Robotics Lab

TurtleBot3

Introduction

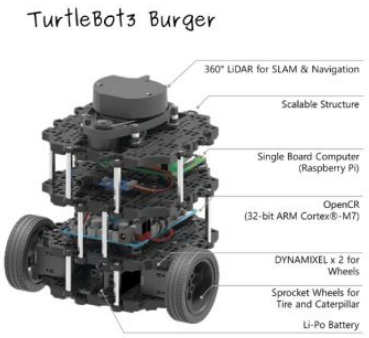
- It is a low-cost, personal robot kit with open-source software.
- It was created at Willow Garage by Melonee Wise and Tully Foote in November 2010.
- TurtleBot1 was developed by Tully (Platform Manager at Open Robotics) and Melonee (CEO of Fetch Robotics) from Willow Garage in 2010.
- In 2012, TurtleBot2 was developed by Yujin Robot based on the research robot, iClebo Kobuki.
- In 2017, TurtleBot3 was developed with features to supplement the lacking functions of its predecessors, and the demands of users.

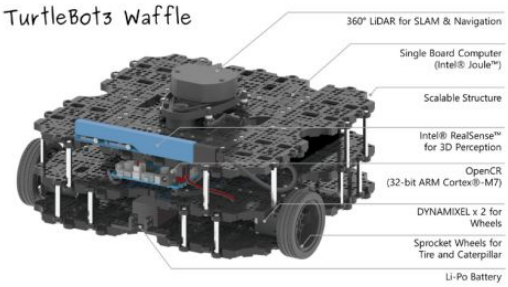
TurtleBot1 TurtleBot2 TurtleBot3

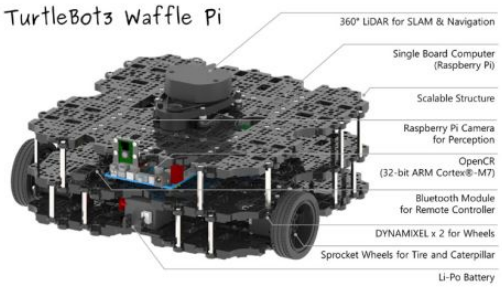
- The goal of TurtleBot3 is to dramatically reduce the size of the platform and lower the price without having to sacrifice its functionality and quality, while at the same time offering expandability.

Turtlebot3 Hardware

- Three official TurtleBot3 models: TurtleBot3 Burger, Waffle, Waffle Pi
- The basic components of TurtleBot3 are actuators, an SBC for operation ROS, a sensor for SLAM and navigation, restructurable mechanism, an OpenCR embedded board used as a sub-controller, sprocket wheels that can be used with tire and caterpillar, and a 3 cell lithium-poly battery.
- Waffle is different from Burger in terms of platform shape which can conveniently mount components, use of higher torque actuators, high-performance SBC with Intel processor, RealSense Depth Camera for 3D recognition from Intel.
- TurtleBot3 Waffle Pi is the same shape as Waffle model, but this model uses Raspberry Pi and Raspberry Pi Camera to make it more affordable.







Turtlebot3 Software

- The TurtleBot3 software consists of firmware (FW) of OpenCR board used as a sub-controller and 4 ROS packages.
- The firmware of TurtleBot3 is also called as ‘turtlebot3_core’ in the sense that it is the core of TurtleBot3
- It uses OpenCR as a sub-controller to estimate the location of the robot by calculating the encoder value of Dynamixel, which is the driving motor of TurtleBot3 or to control the velocity according to the command published by the upper-level software.
- Acceleration and angular velocity are obtained from 3-axis acceleration and 3-axis gyro sensor mounted on OpenCR to estimate the direction of the robot, and the battery state is also measured and transmitted via topics.
- TurtleBot3’s ROS package includes 4 packages which are ‘turtlebot3’, ‘turtlebot3_msgs’, ‘turtlebot3_simulations’, and ‘turtlebot3_applications’
- The ‘turtleBot3’ package contains TurtleBot3’s robot model, SLAM and navigation package, remote control package, and bring up package.
- The ‘turtlebot3_msgs’ package contains message files used in turtlebot3, ‘turtlebot3_simulations’ contains packages related to simulation, and ‘turtlebot3_applications’ package contains applications.

