

Họ và tên: Đỗ Đức Mạnh

Mã sinh viên: 20020688

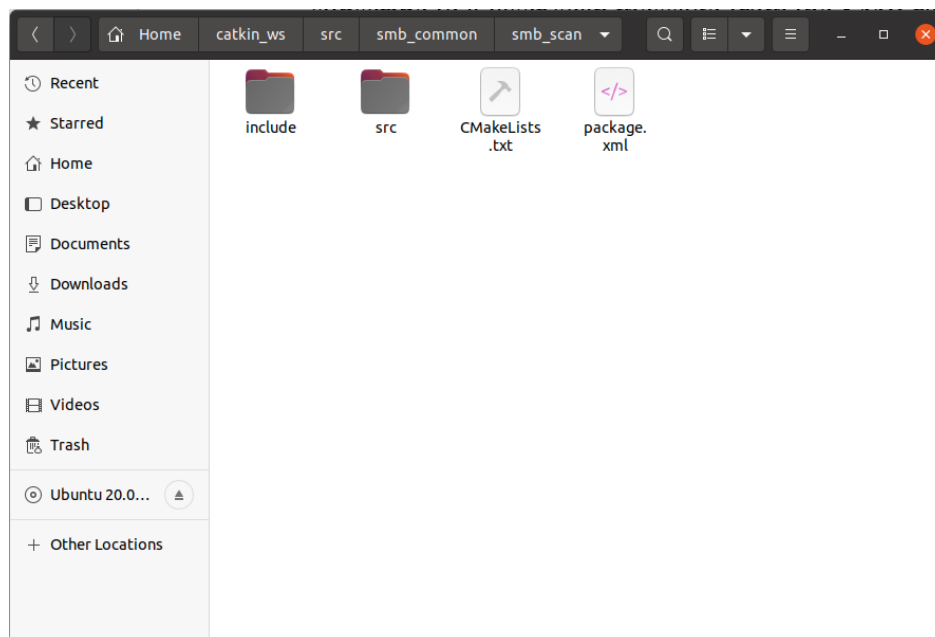
Báo cáo thực hành tuần 2

Môn học: Lập trình ROS

Bài 1: Yêu cầu: Tạo 1 package để làm việc phụ thuộc roscpp , rospy và sensor_msgs.

Tạo package “smb_scan” tại “smb_common” với dòng lệnh:

```
~$ catkin_create_pkg smb_scan roscpp rospy
```



Hình 1. Package “smb_scan”

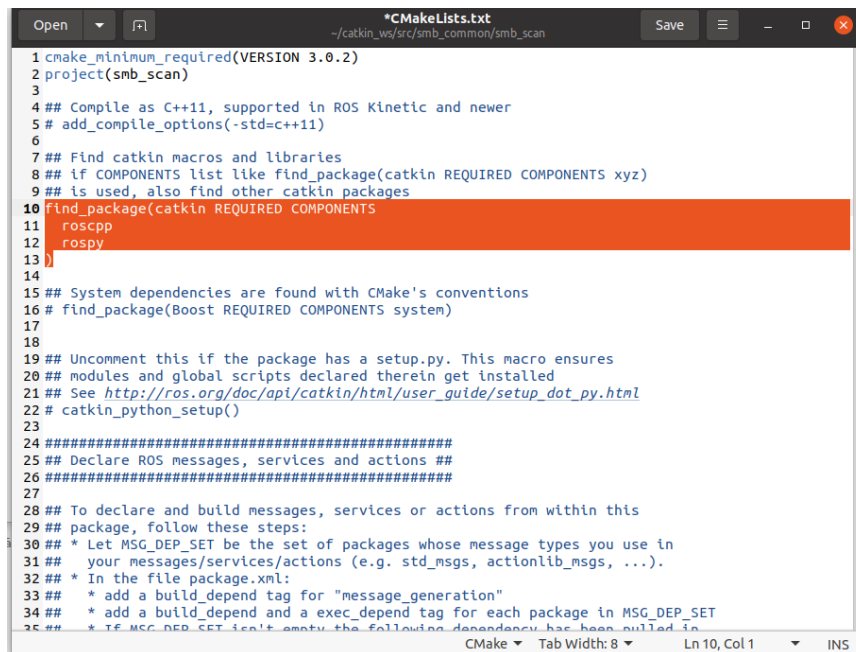
Tại đây ta sẽ thêm các file cpp, py tại src và chỉnh sửa, gọi các file tại CMakeLists.txt.

Bài 2: (Tùy chọn) Yêu cầu: Tải file smb_highlevel_controller để sử dụng
Ở bài báo cáo này, tôi không sử dụng file trên.

Bài 3: Yêu cầu: xem file CMakeList.txt và package.xml.

Tại file CMakeList.txt cần chú ý đến 2 phần.

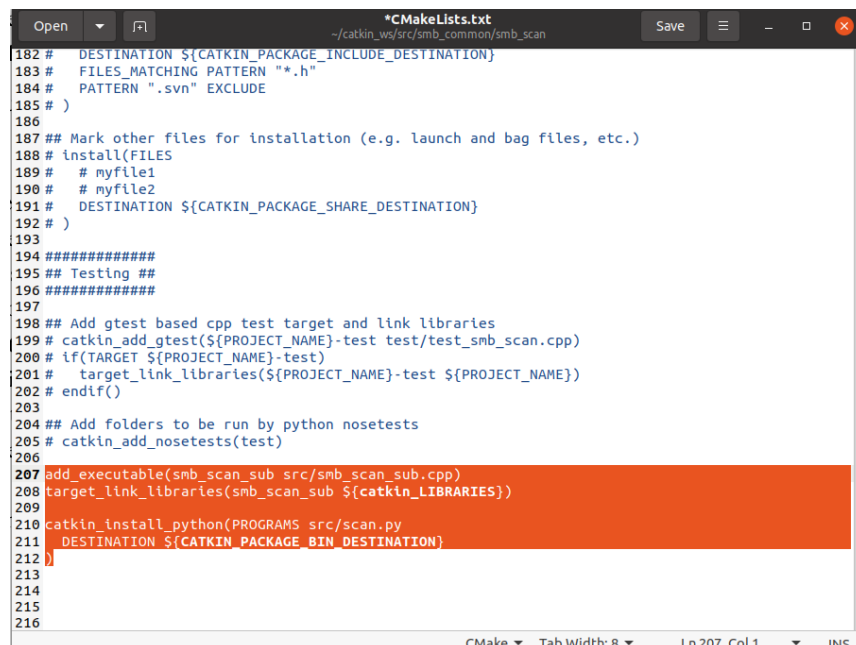
Phần 1: Nơi các gói cần có để có thể thao tác trong package như roscpp, rospy, msg, ... Ta sẽ thêm bớt tại `find_package()`.



```
1 cmake_minimum_required(VERSION 3.0.2)
2 project(smb_scan)
3
4 ## Compile as C++11, supported in ROS Kinetic and newer
5 # add_compile_options(-std=c++11)
6
7 ## Find catkin macros and libraries
8 ## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
9 ## is used, also find other catkin packages
10 find_package(catkin REQUIRED COMPONENTS
11   roscpp
12   rospy
13 )
14
15 ## System dependencies are found with CMake's conventions
16 # find_package(Boost REQUIRED COMPONENTS system)
17
18
19 ## Uncomment this if the package has a setup.py. This macro ensures
20 ## modules and global scripts declared therein get installed
21 ## See http://ros.org/doc/api/catkin/html/user\_guide/setup\_dot\_py.html
22 # catkin_python_setup()
23
24 #####
25 ## Declare ROS messages, services and actions ##
26 #####
27
28 ## To declare and build messages, services or actions from within this
29 ## package, follow these steps:
30 ## * Let MSG_DEP_SET be the set of packages whose message types you use in
31 ##   your messages/services/actions (e.g. std_msgs, actionlib_msgs, ...).
32 ## * In the file package.xml:
33 ##   * add a build_depend tag for "message_generation"
34 ##   * add a build_depend and a exec_depend tag for each package in MSG_DEP_SET
35 ##   * If MSG_DEP_SET isn't empty, the following dependency has been pulled in
```

Hình 2. Nơi gọi các gói cần thiết trong *CmakeLists.txt*

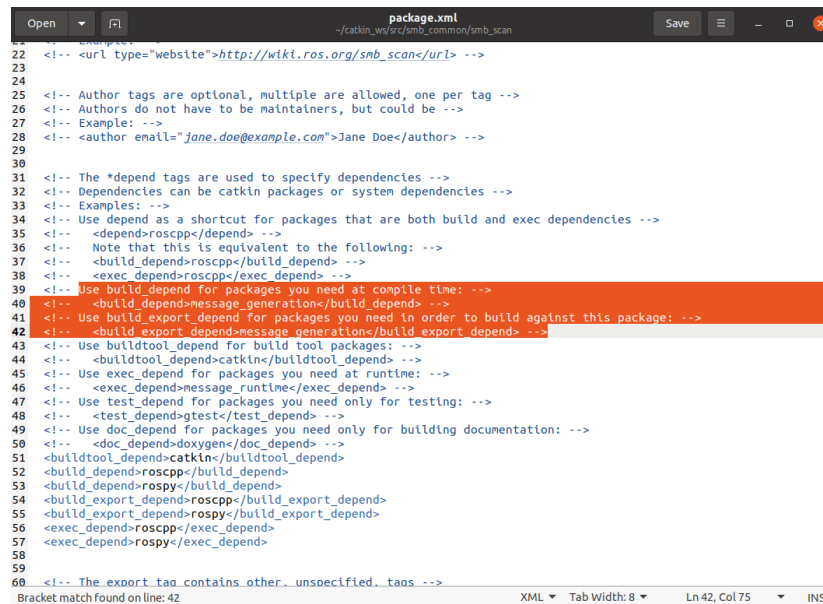
Phần 2: Gọi các file trong package qua đường dẫn. Trong hình là cách gọi file roscpp và rospy theo thứ tự lần lượt.



```
182 # DESTINATION ${CATKIN_PACKAGE_INCLUDE_DESTINATION}
183 # FILES_MATCHING PATTERN "*.h"
184 # PATTERN ".svn" EXCLUDE
185 )
186
187 ## Mark other files for installation (e.g. launch and bag files, etc.)
188 # install(FILES
189 #   myfile1
190 #   myfile2
191 #   DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}
192 # )
193
194 #####
195 ## Testing ##
196 #####
197
198 ## Add gtest based cpp test target and link libraries
199 # catkin_add_gtest(${PROJECT_NAME}-test test/test_smb_scan.cpp)
200 # if(TARGET ${PROJECT_NAME}-test)
201 #   target_link_libraries(${PROJECT_NAME}-test ${PROJECT_NAME})
202 # endif()
203
204 ## Add folders to be run by python nosetests
205 # catkin_add_nosetests(test)
206
207 add_executable(smb_scan_sub src/smb_scan_sub.cpp)
208 target_link_libraries(smb_scan_sub ${catkin_LIBRARIES})
209
210 catkin_install_python(PROGRAMS src/scan.py
211   DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION})
212 )
213
214
215
216
```

Hình 3. Gọi các file roscpp, rospy qua đường dẫn của chúng

Trong file package.xml cần chú ý đến phần thêm bớt các msg bất kì.

