PRE-REQUISITES

- Download UBUNTU 20.04 and Install on your PC
- Install ROS Noetic on Remote PC:

```
 $ sudo apt update
 $ sudo apt upgrade
 $ wget https://raw.githubusercontent.com/ROBOTIS-GIT/robotis_tools/master/install_ros_noetic.sh
 $ chmod 755 ./install_ros_noetic.sh
 $ bash ./install_ros_noetic.sh
```

- Install Dependent ROS Packages:

```
 $ sudo apt-get install ros-noetic-joy ros-noetic-teleop-twist-joy \
   ros-noetic-teleop-twist-keyboard ros-noetic-laser-proc \
   ros-noetic-rgbd-launch ros-noetic-rosserial-arduino \
   ros-noetic-rosserial-python ros-noetic-rosserial-client \
   ros-noetic-rosserial-msgs ros-noetic-amcl ros-noetic-map-server \
   ros-noetic-move-base ros-noetic-urdf ros-noetic-xacro \
   ros-noetic-compressed-image-transport ros-noetic-rqt* ros-noetic-rviz \
   ros-noetic-gmapping ros-noetic-navigation ros-noetic-interactive-markers
```

- Install Turtlebot3 Packages:

```
 $ sudo apt install ros-noetic-dynamixel-sdk
 $ sudo apt install ros-noetic-turtlebot3-msgs
 $ sudo apt install ros-noetic-turtlebot3
```

- `$ sudo apt-get update`
- `$ sudo apt-get upgrade`
- Install Simulation Package:

```
$ cd ~/catkin_ws/src/
$ git clone -b noetic-devel https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
$ cd ~/catkin_ws && catkin_make
```

Launch Simulation World in Gazebo

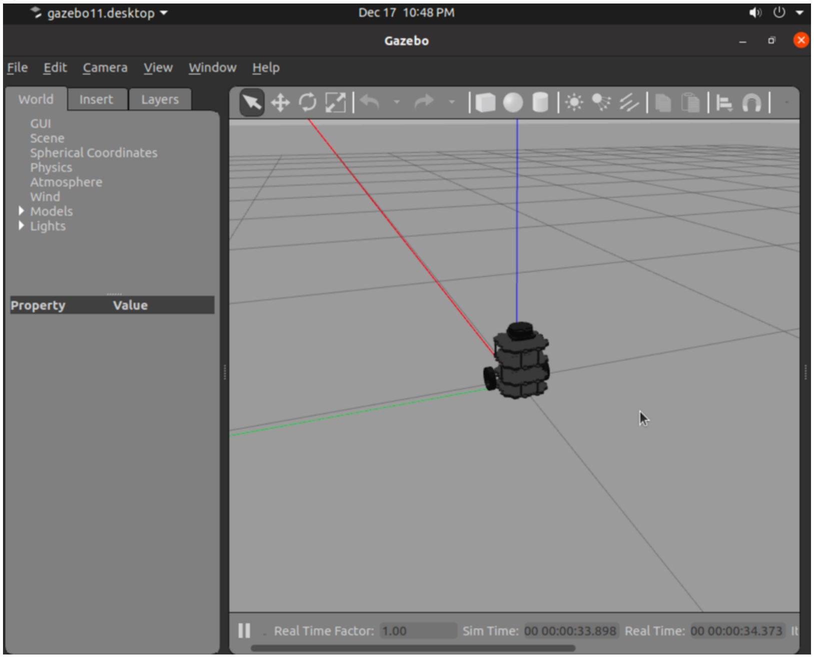
- Run **Roscore** from PC: *//run it everytime before bringup.*

```
 $ roscore
```

- **Bringup** Turtlebot3 in empty world: (New Terminal)

```
 $ export TURTLEBOT3_MODEL=${TB3_MODEL}  // use burger , waffle in place of ( ${TB3_MODEL}  )
 $ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
```

