

Уроци по теле-контрол, картографиране и навигация на робот в ROS

Настоящите уроци са предназначени за ученици в средното образование и за студенти. Те покриват материал за работа с Операционната система за роботи ROS.

Всички съвременни електронни и мехатронни устройства се управляват от компютърни програми – софтуер. Роботите са комплексни машини, които обикновено изискват по-специални и сложни софтуерни решения. Поради това е създадена операционната система за роботи ROS. Нейната цел е да предостави готови софтуерни библиотеки и решения за лесно програмиране на роботите.

• Какво е ROS?

Операционната система за роботи ROS е предназначена за контрол на роботи, задвижващи устройства, машини и други. Основните компоненти в ROS са пакети, възли, топици, сървизи и съобщения. ROS е мета-операционна система и се инсталира върху други операционни системи като Linux (препоръчително) или Windows.

Софтуерът в ROS е организиран в пакети. Пакетът може да съдържа ROS възли, независима от ROS библиотека, набор от данни, конфигурационни файлове, софтуер на трета страна или нещо друго, което логически представлява полезен модул. Целта на тези пакети е да осигури тази полезна функционалност по лесен за използване начин, така че софтуерът да може лесно да се използва повторно. Като цяло ROS пакетите следват принцип „Goldilocks“: достатъчно функционалност, за да бъде полезна, но не прекалено, че пакетът да е тежък и труден за използване от друг софтуер.

Възелът е процес, който извършва изчисления (програма, алгоритъм). Възлите са обединени заедно в граф и комуникират помежду си, като използват топици за стрийминг, RPC услуги (сървизи) и сървър за параметри. Тези възли са предназначени да работят в отлично-организиран мащаб; система за управление на един робот, обикновено включва много възли. Например, един възел контролира лазерен далекомер, един възел контролира колесните двигатели на робота, един възел извършва локализация, един възел извършва планиране на пътя, един възел осигурява графичен изглед на системата и т. н.

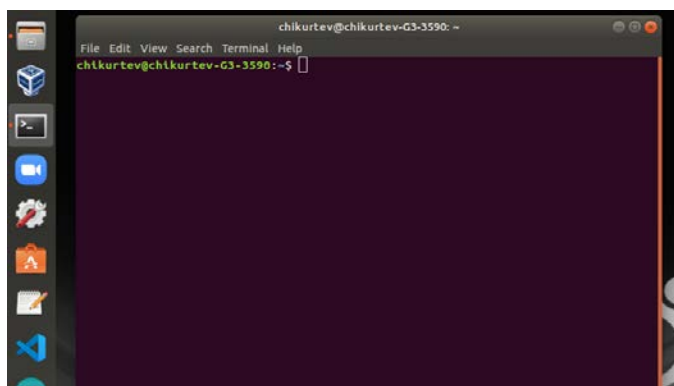
ROS сървизите са предназначени да предоставят двупосочна комуникация между възли, базирана на принципа заявка/отговор. Те се използват за автоматизиране на някои процеси по време на управление на роботите.

Възлите комуникират помежду си чрез публикуване на съобщения по канали (топици). Съобщението е проста структура от данни, съдържаща въведени полета. Поддържат се стандартни примитивни типове (цяло число, плаваща запетая, булева стойност и т.н.), както и масиви от примитивни типове. Съобщенията могат да включват произволно вложени структури и масиви (подобно на C структури).

Възлите също могат да обменят съобщение за заявка и отговор като част от повикване на услуга за ROS. Тези съобщения за заявка и отговор са дефинирани във `sgv` файлове.

- **Убунту терминала и ROS команди.**

Понеже Убунту Линукс е препоръчваната операционна система за работа с ROS, в тези уроци използваме нея. За да работим с ROS трябва да използваме Линукс терминала показан на фигура 1. Терминалът в Линукс е основно средство за достъп, обработване на файлове, работа с устройства, инсталиране/деинсталиране на програми, стартиране на програми и други. В терминала чрез специални команди можем да стартираме програми в ROS, да проверяваме активните възли, топици, сървизи и да визуализираме съобщения с данни от различните топици.