Turtlebot Package Installation(RemotePC)

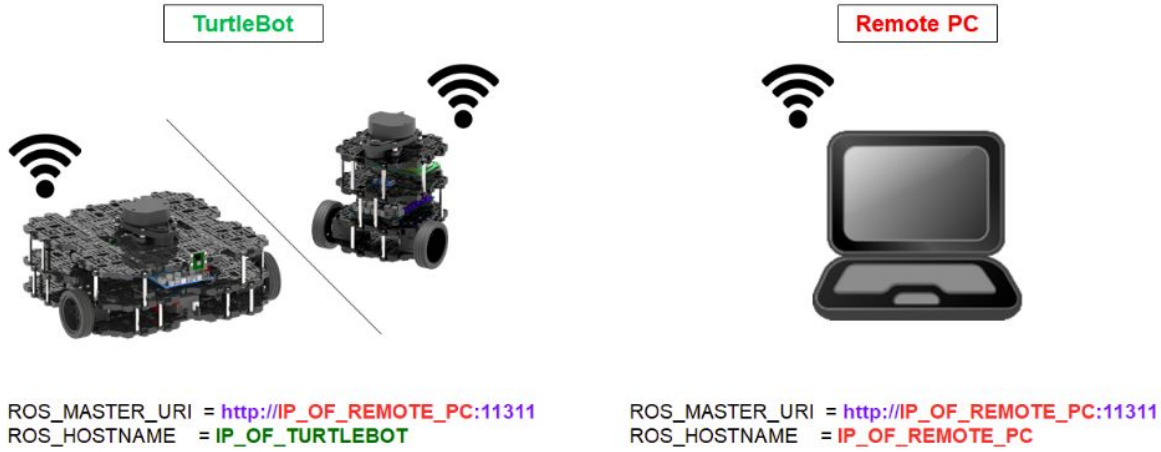
```
$ cd ~/catkin_ws/src/
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
$ cd ~/catkin_ws && catkin_make
```

Turtlebot Package Installation(TurtleBot3)

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
$ git clone https://github.com/ROBOTIS-GIT/hls_lfcd_lds_driver.git
$ cd ~/catkin_ws && catkin_make
```

Network Configuration

ROS requires IP addresses in order to communicate between TurtleBot PC and the remote PC. The remote PC and TurtleBot PC should be connected to the same wifi router.



Check the IP address of the remote PC using 'ifconfig' command in terminal

```
$ ifconfig
```

```
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 94726  bytes 62048119 (62.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 94726  bytes 62048119 (62.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlo1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.100.8.158  netmask 255.255.240.0  broadcast 10.100.15.255
        inet6 fe80::e871:20c7:fcdf:4130  prefixlen 64  scopeid 0x20<link>
        ether dc:f5:05:fd:25:bb  txqueuelen 1000  (Ethernet)
        RX packets 8296736  bytes 5899114052 (5.8 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1257030  bytes 305445703 (305.4 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Configuration in Remote PC

Note the IP address of remote PC and in terminal open

```
$ gedit ~/.bashrc
```

Update the IP address and configuration on that file as

```
...
source /opt/ros/melodic/setup.bash
source ~/catkin_ws/devel/setup.bash

export ROS_MASTER_URI=http://<REMOTE_IP_ADD>:11311
export ROS_HOSTNAME=<REMOTE_IP_ADD>
export TURTLEBOT3_MODEL=burger
```

After this source the bashrc with the following command

```
$ source ~/.bashrc
```

Configuration in TurtleBot3 PC

Get the IP address of Turtlebot also using ifconfig in the terminal of the turtlebot computer.
Than modify the .bashrc file in turtlebot computer to update ROS_MASTER_URI which will take the IP address of Remote PC and ROS_HOSTNAME which should hold IP address of turtlebot

```
$ gedit ~/.bashrc
```

```
...

export ROS_MASTER_URI=http://<REMOTE_IP_ADD>:11311
export ROS_HOSTNAME=<TURTLEBOT_IP_ADD>
```

After this source the bashrc with the following command

```
$ source ~/.bashrc
```