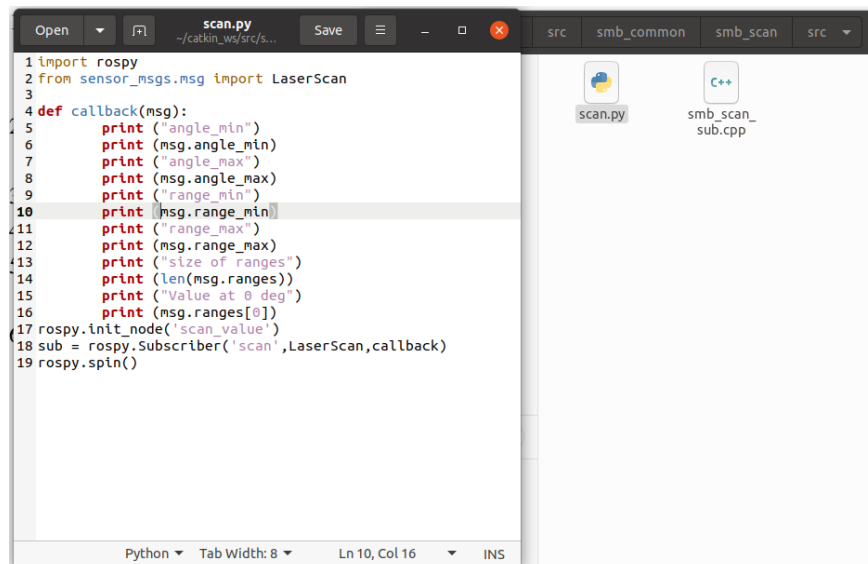


Hình 4. File package.xml

Bài 4: Yêu cầu: tạo 1 file đọc các giá trị của laser tại topic /scan.

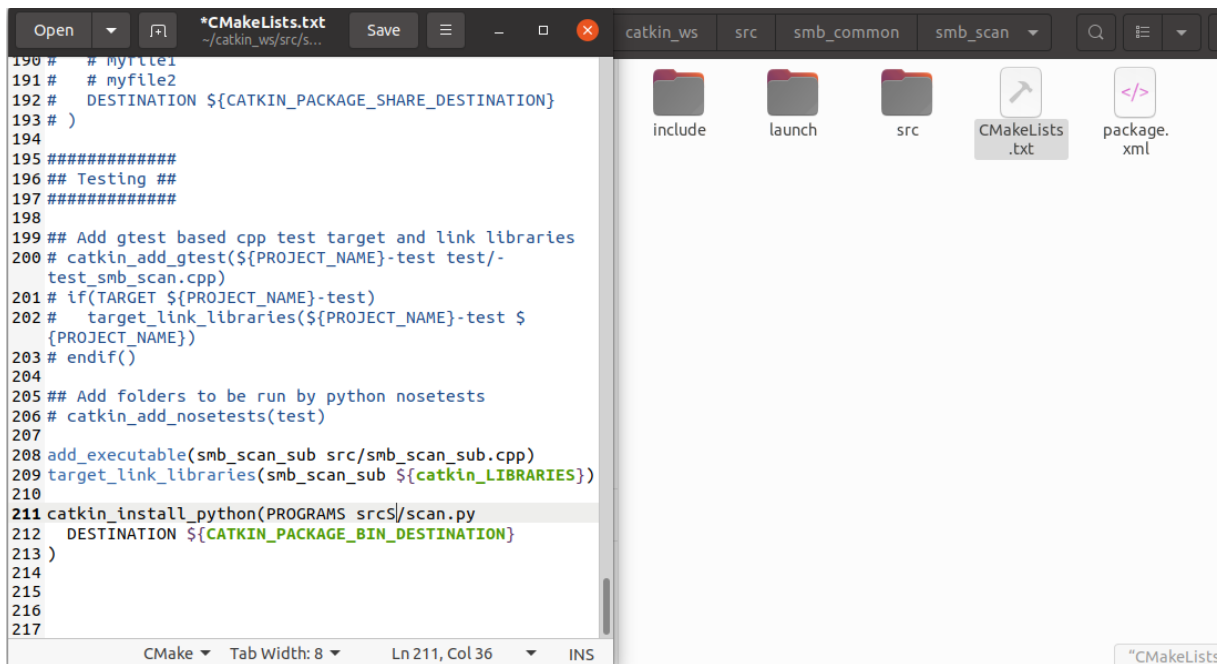
Cách 1: Sử dụng bằng file rospy.

B1: Tạo file python scan.py để nhận các giá trị từ topic /scan và lưu vào /smb_scan/src. Ta sẽ đọc được các giá trị giới hạn góc quét và khoảng cách, một mảng lưu khoảng cách thu được từ Lidar có 362 phần tử.

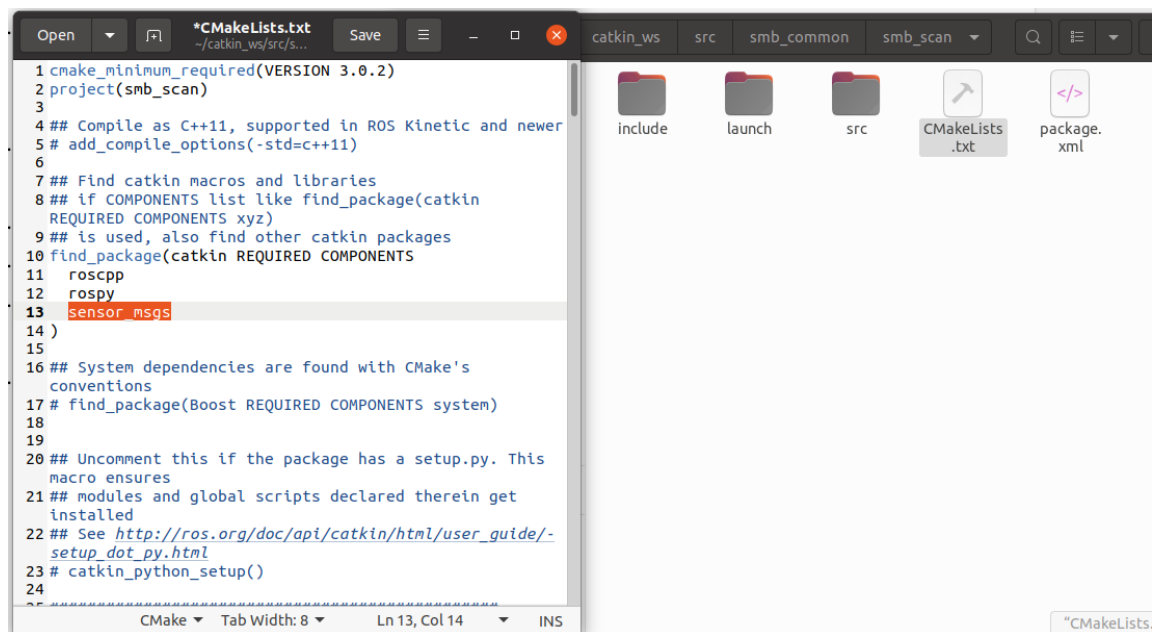


Hình 5. File scan.py

B2: Thêm vào file CMakeLists.txt để thực hiện chương trình python từ B1 và gọi thêm msg sensor_msgs.



Hình 6. Thêm các dòng gọi file trong CmakeLists.txt



Hình 7. Gọi file msg để đọc giá trị từ LiDAR

B3: Chạy file smb_gazebo.launch và file scan.py tạo từ B1:

```

manh@manh: ~/catkin_ws
[ INFO] [1677140874.083090568, 0.337000000]: Publishing to tf is enabled
[ INFO] [1677140874.133620731, 0.349000000]: left wheel to origin: 0.2159, 0.285, -0.0185
[ INFO] [1677140874.133691465, 0.349000000]: right wheel to origin: 0.2159, -0.285, -0.0185
[ INFO] [1677140874.133778160, 0.349000000]: Odometry parameters : wheel separation 0.57, left wheel radius 0.19, right wheel radius 0.19
[ INFO] [1677140874.152672377, 0.353000000]: Adding left wheel with joint name: LF_WHEEL_JOINT and right wheel with joint name: RF_WHEEL_JOINT
[ INFO] [1677140874.159254201, 0.355000000]: Adding left wheel with joint name: LH_WHEEL_JOINT and right wheel with joint name: RH_WHEEL_JOINT
[ INFO] [1677140874.238721907, 0.386000000]: Dynamic Reconfigure:
DynamicParams:
  Odometry parameters:
    left wheel radius multiplier: 1
    right wheel radius multiplier: 1
    wheel separation multiplier: 1
  Publication parameters:
    Publish executed velocity command: disabled
    Publication rate: 50
    Publish frame odom on tf: enabled
[Wrn] [Publisher.cc:135] Queue limit reached for topic /gazebo/default/user_camera/pose, deleting message. This warning is printed only once.
Value at 0 deg
inf
Value at 90 deg
inf
Value at 180 deg
inf
angle_min
-1.5707999467849731
angle_max
1.5707999467849731
range_min
0.44999998807907104
range_max
50.0
size of ranges
362
Value at 0 deg
inf
Value at 90 deg
inf
Value at 180 deg
inf
angle_min
-1.5707999467849731
angle_max
1.5707999467849731
range_min
0.44999998807907104
range_max
50.0
size of ranges

```

Hình 8. Các giá trị thu được từ topic /scan

Cách 2: Sử dụng file roscpp.

