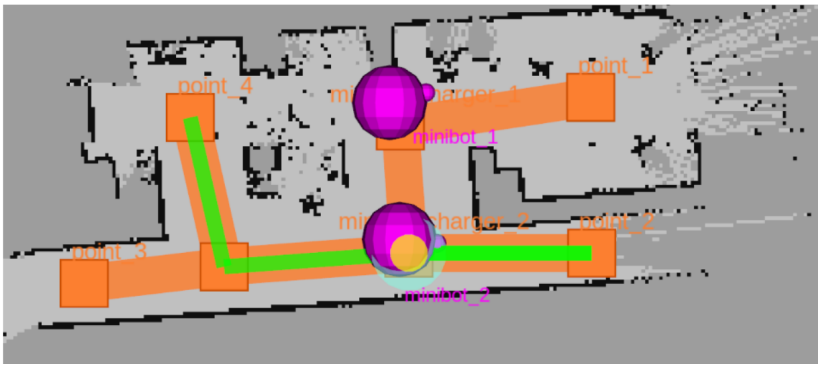
3.1.8 그 다음 RMF를 실행하면...

```
ros2 launch rmf_demos pinklab_real.launch.xml
```

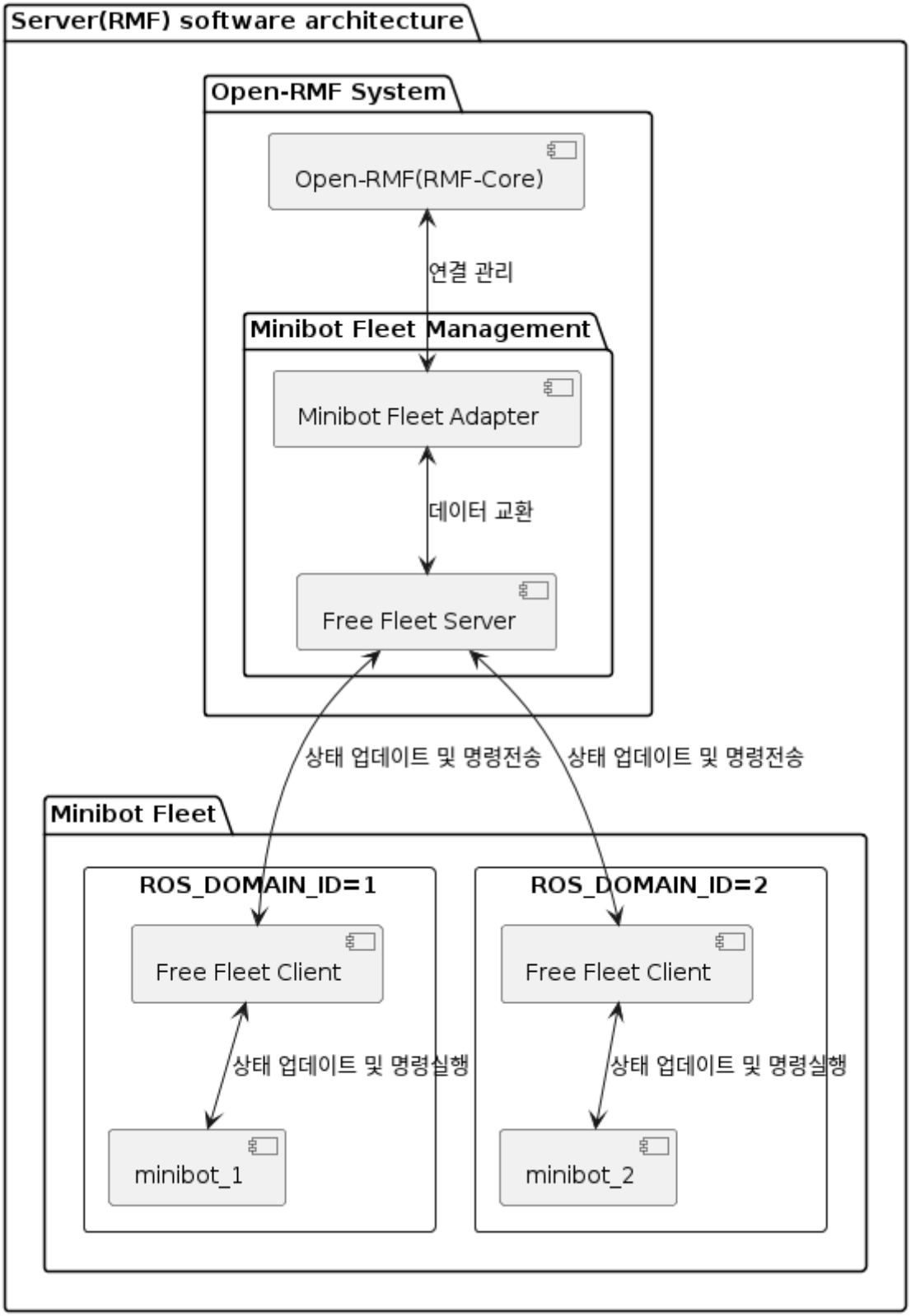
3.1.9 Panel로 Task를 주면 입찰받은 minibot이 움직이게 된다.