Note: Time sync between TurtleBot3 and Remote PC
Install ntpdate and synchronize to NTP server on both Turtlebot and Remote PC

```
$ sudo apt-get install ntpdate
$ sudo ntpdate ntp.ubuntu.com
```

Bringup TurtleBot

1. Run roscore
   [Remote PC] Run roscore

   ```
   $ roscore
   ```

2. Bringup TurtleBot3
   [Turtlebot] Bring Up basic packages to start TurtleBot3 applications

   ```
   $ roslaunch turtlebot3_bringup turtlebot3_robot.launch
   ```
   <image from terminal>

Note: If you want to launch Lidar sensor, Raspberry Pi Camera, Intel® RealSense™ R200 or core separately, please use below commands.

```
$ roslaunch turtlebot3_bringup turtlebot3_lidar.launch
$ roslaunch turtlebot3_bringup turtlebot3_rpicamera.launch
$ roslaunch turtlebot3_bringup turtlebot3_realsense.launch
$ roslaunch turtlebot3_bringup turtlebot3_core.launch
```

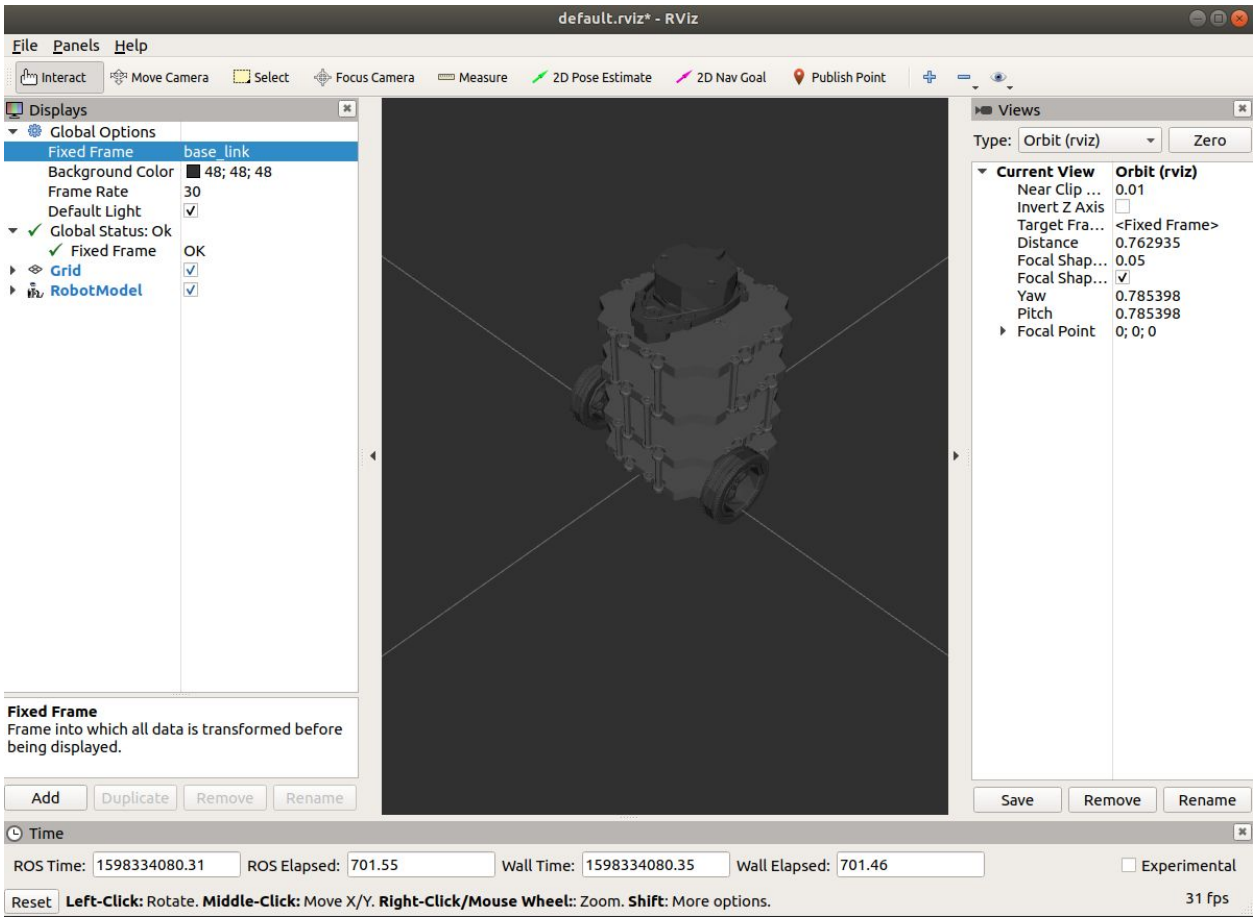
3. Load a TurtleBot3 on Rviz
   [Remote PC] Run the command below

