

# Building a map using cartographer

---

## Table of Contents

- Table of Contents
- Preparation
- Mapping
- Virtual walls, points arrangement and recommendations
  - Virtual Walls
  - Points arrangement
    - 1. Semi-automatic method
      - Automatic mode
    - 2. Manual method
    - Initial position
  - Finish
  - Recommendations
  - Typical cartography problems
  - Additional links

## Preparation

Before mapping you need to make sure:

- That the robot is [calibrated](#);
- That the correct wheels radius values are specified in <https://pb2.icmm.ru/robotsettings/index>

∨ More details...

Example for standard and identical wheels:

Parameter	Value
robot_settings/driving/wheelRadius	0.084
robot_settings/driving/wheelRadiusLeft	0.084
robot_settings/driving/wheelRadiusRight	0.084

- The robot has a lidar;
- The robot has IMU installed;
- The floor and walls are suitable for the robot and meet [these requirements](#);
- Old maps have been removed from the directory **.promobot/resources/maps/**;
- The files **points.json** and **map.json** have been removed from the directory **.promobot/resources/**;
- The lidar is configured

▾ More details...

1. Make these config files:

```
/opt/promobot/share/promobot_cartographer/configuration_files/promobot_urg_lidar_2d_localization.lua
```

```
/opt/promobot/share/promobot_cartographer/configuration_files/promobot_urg_lidar_2d_map_create.lua
```

contain these lines with following values:

```
TRAJECTORY_BUILDER_2D.max_range = lidar's working distance (measured in meters) +1.
```

```
TRAJECTORY_BUILDER_2D.missing_data_ray_length = lidar's working distance (measured in meters).
```

▾ Config file example...

There is a config file (for 5 meters lidar) screenshot attached below. The screenshot shows that **max\_range** is set to 6 and **missing\_data\_ray\_length** is set to 5.

```
Activities Text Editor Thu 18:00 en
promobot_urg_lidar_2d_localization.lua
/opt/promobot/share/promobot_cartographer/configuration_files
Save

use_landmarks = false,
num_laser_scans = 1,
num_multi_echo_laser_scans = 0,
num_subdivisions_per_laser_scan = 1,
num_point_clouds = 0,
lookup_transform_timeout_sec = 0.2,
submap_publish_period_sec = 0.3,
pose_publish_period_sec = 5e-3,
trajectory_publish_period_sec = 5e-1,
rangefinder_sampling_ratio = 1.0,
odometry_sampling_ratio = 1.0,
fixed_frame_pose_sampling_ratio = 1.,
imu_sampling_ratio = 1.0,
landmarks_sampling_ratio = 1.,
}

MAP_BUILDER.use_trajectory_builder_2d = true

POSE_GRAPH.optimize_every_n_nodes = 30

--TRAJECTORY_BUILDER_2D.ceres_scan_matcher.ceres_solver_options.use_nonmonotonic_steps = true
TRAJECTORY_BUILDER_2D.num_accumulated_range_data = 2.
TRAJECTORY_BUILDER_2D.min_range = 0.1
TRAJECTORY_BUILDER_2D.max_range = 6.
TRAJECTORY_BUILDER_2D.missing_data_ray_length = 5.
TRAJECTORY_BUILDER_2D.use_imu_data = true
TRAJECTORY_BUILDER_2D.use_online_correlative_scan_matching = false

TRAJECTORY_BUILDER_2D.submaps.num_range_data = 30.

--TRAJECTORY_BUILDER_2D.submaps.range_data_inserter.probability_grid_range_data_inserter.hit_probability = 0.57
--TRAJECTORY_BUILDER_2D.submaps.range_data_inserter.probability_grid_range_data_inserter.miss_probability = 0.45

TRAJECTORY_BUILDER_2D.motion_filter.max_angle_radians = math.rad(0.15)
TRAJECTORY_BUILDER_2D.motion_filter.max_distance_meters = 0.05

-- more points
--TRAJECTORY_BUILDER_2D.adaptive_voxel_filter.max_length = 0.2
--TRAJECTORY_BUILDER_2D.adaptive_voxel_filter.min_num_points = 400

-- wheel odometry is fine

Lua Tab Width: 8 Ln 1, Col 1 INS
```

Thus, for a 20 meters lidar:

- - `max_range = 21`
  - `missing_data_ray_length = 20`

Please put a period after the integer number, as it is done on the screenshot!

2. If lidar is connected via Ethernet, then set this parameter

`robot_settings/driving/useIpLidar` to `true`.

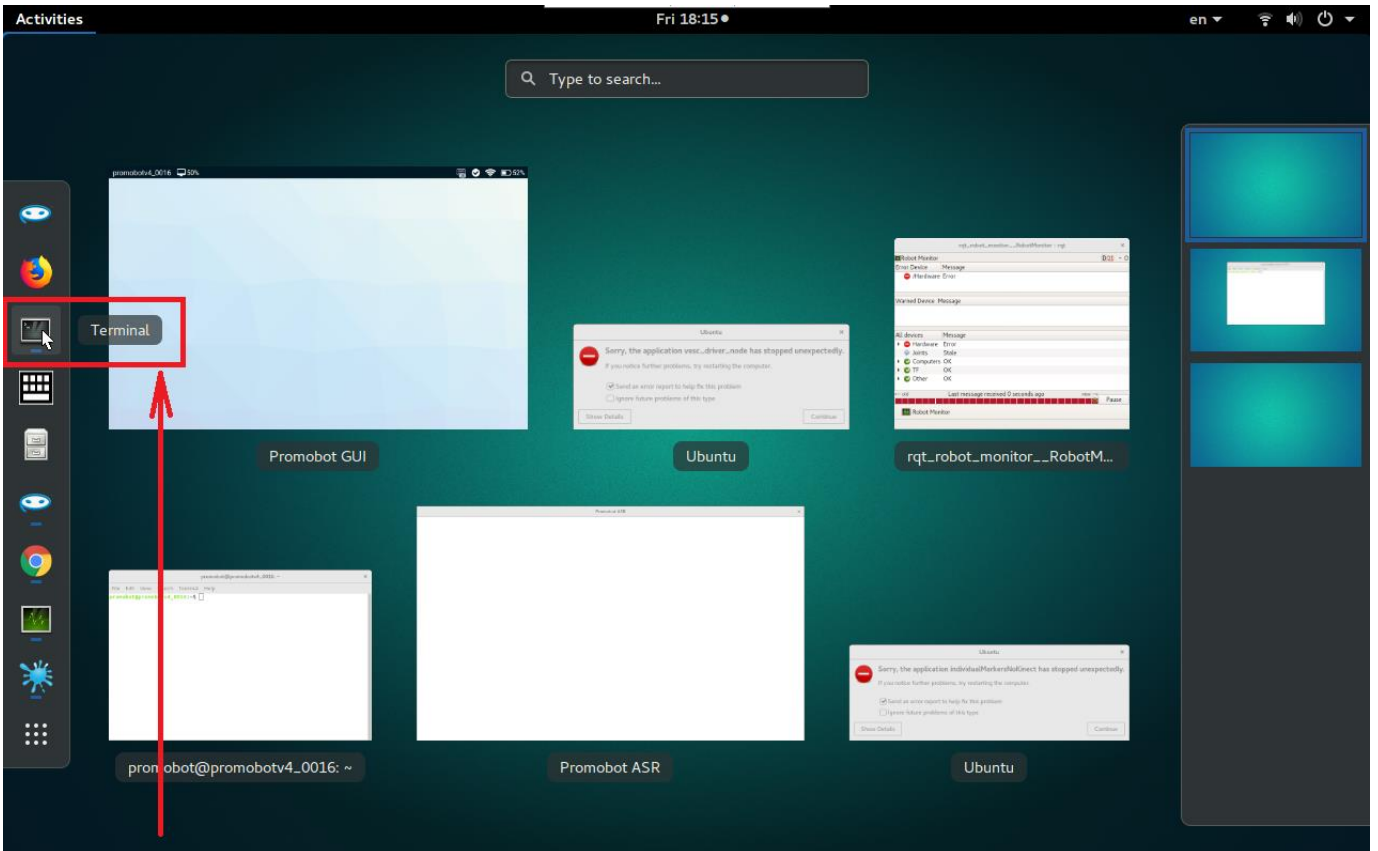
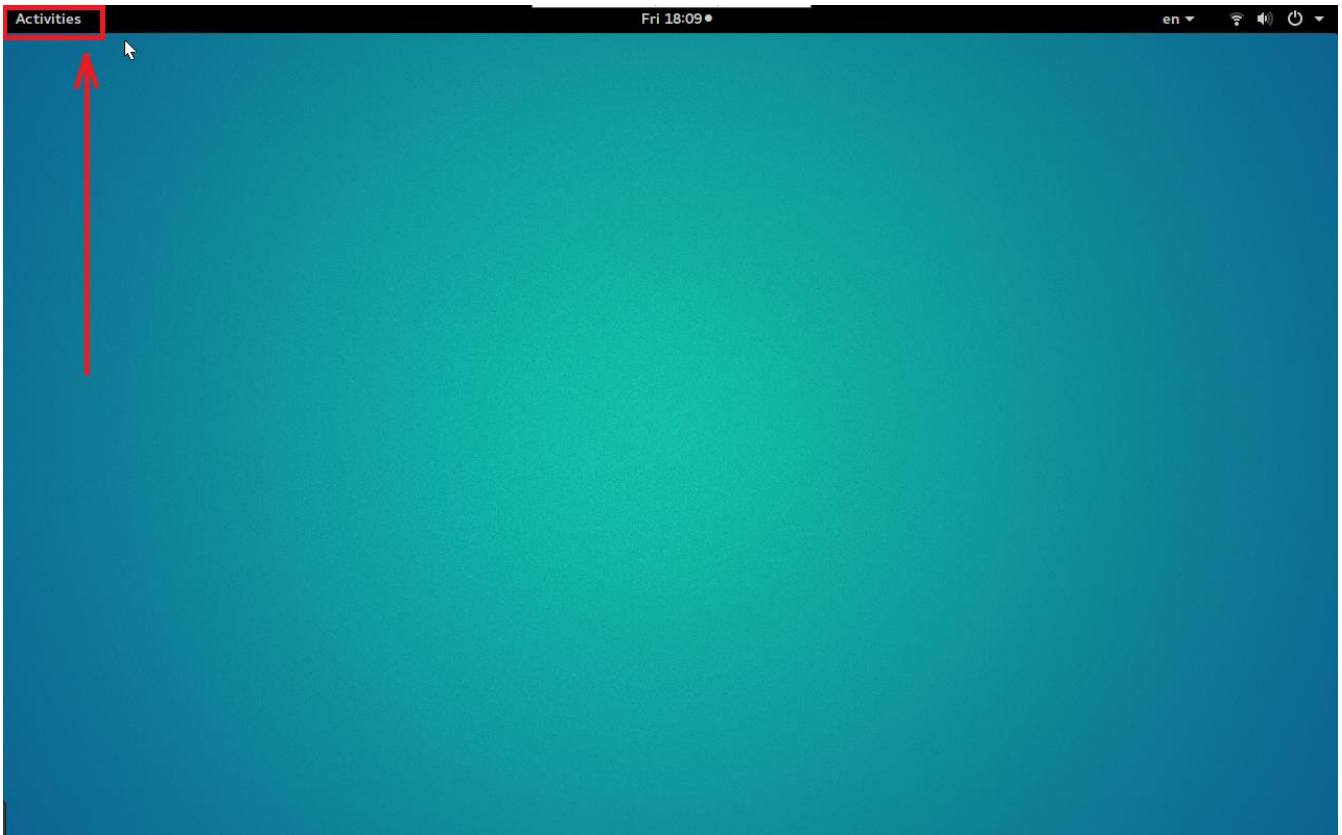
3. Edit file `home/.promobot/.bashrc`, set lidar type according to the lidar set on robot:

```
export LIDAR_TYPE=
```

Options:

- `FS_IP_250`
- `SICK_IP_250`
- `URG_IP_250`
- `URG_IP`
- `FS_IP`





2. Check data flow from IMU:
3. 

```
rostopic echo /imu  
rostopic echo /imu/data
```

The terminal should show real-time IMU data. If the data is not available, check IMU connection and settings.

4. Check data flow from Lidar:

```
rostopic echo /scan
```

The terminal should show real-time LIDAR data. If the data is not available, check LIDAR connection and settings.

5. Shut down robot's processes:
6. 

```
killall promobot_gui_op
```
7. 

```
killall core.sh  
killall roscore
```

After completing these commands, the robot GUI will be closed.

