4) MAKE ROBOT NAVIGATE WITH Nav 2

- we have briefly described the 1st step of Robot Navigation
  creating a map ☑ achieved by SLAM algorithms.
  ↓
  we generate a map { <map>.pgm
  <map>.yaml

- now it's time for the 2nd step: Navigate from A to B using the map created ...
  again we will start the robot on the mapped environment simulation / real robot
  ⇓
  On this, we will start Navigation Stack giving as input the map previously generated → in RViz, we will have a new interface to use in RViz to give navigation commands

- Go to any point in the map (feasible) avoiding existing obstacles + new obstacles added dynamically during navigation

---

FIX BEFORE STARTING, to make Nav2 WORK IN HUMBLE

when loading map, the map may be NOT loaded properly in time, preventing successive steps...
↓ Fixing the bugs... in two main steps

(Data Distribution System, ROS 2 communication is based on this)

1) change DDS from FAST to CYCLONE

this will allow ROS2 Nav2 running properly

install: ros-humble-rmw-cyclonedds-cpp

and once installed, tell ROS2 to use it: (exporting env variable)
editing bashrc, we add to it:

export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp

im Humble the map and navigation is NOT correctly working with default DDS...
this 1ST FIX should solve the map issue

2) modify parameter file where turtlebot3 navigation package is installed...

NOTICE: this is NOT good practice since if im the future this package get updated, you will need to do it again
(do it only temporary, probably this issue will be fixed)

go to /opt/ros/humble where ROS is installed,
then the parameters files will be in .../share folder,
move to .../turtlebot3_navigation2
on this folder, go to .../param and edit the model.yaml with the model of TURTLEBOT3 im use
(im our case waffle.yaml) ⇐ edit it with sudo (super user needed)

Im this file, edit it to:

robot_model_type: "nav2_amcl::DifferentialMotionModel"

(reboot the computer after this modification)

MAKE ROBOT NAVIGATE in the generated map

For now we will work with turtlebot3, later on we will run navigation on a custom robot

- FIRST, we start the robot simulation (gazebo) as usual
- start navigation, by launching

containing all nodes, etc. to run navigation

`turtlebot3_navigation2` `navigation2.launch.py`

`use_sim_time := True` `map := .../<map>.yaml`

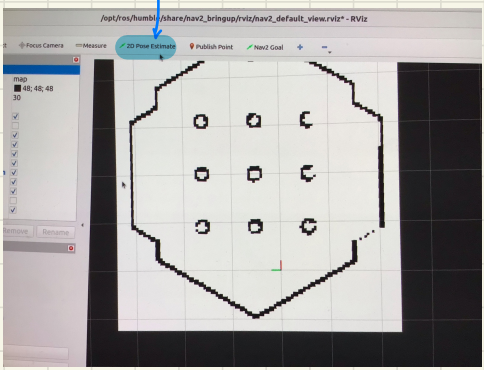
working in sim environment

provide the saved map to the navigation stack, by relative path to the map

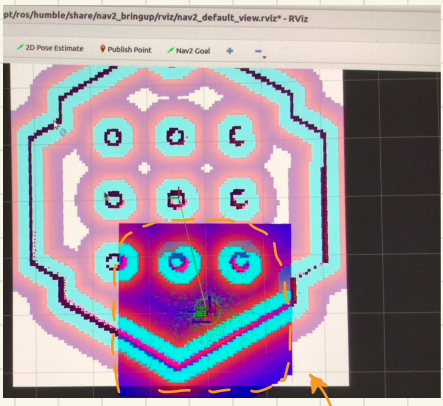
⇓

this will start Rviz with the map, with origin as specified in yaml file
(IF map NOT loading, try to restart)

- On Rviz, to start navigation, we need to give an **initial 2D Pose estimate**, estimation of where robot can be



click on it, and then place the robot pose according to simulation initial pose



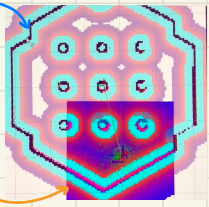
we have communicated to Rviz where is the Robot now...
So, navigation localize the ROBOT there, and using the data from sensors (gazebo), the environment is matched
When robot move, localize itself in the map

IF I give a WRONG 2D pose estimate... laser scan data will NOT correspond to map in Rviz

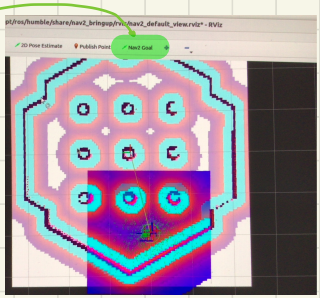
Be as close as possible to real pose!

the navigation is based on { global planner, local planner (controller) }

(details later on...)