Thao tác tương tự như cách 2. Cần lưu ý khi chạy file roscpp sử dụng câu lệnh: `~$ rosruncatkin_ws/src/smb_common/smb_scan_sub`.

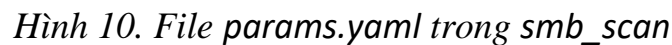
```

smb_scan_sub.cpp
~/catkin_ws/src/smb_common/smb_scan/src
Save
1 #include "ros/ros.h"
2 #include "sensor_msgs/LaserScan.h"
3
4
5 void chatterCallback(const sensor_msgs::LaserScan::ConstPtr& msg)
6 {
7     ROS_INFO("range_min: [%f]", msg->range_min);
8     ROS_INFO("range_max: [%f]", msg->range_max);
9     ROS_INFO("angle_min: [%f]", msg->angle_min);
10    ROS_INFO("angle_max: [%f]", msg->angle_max);
11    ROS_INFO("ranges: [%ld]", sizeof(msg->ranges)/sizeof(msg->ranges[0]));
12 }
13
14 int main(int argc, char **argv)
15 {
16
17     ros::init(argc, argv, "smb_scan_sub");
18
19     ros::NodeHandle n;
20
21     ros::Subscriber sub = n.subscribe("scan", 1000, chatterCallback);
22
23     ros::spin();
24
25     return 0;
26 }
27
28
Bracket match found on line: 15
C++ Tab Width: 8 Ln 27, Col 2 INS

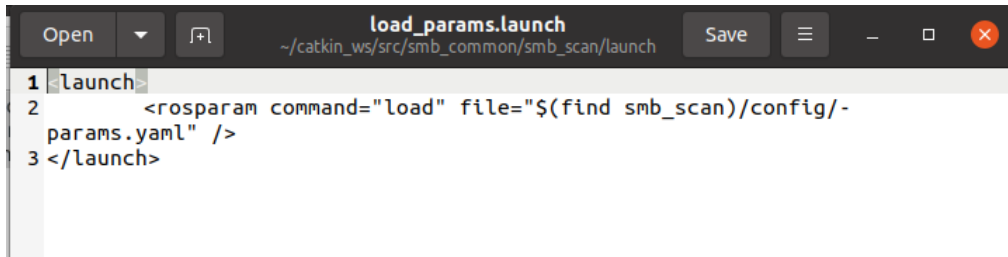
```

Hình 9. File smb_scan_sub.cpp

B1: Tạo package “config” chứa file “params.yaml”

[illegible]

B3: Tạo file launch “load_params.launch” để gọi các thông số trên.



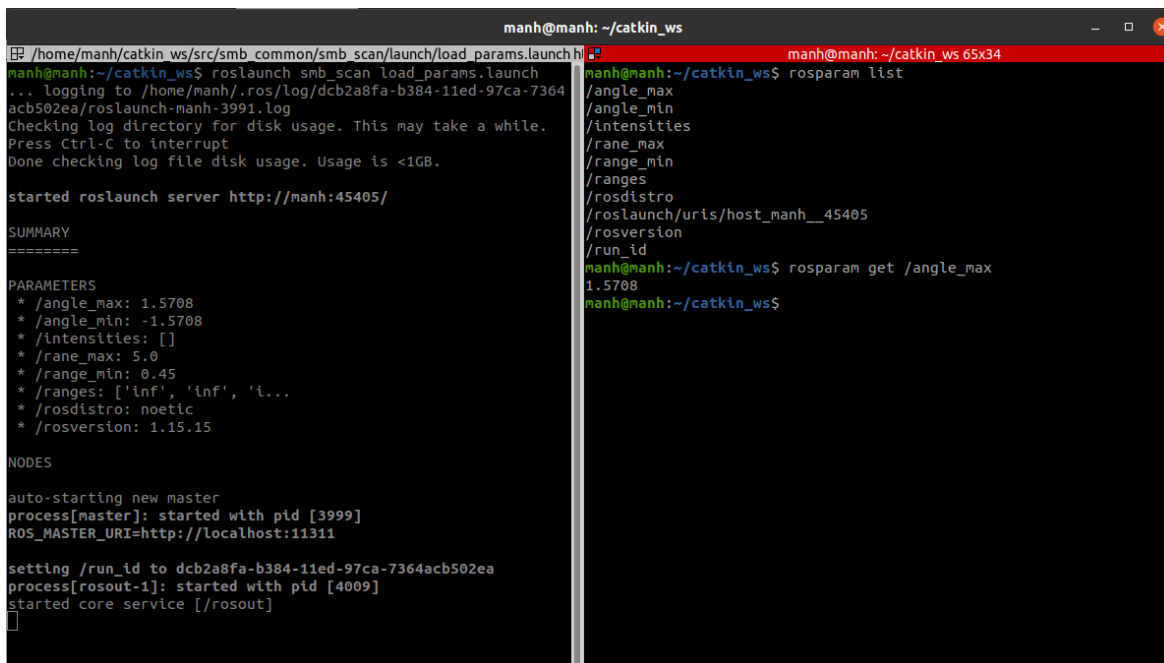
```
1 launch
2   <rosparam command="load" file="${find smb_scan}/config/-
  params.yaml" />
3 </launch>
```

Hình 12. File `load_params.launch`

B4: Chạy file “`load_params.launch`” và gọi các thông số bằng “`rosparam`”.

Gọi các thông số qua lệnh `~$ rosparam get /param_name`

Đặt các thông số qua lệnh `~$ rosparam set /param_name value`



```
manh@manh: ~/catkin_ws
manh@manh:~/catkin_ws$ roslaunch smb_scan load_params.launch
... logging to /home/manh/.ros/log/dcb2a8fa-b384-11ed-97ca-7364
acb502ea/roslaunch-manh-3991.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://manh:45405/

SUMMARY
=====
PARAMETERS
* /angle_max: 1.5708
* /angle_min: -1.5708
* /intensities: []
* /range_max: 5.0
* /range_min: 0.45
* /ranges: ['inf', 'inf', 'i...
* /roscdistro: noetic
* /rosversion: 1.15.15

NODES
auto-starting new master
process[roscdistro]: started with pid [3999]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to dcb2a8fa-b384-11ed-97ca-7364acb502ea
process[roscdistro-1]: started with pid [4009]
started core service [/roscdistro]
^C
```

```
manh@manh:~/catkin_ws$ rosparam list
/angle_max
/angle_min
/intensities
/range_max
/range_min
/ranges
/roscdistro
/rosdistro
/roslaunch/uris/host_manh__45405
/rosversion
/run_id
manh@manh:~/catkin_ws$ rosparam get /angle_max
1.5708
manh@manh:~/catkin_ws$
```

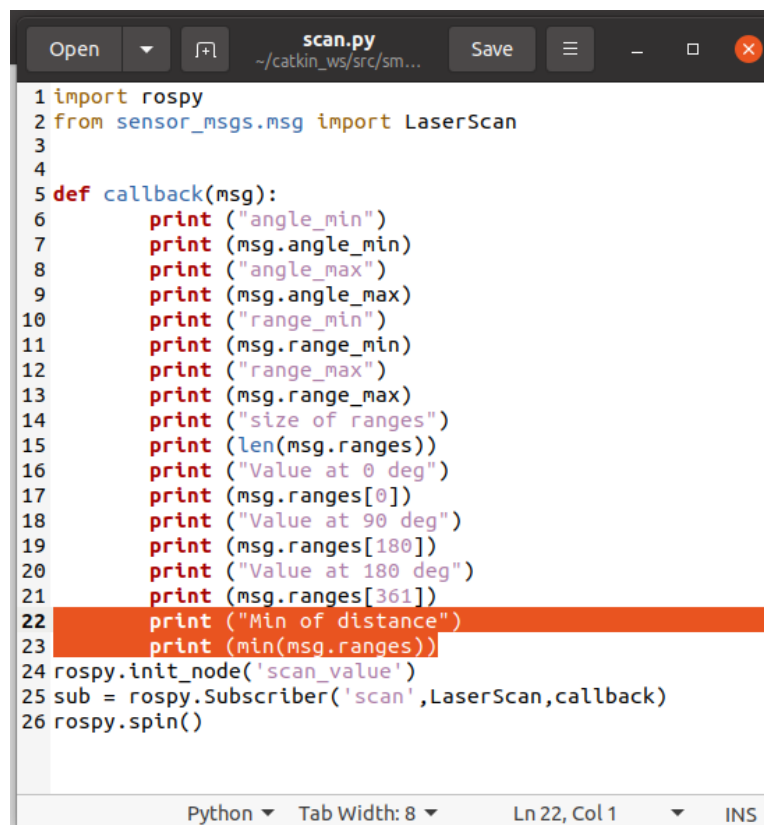
Hình 13. Các câu lệnh và kết quả khi gọi lên các thông số

Trước khi làm từ bài 6 – 11, các môi trường trong file `smb_gazebo` (`big_map_summer_school.world`, `planner_tutorial.world`) không thể khởi tạo do tôi dùng máy ảo để chạy ubuntu nên tôi cần tạo một môi trường khác nhẹ hơn tại mục Note trong phần cuối bài báo cáo này.