8. Put the robot in the zero point location place (it is recommended to zero point be charging station).

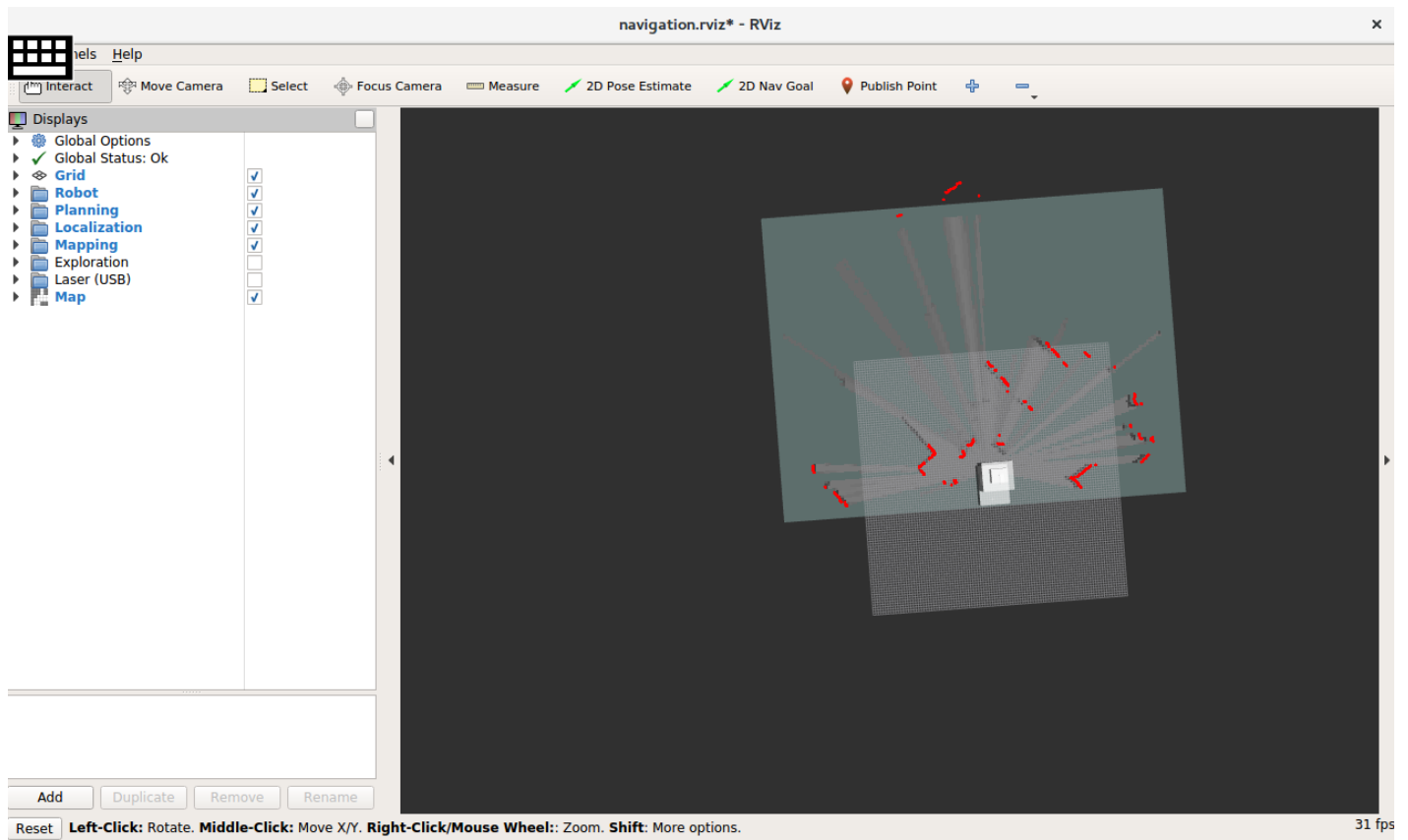
Put the robot in an desired angle of rotation! That means, if you are going to place the marker of charging station with its backside to a certain wall, then you should also put the robot with its back to this wall.

9. Start the process of building a map using this command:

```
roslaunch promobot_bringup map_create.sh
```

**Starting map building via ssh is not supported!**

After that you should see the **Rviz** window.



Make sure that obstacles are painted red, robot's icon is shown fully and **there are no error messages**.

▼ Errors look like these...

**Displays**

- Global Options
- Global Status: Ok
- Grid
- TF
- Submaps
- PointCloud2
- MarkerArray
- Map
- Map
- RobotModel**
  - Status: Error
  - URDF
    - URDF parsed OK
    - base\_footprint No transform from [base...
    - base\_link No transform from [base...
    - base\_top\_link No transform from [base...
    - bwheel No transform from [bwh...
    - camera No transform from [cam...
    - camera\_link No transform from [cam...
    - hark\_link No transform from [hark...
    - head\_1\_link No transform from [head...
    - head\_2\_link No transform from [head...
    - head\_base\_link No transform from [head...
    - head\_pan\_F3\_0\_link No transform from [head...
    - head\_pan\_s1\_0\_link No transform from [head...
    - head\_tilt\_F2\_0\_link No transform from [head...
    - head\_top\_link No transform from [head...
    - hedgehog\_link No transform from [hedg...
    - imu link No transform from [imu...

Add Duplicate Remove Rotate

Time  
ROS Time: 1610720773.44 ROS Elapsed: 31.03 Wall T  
Reset



Rviz has built-in configuration that is most suitable for mapping. To use this configuration press `ctrl+o` or **File** → **Open**, then follow the path `/opt/promobot/share/promobot_cartographer/configuration_files` and choose `demo_promobot.rviz`

If you can't find the robot's model in the Rviz interface check the `cartographer_ros` folder in `/opt/promobot/share`.

10. Open two additional terminal windows. In the first terminal enter the command to view data from the left wheel:

```
rostopic echo /lwheel
```

In the second terminal enter the command to view data from the right wheel:

```
rostopic echo /rwheel
```

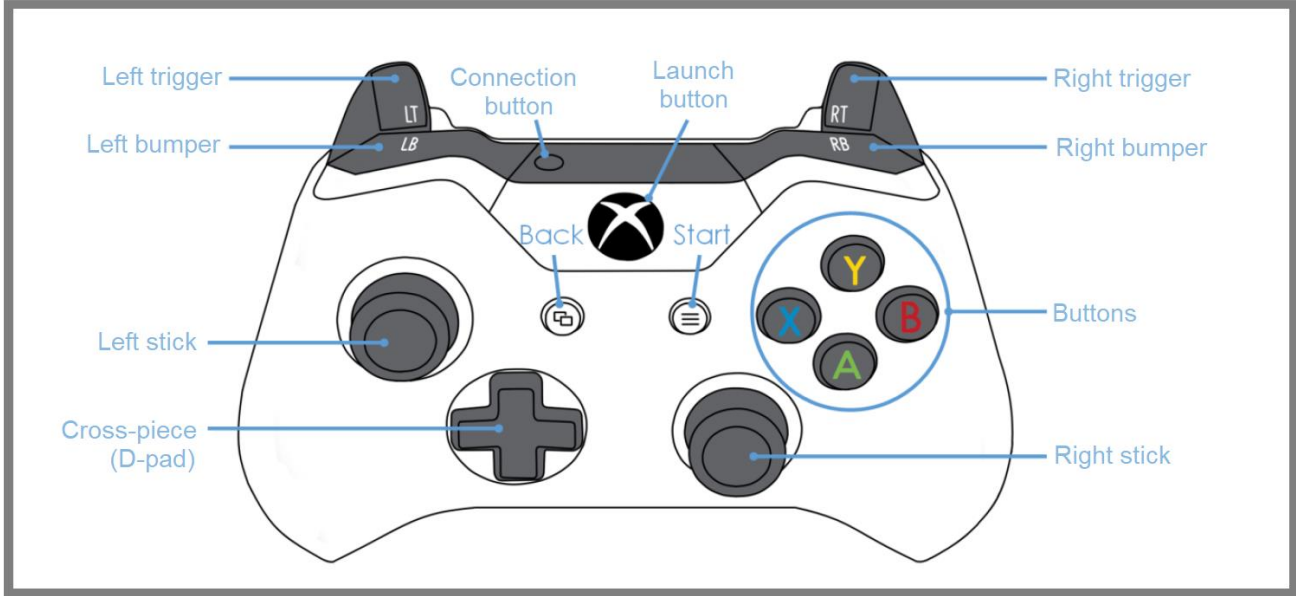
Keep these terminals open all the time during the process of building the map and watch that the values increase during the driving.

▼ Why is it necessary?

There was a case when, after three hours of driving, the map was not saved (`map.pgm` looked like a white sheet). During the ride there was a vesc (wheels board) failure. When such a failure occurs, the map is not saving. The map is building until the first vesc failure.

Therefore, it makes sense to observe whether the data comes from the wheels or not, so that in case of a failure, you do not continue to drive around the room in vain.





Combinations to control the robot:

Combination	Description
Back + Start	Switching auto movement mode
Back + Y	Controller activated phrases mode
A + B + Left stick	Moving the robot with a very fast speed
A + Left stick	Moving the robot with a fast speed
B + Left stick	Moving the robot with a normal speed
A/B + Cross-piece	Head control (when moving up and down, the actuator additionally works)
X + Left bumper	Increase speaker volume by 5%
X + Left trigger	Decrease speaker volume by 5%
X + Right bumper	Increase microphone sensitivity by 2%
X + Right trigger	Decrease microphone sensitivity by 2%
Available only on controller activated phrases mode	
Y + Right trigger	Next phrase
Y + Right bumper	Previous phrase

✓ Map building algorithm...

Terms:

**Corridor** is any space, where the distance between walls is less than the lidar working distance (lidar can see both walls at the same time).

**Room** is any space connected by corridors.

**Teleportation** is a situation when there are identical areas on the map, and, while being on one of them, the robot believes that he is on another one.

1) If the robot is standing close to the wall, then move it slightly away from the wall so that nothing prevents it from making a 360-degree rotation around its axis. Rotate the robot 360 degrees around its axis.

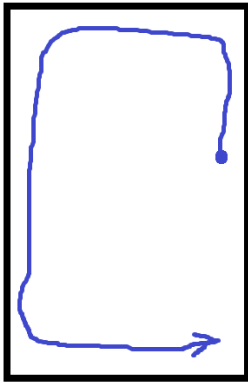
✓ Why do I need to make a rotation?

This is a preparation for the next step. In the end of building the map the robot returns to the same point from which the mapping was started. In order to lines (which represent walls) are correctly connected to each other it is necessary for starting point to be as much scanned with lidar as possible (which is what we do with this rotation).

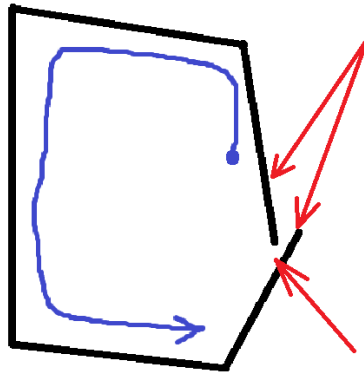
2) Select the closest to the robot wall and start driving along it, ignoring all other objects. Make a full circle around the room.

3) If there is a place on the map where the lines (which represent walls) didn't connected (the cycle did not closed), then you need a robot to spin around its axis (1-2 circles) until the lines connect.

✓ What does it mean - the cycle didn't closed?