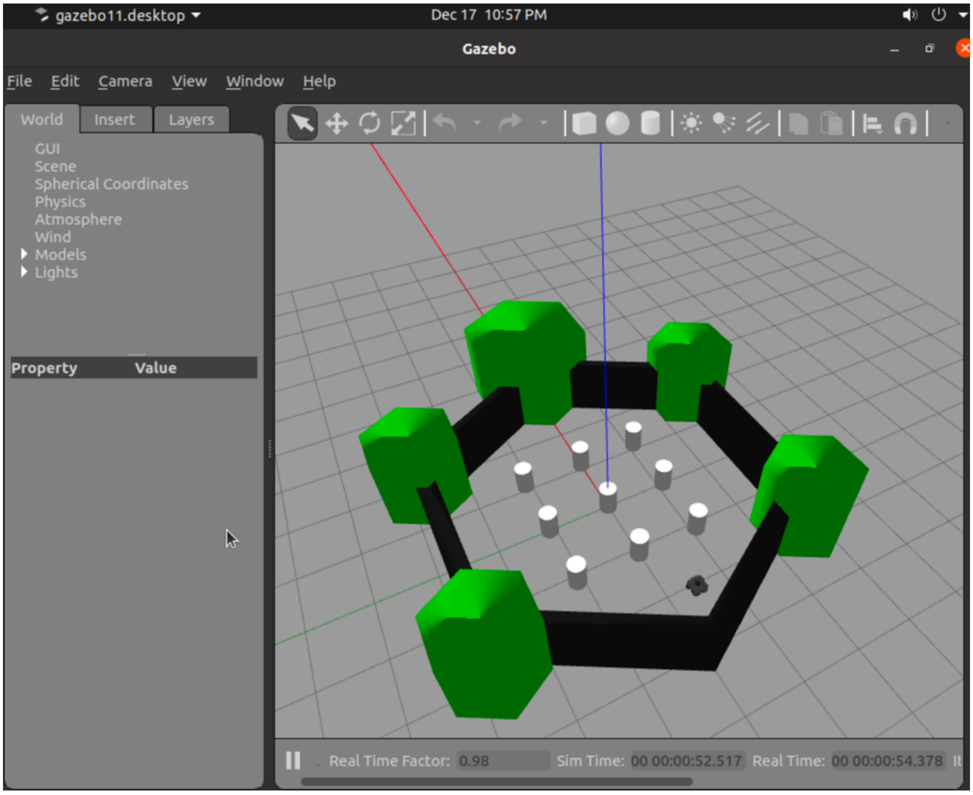
//Turtlebot3 will launch in Gazebo



Real Time Factor: 1.00 Sim Time: 00 00:00:33.898 Real Time: 00 00:00:34.373

OR

- Bringup Turtlebot3 in world: (New Terminal)
- $ export TURTLEBOT3_MODEL=waffle *// use burger , waffle in place of (${TB3_MODEL})*
- $ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
- *//Turtlebot3 will launch in Gazebo*