Bài 6: Yêu cầu: Viết hàm tìm ra khoảng cách nhỏ nhất laser đo được.

Cách 1: Dùng `rospy`.

B1: Tại file `scan.py` dùng hàm `min()` trong python để tìm ra giá trị nhỏ nhất trong tập `ranges[]`.



```
1 import rospy
2 from sensor_msgs.msg import LaserScan
3
4
5 def callback(msg):
6     print ("angle_min")
7     print (msg.angle_min)
8     print ("angle_max")
9     print (msg.angle_max)
10    print ("range_min")
11    print (msg.range_min)
12    print ("range_max")
13    print (msg.range_max)
14    print ("size of ranges")
15    print (len(msg.ranges))
16    print ("Value at 0 deg")
17    print (msg.ranges[0])
18    print ("Value at 90 deg")
19    print (msg.ranges[180])
20    print ("Value at 180 deg")
21    print (msg.ranges[361])
22    print ("Min of distance")
23    print (min(msg.ranges))
24 rospy.init_node('scan_value')
25 sub = rospy.Subscriber('scan', LaserScan, callback)
26 rospy.spin()
```

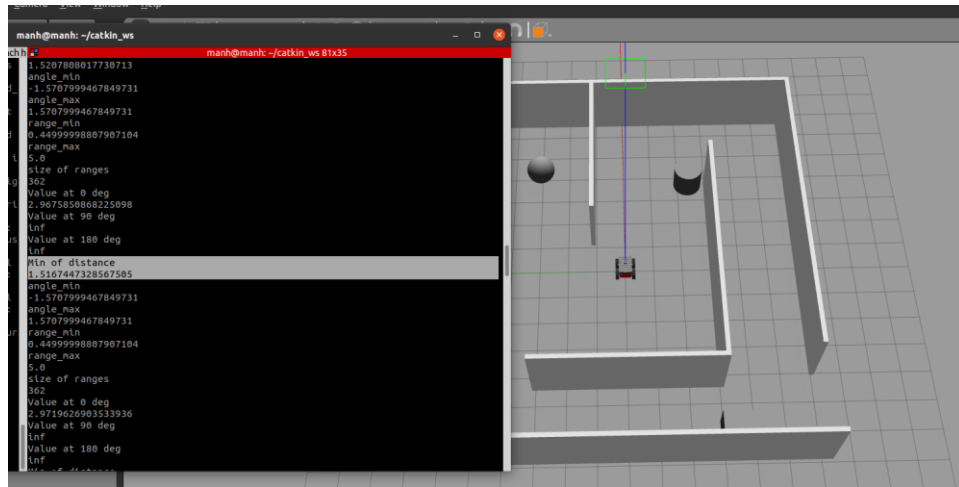
Hình 14. File `scan.py` tìm khoảng cách nhỏ nhất thu được từ laser

B2: Chạy file `smb_gazebo.launch` để khởi tạo môi trường và robot. Sau đó chạy file `scan.py` để in ra khoảng cách nhỏ nhất thu được bằng câu lệnh:

```
~$ rosrn smb_scan scan.py
```

Ta có thể kiểm tra lại bằng cách xem tập giá trị `ranges[]` bằng câu lệnh:

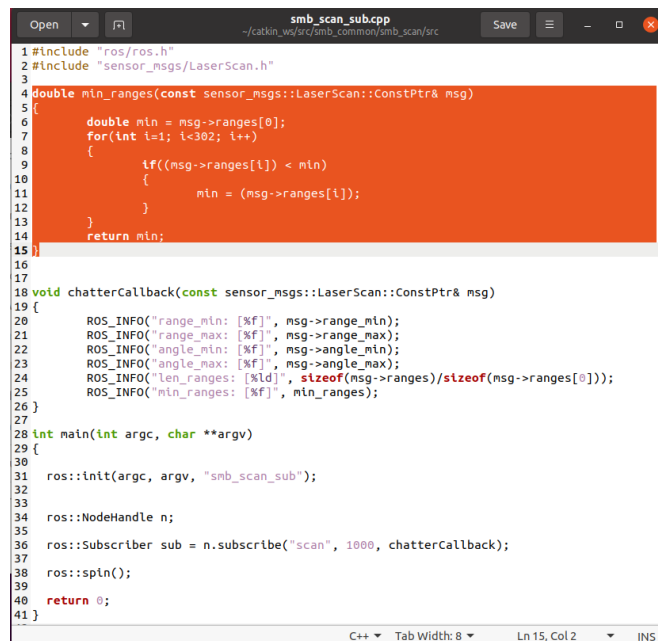
~\$ rostopic echo /scan



Hình 15. Khoảng cách nhỏ nhất thu được từ laser

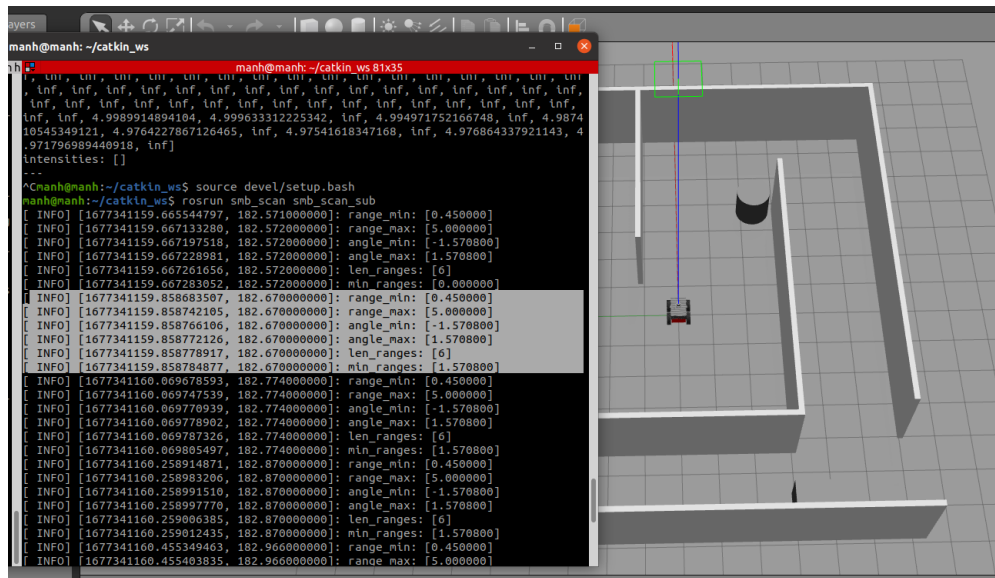
Cách 2: Dùng roscpp

B1: Tại file smb_scan_sub.cpp tạo hàm min_ranges() để tìm ra giá trị nhỏ nhất trong tập ranges[]. Tập ranges[] trong thư viện sensor_msgs/LaserScan có 302 phần tử, để tìm giá trị nhỏ nhất ta sẽ gán biến min là phần tử đầu tiên, ta sẽ kiểm tra từng phần tử trong ranges[] với biến min, nếu phần tử nào nhỏ hơn min thì gán min bằng phần tử đó. Ta sẽ tìm được giá trị nhỏ nhất trong tập ranges[].



Hình 16. Hàm tìm khoảng cách nhỏ nhất của laser trong smb_scan_sub.cpp

B2: Chạy file `smb_gazebo.launch` để khởi tạo môi trường và robot. Sau đó chạy file `smb_scan_sub.cpp` để hiển thị khoảng cách nhỏ nhất laser đo được qua câu lệnh: `~$ rosrund smb_scan smb_scan_sub`



Hình 17. Khoảng cách nhỏ nhất thu được từ laser

Bài 7: Yêu cầu: Dùng file launch từ tuần 1 để gọi thêm các thông số từ *Bài 5*.

B1: Tôi sẽ chỉnh sửa tại file `smb_gazebo.launch` để chỉ qua một câu lệnh sẽ gọi lên môi trường, robot, các thông số (parameters) và điều khiển robot bằng bàn phím (teleop_key). File `smb_gazebo.launch` được thêm như *Hình 18*.

Các thông số sẽ được lấy từ file `params.yaml` như *Bài 5*.

```

70 <remap from="scan" to="/scan"/>
71 <rparam>
72   target_frame: rslidar # Leave disabled to output scan in pointcloud frame
73   transform_tolerance: 0.01
74   min_height: $(arg laser_scan_min_height)
75   max_height: $(arg laser_scan_max_height)
76
77   angle_min: -1.5708 # -M_PI/2
78   angle_max: 1.5708 # M_PI/2
79   angle_increment: 0.0087 # M_PI/360.0
80   scan_time: 0.03333
81   range_min: 0.45
82   range_max: 5.0
83   use_inf: true
84   inf_epsilon: 1.0
85
86   # Concurrency level, affects number of pointclouds queued for processing
87   # 0 : Detect number of cores
88   # 1 : Single threaded
89   # 2->Inf : Parallelism level
90   concurrency_level: 1
91 </rparam>
92 </node>
93
94 <include file="$(find smb_control)/launch/control.launch">
95   <arg name="simulation" default="true"/>
96   <arg name="robot_namespace" default="$(arg robot_namespace)"/>
97   <arg name="robot_description" default="$(arg robot_description)"/>
98   <arg name="enable_ekf" default="$(arg enable_ekf)" />
99 </include>
100
101 <roslaunch command="load" file="$(find smb_scan)/config/params.yaml" />
102
103 <node pkg="smb_teleop" type="smb_teleop_key" name="smb_teleop_keyboard"
104   output="screen">
105
106 </launch>