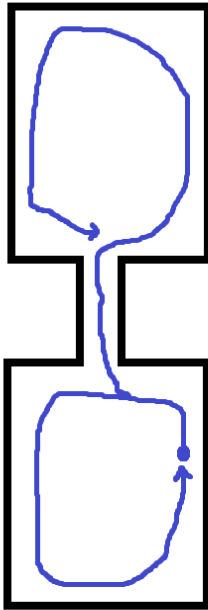
How the room really looks



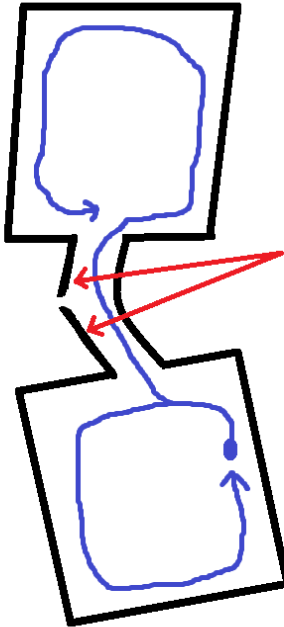
How lidar sees it

In fact, this are the one same wall, but it seems like this are two walls

Here is that spot where the cycle didn't closed, the lines didn't connected



How the room really looks



How lidar sees it

In fact, this are the one same wall, but it seems like this are two walls

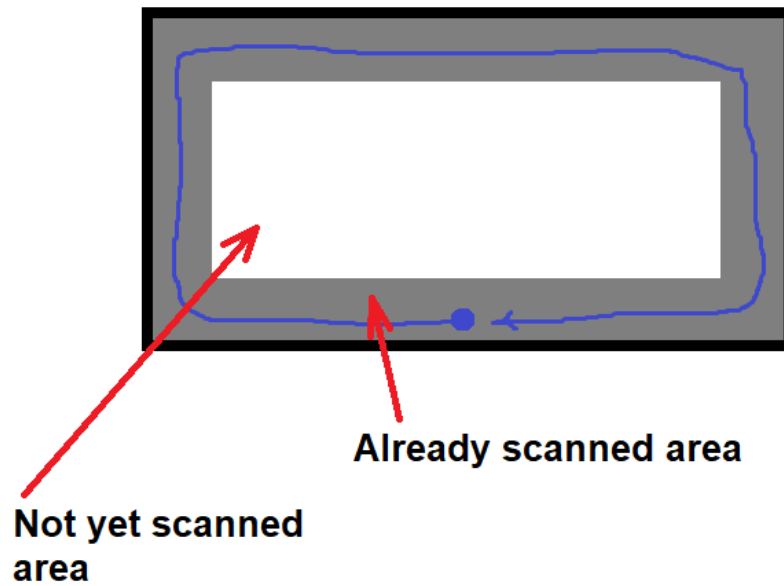
Here is that spot where the cycle didn't closed, the lines didn't connected

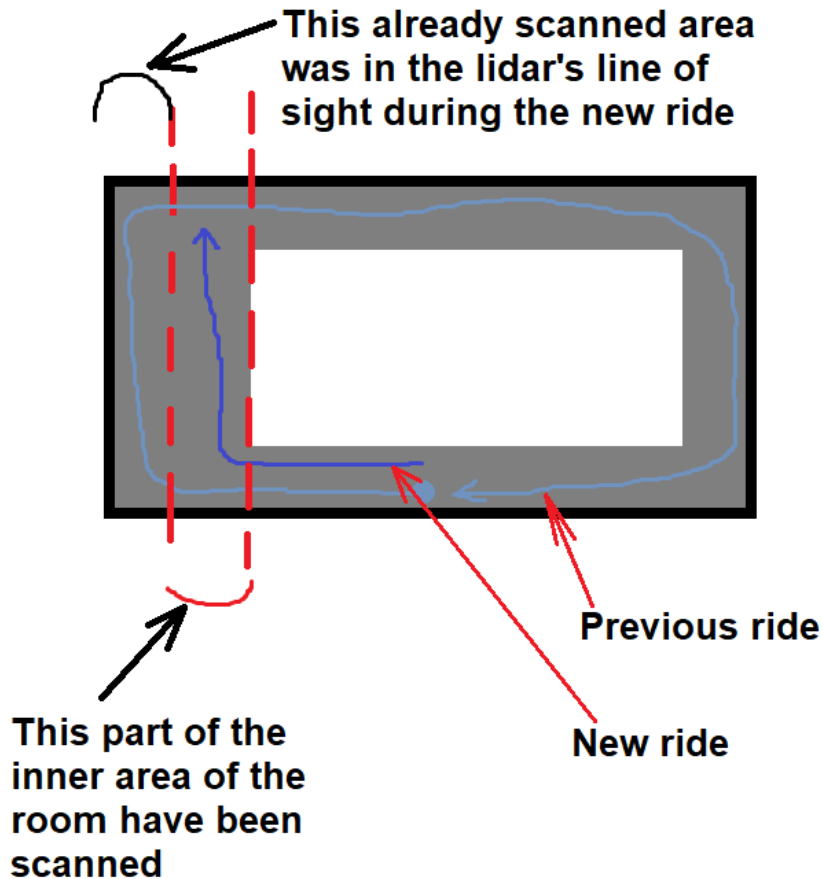
4) Scan the inner part of the room keeping the wall (or an already scanned area) within the lidar line of sight.

∨ Further...

If the room is small (for example 10 x 10 m with a lidar working distance of 5 m), then the inner part of the room can have been scanned after the first drive along the walls. In this case, no additional scanning is required.

If the room is greater than the lidar's working distance, then it is required to scan the inner part of the room keeping the wall or an already scanned area within the lidar line of sight.





Attention!

It is better to keep in sight **NOT ANY** scanned area, but some **DISTINCTIVE** one. If it is not possible, it is recommended for you to make robot to come to the corner of the room from time to time and make sure that the robot is displayed in the correct place on the map.

This will help avoid cases of "teleportation".

- 5) If there is a previous room where the map has already been built (a room connected by a corridor to the current room), then return there and correct the shifts of the walls and corridor (using the instructions from step 3).
- 6) If there are rooms with an unbuilt map then select one of them and repeat steps 1-5, otherwise go to step 7.
- 7) Drive across the entire map and check the localization (make sure that the real position of the robot matches the position of the robot on the map). If the

localization is incorrect, then move around the map looking for disjoints in the walls until the localization starts working correctly.

It is recommended to drive slowly. Mapping ending should be in the same place with mapping beginning (robot position and rotation angle should be the same).  
**To cancel mapping press ctrl+c in the terminal.**

## 12. Save the map.

**Run secondary terminal and save the map using:**

```
roslaunch promobot_bringup map_save.sh
```

Close the secondary terminal. To finish mapping press **ctrl+c** in the terminal that is used to run **Rviz**.

If map is not saved after mapping process, robot won't move in auto-mode.

## 13. Set the following parameters in the database web-interface:

▼ Detailed instructions...

You need to go to the database web-interface using [this link](#) and log in using your login and password.

Then go to **Configuration** → **Robot settings**, select robot from the drop-down list.



pb2.icmm.ru/robotsettings/index

Promobot DB v3

ЛингвоБаза ▾ Аналитика ▾ Конфигурация ▾ Администрирование ▾ ... ▾

Home / Robotsettings

Создать [Иконка] [Иконка]

Enter parameter ID (or part of ID)...

Выберите владельца...

Выберите робота...

Выберите приложение...

Выберите группу...

Выберите параметр...

Выберите тип...

Include common settings

Время создания

Время изменения

Показать Сброс

Выберите робота...

Выберите робота...

Сравнить Сброс

First « 1 2 3 4 5 6 7 8 9 10 11 12 » Last Всего записей: 485206 Всего страниц: 9705 Показаны записи: 1-50

#	Робот	Параметр ↕	Значение	Создатель	Created	Редактор	Edited	
1	Промобот 4.28	robot_settings/driving/useMap		Вячеслав Логинов	02 March 2020 19:01:24	(not set)	02 March 2020 19:01:24	👁️ 🗑️ 📄
2	Промобот 4.34	Aqui está a sua linda foto!		Вячеслав Логинов	02 March 2020 19:01:24	Elena Senik	02 March 2020 19:43:20	👁️ 🗑️ 📄
3	Промобот 4.40			Вячеслав Логинов	02 March 2020 19:01:24	(not set)	02 March 2020 19:01:24	👁️ 🗑️ 📄
4	Промобот 4.46			Вячеслав Логинов	02 March 2020 19:02:35	(not set)	02 March 2020 19:02:35	👁️ 🗑️ 📄

Next, you need to copy the name of the parameter from the table below and paste it into the searching field by in the database web-interface (then press Enter or click on any free space of the form to find the parameter) (1 in the screenshot).

Click on the pencil icon to open parameter value editing window (2 in the screenshot).

pb2.icmm.ru/robotsettings/index

Promobot DB v3

ЛингвоБаза ▾ Аналитика ▾ Конфигурация ▾ Администрирование ▾ ... ▾

Home / Robotsettings

Создать [Иконка] [Иконка]

robot\_settings/driving/useMap

Выберите владельца...

promobot4\_0063 (Промобот 4.63)

Выберите приложение...

Выберите группу...

Выберите параметр...

Выберите тип...

Include common settings

Время создания

Время изменения

Показать Сброс

Выберите робота...

Выберите робота...

Сравнить Сброс

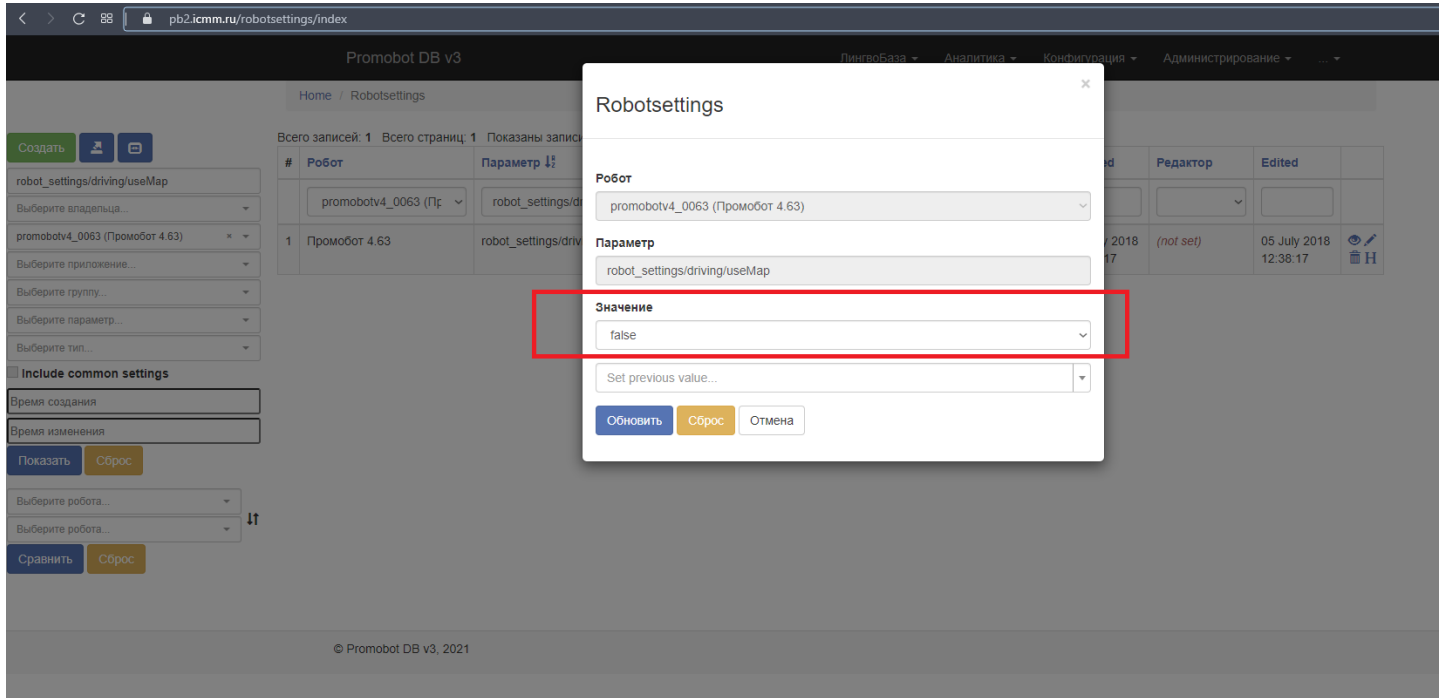
Всего записей: 1 Всего страниц: 1 Показаны записи: 1-1

#	Робот	Параметр ↕	Значение	Создатель	Created	Редактор	Edited	
1	Промобот 4.63	robot_settings/driving/useMap	false	Андрей Будрин	05 July 2018 12:38:17	(not set)	05 July 2018 12:38:17	👁️ 🗑️ 📄 ✎