Фигура 1. Терминален прозорец в Линукс.

Някои от най-използваните команди за работа с ROS са:

roslaunch package_name node_name – стартира се програма

roslaunch package_name launch_file_name – стартира се launch файл

roslaunch list – извежда списък на всички стартирани програми

roslaunch info /node_name – показва информация за даден възел

rostopic list - извежда списък на всички активни топици

rostopic info /topic_name – показва информация за даден топик

rostopic echo /topic_name – визуализира данните предавани в даден топик

rostopic pub /topic_name message_type „message data“ - публикува съобщение в даден топик

rosservice list – извежда списък на наличните сървизи

rosservice call /service_name param – извиква сървиз с даден параметър (някои сървизи се извикват без параметри)

Ако искаме да стартираме програма от даден пакет в ROS, като пример може да използваме следната команда:

roslaunch turtlesim turtlesim_node

Забелязваме, че името на пакета, който използваме е **turtlesim**, а програмата/възела, която стартираме е **turtlesim_node**. По този начин можем да стартираме всяка една програма намираща се, в който и да е пакет. Този метод обаче стартира само една програма и не е подходящ при случай, когато се налага едновременно да се работи с много програми.

РОС предлага метод за стартиране на много програми, чрез използване на файлове от тип **launch**. За да стартираме файлове от типа **launch**, трябва да използваме командата **roslaunch**. Пример за използване на тази команда е следният:

roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch

И тук както при стартиране на програма първо се дава името на пакета, в който се намира файла, а след това са дава името на самия файл, заедно с разширението **‘.launch’**. Този тип файлове са често използвани, понеже дават възможност за едновременно стартиране на множество програми и задаване на специфични параметри за всяка програма.

Освен командите за стартиране на програми, много полезни са и командите за предоставяне на информация. Тези команди помагат на разработчика да се ориентира в общата картина и да провери кои програми са стартирани, кои топици са активни и допълнителна информация за тях. Например командата **rostopic list**, показва всички стартирани топици. А командата **rostopic info /cmd_vel** ще покаже цялата информация за топика като: програми, които го публикуват, програми, които са абонирани за топика, типа на съобщенията, който се публикува и други. По този начин може да се извлича информация за всеки активен топик и да се проследяват неговите връзки. Това важи и за възлите и услугите.

Третият тип команди служат за директно взаимодействие с активните програми. Те дават възможност за показване на данните от топиците или за публикуване на данни по топик, както и за извикване на услуги. Пример за публикуване на данни по топик е следния: **rostopic pub -r 10 /cmd_vel geometry_msgs/Twist '{linear: {x: 0.1, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}'**

В този пример публикуваме съобщение в топика **cmd_vel**, типа на съобщението е **geometry_msgs/Twist** и параметрите на съобщението са два вектора **linear** и **angular**. Всеки вектор съдържа данни за движение на робота. Друг важен параметър при публикуване на данни в топик е **-r 10**, този параметър активира режим на непрекъснато публикуване с дадена честота (числото задава честотата). Ако не използваме параметъра **-r**, тогава режима на публикуване е блокиращ (режим по подразбиране). При този режим съобщението се публикува еднократно и след това командата се прекратява.

Вече се запознахме с основните компоненти за работа с РОС и можем да преминем към работа с роботи и някои от популярните РОС пакети.

- **РОС инструменти и приложения**

Като част от РОС често се налага да се използват някои специализирани инструменти и приложения. Най-често използваните такива са:

- Приложение за симулиране на роботи **Gazebo**.

Това приложение е специализирано в симулирането на роботи и създаване на симулирани среди. Чрез него можем да тестваме и експериментираме с виртуални модели на роботи, които искаме да управляваме. Също така можем да създадем собствен виртуален робот.