```

Hình 18. File `smb_gazebo.launch`

B2: Sau khi chạy file `smb_gazebo.launch`, terminal sẽ hiển thị các thông số (PARAMETERS) như hình 19 được sắp xếp theo bảng chữ cái. Ta có thể thấy thông số giới hạn góc quét và khoảng cách của laser từ file `params.yaml`.

```

PARAMETERS
* /angle_max: 1.5708
* /angle_min: -1.5708
* /ekf_localization/base_link_frame: base_link
* /ekf_localization/frequency: 50
* /ekf_localization/imu0: imu/data
* /ekf_localization/imu0_config: [false, false, Fa...
* /ekf_localization/imu0_differential: true
* /ekf_localization/imu0_queue_size: 10
* /ekf_localization/imu0_remove_gravitational_acceleration: true
* /ekf_localization/odom0: smb_velocity_cont...
* /ekf_localization/odom0_config: [false, false, Fa...
* /ekf_localization/odom0_differential: false
* /ekf_localization/odom0_queue_size: 100
* /ekf_localization/odom_frame: odom
* /ekf_localization/two_d_mode: true
* /ekf_localization/world_frame: odom
* /gazebo/enable_ros_network: true
* /gazebo_ros_control/angular/z/has_acceleration_limits: true
* /gazebo_ros_control/angular/z/has_velocity_limits: true
* /gazebo_ros_control/angular/z/max_acceleration: 3.0
* /gazebo_ros_control/angular/z/max_velocity: 1.1
* /gazebo_ros_control/base_frame_id: base_link
* /gazebo_ros_control/enable_odom_tf: false
* /gazebo_ros_control/estimate_velocity_from_position: false
* /gazebo_ros_control/linear/x/has_acceleration_limits: true
* /gazebo_ros_control/linear/x/has_velocity_limits: true
* /gazebo_ros_control/linear/x/max_acceleration: 2.0
* /gazebo_ros_control/linear/x/max_velocity: 1.1
* /gazebo_ros_control/pid_gains/LF_WHEEL_JOINT/p: 100.0
* /gazebo_ros_control/pid_gains/LH_WHEEL_JOINT/p: 100.0
* /gazebo_ros_control/pid_gains/RH_WHEEL_JOINT/p: 100.0
* /gazebo_ros_control/pid_gains/RH_WHEEL_JOINT/p: 100.0
* /gazebo_ros_control/wheel_radius_multiplier: 1.0
* /gazebo_ros_control/wheel_separation_multiplier: 1.875
* /intensities: []
* /pointcloud_to_laserscan/angle_increment: 0.0087
* /pointcloud_to_laserscan/angle_max: 1.5708
* /pointcloud_to_laserscan/angle_min: -1.5708
* /pointcloud_to_laserscan/concurrency_level: 1
* /pointcloud_to_laserscan/inf_epsilon: 1.0
* /pointcloud_to_laserscan/max_height: $(arg laser_scan_...
* /pointcloud_to_laserscan/min_height: $(arg laser_scan_...
* /pointcloud_to_laserscan/range_max: 5.0
* /pointcloud_to_laserscan/range_min: 0.45
* /pointcloud_to_laserscan/scan_time: 0.03333
* /pointcloud_to_laserscan/target_frame: rslidar
* /pointcloud_to_laserscan/transform_tolerance: 0.01
* /pointcloud_to_laserscan/use_inf: true
* /range_max: 5.0
* /range_min: 0.45
* /ranges: ['inf', 'inf', 'i...
* /robot_description: <?xml version="1"...
* /roslint: none
* /rosversion: 1.15.15
* /smb_joint_publisher/publish_rate: 50
* /smb_joint_publisher/type: joint_state_contr...
* /smb_robot_state_publisher/publish_frequency: 50
* /smb_robot_state_publisher/use_tf_static: true
* /smb_velocity_controller/cnd_vel_timeout: 0.25
* /smb_velocity_controller/left_wheel: ['LF_WHEEL_JOINT'...
* /smb_velocity_controller/pose_covariance_diagonal: [0.1, 0.001, 0.00...
* /smb_velocity_controller/publish_rate: 50
* /smb_velocity_controller/right_wheel: ['RH_WHEEL_JOINT'...
* /smb_velocity_controller/twist_covariance_diagonal: [0.001, 0.001, 0.0...
* /smb_velocity_controller/type: diff_drive_contro...
* /smb_velocity_controller/velocity_rolling_window_size: 2
* /twist_mux/locks: [{'name': 'e_stop'...
* /twist_mux/topics: [{'name': 'joy', ...
* /use_sim_time: true


NODES

```

Hình 19. Các thông số hiển thị trên terminal

Bài 8: Yêu cầu: Chỉnh trạng thái của “laser_enabled” trong file “smb_gazebo.launch”.

Ta mở file smb_gazebo.launch và tìm đến dòng 18 để chỉnh sửa như *Hình 20*.

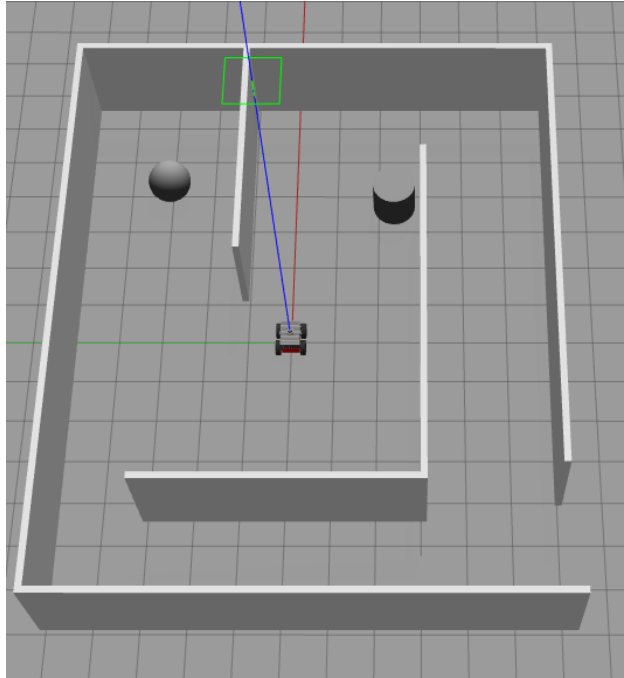


```
1 <?xml version="1.0" encoding="utf-8"?>
2
3 <launch>
4
5   <!-- GAZEBO ARGUMENTS -->
6   <!-- Run Gazebo headless -->
7   <arg name="headless" default="false"/>
8   <!-- Model path -->
9   <arg name="model_path" default="$(find smb_gazebo)/"/>
10  <arg name="robot_namespace" default="/" />
11  <arg name="robot_model_name" default="smb"/>
12  <arg name="enable_ekf" default="true"/>
13
14  <!-- Name of the world -->
15  <arg name="world" default="empty"/>
16  <!-- Path to the world file -->
17  <arg name="world_file" default="$(find smb_gazebo)/worlds/$(arg world).world"/>
18  <arg name="laser_enabled" default="true"/>
19
20  <!-- Set the initial pose of the robot's main body -->
21  <arg name="x" default="0.0"/>
22  <arg name="y" default="0.0"/>
23  <arg name="z" default="0.4"/>
24  <arg name="roll" default="0.0"/>
25  <arg name="pitch" default="0.0"/>
26  <arg name="yaw" default="0.0"/>
27  <!-- Start paused -->
28  <arg name="paused" default="false"/>
29  <!-- Use simulation clock -->
30  <arg name="use_sim_time" default="true"/>
31  <!-- Debug mode -->
32  <arg name="debug" default="false"/>
```

Hình 20. Chỉnh trạng thái của laser_enabled

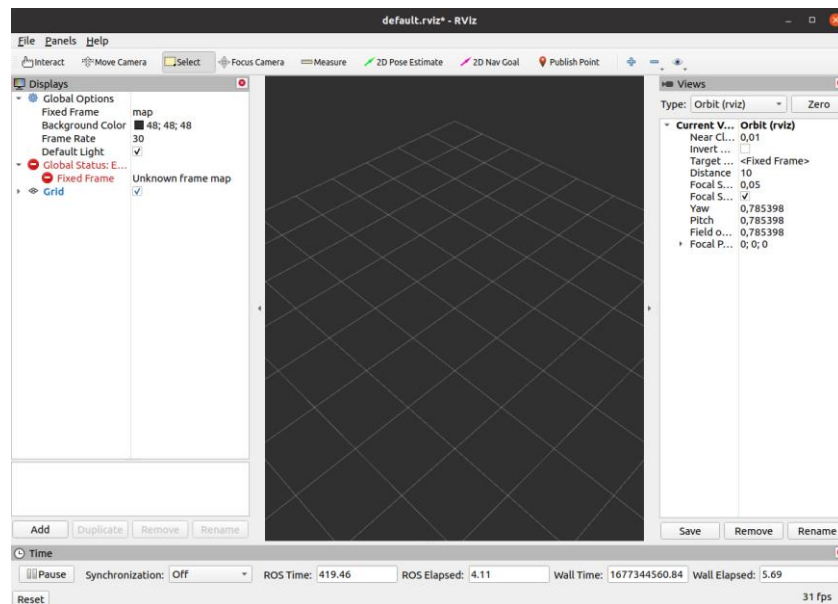
Bài 9: Yêu cầu: Hiện thị robot và laserscan trên Rviz ; thêm Rviz vào file launch.