   ```
   $ roslaunch turtlebot3_bringup turtlebot3_remote.launch
   ```

```
niraj@niraj-VivoBook:~/installation/ros/catkin_ws$ roslaunch turtlebot3_bringup turtlebot3_remote.launch
... logging to /home/niraj/.ros/log/3bb453c4-e693-11ea-a369-dcf505fd25bb/roslaunch-niraj-VivoBook-16067.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

xacro: in-order processing became default in ROS Melodic. You can drop the option.
started roslaunch server http://172.16.1.188:40205/

SUMMARY
========

PARAMETERS
 * /robot_description: <?xml version="1....
 * /robot_state_publisher/publish_frequency: 50.0
 * /robot_state_publisher/tf_prefix:
 * /rosdistro: melodic
 * /rosversion: 1.14.7

NODES
  /
    robot_state_publisher (robot_state_publisher/robot_state_publisher)

ROS_MASTER_URI=http://172.16.1.188:11311

process[robot_state_publisher-1]: started with pid [16085]
```

In new terminal window enter the following command

```
$ rosrun rviz rviz -d `rospack find turtlebot3_description`
/rviz/model.rviz
```



Teleoperation

WARNING: Make sure to run the Bringup instruction before performing these examples, and be careful when testing the robot on the table as it might fall.

After bringup use the following command in the new tab

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

```
niraj@niraj-VivoBook:~/installation/ros/catkin_ws$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
... logging to /home/niraj/.ros/log/3bb453c4-e693-11ea-a369-dcf505fd25bb/roslaunch-niraj-VivoBook-16415.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://172.16.1.188:32791/

SUMMARY
========

PARAMETERS
 * /model: burger
 * /rosdistro: melodic
 * /rosversion: 1.14.7

NODES
  /
    turtlebot3_teleop_keyboard (turtlebot3_teleop/turtlebot3_teleop_key)

ROS_MASTER_URI=http://172.16.1.188:11311

process[turtlebot3_teleop_keyboard-1]: started with pid [16431]

Control Your TurtleBot3!
---------------------------
Moving around:
        w
   a    s    d
        x

w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi : ~ 0.26)
a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi : ~ 1.82)

space key, s : force stop

CTRL-C to quit
```

You can now use the keyboard to move the turtlebot but make sure the terminal which runs the teleop operation should be selected.

SLAM

- Stands for Simultaneous localization and mapping
- It is a technique to draw a map by estimating current location in an arbitrary space