https://open-rmf.github.io/rmf-panel-js/


Loop
0
0
Schedule a Loop Request
point_2
point_4
1
SUBMIT REQUEST
Robots & T
Robots
REFRESH
minibot_1
minibot
Assigned Tasks
Status
Idle 0
Battery
minibot_2
minibot
Assigned Tasks
Status
Idle 0
Battery


point_4
point_1
minibot_1
point_3
point_2
minibot_2

3.1.10 같은 Fleet만 사용할 경우 RMF시스템의 소프트웨어 아키텍쳐 (예시)


Server(RMF) software architecture
Open-RMF System
Open-RMF(RMF-Core)
연결 관리
Minibot Fleet Management
Minibot Fleet Adapter
데이터 교환
Free Fleet Server
상태 업데이트 및 명령전송
상태 업데이트 및 명령전송
Minibot Fleet
ROS_DOMAIN_ID=1
Free Fleet Client
상태 업데이트 및 명령실행
minibot_1
ROS_DOMAIN_ID=2
Free Fleet Client
상태 업데이트 및 명령실행
minibot_2

3.2 다른 Fleet를 추가하기.

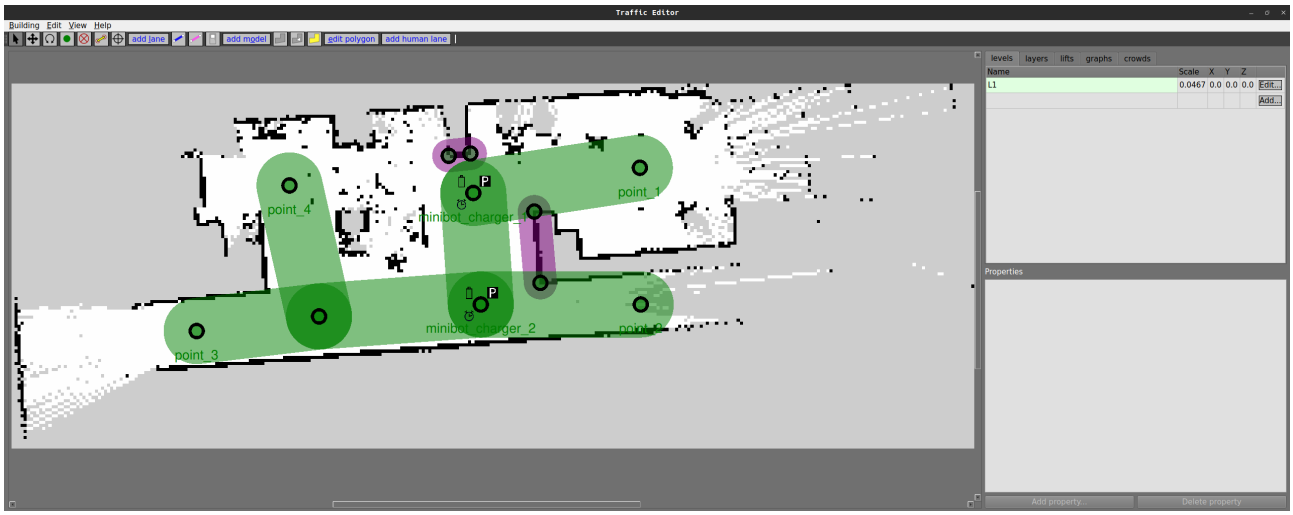
3.2.1 Minibot과 다른 로봇인 Turtlebot3를 RMF시스템에 추가해보자.