1

2

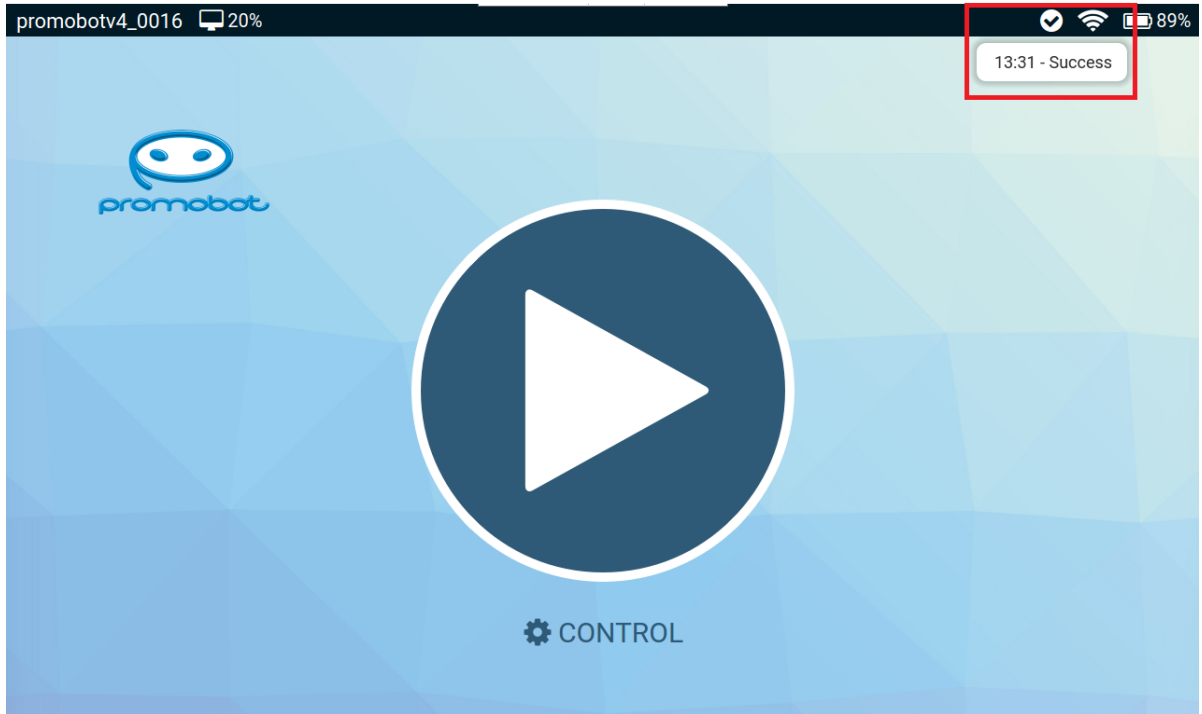
Enter the value from the table in the appropriate field of opened window and click **Update**.



Repeat these steps for all table parameters.

Make sure that parameters update on the robot was successful (replication status - Success).

▼ Screenshot...



Parameter	Value
robot_settings/driving/useMap	true
robot_settings/driving/usePeopleSearch	false
robot_settings/driving/useRadius	false
robot_settings/driving/skipStationScanning	true false (for release 1.9.3 and above)
robot_settings/tf_switcher/default_in_frame	cartographer_frame

## Virtual walls, points arrangement and recommendations

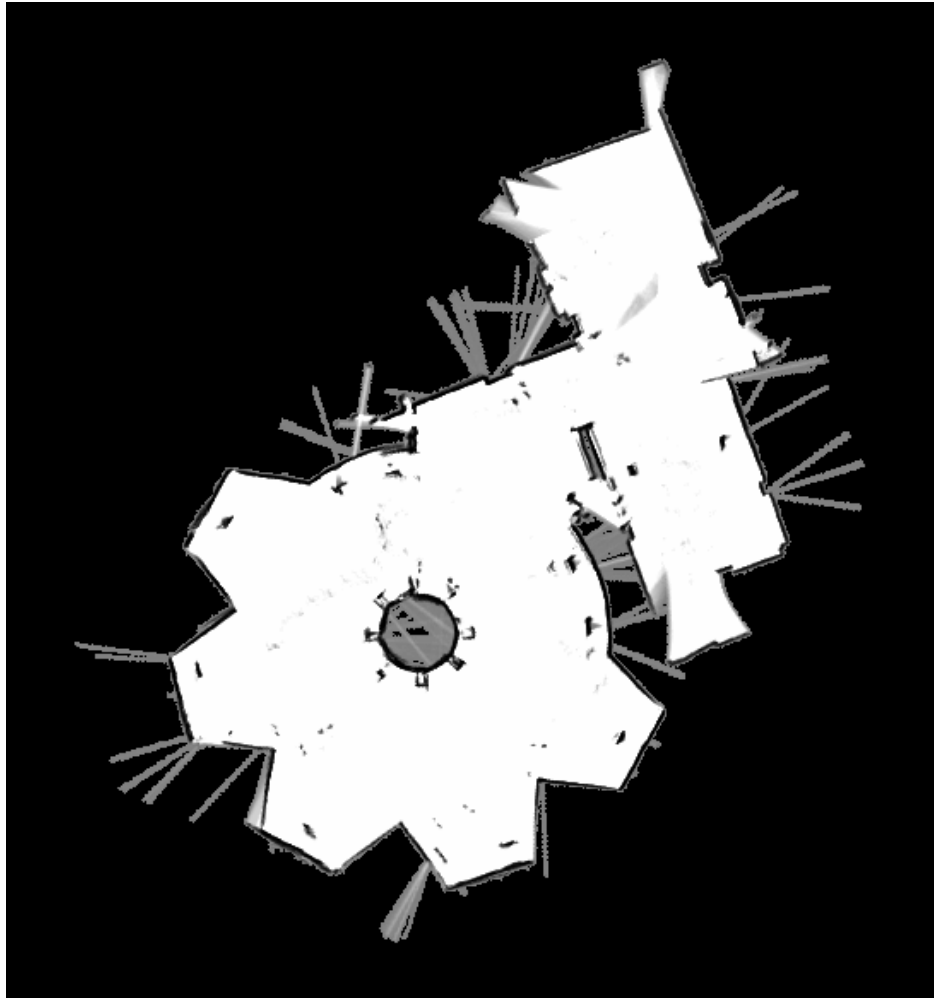
### Virtual Walls

Saved map will be located at:

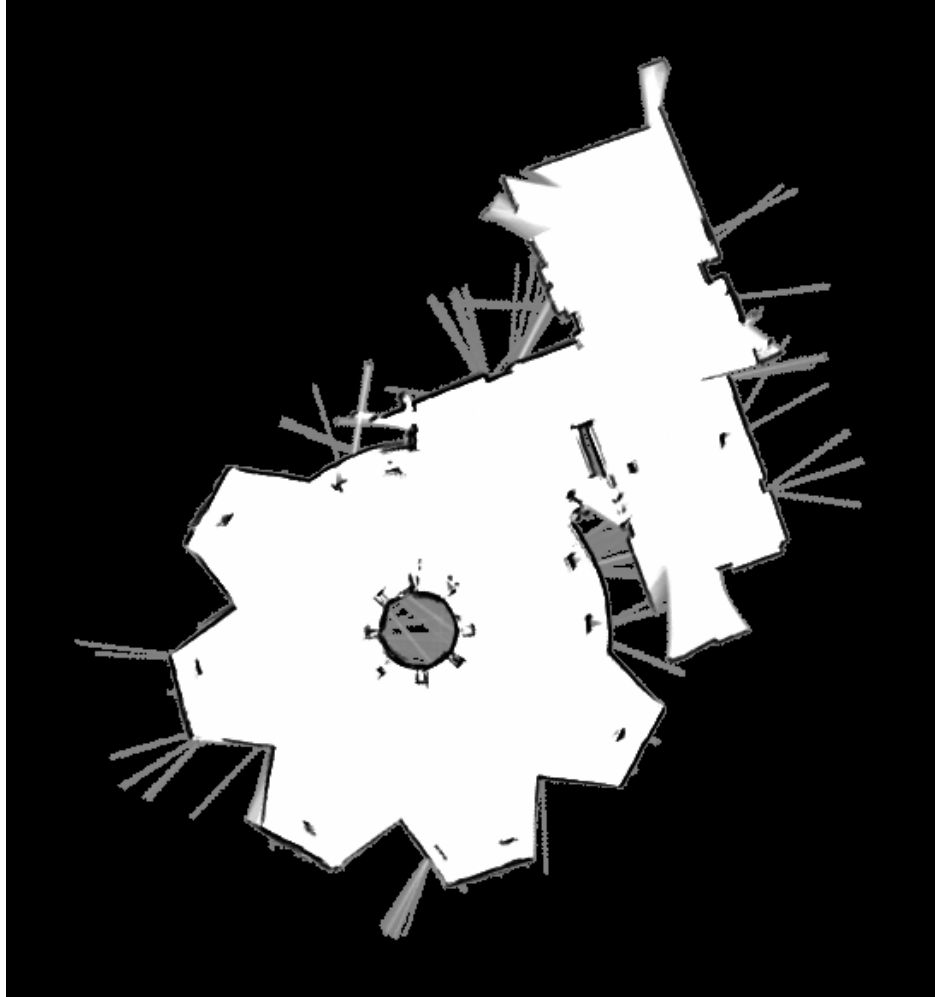
`/home/promobot/.promobot/resources/maps/map.pgm`

**You can open the map in the graphics editor (GIMP, Inkscape, etc.). Dark color is for walls and obstacles and white is for the space where the robot will be able to move.**

The automatically generated map file is likely to be filled with “phantom”, non-existing obstacles that should be cleaned in the graphics editor. The example below shows the initial map and the cleaned map.