1. Run SLAM Nodes
   [Remote PC] Run roscore

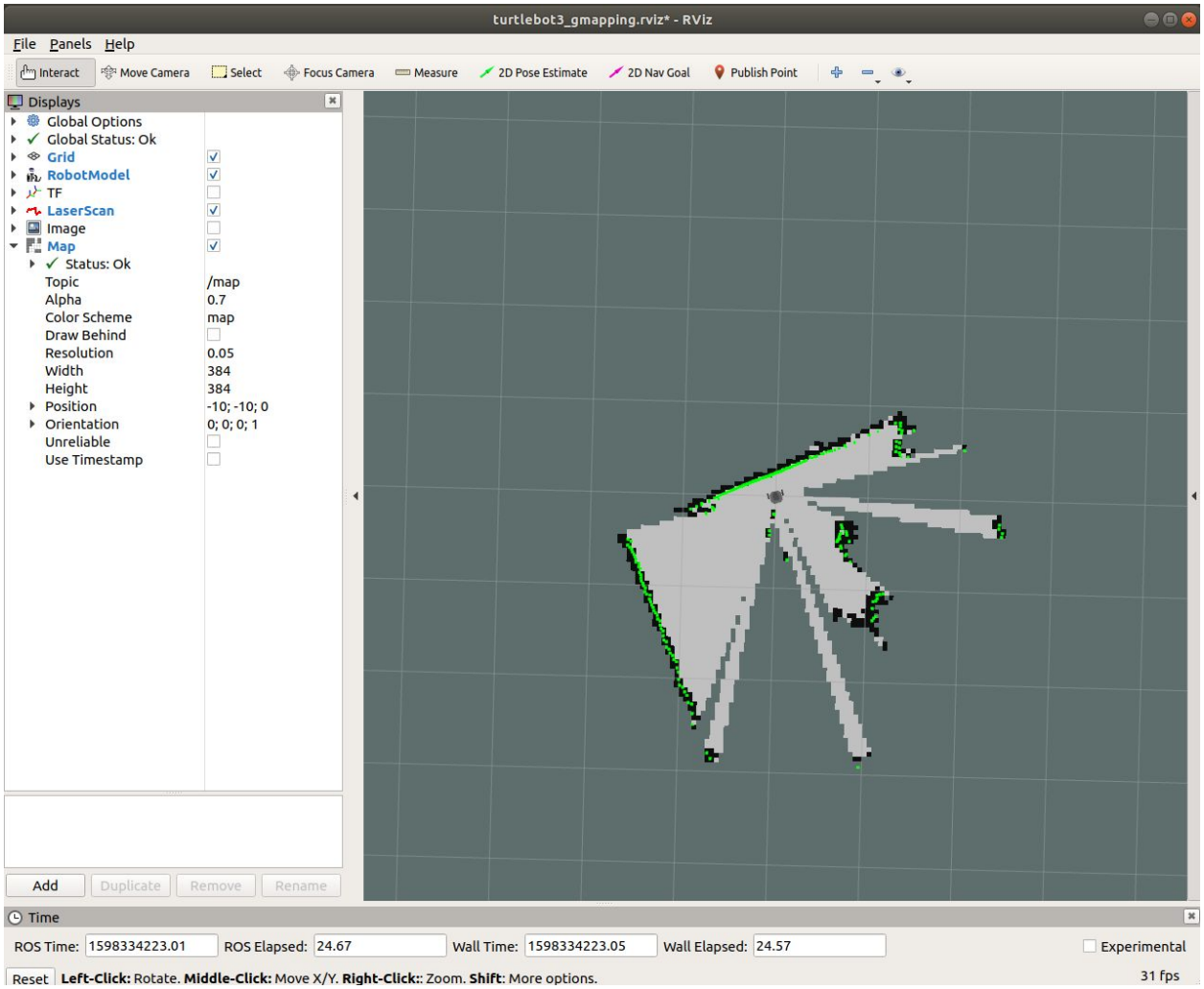
   ```
   $ roscore
   ```

   [TurtleBot] Bring up basic packages to start TurtleBot3 applications

```
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

[Remote PC] Open a new terminal and launch the SLAM file