B1: Chạy file smb_gazebo.launch để khởi tạo môi trường và robot



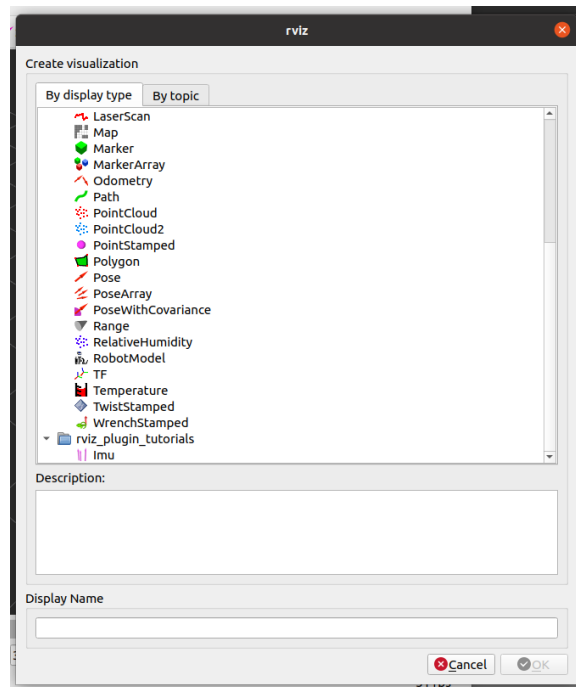
Hình 21. Robot trong môi trường gazebo

B2: Khởi chạy Rviz bằng câu lệnh: `~$ rosrun rviz rviz`



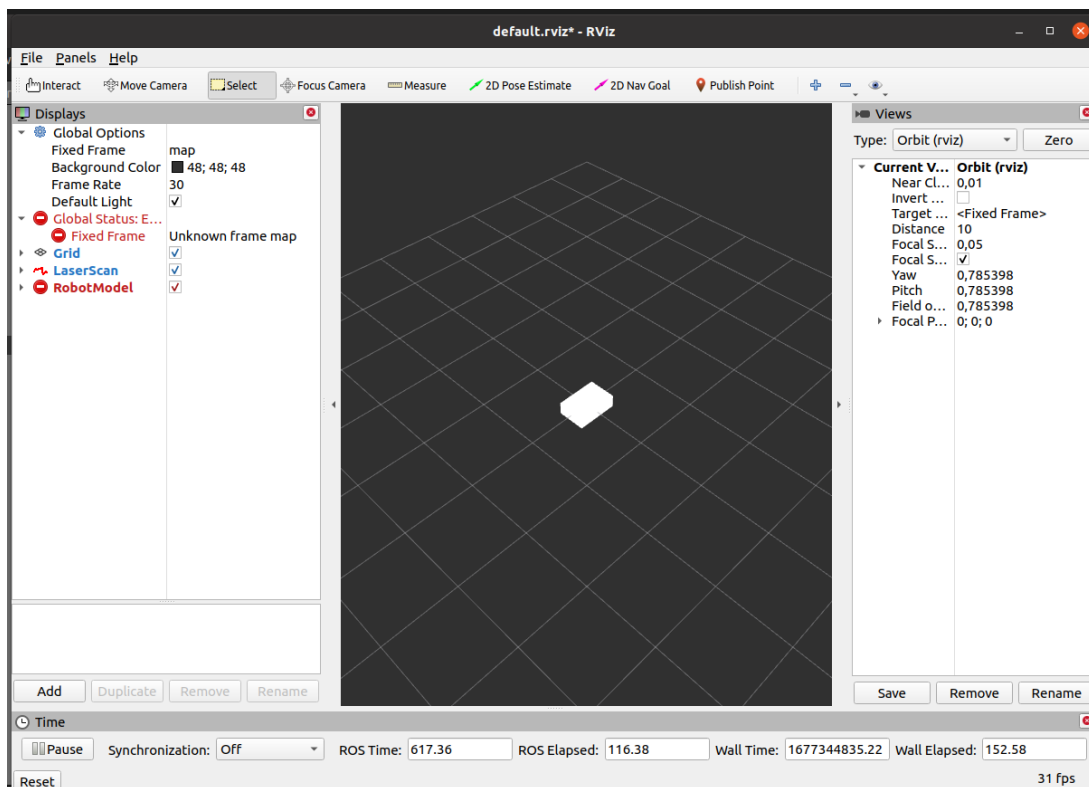
Hình 22. Môi trường Rviz

B3: Gọi các node để hiển thị. Ta sẽ chọn “Add” để thêm các node như “LaserScan” và “RobotModel” như Hình 23.



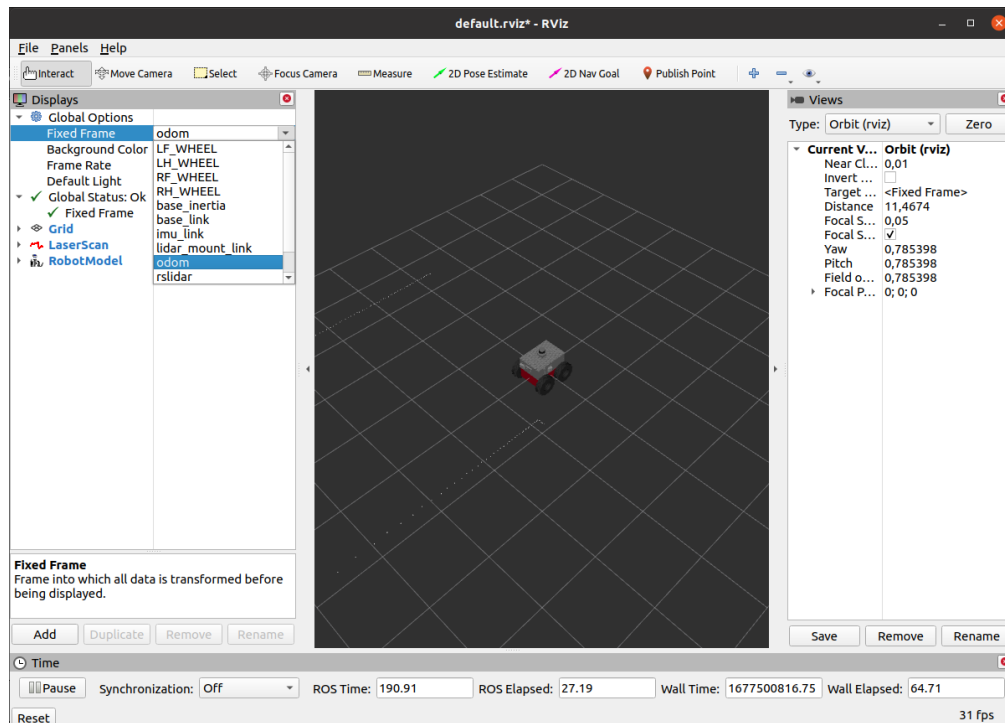
Hình 23. Khung “Add” thêm các topic

Rviz sẽ hiển thị như Hình 24.

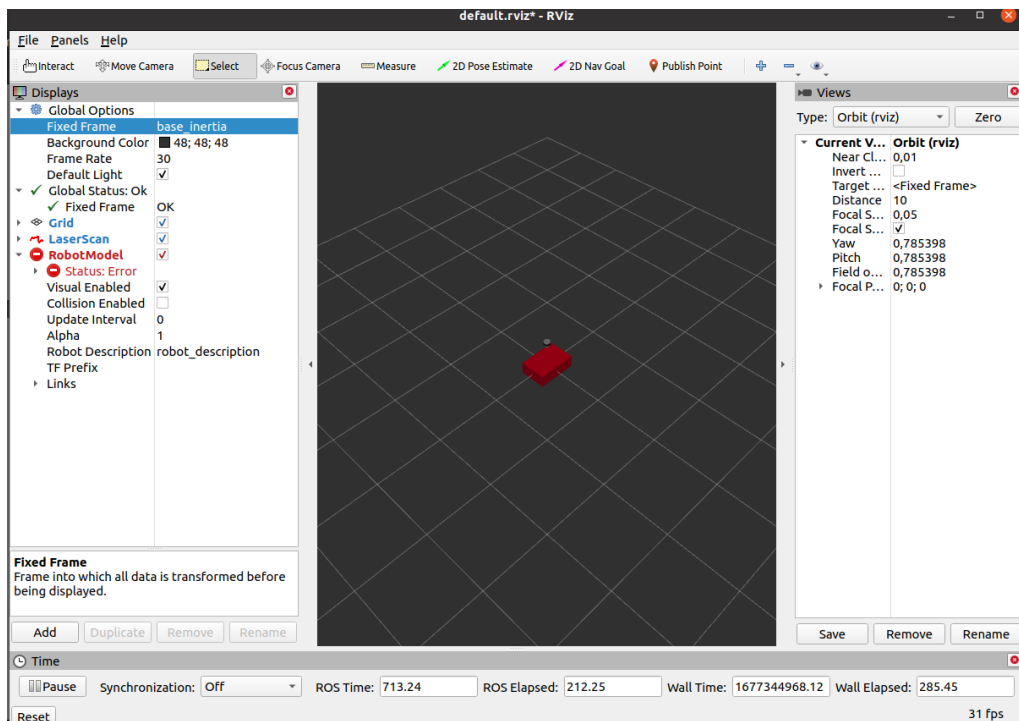


Hình 24. Hiển thị trên Rviz sau khi thêm các topic

B4: Ta sẽ chọn “odom” tại “Fixed Frame” như *Hình 25* để có được kết quả như *Hình 26*.