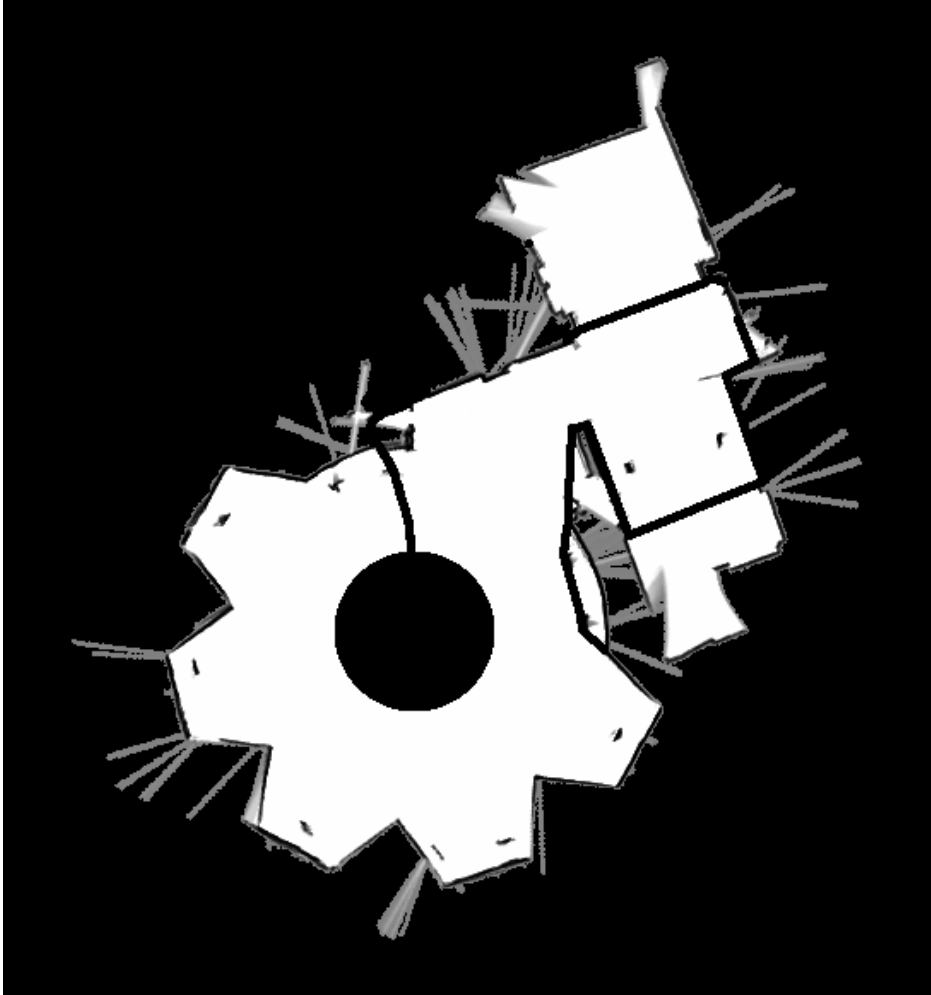


Initial map

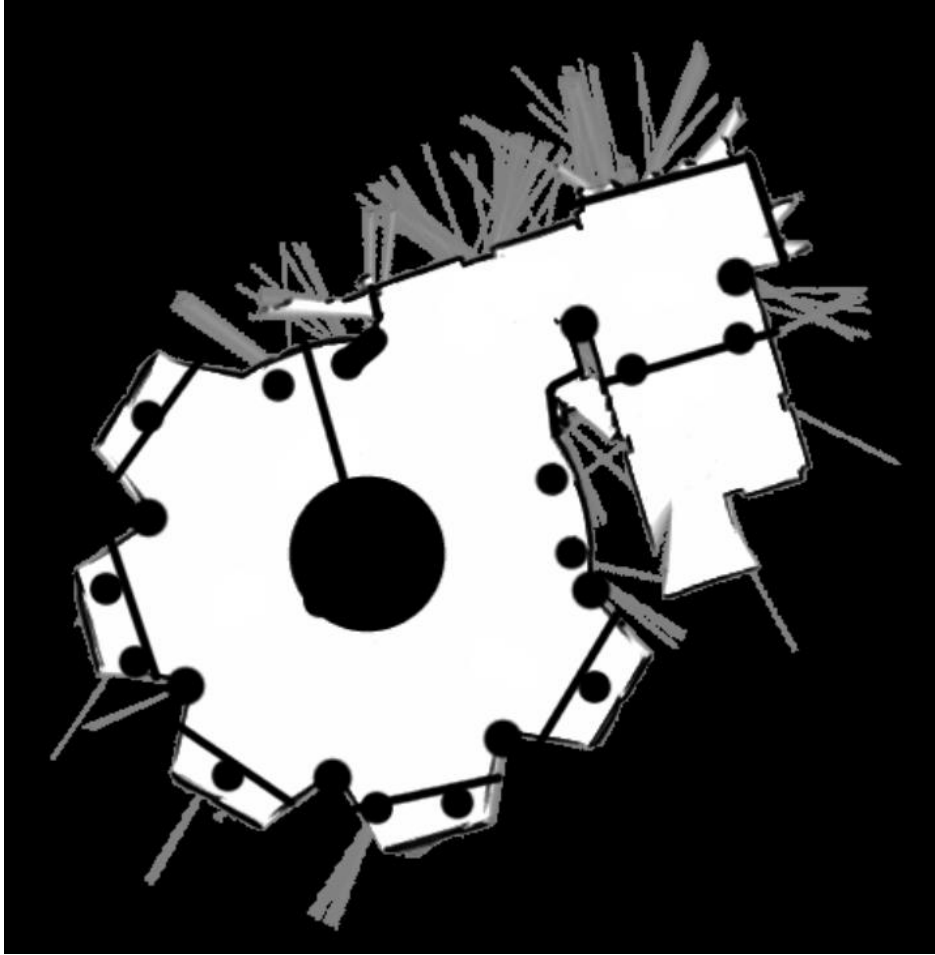


Cleaned map

If you want to restrict robot's movement you can add "phantom", virtual walls the same way you delete "phantom" obstacles: by drawing them in the graphics editor. The example below shows the initial map and the map with some virtual walls.



When adding virtual walls, it is recommended to smooth out sharp corners as much as possible, especially those protruding into the intended navigation area.



Map with sharp corners smoothed out

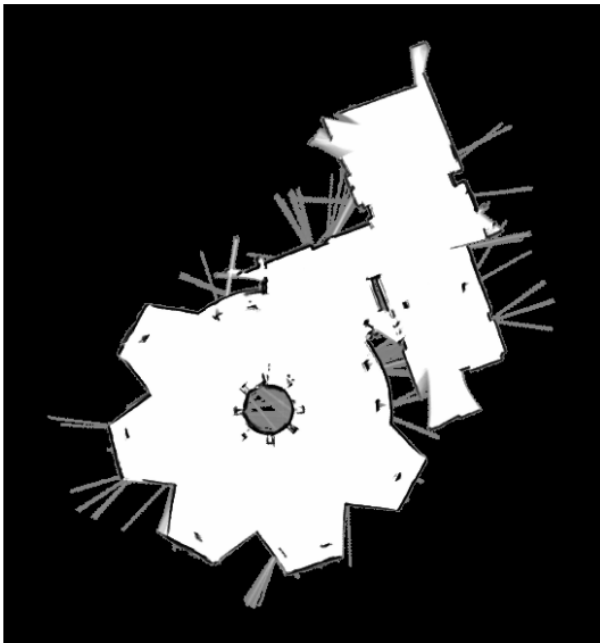
∨ What has changed on the maps? I can't see any differences..



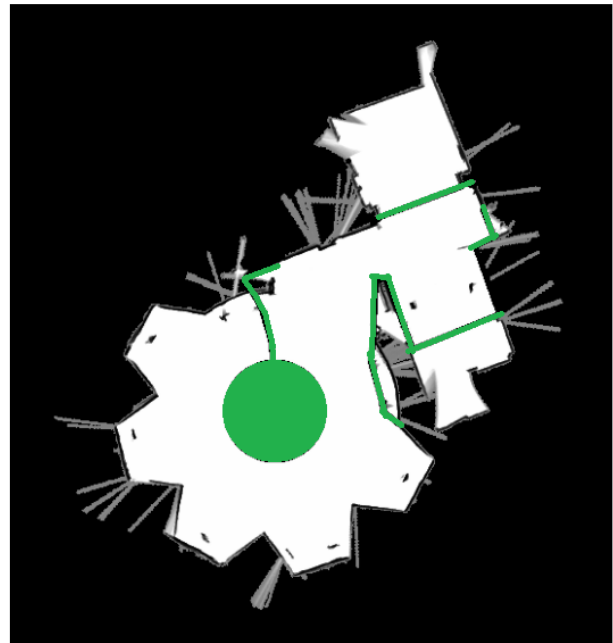
Initial map



Cleaned map



Cleaned map



Virtual walls added

Virtual walls shouldn't have any gaps because they can be treated as a free space. That may cause unpredicted robot's movement and should be avoided. Check the map using Bucket Fill tool. If you want to limit robot's movement you can add "phantom", virtual walls the same way you delete "phantom" obstacles: by drawing them in the graphics editor. The example above shows the initial map and the map with some virtual walls.



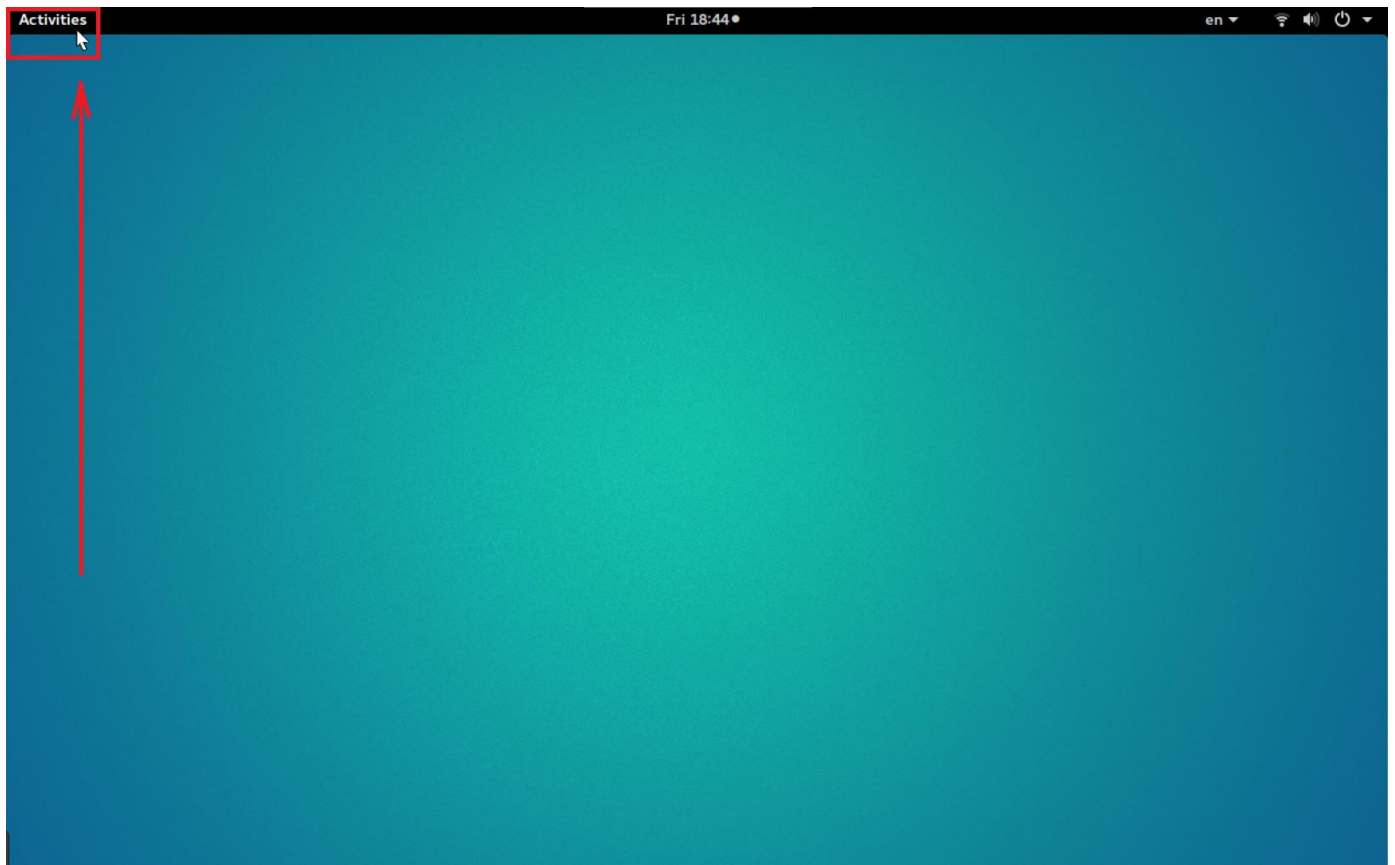
I would like to note that after the first editing in GIMP, the map will not necessarily be final, there may be several iterations when the walls are drawn when the robot gets stuck or other circumstances.

## Points arrangement

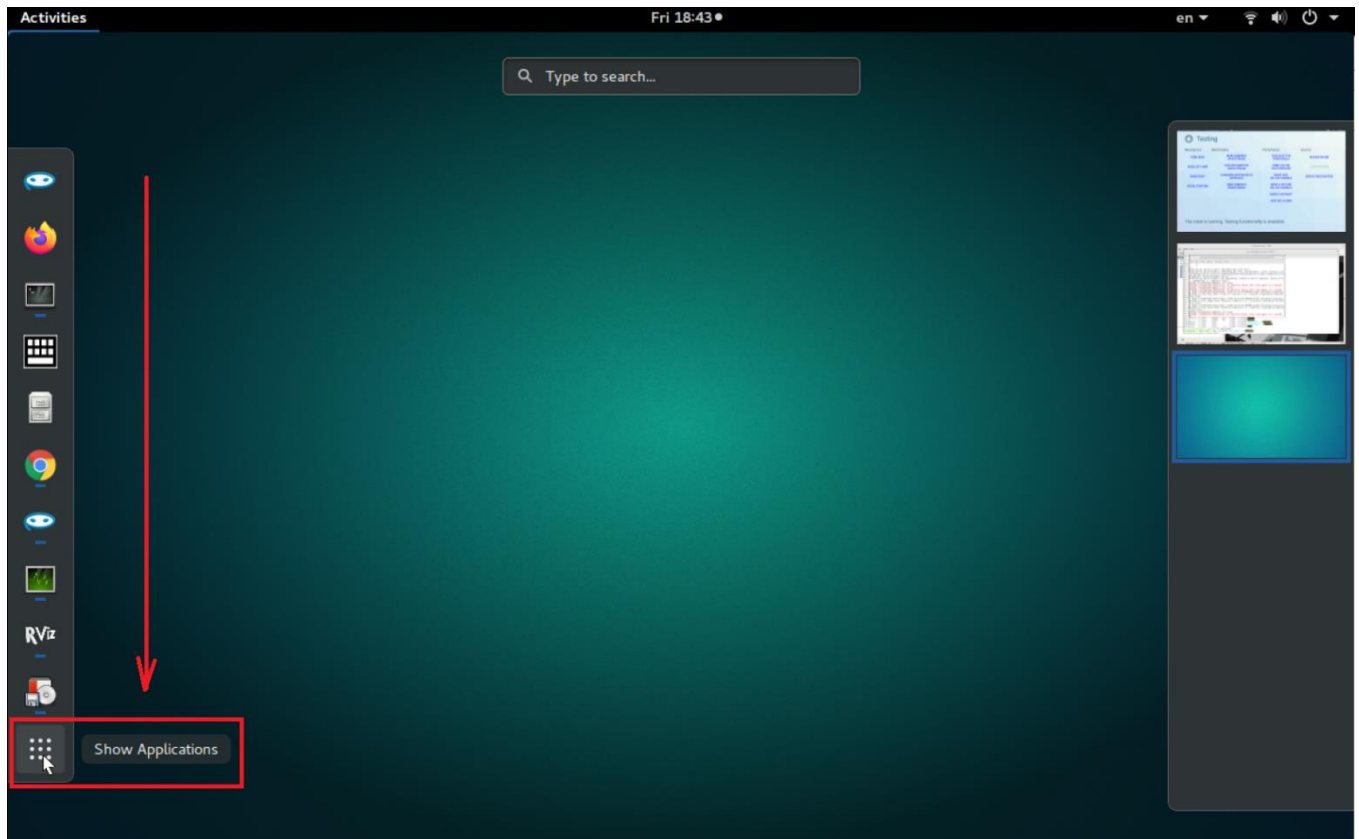
GUI should be running! Run the GUI.