```
$ roslaunch turtlebot3_slam turtlebot3_slam.launch
slam_methods:=gmapping
```



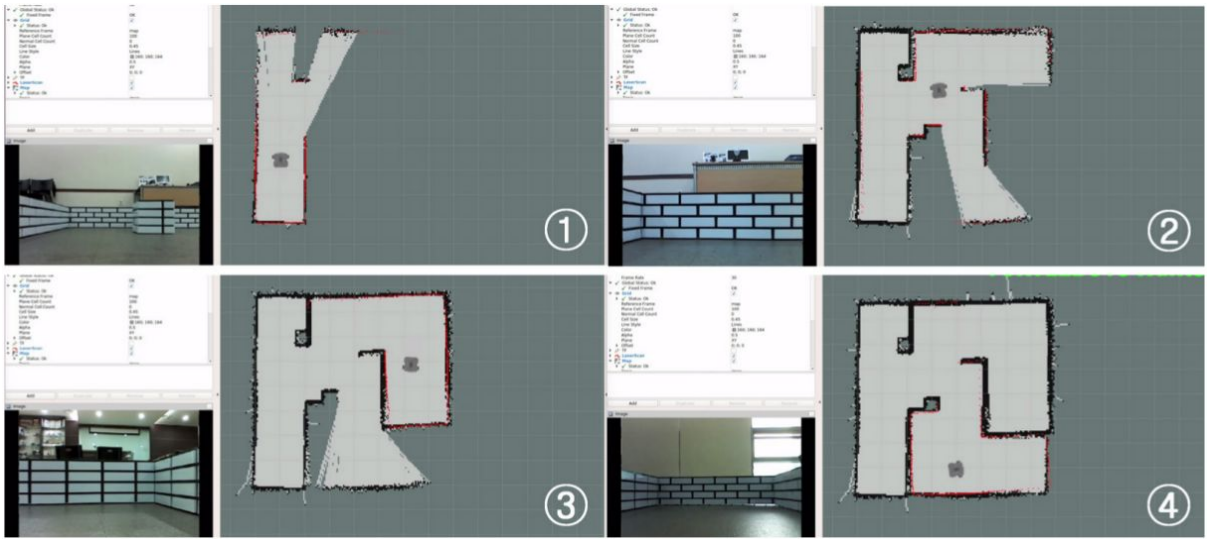
Note:

- TurtleBot3 supports Gmapping, Cartographer, Hector, and Karto among various SLAM methods. You can do this by changing the slam_methods:=xxxxx option.

- The slam_methods options include gmapping, cartographer, hector, karto, frontier_exploration, and you can choose one of them.
- For example, to use Karto, you can use the following:
  `$ roslaunch turtlebot3_slam turtlebot3_slam.launch slam_methods:=karto`

2. Run Teleoperation Node
   [Remote PC] Open a new terminal and run the teleoperation node
   - This command allows the user to control the robot to perform SLAM operation manually.
   - It is important to avoid vigorous movements such as changing the speed too quickly or rotating too fast.
   - When building a map using the robot, the robot should scan every corner of the environment to be measured.

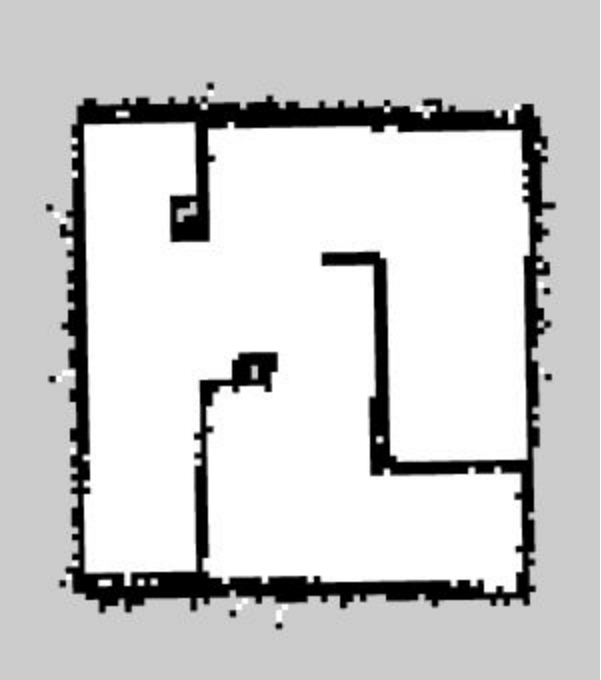
```
$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```



3. Save Map
   [Remote PC] After we see the map on the screen, let's save the map. We have to run the map_saver node to create a map file. The map is drawn based on the robot's odometry, tf information, and scan information of the sensor when the robot moves.

```
$ rosrun map_server map_saver -f ~/map
```

4. Map
    - After running the command in the previous step we can see two files map.pgm and map.yaml is saved in the defined directory.
    - The white section in the map is the free area in which the robot can move, black is the occupied area in which the robot cannot move and gray is the unknown area.



Navigation

- **Navigation** is to move the robot from one location to the specified destination in a given environment.
- For this purpose, a map that contains geometry information of furniture, objects, and walls of the given environment is required.
- The navigation enables a robot to move from the current pose to the designated goal pose on the map by using the map, robot's encoder, IMU sensor, and distance sensor.