3.2.2 Turtlebot3 ROS2 PKG 다운로드

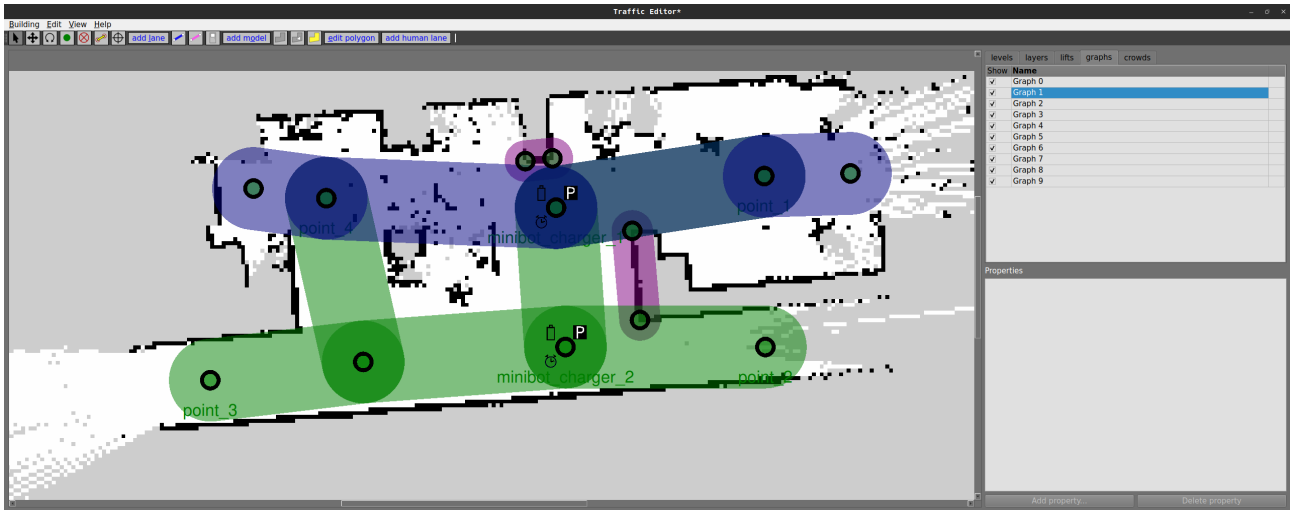
```
cd ~/rmf_ws/src/
git clone -b humble-devel https://github.com/ROBOTIS-GIT/DynamixelSDK.git
git clone -b humble-devel https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
git clone -b humble-devel https://github.com/ROBOTIS-GIT/turtlebot3.git
cd ~/rmf_ws
```

```
colcon build --symlink-install
echo 'export TURTLEBOT3_MODEL=burger' >> ~/.bashrc
source ~/.bashrc
```

3.2.3 Traffic editor를 열고...



3.2.4 Graph 1번으로 Minibot Fleet와는 다른 경로를 지정하자.



3.2.5 Spawn 지점과 관련된 설정을 하고, 저장 및 종료. 링크5[23]

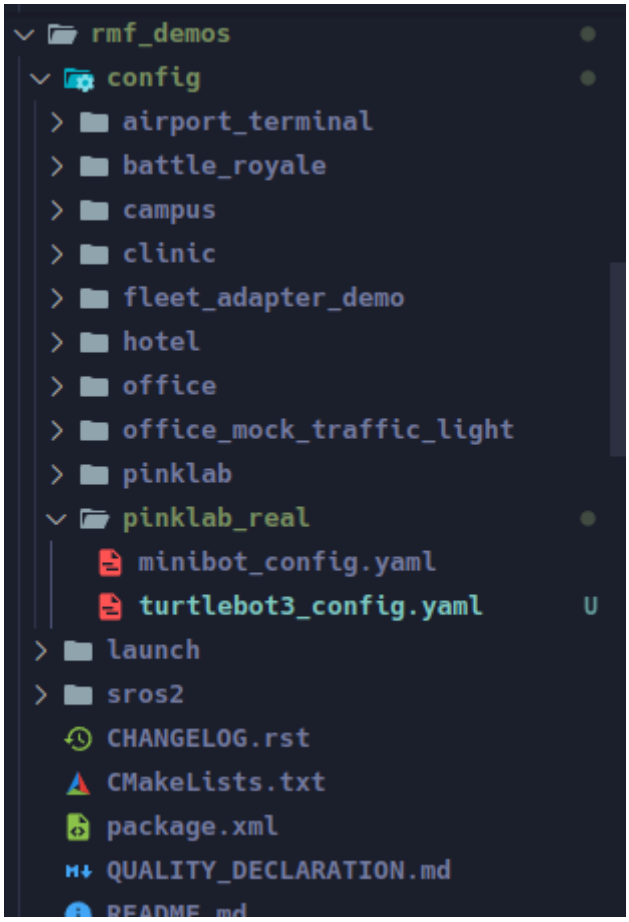
3.2.6 그 다음 Fleet adapter를 구축해야한다.