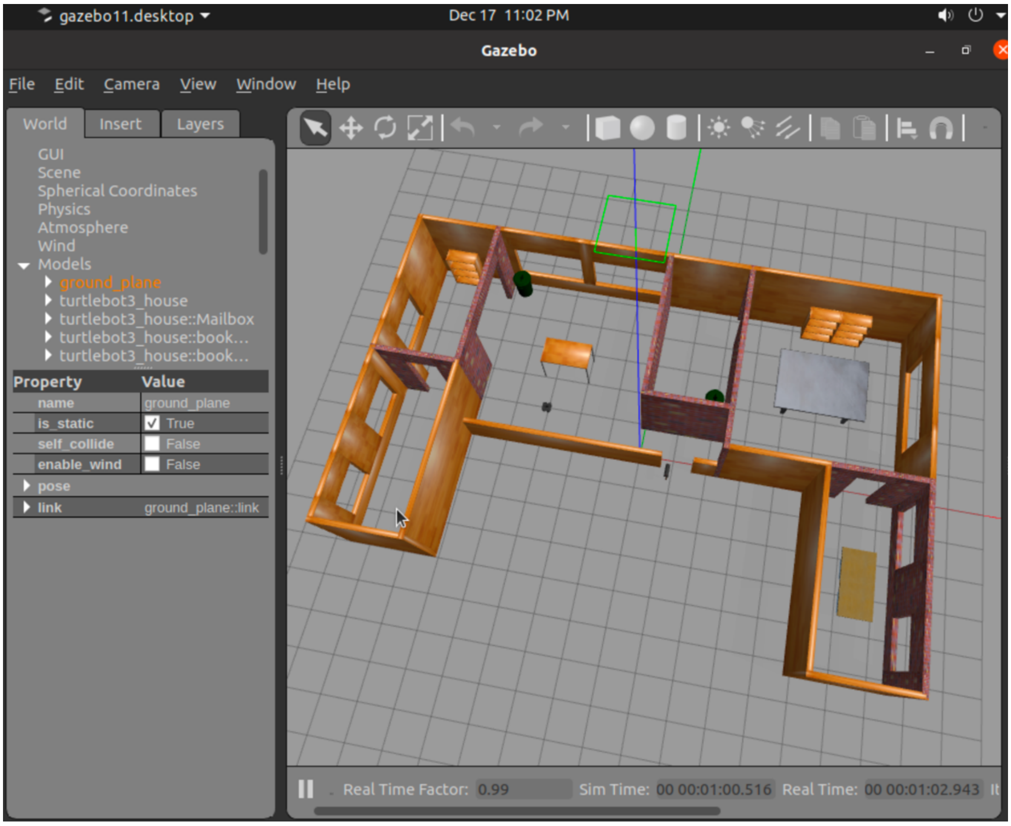


Real Time Factor: 0.98 Sim Time: 00 00:00:52.517 Real Time: 00 00:00:54.378

OR

- Bringup Turtlebot3 in House: (New Terminal)
- $ export TURTLEBOT3_MODEL=waffle_pi *// use burger,waffle in place of (${TB3_MODEL})*
- $ roslaunch turtlebot3_gazebo turtlebot3_world.launch

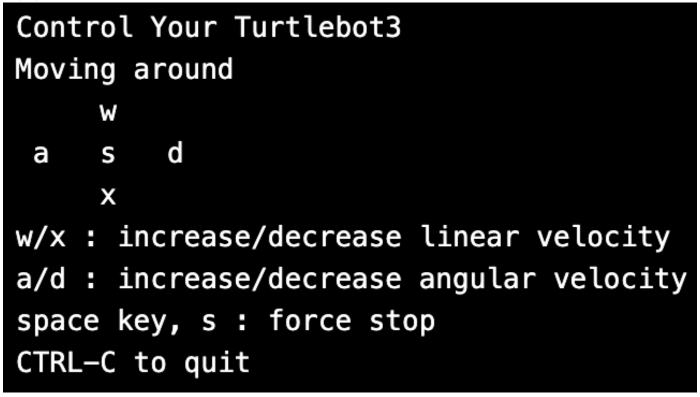
//Turtlebot3 will launch in Gazebo



Property	Value
name	ground_plane
is_static	☑ True
self_collide	□ False
enable_wind	□ False
pose	
link	ground_plane::link