▼ How to run the GUI?

1. Open **Activities** menu.



2. Click on the **Show Applications** menu icon.



3. Open Promobot GUI.



There are 2 ways to arrange points:

1. Semi-automatic method.
2. Manual method.

**Advice.** Use the 1st method as the main one, and the second method only if you need to shift a particular point.

### 1. Semi-automatic method

Before the point positioning you should run the GUI and switch the robot to the Manual mode by pressing **back+start** on the controller.

Important!

It is highly recommended to set the zero (starting) point at the charging station. Otherwise, the automatic return to charging station algorithm will send the robot not to the charging station.

Important!

After the point positioning robot should be in the starting point in the exact position it was in the beginning of the positioning (it's initial turn angle deviation should be +/- 5 °). Use the charger as a reference point.

To position points in the Semi-automatic mode proceed as follows:

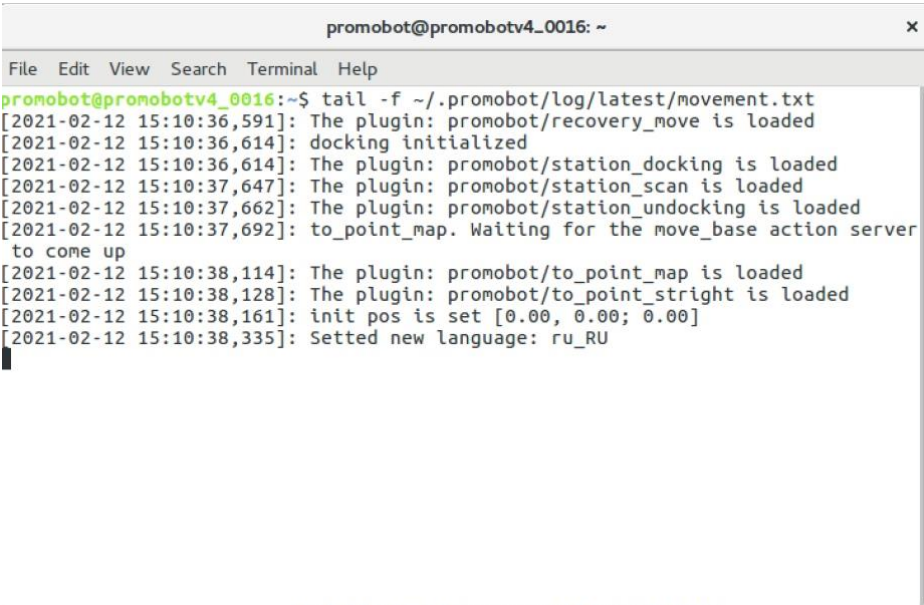
1. Minimize GUI.
2. Robot should stand in zero point position.
3. Open new terminal and run:

```
tail -f ~/.promobot/log/latest/movement.txt
```

Information on the screen will show the points positioning process.

▼ Terminal output...

After executing this command, the output in the terminal will look something like this:



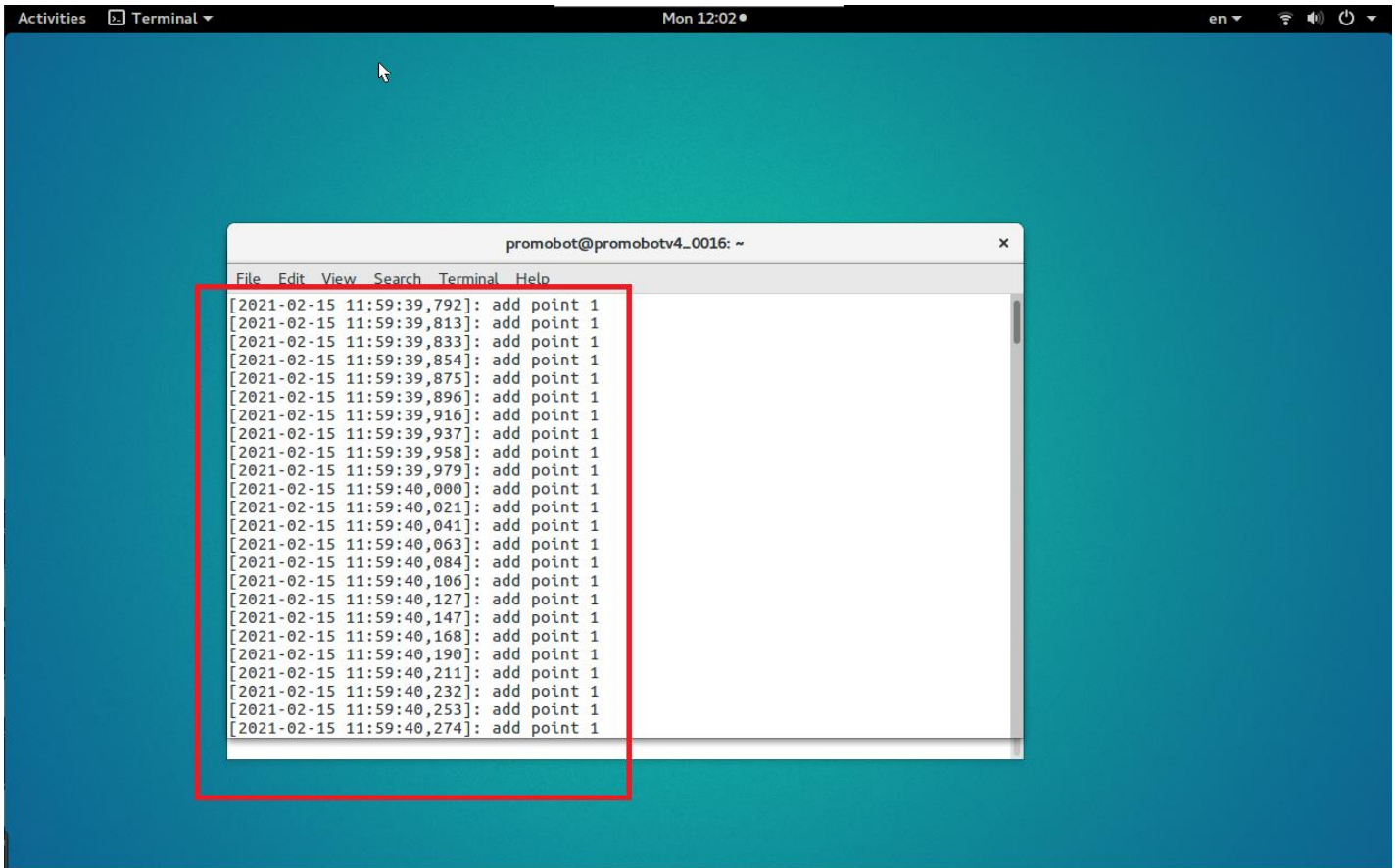
```
promobot@promobotv4_0016: ~
File Edit View Search Terminal Help
promobot@promobotv4_0016:~$ tail -f ~/.promobot/log/latest/movement.txt
[2021-02-12 15:10:36,591]: The plugin: promobot/recovery_move is loaded
[2021-02-12 15:10:36,614]: docking initialized
[2021-02-12 15:10:36,614]: The plugin: promobot/station_docking is loaded
[2021-02-12 15:10:37,647]: The plugin: promobot/station_scan is loaded
[2021-02-12 15:10:37,662]: The plugin: promobot/station_undocking is loaded
[2021-02-12 15:10:37,692]: to_point_map. Waiting for the move_base action server
to come up
[2021-02-12 15:10:38,114]: The plugin: promobot/to_point_map is loaded
[2021-02-12 15:10:38,128]: The plugin: promobot/to_point_straight is loaded
[2021-02-12 15:10:38,161]: init pos is set [0.00, 0.00; 0.00]
[2021-02-12 15:10:38,335]: Setted new language: ru_RU
```

4. Now you need to tell the robot to record the current point:
  1. if you are using a controller: press **back+a**
  2. if you are using a terminal window (**this must not be the same window as for tail command**):

```
rostopic pub /drive/points/save std_msgs/Bool "data: true"
```

The *add point N* lines will quickly start be displayed in terminal.

▼ Screenshot...

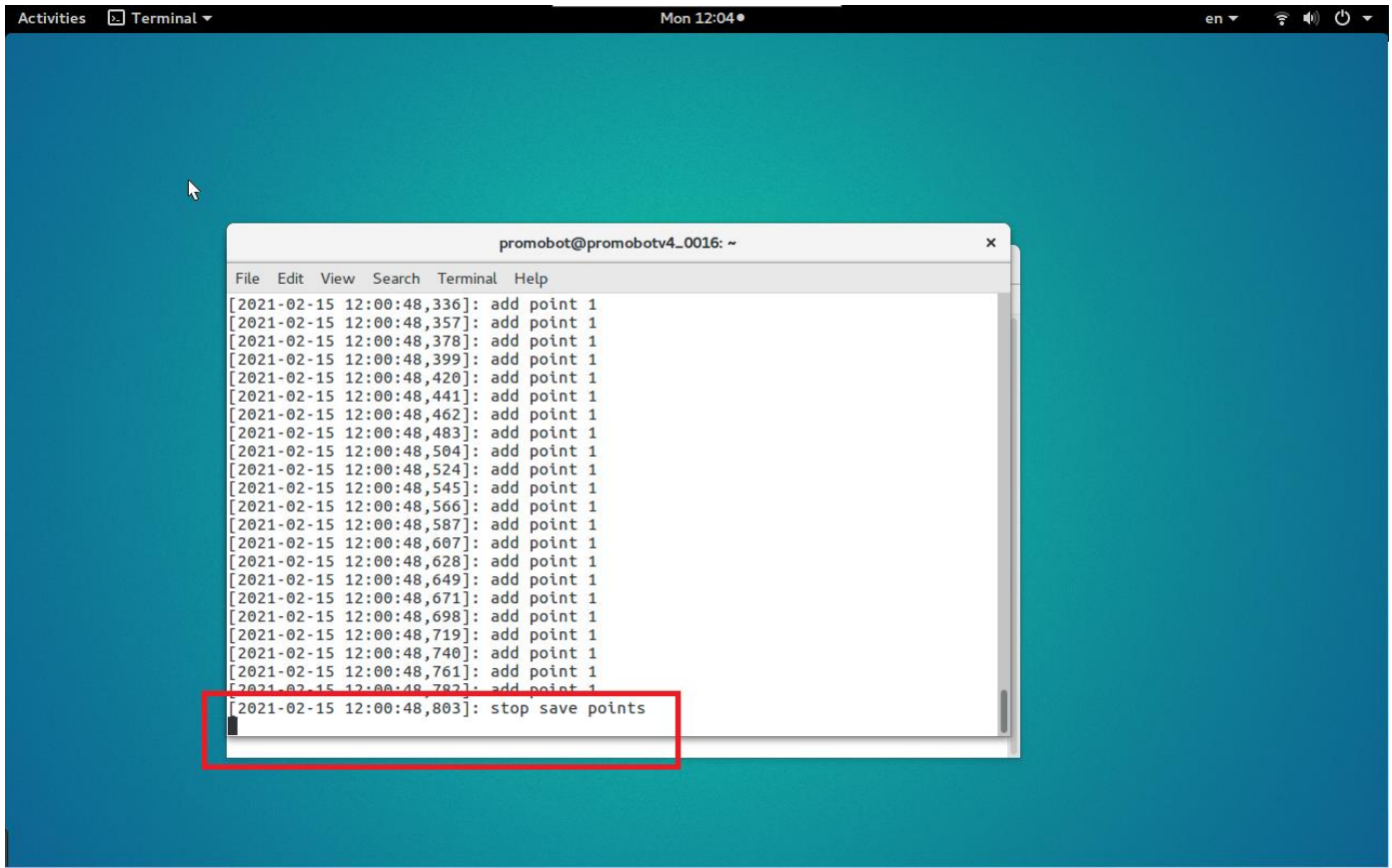


5. Wait for **5 seconds** and then either
  1. press **back+a**, if you are using a controller
  2. or type the following in the terminal (**this must not be the same window as for tail command**):

```
rostopic pub /drive/points/save std_msgs/Bool "data: false"
```

The *stop save points* line will be displayed in terminal.