- minibot fleet adapter와 다른점은 turtlebot fleet adapter는 배터리정보를 포함한다.
- 그 외에는 똑같다..

```
cd ~/rmf_ws
git clone https://github.com/pinklabstudio/turtlebot3_fleet_adapter.git -b main
colcon build --symlink-install
```

23 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/16/16-5pinklab_real.building.yaml

3.2.7 Turtlebot3의 RMF정보를 작성할 yaml파일을 만들고...



3.2.8 Turtlebot3의 위치정보, 개체, 좌표변환을 작성. 링크6[24]

```yaml
rmf_fleet:
  name: "turtlebot3"
  fleet_manager:
    prefix: "http://127.0.0.1:22017"
    ip: "127.0.0.1"
    port: 22017
    user: "some_user"
    password: "some_password"
  limits:
    linear: [0.5, 0.75] # velocity, acceleration
    angular: [0.6, 2.0] # velocity, acceleration
  profile: # Robot profile is modelled as a circle
    footprint: 0.3 # radius in m
    vicinity: 0.5 # radius in m
  reversible: True # whether robots in this fleet can reverse
  battery_system:
    voltage: 12.0 # V
    capacity: 24.0 # Ahr
    charging_current: 5.0 # A
  mechanical_system:
    mass: 20.0 # kg
    moment_of_inertia: 10.0 #kgm^2
    friction_coefficient: 0.22
  ambient_system:
    power: 20.0 # W
  tool_system:
    power: 0.0 # W
  recharge_threshold: 0.10 # Battery level below which robots in tl
  recharge_soc: 1.0 # Battery level to which robots in this fleet s
  publish_fleet_state: 10.0 # Publish frequency for fleet state, er
  account_for_battery_drain: True
  task_capabilities: # Specify the types of RMF Tasks that robots i
    loop: True
    delivery: False
    clean: False
    finishing_request: "park" # [park, charge, nothing]
  actions: ["teleop"]

  responsive_wait: True # Should responsive wait be on/off for the
```

```yaml
# turtlebot3 CONFIG ====================================
robots:
  turtlebot3_1:
    robot_config:
      max_delay: 10.0 # allowed seconds of delay of the c
    rmf_config:
      robot_state_update_frequency: 1.0
      charger:
        waypoint: "turtlebot3_charger_1"
      start:
        map_name: "L1"
        waypoint: "turtlebot3_charger_1"
        orientation: 0.0 # radians

# TRANSFORM CONFIG =====================================
# For computing transforms between Robot and RMF coordina

# Optional
reference_coordinates:
  rmf: [[0.0, 0.0],
        [14.9834, 0.0],
        [14.9834, -5.5079],
        [0.0, -5.5079]]
  robot: [[ -7.311037063598633, 1.4209141731262207],
          [ 8.718361854553223, 1.4333285093307495],
          [ 8.712226867675781, -4.453343868255615],
          [ -7.3213982582092285, -4.454360485076904]]
```

3.2.9 RMF시스템 시작 시 turtlebot3 Fleet adapter를 같이 시작하도록 설정. 링크7[25]

- RMF시스템을 시작하는 파일은 rmf_demos/launch/pinklab_real.launch.xml에 있다.

```xml
<!-- turtlebot3 fleet adapter -->
<include file="$(find-pkg-share turtlebot3_fleet_adapter)/launch/fleet_turtlebot3_adapter.launch.xml">
    <arg name="use_sim_time" value="$(var use_sim_time)" />
    <arg name="nav_graph_file"
        value="$(find-pkg-share rmf_demos_maps)/maps/pinklab_real/nav_graphs/1.yaml" />
    <arg name="config_file"
        value="$(find-pkg-share rmf_demos)/config/pinklab_real/turtlebot3_config.yaml" />
</include>
```