1. Run Navigation Nodes
   [Remote PC] Run roscore

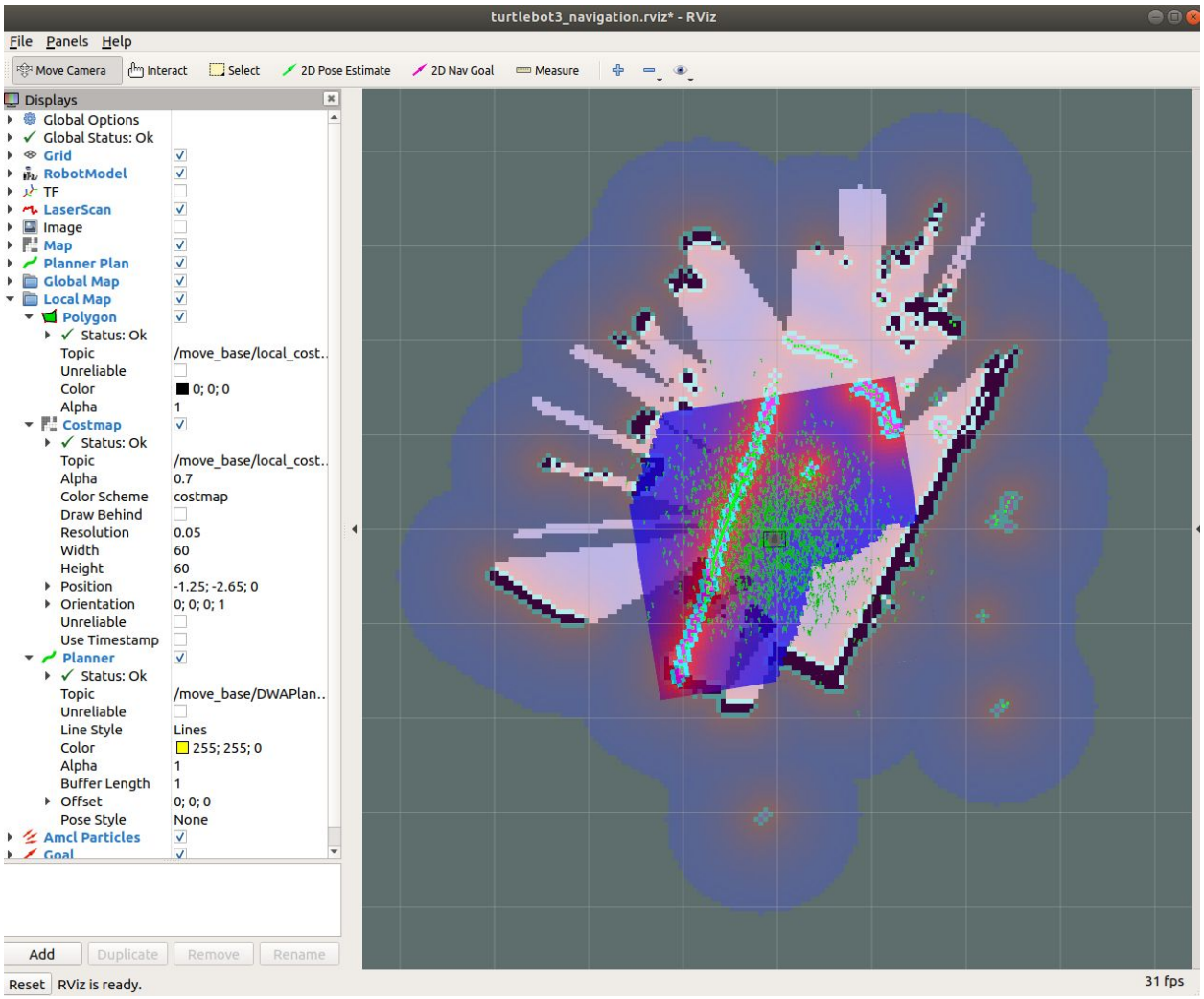
```
$ roscore
```

[TurtleBot] Bring up basic packages to start TurtleBot3 applications

```
$ roslaunch turtlebot3_bringup turtlebot3_robot.launch
```