- Инструмент за визуализиране на данни в реално време **Rviz**.

Rviz е много полезен инструмент, който визуализира данните от всички топици в РОС. Чрез този инструмент можем да наблюдаваме какви данни се получават от различни сензори, да ги сравняваме и анализираме. Тук се визуализира и модела на робота, който използваме, направената карта ако има такава, измината траектория и други.

- Инструмент за визуализиране на РОС възли, топици и сървизи и връзките помежду им **rqt_graph**.

С помощта на този инструмент можем лесно да се ориентираме в текущото състояние на всички активни възли, топици и други. Инструментът показва взаимовръзките между възлите, и дава ясна представа за принципа на работа на една система за управление. Чрез този инструмент може да се откриват нередности, да се подобряват алгоритми и да се проследява работата на една система.

- **Теле-контрол на мобилен робот – Контрол на симулиран мобилен робот чрез клавиатура или джойстик.**

Теле-контролът е метод за управление на робот, директно от оператор. Обикновено този тип управление се използва за първоначални тестове на даден робот, за обучение и запаметяване на движения и за картографиране. Процесът на теле-контрол се изразява в директно подаване на команди към робота. Например изпращаме команда към мобилен робот да се движи напред със определена скорост и това продължава докато не подадем команда за стоп. По този начин могат да се управляват симулирани роботи в симулационна среда.

В РОС е прието за задвижване на мобилен робот да се използва стандартен топик наречен **cmd_vel**. За да се реализира управлението на мобилен робот, трябва да имаме програма, която да публикува данни в този топик. Има различни начина за теле-управление на робот: чрез команди от клавиатурата и чрез команди от джойстик.

Управлението чрез клавиатура се реализира чрез програмата **teleop_twist_keyboard**. Тази програма прочита бутоните на клавиатурата и според предварително зададени параметри изпраща данни в топика **cmd_vel**.

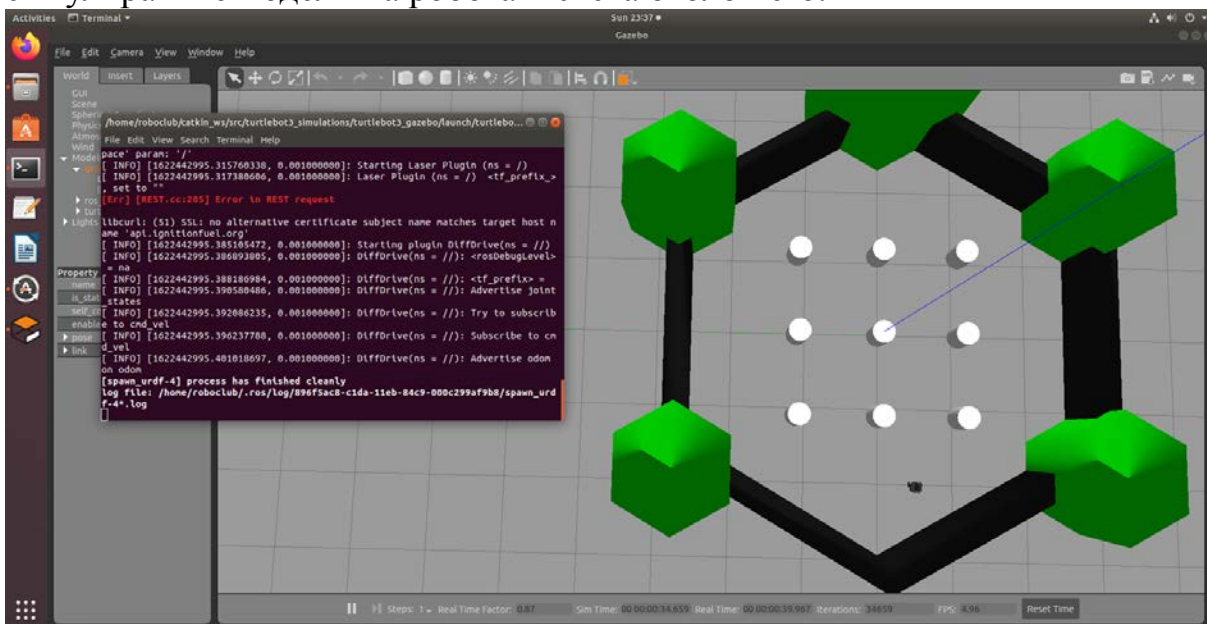
По същия начин работи и програмата за управление чрез джойстик. Тя се нарича **teleop_twist_joy**. Тази програма прочита данните от бутоните на джойстик и ги преобразува в данни за управление на робот.