24 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/16/16-6turtlebot3_config.yaml
25 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/16/16-7pinklab_real.launch.xml

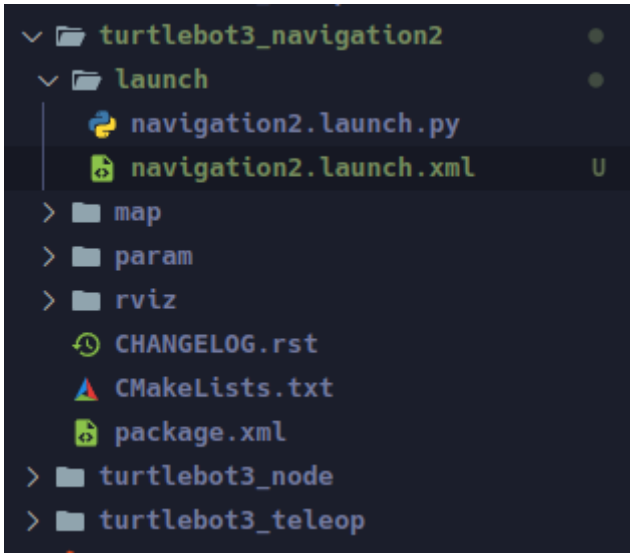
3.2.10 RMF시스템에 turtlebot3의 navigation2정보를 취합할 Free Fleet Server코드 작성. 링크8[26]

```
<!-- launch the free fleet server -->
<node pkg="free_fleet_server_ros2"
    exec="free_fleet_server_ros2"
    name="turtlebot3_fleet_server_node"
    output="both">

    <param name="fleet_name" value="turtlebot3" />

    <param name="fleet_state_topic" value="turtlebot3/fleet_states" />
    <param name="mode_request_topic" value="turtlebot3/robot_mode_requests" />
    <param name="path_request_topic" value="turtlebot3/robot_path_requests" />
    <param name="destination_request_topic" value="turtlebot3/robot_destination_requests" />

    <param name="dds_domain" value="44" />
    <param name="dds_robot_state_topic" value="robot_state" />
    <param name="dds_mode_request_topic" value="mode_request" />
    <param name="dds_path_request_topic" value="path_request" />
    <param name="dds_destination_request_topic" value="destination_request" />

    <param name="update_state_frequency" value="20.0" />
    <param name="publish_state_frequency" value="2.0" />

    <param name="translation_x" value="0.0" />
    <param name="translation_y" value="0.0" />
    <param name="rotation" value="0.0" />
    <param name="scale" value="1.0" />

</node>
```

3.2.11 Navigation2을 시작하면서, Free fleet client를 같이 시작하기 위해 파일을 한 개 만들고...

- turtlebot3 pkg중 turtlebot3_navigation2의 navigation2.launch.xml을 만들자.

26 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/16/16-8pinklab_real.launch.xml



3.2.12 Navigation2과 Free Fleet client 코드를 작성. 링크9[27]

```xml
<?xml version='1.0'?>
<launch>
    <!-- Arguments -->
    <arg name="map" default="" />

    <!-- launch the navigation2 stack -->
    <include file="$(find-pkg-share turtlebot3_navigation2)/launch/navigation2.launch.py">
        <arg name="map" value="$(var map)" />
    </include>

    <!-- launch the free fleet client first -->
    <node name="turtlebot3_free_fleet_client_node" pkg="free_fleet_client_ros2"
        exec="free_fleet_client_ros2" output="both">
        <param name="fleet_name" value="turtlebot3" />
        <param name="robot_name" value="turtlebot3_1" />
        <param name="robot_model" value="turtlebot3" />
        <param name="level_name" value="L1" />
        <param name="dds_domain" value="44" />
        <param name="max_dist_to_first_waypoint" value="10.0" />
        <param name="map_frame" value="map" />
        <param name="robot_frame" value="base_footprint" />
        <param name="nav2_server_name" value="/navigate_to_pose" />
        <param name="use_sim_time" value="False" />
    </node>
</launch>
```