▼ Screenshot...



6. Move the robot to the next point using the controller and repeat steps 3 through 5;
7. After you create all the points, finish the point positioning by returning the robot to the starting point;
8. Make sure there is a **points.json** file in **.promobot/resources** directory;
9. To enable automatic movement copy **points.json** and rename it to **map.json**

NOTE: you should now have 2 identical files: points.json and map.json

The first point should be not closer than 2 meters to the charging station and the rest – not closer than 1 meter to the charging station.

### Automatic mode

Movement order is set in the map.json file.

### Map.json requirements:

The file consists of multiple commands and points' parameters:

- - **angle** - w and z coordinates are used for robot's turning on the point;
  - **id** – point number;
  - **pos** - x and y coordinates of the point in 2d space
  - **text\_start** – action set that is used by a robot before it moves to the next point (robot performs actions on “previous” point after it receives command to move to that point). Consists of several mandatory parameters:
    - **action** – action (see action description) that robot will perform when it will be at the point. You can only set one action of each type (for example, Navigation + Script , but not Script + Script). All actions are performed simultaneously;
    - **anchor** – activates a linguistic base anchor (see anchor description);
    - **animation** – facial expression that robot shows on it's screen (see facial expressions description);
    - **text** – phrase that the robot will say;
    - **url** – web page that the robot will open and show on it's screen.

**If you are not using a parameter you should set it to 0 (numeric) or null (text).**

- - **text\_finish** – action set that is used by a robot when it will arrive to the point. Parameters are the equal to “text\_start”.

File should be valid. You can check it at the third-party services like <https://jsonlint.com/>

File could be valid but have a wrong structure that won't work for the robot's software.

**Map.json** file should correspond to the following structure:

▼ Click to see Map.json structure....

```
\\начало файла, открываем фигурную скобку, в рамках которой один массив
points
{
  "points":
    \\ открываем массив точек с помощью квадратной скобки
    [
      \\ первый элемент в массиве точек, открываем описание элемента
      фигурной скобкой
      {
        "angle":
          \\ первый параметр, который состоит из нескольких значений,
          открываем фигурную скобку
          {
            "w": 0.376174405040294,
            "z": 0.92654887458384
```

```

        \\ закончились значения первого параметра, закрываем фигурную
скобку
        },
        "id": 0,
        "pos":
        \\ следующий параметр, который состоит из нескольких
значений, открываем фигурную скобку
        {
            "x": -0.906758110520606,
            "y": -5.17015431614186
        }
        \\ закончились значения параметра, закрываем фигурную скобку
        },
        "text_start": null,
        "text_finish":
        \\ данный параметр является массивом, так что открываем
квадратную скобку. внутри данного массива только один список значений, так
что внутри квадратных скобок вписываем этот список значений в фигурных
скобках
        [{
            "action": null,
            "anchor": 0,
            "animation": 0,
            "text": null,
            "url": null
        }]
        \\ заканчиваем описание первой точки. закрываем фигурную скобку
    },
    \\ аналогично описываем следующую точку
    {
        "angle":
        {
            "w": 0.429732129790555,
            "z": 0.902956420114323
        },
        "id": 1,
        "pos": {
            "x": -2.12233353289366,
            "y": -3.59353399042218
        },
        "text_start": null,
        "text_finish":
        \\ в данном случае в массиве есть несколько списков, каждый
из которых заключен в фигурные скобки и разделен запятыми
        [
            {
                "action": "navigation:2",
                "anchor": 0,
                "animation": 0,
                "text": null,
                "url": null
            },
            {
                \\ Еще один массив, теперь это несколько действий
                "action": "[lingvoCase:default],[script:get_hand_boy]",
                "anchor": 0,
                "animation": 0,
                "text": null,
            }
        ]
    }

```



```

        "url": null
      },
      {
        "action": "navigation:6",
        "anchor": 0,
        "animation": 0,
        "text": null,
        "url": null
      }
    ]
  }
}

```

\\ заканчиваем описание данного массива, состоящего из  
 нескольких списков, закрываем квадратную скобку  
 ]  
 \\ заканчиваем описание данной точки  
 }  
 \\ заканчиваем массив точек  
 ]  
 \\ заканчиваем файл  
 }

Problems that may occur during the automatic mode setup and their solutions:

Problem	Solution
Robot is not moving to any point	Check if map.json is valid
Robot is moving to the wrong place after it is sent to a certain point	Check the map.json's structure
Robot doesn't perform a script when it arrives to the point	Check the map.json's structure, try to change action's order if there are several different actions

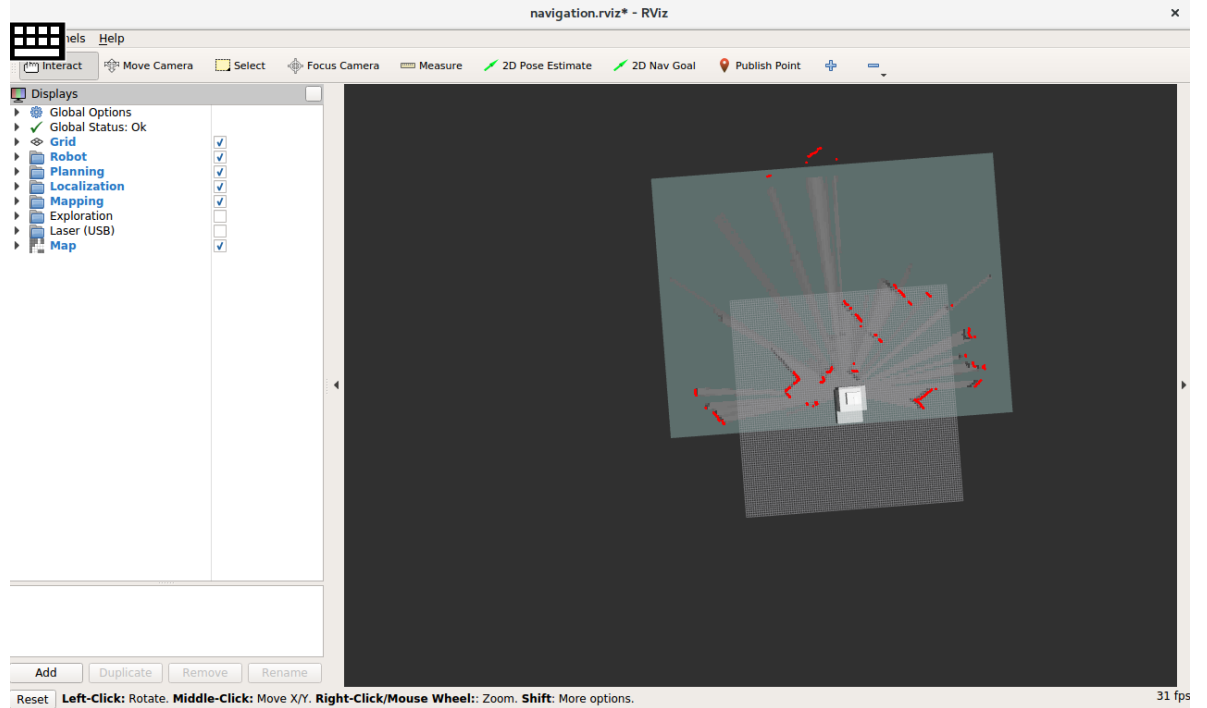
## 2. Manual method

- 
- Open **Rviz**.

```
roslaunch promobot_bringup rviz.launch
```

▼ Expected result...

Rviz has started.

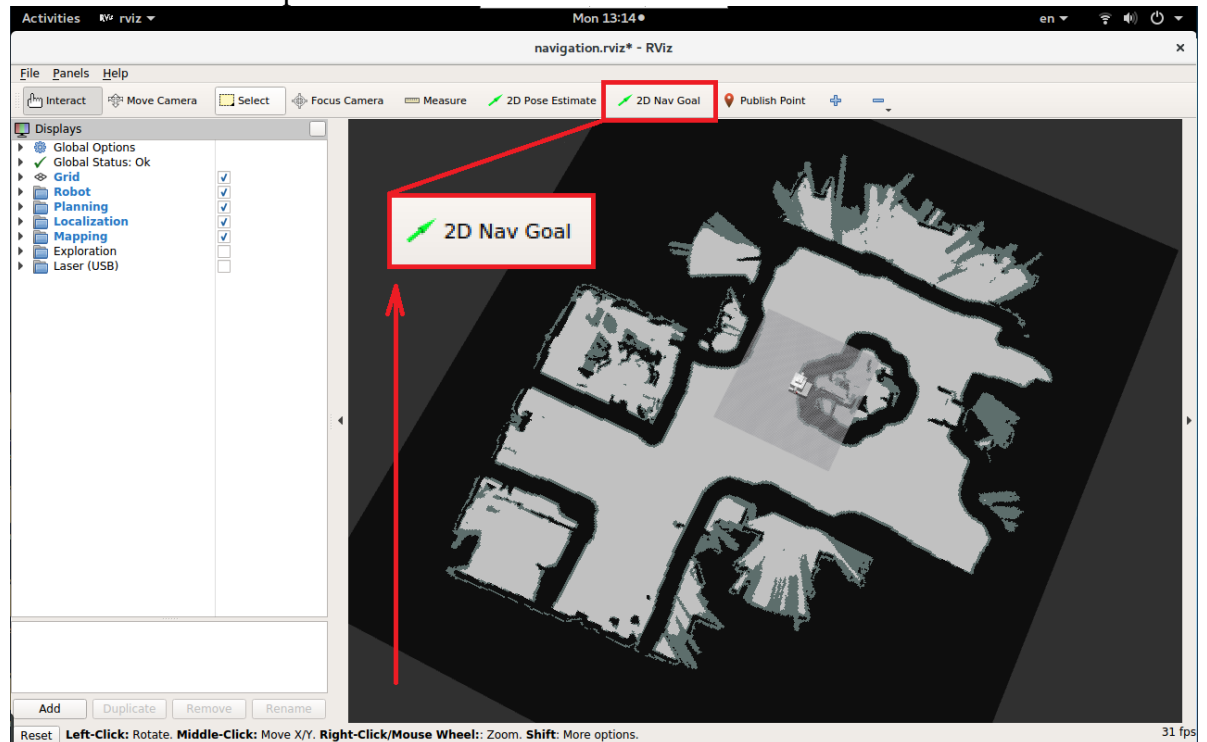


Open the topic in the terminal by calling this command:

```
/drive/destination rostopic echo /drive/destination
```