Real Time Factor: 0.99 Sim Time: 00 00:01:00.516 Real Time: 00 00:01:02.943

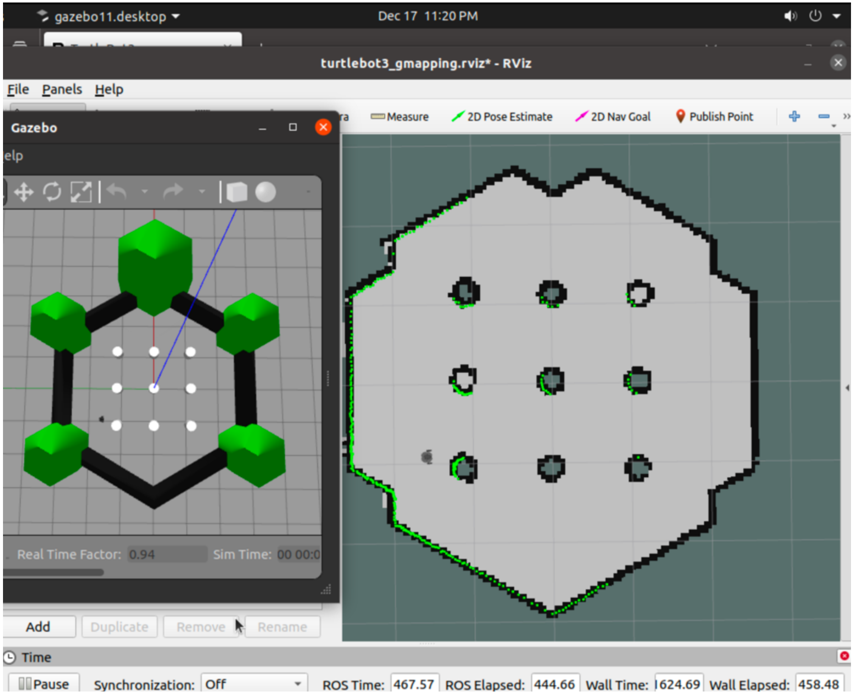
- Launch **Teleoperation** Node: (New Terminal)
  $ export TURTLEBOT3_MODEL=${TB3_MODEL} *// use burger , waffle in place of (${TB3_MODEL})*
  $ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
  //Turtlebot3 teleop key will launch and we can control it using keyboard keys while keeping the terminal open in one corner. If the node is successfully launched, the following instruction will be appeared to the terminal window.



```
Control Your Turtlebot3
Moving around
    w
 a  s  d
    x
w/x : increase/decrease linear velocity
a/d : increase/decrease angular velocity
space key, s : force stop
CTRL-C to quit
```

Launch SLAM Simulation

- Launch simulation world: (New Terminal)
  $ export TURTLEBOT3_MODEL=burger
  $ roslaunch turtlebot3_gazebo turtlebot3_world.launch
- Launch **SLAM Node**: (New Terminal)
  $ export TURTLEBOT3_MODEL=burger
  $ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=gmapping
- Launch Teleoperation Node: (New Terminal)
  $ export TURTLEBOT3_MODEL=burger
  $ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch *//Turtlebot3 SLAM will launch in RViZ. If the node is successfully launched, the following instruction will be appeared to the terminal window.Once SLAM node is successfully up and running, TurtleBot3 will be exploring unknown area of the map using teleoperation.Start exploring and drawing the map.*



Real Time Factor: 0.94 Sim Time: 00 00:0

Synchronization: Off ROS Time: 467.57 ROS Elapsed: 444.66 Wall Time: 1624.69 Wall Elapsed: 458.48

- **Save Map**:
  $ rosrun map_server map_saver -f ~/map. *// See the drawn map in the attached file.*