27 https://github.com/pinklabstudio/rmf_demos/blob/dev_edudata/education_code/16/16-9navigation2.launch.xml

3.2.13 빌드한 뒤, 먼저 Minibot을 다 실행하고...

```
ros2 launch minibot_bringup bringup_robot.launch.xml # Minibot에서 (SSH로 minibot_1접속
후)
ros2 launch minibot_navigation2 bringup_launch.xml \
    map:=$HOME/rmf_ws/src/rmf_demos/rmf_demos_maps/maps/pinklab_real/pinklab_L1.yaml
#PC에서 실행
rviz2 -d `ros2 pkg prefix minibot_navigation2`/share/minibot_navigation2/rviz/
nav2_view.rviz
```

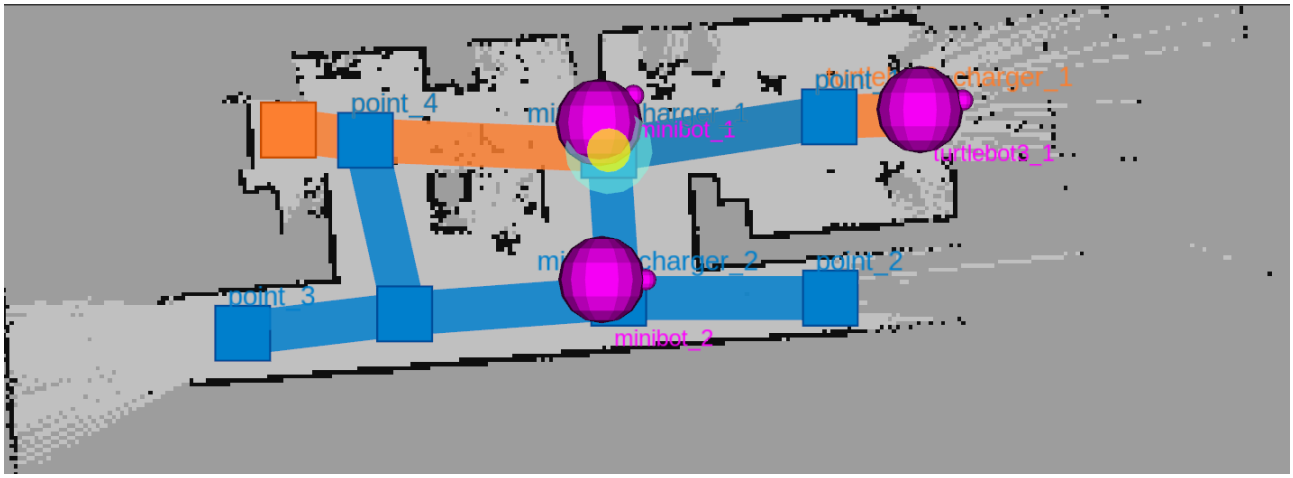
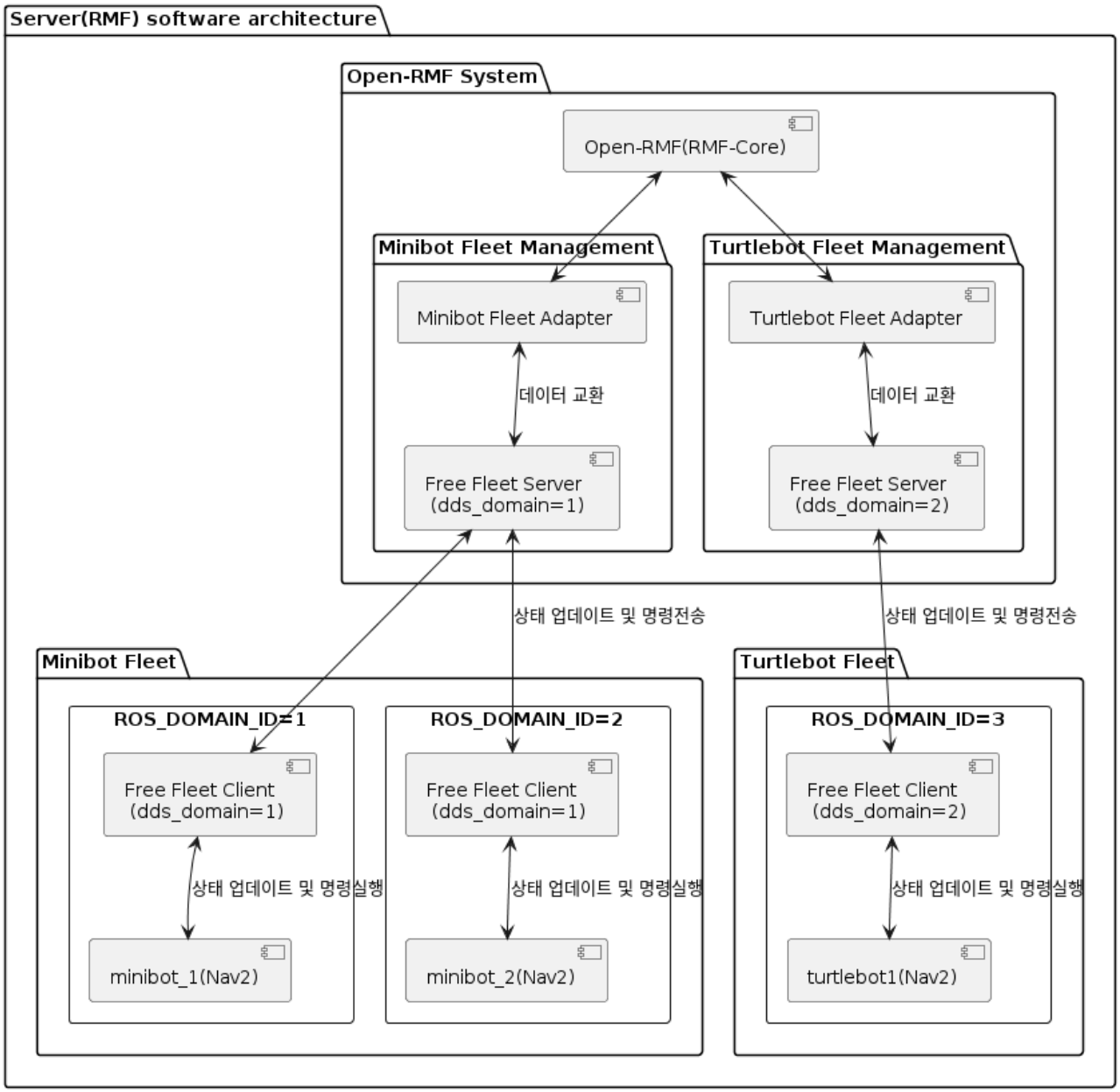
```
# minibot에서
export ROS_DOMAIN_ID=33 #Minibot에서 (SSH로 minibot_2접속 후)
ros2 launch minibot_bringup bringup_robot.launch.xml # Minibot에서 (SSH로 minibot_2접속
후)

# PC에서
export ROS_DOMAIN_ID=33 # PC에서 실행
ros2 launch minibot_navigation2 bringup_launch.xml \