В този урок ще използваме симулацията на робота turtlebot3, по подобен начин могат да се използват и симулациите на роботите husky и turtlebot2.

Първоначално трябва се стартира симулацията на робота в симулационна среда. Това става като се изпълнят следните команди в терминала:

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

След изпълнението на дадената команда се зарежда симулирания свят на робота, показан на фигура 2. Това, което се вижда на фигурата е приложението Gazebo и симулираните модели на робота и света около него.

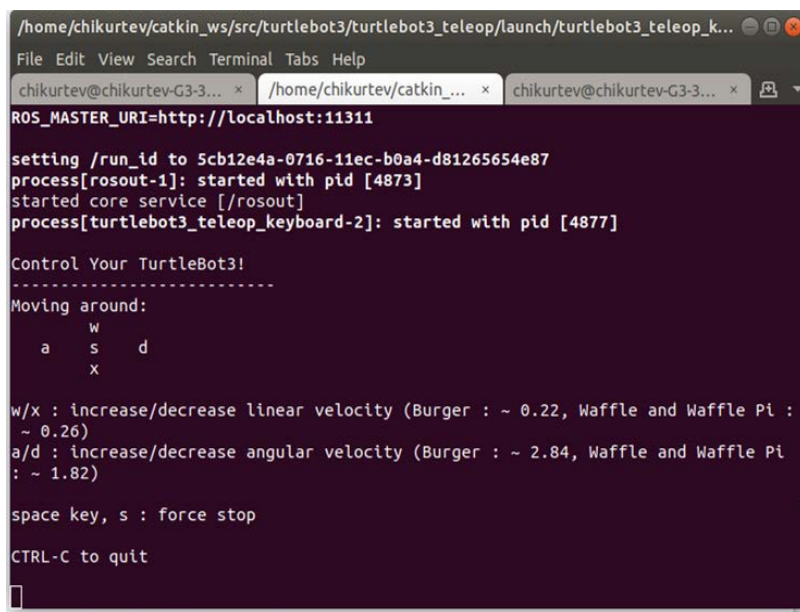


Фигура 2. Gazebo и симулиран модел на робот.

Когато имаме стартиран робот вече можем да стартираме някоя от програмите за теле-управление. В този урок ще разгледаме теле-управлението с клавиатура, понеже то не изисква наличието на джойстик. За да стартираме програмата за управление с клавиатура в отделен терминал трябва да изпълним следната команда:

```
export TURTLEBOT3_MODEL=burger  
roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

Когато командата се изпълни, в терминала се показват инструкции за работа с клавишите на клавиатурата (виж фигура 3). Както се вижда на фигурата използваните клавиши са: w, a, s, d, x.