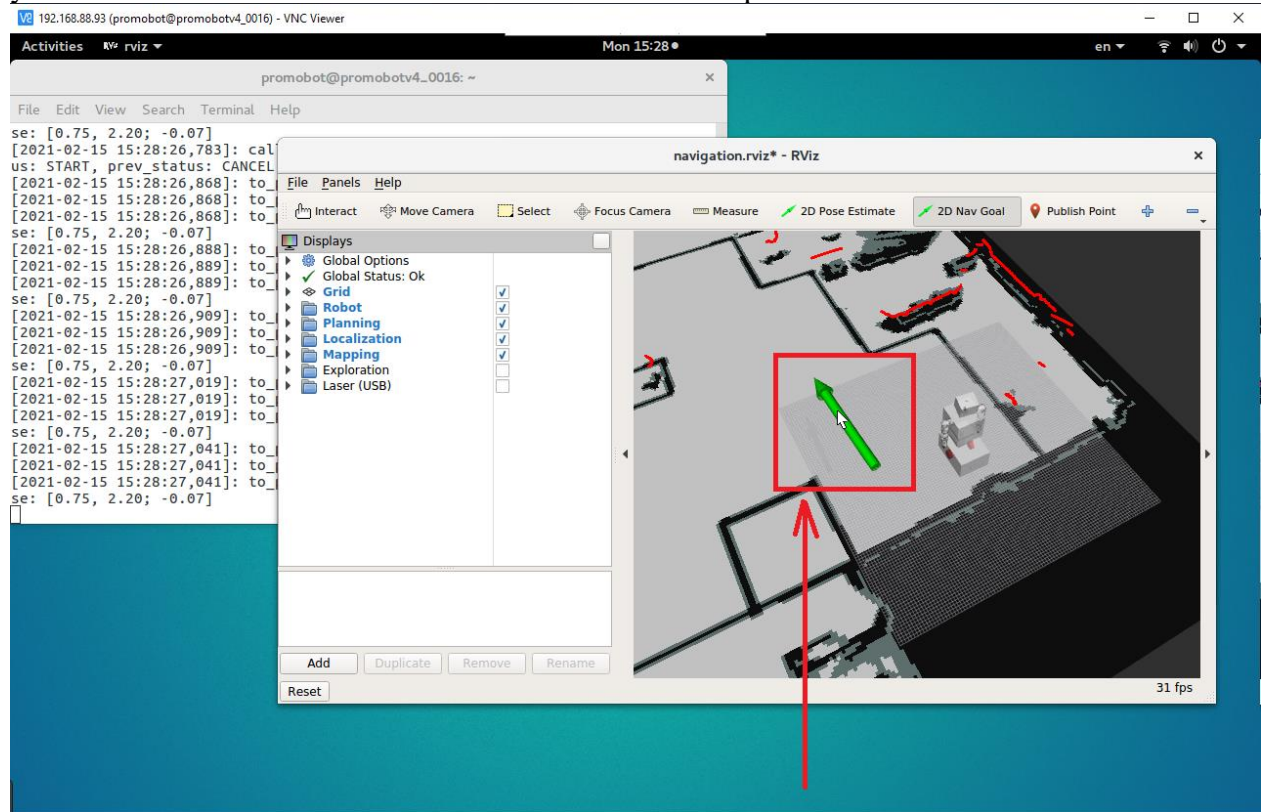
After executing this command, the terminal will NOT start outputting any lines.  
**DO NOT close this terminal window!**

- Go back to **Rviz** and push the **2D Nav Goal** button.



- Next, click and hold the mouse cursor on the map place where you want to set the point. You will see a green arrow appeared in the selected location. The beginning (blunt end) of the arrow will contain the coordinates of the point, and the direction of the arrow will indicate where the robot will look. Release the held mouse cursor when

you have chosen the desired direction of the robot at this point.



- After that, the terminal will display information about the selected point.

‣ More details...

Example for data from `/drive/destination` topic:

Only data structure will be the same, the values will be different.

```
header:
  seq: 0
  stamp:
    secs: 1523636206
    nsecs: 69873363
  frame_id: "map"
pose:
  position:
    x: -1.20477819443
    y: 0.55451887846
    z: 0.0
  orientation:
    x: 0.0
    y: 0.0
    z: 0.998724793097
    w: 0.0504855192539
---
```

- Write topic data to the file: `~/ .promobot/resources/points.json`

▾ More details...

Open file `~/ .promobot/resources/points.json`, scroll down to the end of file, place the mouse cursor **before** closing bracket ("]").

Insert the following block, substituting the required values instead of **%1**, **%2**, **%3**, **%4**, **%5**:

**Attention! Copy this entire block. The comma at the beginning of the line is not an error, it is needed.**

```
,{
  "angle": {
    "w": %1,
    "z": %2
  },
  "id": %3,
  "pos": {
    "x": %4,
    "y": %5
  }
}
```

The "angle" values (%1 and %2) in the `points.json` file correspond to the "orientation" values from the topic.

The "pos" values (%4 and %5) in the `points.json` file correspond to the "position" values from the topic.

▾ More details...

```

header:
  seq: 0
  stamp:
    secs: 1523636206
    nsecs: 69873363
  frame_id: "map"
pose:
  position:
    x: -1.20477819443
    y: 0.55451887846
    z: 0.0
  orientation:
    x: 0.0
    y: 0.0
    z: 0.998724793097
    w: 0.0504855192539

```

Diagram illustrating the mapping of pose data to JSON structure:

- `x: -1.20477819443` maps to `"x":`
- `y: 0.55451887846` maps to `"y":`
- `z: 0.0` maps to `"z":`
- `x: 0.0` maps to `"w":`
- `y: 0.0` maps to `"z":`
- `z: 0.998724793097` maps to `"w":`
- `w: 0.0504855192539` maps to `"z":`

The resulting JSON structure is:

```

{id": 11,
"pos": {
"x":
"y":
}
"angle": {
"w":
"z":
},

```

To find out the id of your new point, look at the id of the last point in the file, and use the number of the last + 1 for your new point. Thus, if you had 15 points in the file before manual addition (id of the last = 15), then for the added one, enter id 16.

▼ Click to see points.json example....

```

{
  "points": [{
    "angle": {
      "w": 0.998923881113163,
      "z": 0.0463797341714684
    },
    "id": 0,
    "pos": {
      "x": -0.0031530077629684,
      "y": 0.0249646781247797
    }
  }, {
    "angle": {
      "w": 0.995720365277429,
      "z": -0.0924172828630208
    },
    "id": 1,
    "pos": {
      "x": -11.9111618540054,

```

```
        "y": -2.47472925560041
    },
    {
        "angle": {
            "w": 0.995720365277429,
            "z": -0.0924172828630208
        },
        "id": 2,
        "pos": {
            "x": -11.9111618540054,
            "y": -2.47472925560041
        }
    },
    {
        "angle": {
            "w": -0.412846618222484,
            "z": 0.910800565339229
        },
        "id": 3,
        "pos": {
            "x": -3.92475407887507,
            "y": -3.54831148959944
        }
    },
    {
        "angle": {
            "w": 0.105501183751213,
            "z": 0.994419177322668
        },
        "id": 4,
        "pos": {
            "x": 4.23771759291041,
            "y": -1.27347722361125
        }
    },
    {
        "angle": {
            "w": 0.3274055621304,
            "z": 0.944883907094452
        },
        "id": 5,
        "pos": {
            "x": 9.61587872167513,
            "y": -3.06686012161428
        }
    },
    {
        "angle": {
            "w": 0.468120240741371,
            "z": 0.883664778186978
        },
        "id": 6,
        "pos": {
            "x": 8.88445106194085,
            "y": -9.08872129180928
        }
    }
}
```

```
    }
  }, {
    "angle": {
      "w": 0.566186994872896,
      "z": 0.824276826580003
    },
    "id": 7,
    "pos": {
      "x": 10.0251608731707,
      "y": -8.4145321788985
    }
  }, {
    "angle": {
      "w": 0.792600260640494,
      "z": 0.609741606611048
    },
    "id": 8,
    "pos": {
      "x": 11.3085748730512,
      "y": -13.4515874544963
    }
  }, {
    "angle": {
      "w": 0.891975396340006,
      "z": 0.452083943891054
    },
    "id": 9,
    "pos": {
      "x": 9.37746717975674,
      "y": -16.6945559421126
    }
  }, {
    "angle": {
      "w": 0.872674403743441,
      "z": 0.488302554827466
    },
    "id": 10,
    "pos": {
      "x": 5.9842699767641,
      "y": -21.3195678819722
    }
  }, {
    "angle": {
      "w": 0.965546256506352,
      "z": 0.260231486462669
    },
    "id": 11,
    "pos": {
      "x": 2.7900275872338,
      "y": -22.447542377038
    }
  }, {
    "angle": {
      "w": 0.99364052244197,
      "z": 0.112598899467309
    },
    "id": 12,
```



```

        "pos": {
            "x": -3.40365830335644,
            "y": -25.2423115222093
        }
    }, {
        "angle": {
            "w": 0.925848332497999,
            "z": -0.377895309855354
        },
        "id": 13,
        "pos": {
            "x": -8.97703753802622,
            "y": -17.7053637622499
        }
    }, {
        "angle": {
            "w": 0.6720507695428,
            "z": -0.740505073012285
        },
        "id": 14,
        "pos": {
            "x": -10.2697658555054,
            "y": -14.1395930708606
        }
    }, {
        "angle": {
            "w": 0.6720507695428,
            "z": -0.740505073012285
        },
        "id": 15,
        "pos": {
            "x": -10.2697658555054,
            "y": -14.1395930708606
        }
    }
]
}

```

## Initial position

The initial position is set by the parameters in the database. This is necessary if, after building the map, the position of the charging station has been changed.

```

/navigation/initial_pose/position/x
/navigation/initial_pose/position/y
/navigation/initial_pose/orientation/w
/navigation/initial_pose/orientation/z

```

To find the coordinates of a point, you can use the first 5 items in the section "**Manual mode**" in the section **Point positioning**.

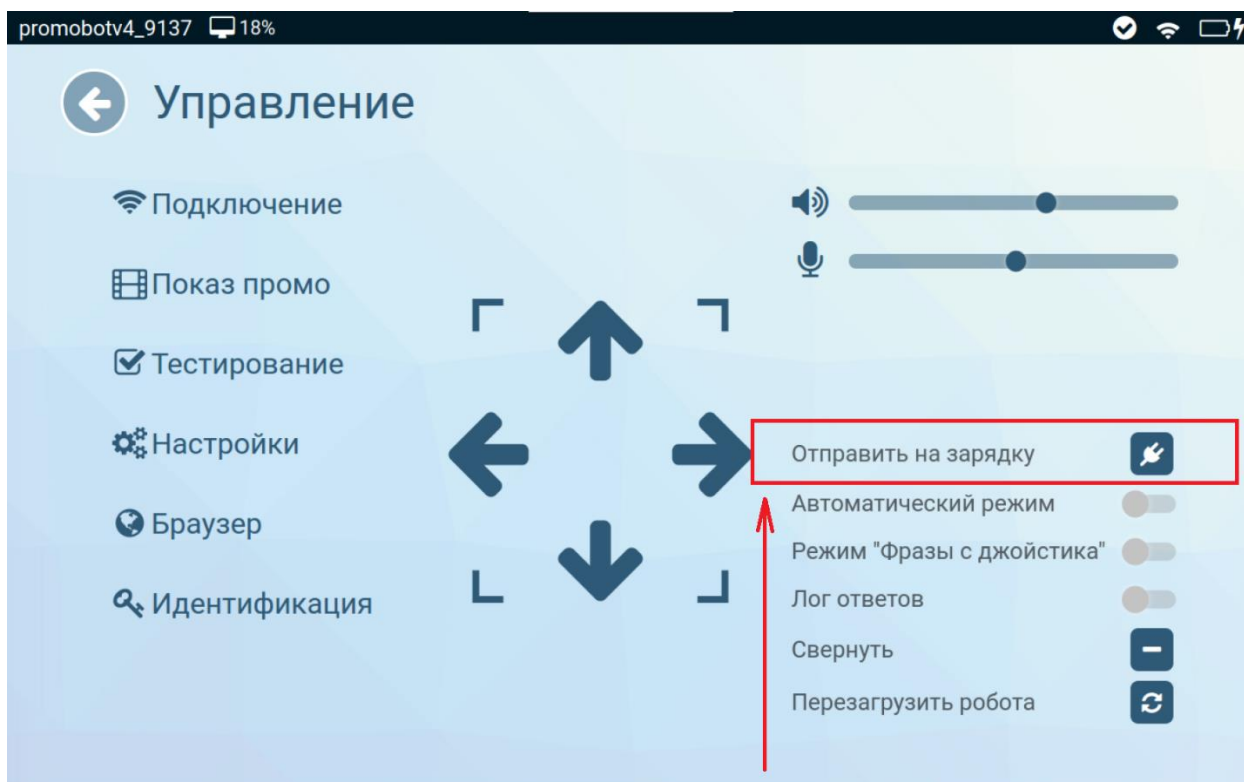
**Charger should be in the starting (zero) point.** If you want robot to move to the charger set `/driving/skipStationScanning` to **true**.

For robots with software version 1.9.3 and higher, set the parameter to **false**.

## Finish

After building a map and setting points, you need to check that the robot is driving correctly over these points. To do so you need to give a command to the robot to drive to point ([instructions](#)). After checking driving to points you need to give a command to the robot to drive to the charging station (from service menu) to make sure that the robot dock to the charging station correctly.

▼ How to give a command to the robot to drive to the charging station?



## Recommendations

- - During the mapping you better move robot slow along the perimeter of the room, the cartographer will combine all the sub-maps into one.

- It is better to start building a map from the zero point (that is, from the place of charging) and finish at the same point.

## **Typical cartography problems**

[Typical cartography problems](#)

## **Additional links**

[Checking the arranged Points](#)

[cartographer](#)

[cartographer\\_ros](#)