3) GENERATE MAP WITH SLAM

To achieve the goal of navigation

" A to B Navigation avoiding collision "

↓ we start from MAP creation of the WORLD (the space where ROBOT can move)

↓

Generating a map is important in next step, so Nav stack can perform Navigation in any point of the world!

↑

Without a MAP could work Navigation, but without the map knowledge will take long time

that's why we separate in 2 main steps as seen before.

To Generate MAP, we rely on SLAM (Simultaneus Localization And Mapping)

↓

Allows Robot to localize in environment, relative to obtacles/walls etc around + map the environment

Once ROBOT and Simulation / Real world are set-up you can start SLAM to generate MAP

MAP is being generated as the ROBOT moves in the world

Once MAP is generated, we save it and use for Navigation!

ROBOT MOVEMENT

FIRST of all, we need to have the ROBOT moving in a simulated world

FOR NOW based on turtlebot 3 ← all already implemented for this

Later on... Nav with custom robot

↓

for NOW, consider we need a ROBOT able to move in an environment with ROS2 + teleop node (move it!)

- how to make turtlebot 3 move ⇒ then we will see SCAM and map generation

↓

first specify model of ROBOT we are going to use, by exporting environment variable for example by modifying bashrc

example of turtlebot 3 model

```
export TURTLEBOT3_MODEL = waffle
```

to launch Gazebo simulation, we launch

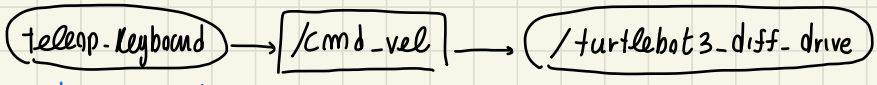
```
turtlebot3_gazebo turtlebot3_world.launch.py
```

⇑

differential drive Robot with a 2D LIDAR (~4 meter scan) ↑ to create an environment map (x,y plane)
spawn in a closed world

Then, to make ROBOT move we run

```
turtlebot3_teleop teleop_keyboard
```



send commands to /cmd_vel

robot listen to this topic with its controller, making robot move

This is all we need to MAP the environment

GENERATE AND SAVE MAP

ROBOT + a way to move it, is what we need for SLAM and MAP creation

- launch the gazebo simulation world

- launch SLAM feature of turtlebot3

  turtlebot3_cartographer cartographer.launch.py use_sim_time := True

  additional argument because we work in Gazebo

  ↳ this will start Rviz2 (visualization tool)
  where the map is generated from the data generated by LASER SCAN

  ↓

  the goal is to generate entire MAP and save it

{ NOTICE: Gazebo replace real robot (simulation physics)
Rviz is a 3D visualization tool }

- launch teleop node to move the robot
  → during motion, in Rviz we see the map updating (while TF moves in Rviz)

MAP ~ { White pixels := free space
Black pixels := obstacles space
Grey pixels := still unknown }

Possible issues:

- Turning too fast create problems during mapping
- Being LaserScan in 2D plane, if ROBOT hit obstacle and oscillate vertically, map data can become noisy

in case of issues, restart SLAM and simulation

Once we finish the map exploration, we recover all map information.

↓

In Rviz we see the final map (minor missing pixels are NOT a problem for navigation)

Now we can SAVE the MAP before closing the SLAM + simulation

have a DIRECTORY for the /maps

then run

provide path and name

```
nav2_map_server map_saver_cli -f maps/<name>
```

no extension required

⇓

this will create a

<name>.pgm and <name>.yaml file

↓ ↓

WHAT IS INSIDE THE GENERATED MAP

Explore what are important things about the saved map.

<name>.pgm map image, White / black / gray pixels

↓ free / obstacles / unknown

this will be loaded when running Navigation to find easily path to destination

<name>.yaml contain different information

- image: relative path to image file (.pgm)
- resolution: in [m / pixel]
  IF for example 0.05 : each pixel is 5 cm (precision)
  the precision requirements depends on application and on the environment into account
- origin: coordinates of lowest left point in the map (bottom left corner)
  It will depends on where the robot mapping started

- negate: 0/1 IF 1 everything will be reversed free ⇆ obstacle

- occupied_thresh: clear separation free/obstacle... BUT in principle MAP is based on probability that pixel is occupied or NOT!
  ex) IF threshold is 0.65, means that IF pixel is occupied with more than 65% chance, it is considered occupied
- free_thresh: If probability of pixel to be occupied is less than this, the pixel is free
  ex) IF 0.25, cell free when less than 25% probability to be occupied

⇓

[IF pixel more than 65% occupied ("into black direction") → black
  less than 25% occupied ("into dark value) → white]

By opening the <names.pgm on the bash with `nano <names.pgm` we can see the pixels of the image

Nx ↑ # pixel horiz Ny ↑ # pixel vertical ⇒ $N_x \cdot \text{resolution} = L_x$, $N_y \cdot \text{resolution} = L_y$ } give you the map dimension in meters

- **ASSIGNMENT 1**

Practice on generating map with SLAM.
Generate a map for a new world "turtlebot3_house"

Launching `turtlebot3_gazebo turtlebot3_house.launch.py`

→ in this world, we have a simple house environment (open in the outside, NOT closed in walls)

Follow the same procedure explained before to generate map and save it (just map around the rooms, NOT open world...)

[NOTICE When mapping, be sure NOT to have "holes" on the walls and other noisy informations that will then effect navigation.]

- RECAP: (pdf provided as course resource)

SLAM Steps (ROS2 Nav2 Course - Section 3)

Those commands are the ones you will run in this section on SLAM. 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 (simulation environment)

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

2. Start SLAM (mapping from sensor data)

```
$ ros2 launch turtlebot3_cartographer cartographer.launch.py use_sim_time:=True
```

3. Make the robot move to generate the map (teleoperate around to retrieve data)

```
$ ros2 run turtlebot3_teleop teleop_keyboard
```

4. Save the map

```
$ ros2 run nav2_map_server map_saver_cli -f ~/my_map
```