[Remote PC] Launch the navigation file

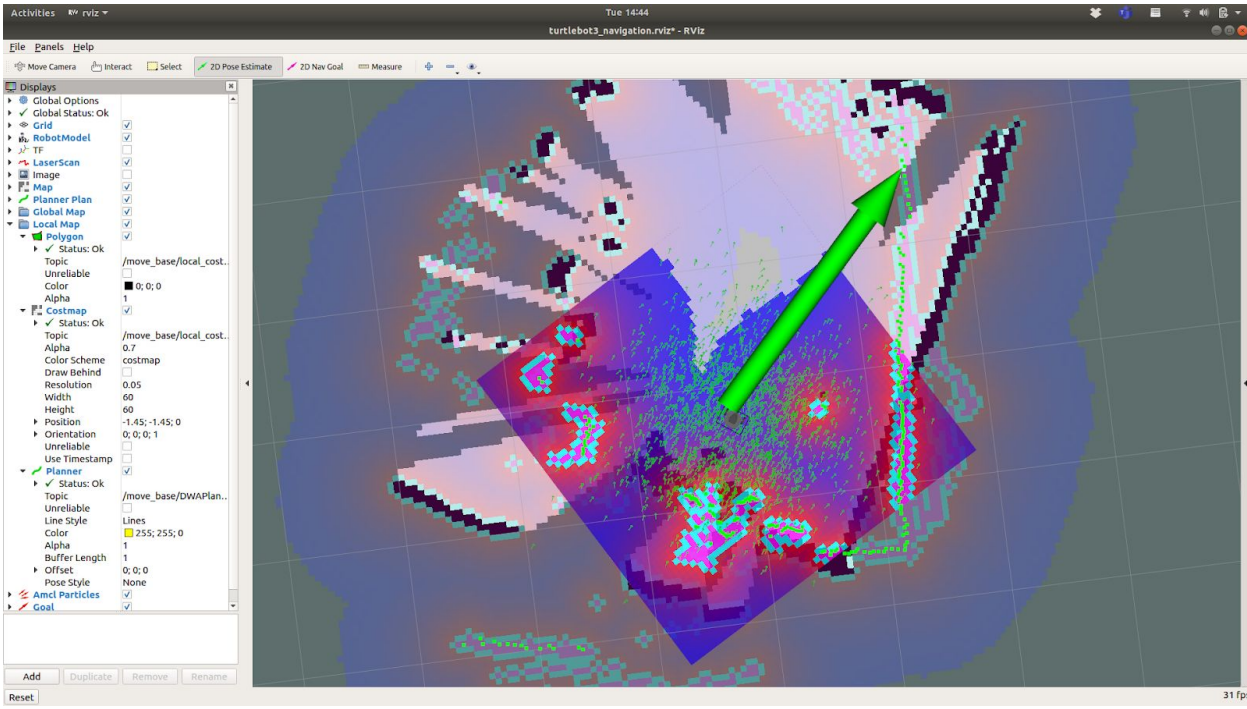
```
$ roslaunch turtlebot3_navigation
turtlebot3_navigation.launch map_file:=$HOME/map.yaml
```



2. Estimate Initial Pose
   [Remote PC]

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

- First, the initial pose estimation of the robot should be performed.
- When you press **2D Pose Estimate** in the menu of RViz, a very large green arrow appears. Move it to the pose where the actual robot is located in the given map, and while holding down the left mouse button, drag the green arrow to the direction where the robot's front is facing
- Then move the robot back and forth with tools like turtlebot3_teleop_keyboard node to collect the surrounding environment information and find out where the robot is currently located on the map



Note: the turtlebot3_teleop_keyboard node used for Estimate Initial Pose should be terminated after use, otherwise the robot will behave strangely because the topic overlaps with /cmd_vel topic from the navigation node

3. Send Navigation Goal
   [Remote PC]
   - In RViz, Click the 2D Nav Goal button.

- Click on a specific point in the map to set a goal position and drag the cursor to the direction where TurtleBot should be facing at the end.
- The robot will create a path to avoid obstacles to its destination based on the map. Then, the robot moves along the path. At this time, even if an obstacle is suddenly detected, the robot moves to the target point avoiding the obstacle.