    map:=$HOME/rmf_ws/src/rmf_demos/rmf_demos_maps/maps/pinklab_real/pinklab_L1.yaml
\
    robot_name:=minibot_2 \
    params_file:=$HOME/rmf_ws/src/pinklab_minibot_robot/minibot_navigation2/params/
nav2_params_minibot2.yaml
```

3.2.14 Turtlebot3 실행

```
# turtlebot3에서
export ROS_DOMAIN_ID=34 #turtlebot에서 (SSH로 turtlebot3_1 접속 후)
ros2 launch turtlebot3_bringup robot.launch.py # turtlebot에서 (SSH로 turtlebot3_1접속
후)
# PC에서
export ROS_DOMAIN_ID=34 #turtlebot에서 (SSH로 turtlebot3_1 접속 후)
ros2 launch turtlebot3_navigation2 navigation2.launch.xml \
    map:=$HOME/rmf_ws/src/rmf_demos/rmf_demos_maps/maps/pinklab_real/pinklab_L1.yaml
```

3.2.15 그 다음 RMF를 실행하면

```
ros2 launch rmf_demos pinklab_real.launch.xml
```


point_4
charger_1
minibot_1
turtlebot3_1
charger_2
point_2
point_3
minibot_2

3.2.16 타 기종을 포함하는 RMF 시스템의 소프트웨어 아키텍쳐 예시는 다음과 같다.