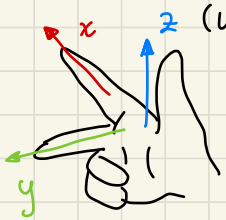


- once given 2D Pose estimate, the match map / sensors is correct
  ↓
  we can send a Nav2 Goal



by selecting it and define desired orientation.
this will start motion in simulation

NOTICE: coordinate system uses right-hand rule (used by TF to represent relative pose)



In Rviz we have special important frame for navigation, we will discuss details later on

IF we give unfeasible Goal pose...
ROBOT try to move, start Recovery behaviour... Feedback: aborted
↓
NO path found, Goal gets aborted (feedback from action)
If goal is feasible, we get Feedback: reached

When relaunching Navigation, from initial 2D Pose estimate, the robot try to match as close as possible to sensor data to localize. and map will converge to correct pose

In summary, to run Navigation { 1) 2D Pose estimate
2) Nav2 Goal

↓

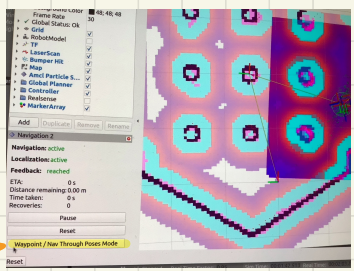
later on we will manage this two steps NOT in Rviz by hands, BUT through code, interacting with Nav2 stack

WAYPOINT FOLLOWER, Multiple Nav2 Goals

Usefull functionality in Nav2, to define different points.
Instead of defining each point as Nav2 Goal, an alternative tools exist

↓

It is possible to send more goals all at once



after selecting this Nav2 tool, go to Nav2 Goal as usual, and you can define "wp_i": waypoint i

then >> start Waypoint Following on bottom left
Will start the navigation going on all the defined waypoints

If we select >> start Nav Through Poses instead, create path through All waypoints without stopping
usefull for fast navigation, but less stable than waypoint Following

DYNAMIC OBSTACLE AVOIDANCE

Important feature of Nav2 stack ⇝ avoid obstacles dynamically

- the Map is already known, so the static obstacles are already known for the Robot
  What if a new object NOT previously in the map?
  ⇓
  Navstack should be able to avoid it and replan path.

- To test it in Gazebo simulation, we can >>insert New obstacles to the world
  ↓
  to see dynamic obstacle avoidance in action,
  we can PAUSE the simulation in Gazebo (during navigation)
  and add an obstacle object in the path

  Robot can still try to go through the obstacle...
  depends on sensor visibility!
  IF we add an obstacle that can't be properly estimated,
  then there will be issues for dynamic adjustment
  ↳anyway new path will be computed ⇒ Bad behaviours can occur!

  IF the obstacle is correctly perceived, even if NOT part of static map, it will update the path at runtime and then map will be cleared if dynamic obstacle get removed or move from previous position

  (this dynamic obstacle avoidance is achieved by the controller / local planner)

- **ASSIGNMENT 2** Using the "house world", experiment with navigation on this environment. Thanks to map of ASSIGNMENT 1
  (remember to load correct map.yaml as argument)
  Objective: Patrol over the whole house by Nav2 Goal or waypoint Follower

NOTE :
- when moving on cluttered environment, navigation may get stucked and even if a feasible path exists, the navigation is aborted.
  Especially when manouvers on narrow environments are required (like "U" turn inside rooms)

- Later on we will explore how to modify navigation parameters.
  This can tune navigation for your specific robot, and optimize it!

Navigation Steps (ROS2 Nav2 Course - Section 4)

Those commands are the ones you will run in this section on Navigation. Use this PDF to easily access them while doing the exercises.

Some of the commands are specific to the Turtlebot3 robot, which we use as an example. Later on in the course (Section 7), you will also get the general commands to run for any robots.

Steps

When you see some text in red, replace it with the correct value.

1. Start your robot stack

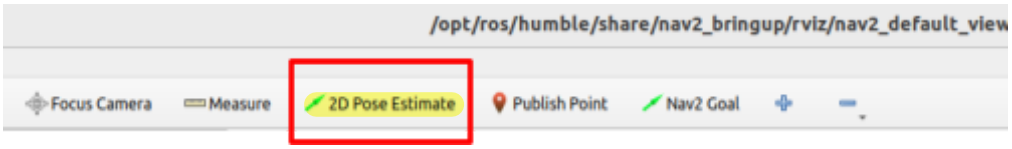
```
$ ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py
```

2. Start Navigation 2

```
$ ros2 launch turtlebot3_navigation2 navigation2.launch.py use_sim_time:=True
map:=path/to/world_map.yaml
```

3. Set 2D Pose Estimate

Click on RViz



4. Send a Navigation2 Goal

Click on RViz