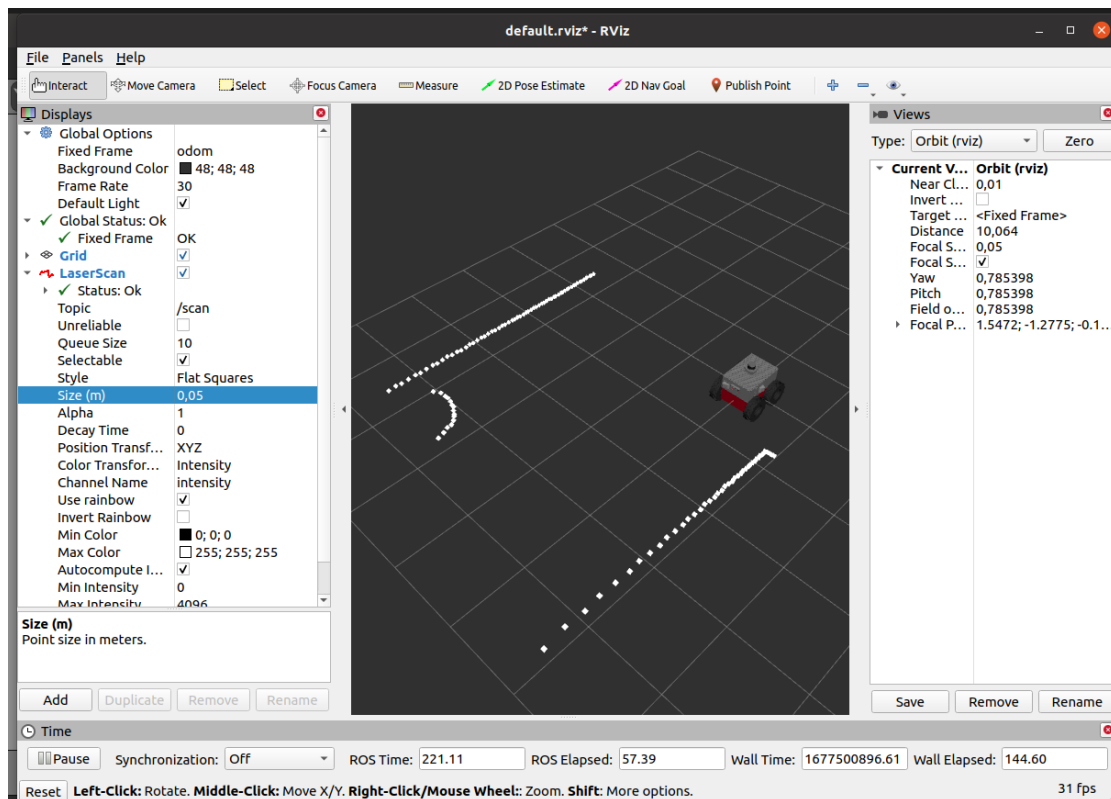
Hình 25. Chọn Fixed Fram phù hợp



Hình 26. Hiện thị trên Rviz sau các bước trên

Nhìn *Hình 26*, ta có thể thấy robot bị lỗi hiển thị và không thấy các tia laser laser. Lỗi ở đây là trước khi khởi chạy Rviz tại bước 2 ta cần thực hiện câu lệnh: `~$ source devel/setup.bash`. Sau đó ta thực hiện các bước tiếp như trên và thu được kết quả đầy đủ như *Hình 27*. Ta có thể chỉnh sửa kích thước, màu sắc pointcloud trong “LaserScan”.

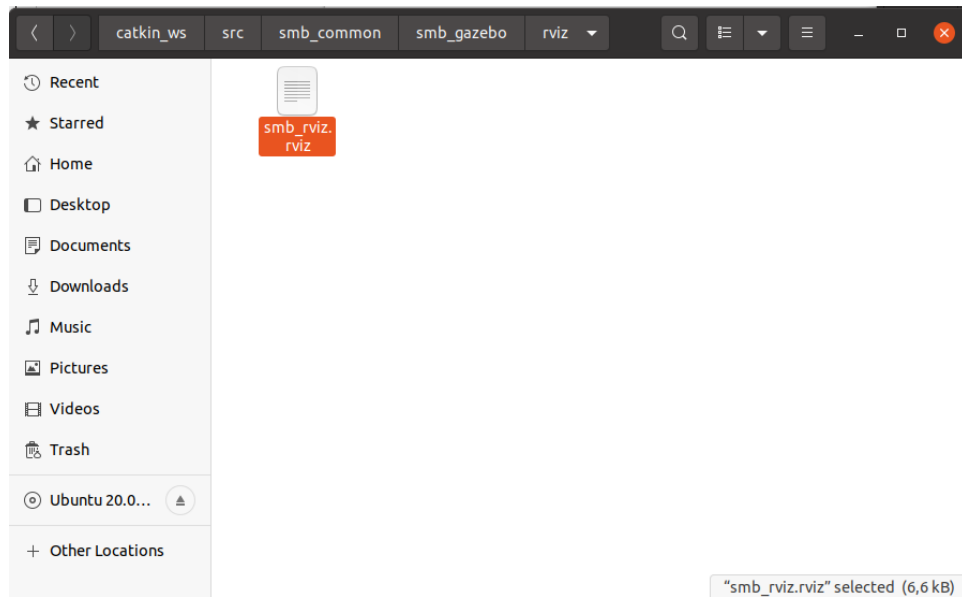
Do file `smb_gazebo.launch` đã tích hợp điều khiển robot bằng bàn phím (`teleop_keybroad`) nên ta có thể điều khiển qua terminal và thấy các pointcloud robot quét được trên đường di chuyển.



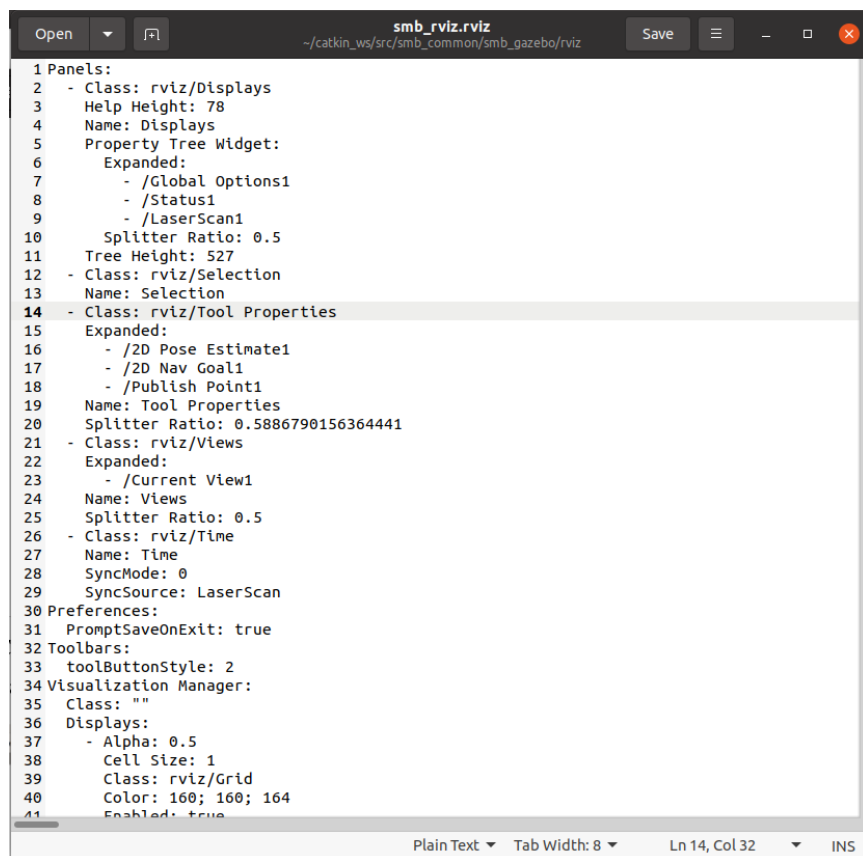
Hình 27. Hiển thị Robot và pointcloud từ laser trên Rviz

B5: Lưu cấu hình Rviz đã cài đặt sau các bước trên.

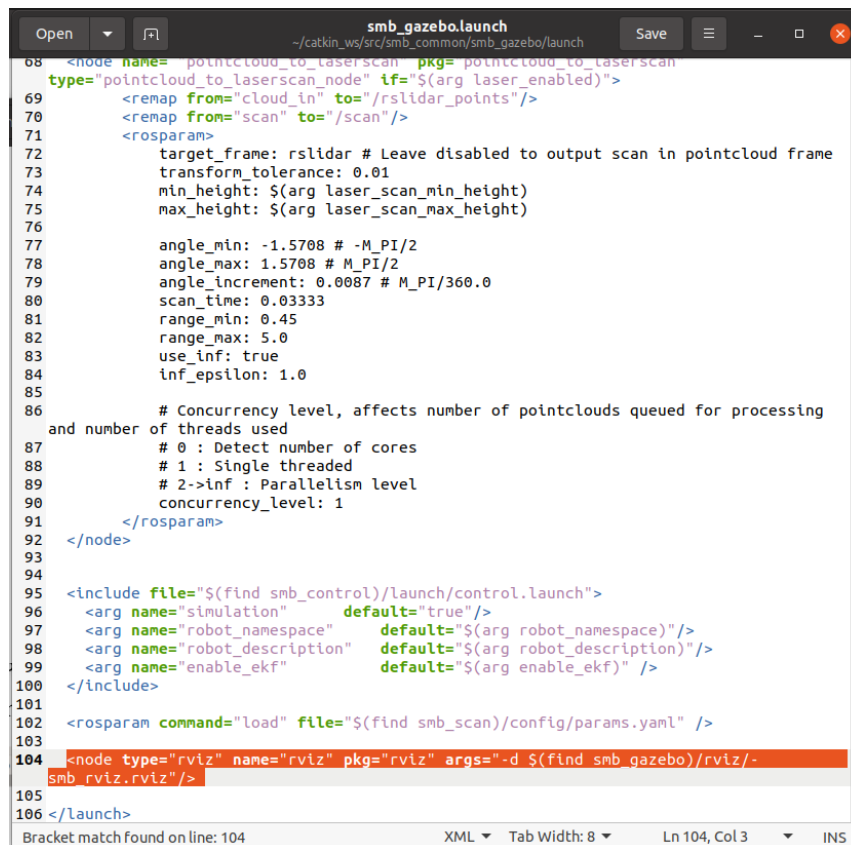
Tôi sẽ lưu cấu hình này thành 1 file có tên `smb_rviz.rviz` và chỉnh sửa file `smb_gazebo.launch` để mỗi khi chạy file `smb_gazebo.launch` sẽ hiện lên luôn cấu hình rviz như trên cùng gazebo. *Hình 28* thể hiện vị trí file `smb_rviz.rviz` và các thông số trong file rviz. và *Hình 29* thể hiện phần chỉnh sửa trong file `smb_gazebo.launch`.