RUN NAVIGATION Simulation

- Launch simulation world: (New Terminal)

```
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

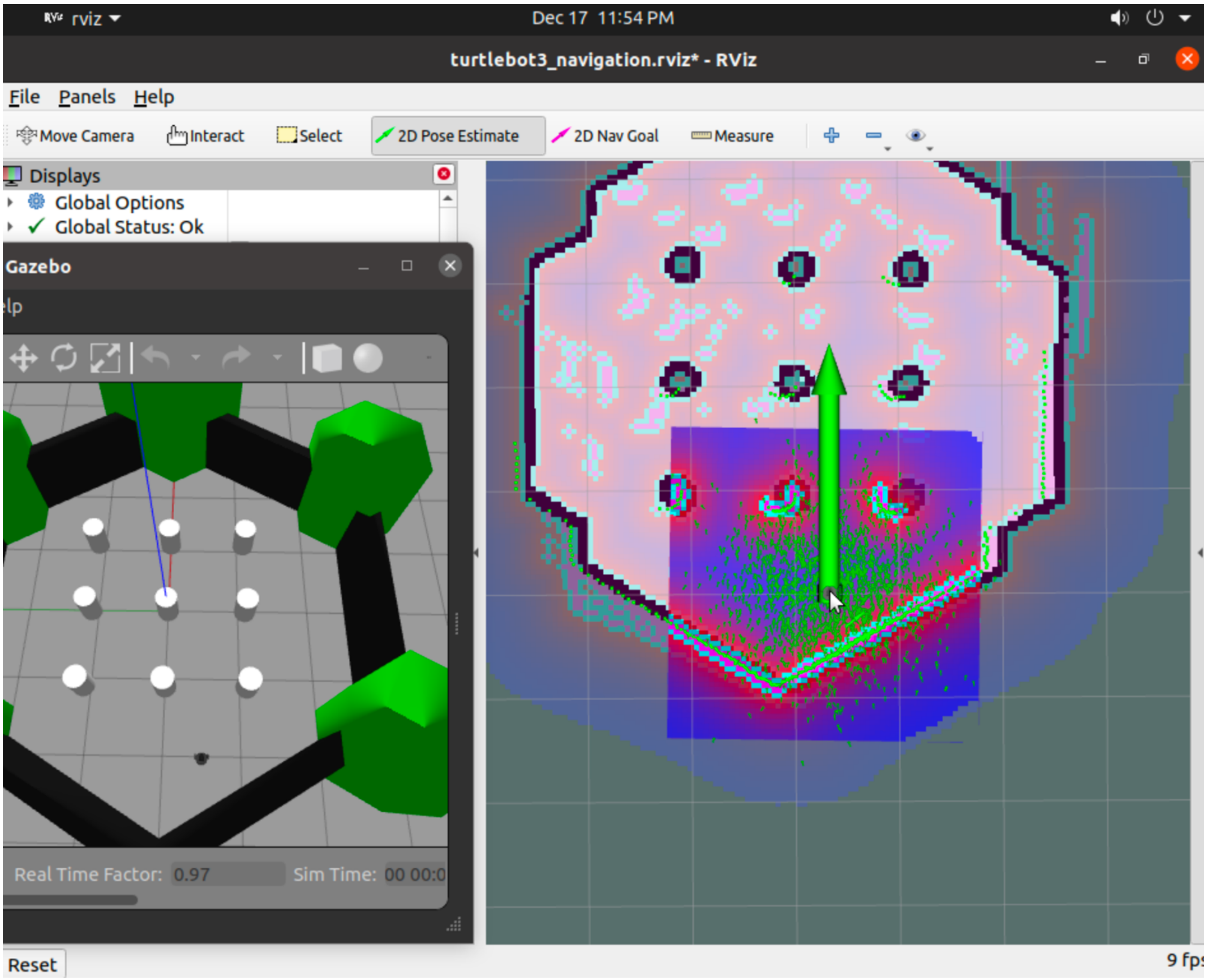
- Launch the **Navigation**:

```
$ export TURTLEBOT3_MODEL=burger
$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/map.yaml
```

// This will open the previous saved map in rviz and further instructions can be performed there.

- Estimate Initial Pose:

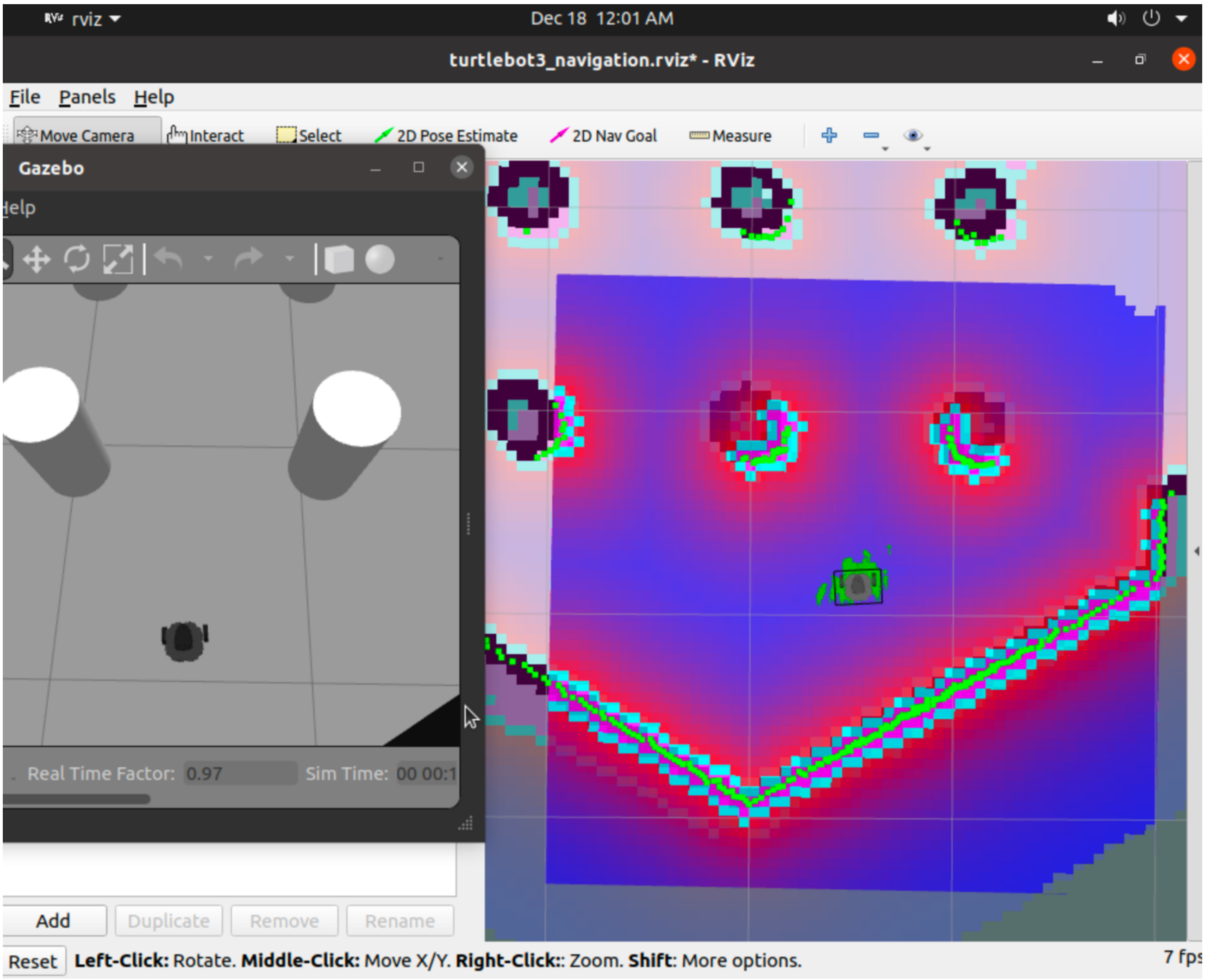
// Click the 2D Pose Estimate button in the RViz menu.
// Click on the map where the actual robot is located and drag the large green arrow toward the direction where the robot is facing.
// Repeat first two steps until the LDS sensor data is overlayed on the saved map.



// Launch keyboard teleoperation node to precisely locate the robot on the map.

```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

// Move the robot back and forth a bit to collect the surrounding environment information and narrow down the estimated location of the TurtleBot3 on the map which is displayed with tiny green arrows.



*//Terminate the keyboard teleoperation node by entering Ctrl + C to the teleop node terminal in order to prevent different **cmd_vel** values are published from multiple nodes during Navigation.*

- Set Navigation goal:
  // Click the 2D Nav Goal button in the RViz menu.
  // Click on the map to set the destination of the robot and drag the green arrow toward the direction where the robot will be facing.
  - *This green arrow is a marker that can specify the destination of the robot.*
  - *The root of the arrow is x, y coordinate of the destination, and the angle ϑ is determined by the orientation of the arrow.*
  - *As soon as x, y, ϑ are set, TurtleBot3 will start moving to the destination immediately.*

  // See the small video in the attached file.