```
/home/chikurtev/catkin_ws/src/turtlebot3/turtlebot3_teleop/launch/turtlebot3_teleop_k...
File Edit View Search Terminal Tabs Help
chikurtev@chikurtev-G3-3... x /home/chikurtev/catkin_... x chikurtev@chikurtev-G3-3... x
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 5cb12e4a-0716-11ec-b0a4-d81265654e87
process[rosout-1]: started with pid [4873]
started core service [/rosout]
process[turtlebot3_teleop_keyboard-2]: started with pid [4877]

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
```

Фигура 3. Терминал за теле-управление на Turtlebot3.

От поясненията в терминала става ясно, че едно натискане на клавиши „w x, a, d” увеличава или намалява скоростта на движение на робота. Така след като започнем да натискаме посочените клавиши, програмата изпраща команди към робота и той започва да се движи. Така вече можем да управляваме робота и да го закараме до където пожелаем. Движението на робота можем да наблюдаваме в симулатора Gazebo.

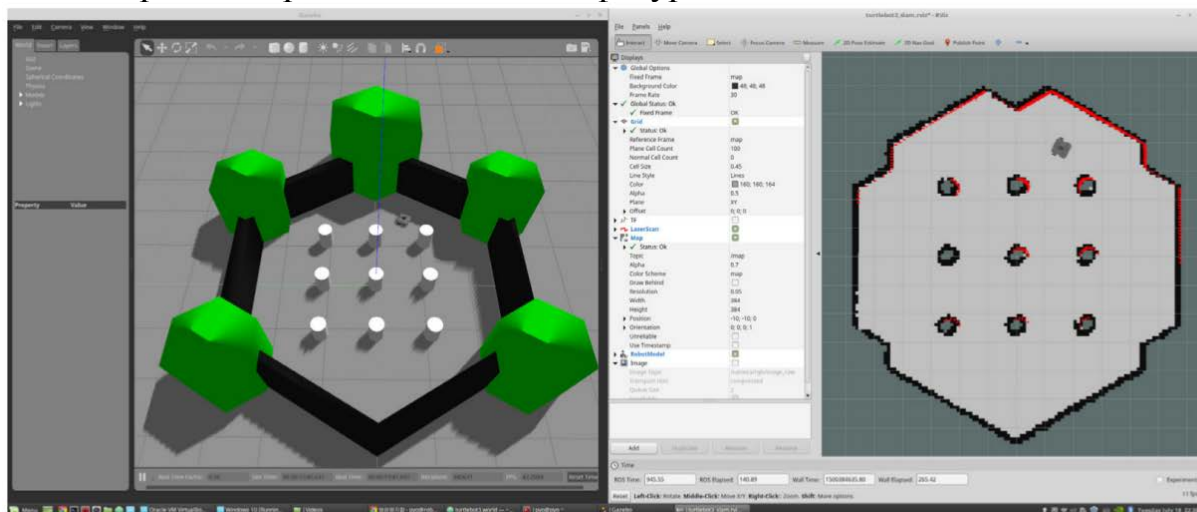
- **Картографиране на помещение чрез теле-контрол на симулирания робот в симулационна среда.**

Както споменахме по-горе теле-контролът може да се използва за картографиране. Картографирането е метод на създаване на карта на дадено помещение или среда, чрез използване на различни софтуерни инструменти и алгоритми. При мобилните работи картографирането служи за основа на Навигационната и локализационната системи. Чрез обикаляне в пространството и използване на сензори за измерване на разстояние (лазерни, инфрачервени, ултразвукови и други), роботът може автоматично да създава карта.

В този урок ще се запознаем с процеса на създаване на карта като използваме пакети на ROS за локализиране и картографиране. Както в по-горния урок ще използваме симулацията на робота turtlebot3. След като сме изпълнили командите и стъпките от предишния урок (заредена е симулацията на turtlebot3 и е стартирана програмата за теле-контрол), сега трябва да се стартира в нов терминал следната команда:

```
export TURTLEBOT3_MODEL=burger
roslaunch turtlebot3_slam turtlebot3_slam.launch
slam_methods:=gmapping
```

Тази команда отваря прозорец на инструмента Rviz, където се визуализират данните от всички топици, включително и създадената карта. Сега трябва чрез метода за теле-управление да обходим цялото пространство около робота, докато се оформи пълна карта с добри детайли като на фигура 4.



Фигура 4. Gazebo и Rviz, направена карта.

Когато имаме добре направена карта, остава последната стъпка от процеса на картографиране – запаметяване на картата. За да изпълним тази стъпка трябва да отворим нов терминал и да изпълним следната команда:

```
roslaunch map_server map_saver -f ~/map
```

Тази команда запаметява данните за направената карта в два файла в “НОМЕ” директорията на операционната система. Можем лесно да задаваме локация, къде да се запазят тези файлове, както и имената на файловете. Това става като след знака ‘~’ въведем желаната локация/директория, а като име на картата се взимат думите след последната наклонена черта.

Така след като вече имаме направена карта, можем да я използваме за да управляваме робота чрез задаване на желана дестинация и използване на навигационния пакет на ROS.

- **Контрол на симулирания робот за да изпълнява Автономна навигация по позната карта до зададена от потребител дестинация.**

Автономната навигация е процес на самостоятелно движение на мобилен робот от едно място до друго. По време на движение системата разпознава наличието на препятствия, изчислява моментната локация на робота и определя най-краткия път до зададената цел. В този урок използваме пакета на ROS за автономна навигация.

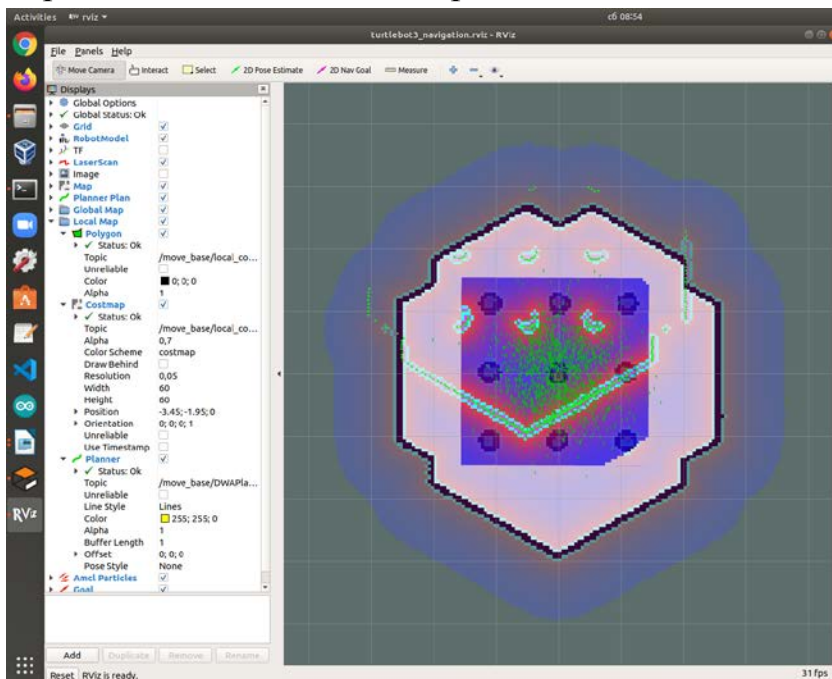
Сега също, както в урока за картографиране трябва да сме стартирали симулацията на робота turtlebot3, като не е необходимо да стартираме програмата за теле-контрол (навигационната система сама ще контролира робота). Следващата стъпка е да

стартираме програмите за навигационната система със следната команда в нов терминал:

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

Тази команда освен, че стартира навигационните пакети, казва на системата за навигация коя да бъде използваната карта. В нашия пример запаметихме карта с име 'map' намираща се в 'HOME' директорията, и сега казваме на навигационната система да използва тази карта.

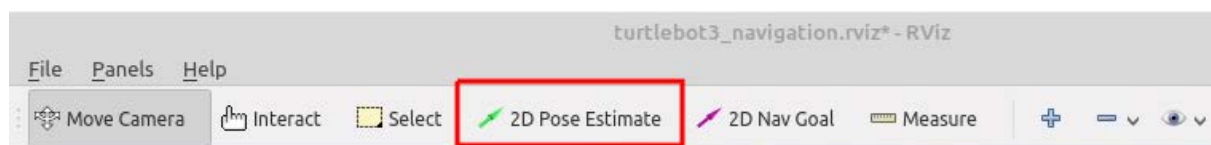
След като сме изпълнили горната команда се отваря прозорец в Rviz, където се визуализира посочената карта, локацията на робота и данните от сензорите (фигура 5). Както се вижда от фигурата данните от лидара (в червено) не съвпадат с очертанията на картата, а има изместване. Това се дължи на разлика в първоначалната позиция на робота и запамената такава при създаване на картата.



Фигура 5. Стартиране на навигационните пакети.

За да работим правилно със системата за навигация трябва коректно да зададем първоначалната локацията на робота. Това става като използваме бутона '2D Pose Estimate'.

1. Натисни бутона 2D Pose Estimate в менюто на Rviz (фигура 6);
2. Избери реалната локация на робота върху картата и завърти голямата зелена стрелка по посоката на ориентация на робота;
3. Повтаряй стъпки 1 и 2 докато очертанията от сензора и тези на картата съвпадат.

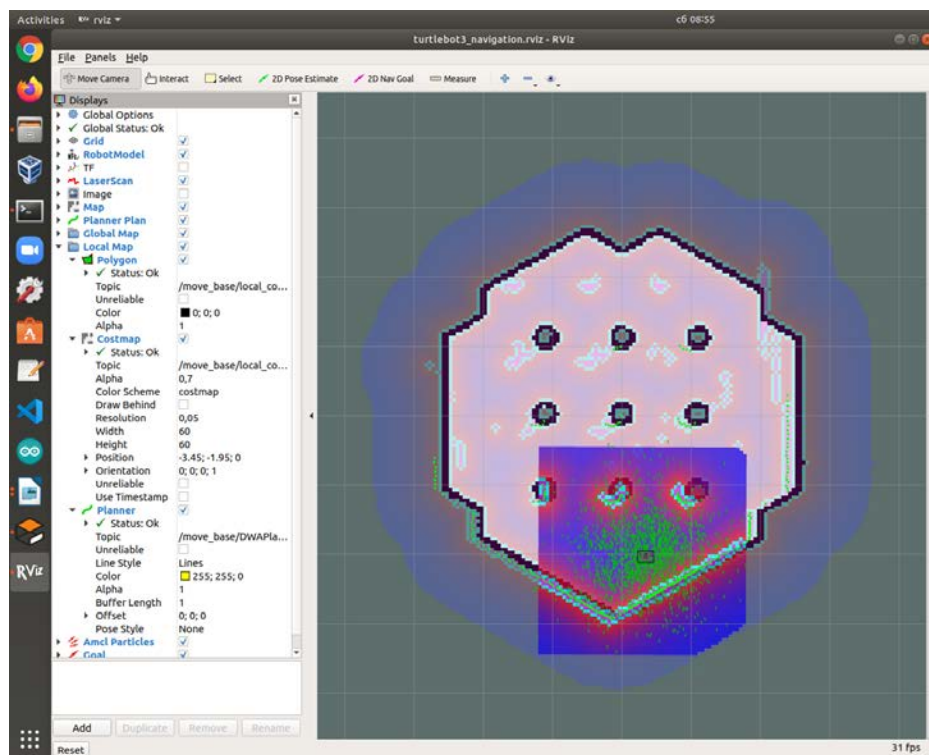


Фигура 6. Меню за работа с Rviz.

Когато успешно зададем правилната локация на робота, тогава очертанията от сензорите и тези на картата трябва да съвпадат напълно. На фигура 7 е показан пример с коректно определена първоначална локация. Обаче върху картата се виждат и петна, които са останали маркирани от предишната локация на робота. За да премахнем стари отпечатыци от данни трябва да изпълним следната команда в нов терминал:

```
rosservice call /move_base/clear_costmaps
```

Тази команда изчиства всички стари данни от картата и след като я изпълним можем да започнем да задаваме желани позиции, където робота да отиде. Следва задаване на желана дестинация за достигане.



Фигура 7. Успешно зададена първоначална позиция.

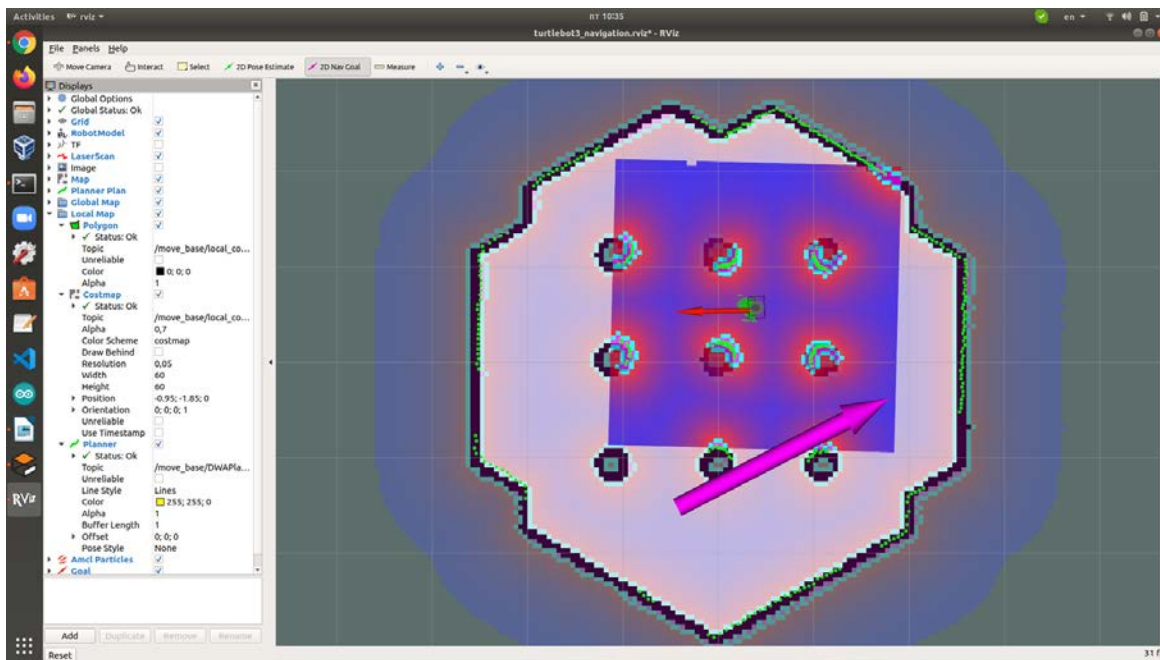
- Задаване на навигационна цел.

1. Натисни бутона 2D Nav Goal в менюто на Rviz (фигура 8);



Фигура 8. Меню за работа с Rviz.

2. Натисни върху картата за да зададеш желана дестинация и завърти лилавата стрелка за задаване на желана ориентация, където робота да се придвижи (фигура 9);



Фигура 9. Задаване на цел към навигацията.

Изчакай докато робота достигне до желаната дестинация и ориентация. Когато робота успешно изпълни заданието в терминала, където е стартиран навигационния възел се изписва съобщението „Goal reached“. След изписване на това съобщение можем да зададем следваща дестинация и ориентация. Върху картата можем да видим локацията и ориентацията на робота, генерирания път до целта и траекторията на движение на робота.

Пример за работа с навигация за Turtlebot3 е достъпен на адрес:

https://www.youtube.com/watch?v=VYIMywwYALU&ab_channel=ROBOTISOpenSourceTeam