Hình 28a. Vị trí file *smb_rviz.rviz*



Hình 28b. Các thông số của file *smb_rviz.rviz*

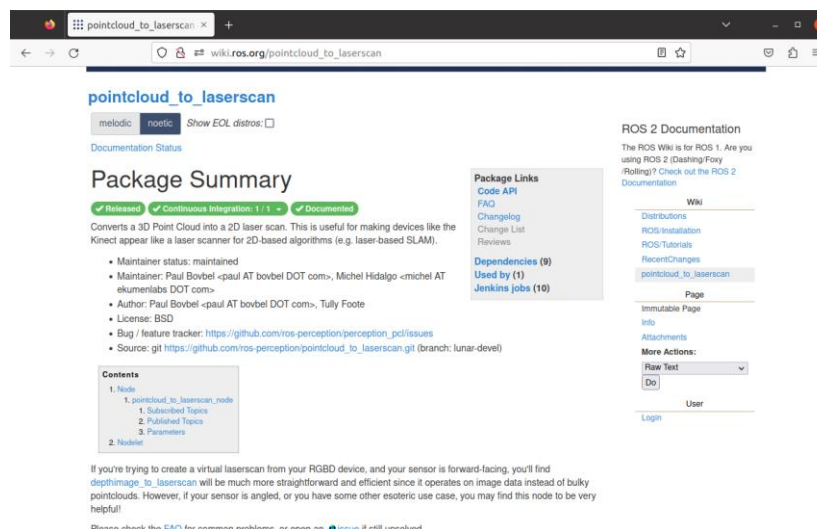


```
68 <node name="pointcloud_to_laserscan" pkg="pointcloud_to_laserscan"
69 type="pointcloud_to_laserscan_node" if="$(arg laser_enabled)">
70 <remap from="cloud_in" to="/rslidar_points"/>
71 <remap from="scan" to="/scan"/>
72 <rosparam>
73   target_frame: rslidar # Leave disabled to output scan in pointcloud frame
74   transform_tolerance: 0.01
75   min_height: $(arg laser_scan_min_height)
76   max_height: $(arg laser_scan_max_height)
77
78   angle_min: -1.5708 # -M_PI/2
79   angle_max: 1.5708 # M_PI/2
80   angle_increment: 0.0087 # M_PI/360.0
81   scan_time: 0.03333
82   range_min: 0.45
83   range_max: 5.0
84   use_inf: true
85   inf_epsilon: 1.0
86
87   # Concurrency level, affects number of pointclouds queued for processing
88   # 0 : Detect number of cores
89   # 1 : Single threaded
90   # 2->inf : Parallelism level
91   concurrency_level: 1
92 </rosparam>
93 </node>
94
95 <include file="$(find smb_control)/launch/control.launch">
96 <arg name="simulation" default="true"/>
97 <arg name="robot_namespace" default="$(arg robot_namespace)"/>
98 <arg name="robot_description" default="$(arg robot_description)"/>
99 <arg name="enable_ekf" default="$(arg enable_ekf)" />
100 </include>
101
102 <rosparam command="load" file="$(find smb_scan)/config/params.yaml" />
103
104 <node type="rviz" name="rviz" pkg="rviz" args="-d $(find smb_gazebo)/rviz/-
105 smb_rviz.rviz" />
106 </launch>
```

Hình 29. Chỉnh sửa trong file `smb_gazebo.launch` để gọi cấu hình `rviz`

Câu 10: Yêu cầu: Phân tích node `pointcloud_to_laserscan`

Tại trang chủ `ros.org` cung cấp thông tin về node `/pointcloud_to_laserscan` trong Hình 30. Đó là node chuyển đổi từ 3D pointcloud thành 2D laserscan.



Hình 30. Thông tin về `/pointcloud_to_laserscan`

Thông tin của node `/pointcloud_to_laserscan` được thể hiện trong *Hình 31*.

Publishers:

`/rosout [msg: rosgraph_msgs/Log]`

`/scan [msg: sensor_msgs/LaserScan]`

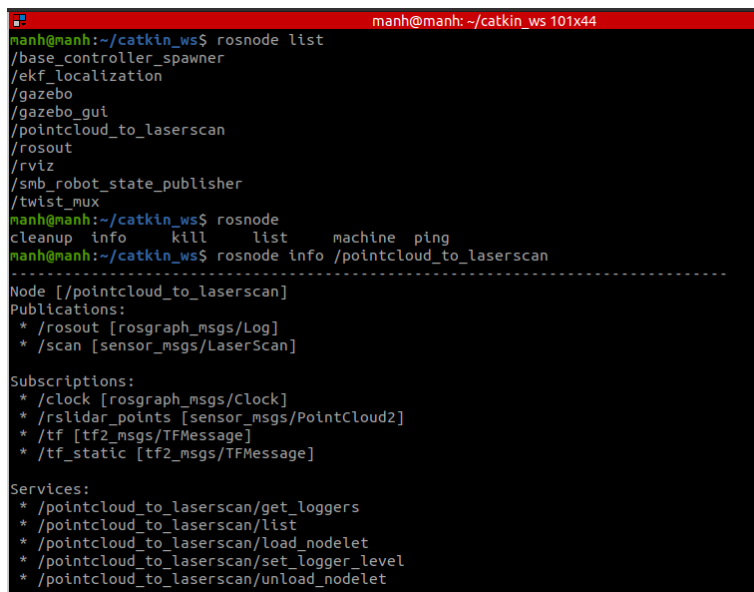
Subscribers:

`/clock [msg: rosgraph_msgs/clock]`

`/rslidar_points [msg: sensor_msgs/PointCloud2]`

`/tf [msg: tf2_msgs/TFMessage]`

`/tf_static [msg: tf2_msgs/TFMessage]`

A terminal window with a red title bar showing the command 'rosnode info /pointcloud_to_laserscan' and its output. The output lists the node's name, publishers, subscribers, and services.

```
manh@manh: ~/catkin_ws 101x44
manh@manh:~/catkin_ws$ rosnode list
/base_controller_spawner
/ekf_localization
/gazebo
/gazebo_gui
/pointcloud_to_laserscan
/rosout
/rviz
/smb_robot_state_publisher
/twist_mux
manh@manh:~/catkin_ws$ rosnode
cleanup info kill list machine ping
manh@manh:~/catkin_ws$ rosnode info /pointcloud_to_laserscan
-----
Node [/pointcloud_to_laserscan]
Publications:
* /rosout [rosgraph_msgs/Log]
* /scan [sensor_msgs/LaserScan]

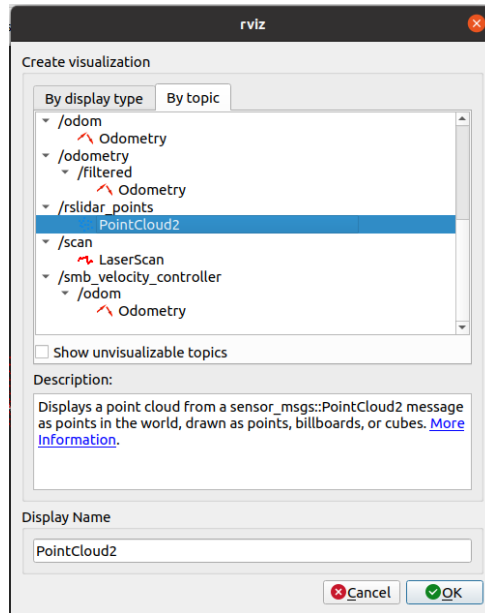
Subscriptions:
* /clock [rosgraph_msgs/Clock]
* /rslidar_points [sensor_msgs/PointCloud2]
* /tf [tf2_msgs/TFMessage]
* /tf_static [tf2_msgs/TFMessage]

Services:
* /pointcloud_to_laserscan/get_loggers
* /pointcloud_to_laserscan/list
* /pointcloud_to_laserscan/load_nodelet
* /pointcloud_to_laserscan/set_logger_level
* /pointcloud_to_laserscan/unload_nodelet
```

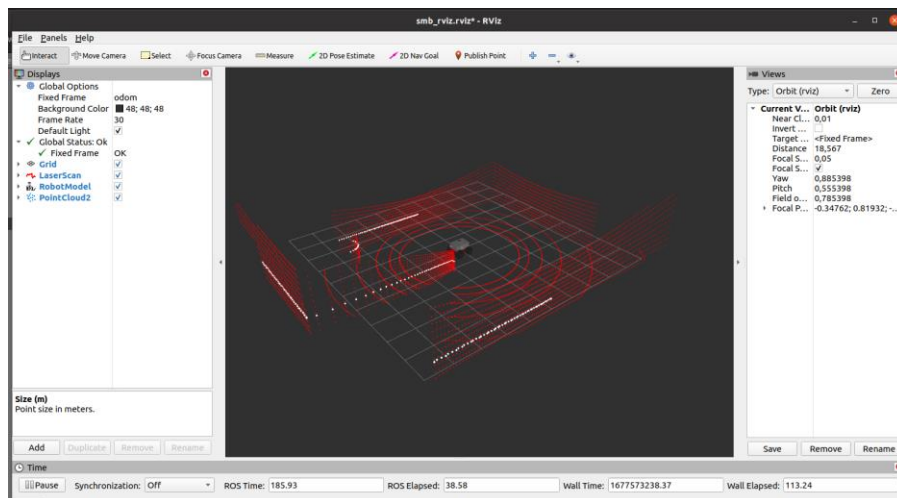
Hình 31. Thông tin node `/pointcloud_to_laserscan`

Trực quan hóa pointcloud và laserscan trên Rviz:

Trực quan hóa laserscan trên Rviz đã làm từ *Bài 10*. Trực quan hóa pointcloud trên rviz sẽ được đọc từ topic `/rslidar_points` qua msg `sensor_msgs/PointCloud2`, thêm topic như *Hình 32* và sẽ đạt kết quả như *Hình 33*.



Hình 32. Thêm topic /rslidar_points



Hình 33. Trực quan hóa laserscan và pointcloud

Câu 11: Yêu cầu: Đếm có bao nhiêu point trong pointcloud

Pointcloud được thể hiện tại topic /rslidar_points với msg là sensor_msgs/PointCloud2.

Thông tin msg sensor_msgs/PointCloud thể hiện tại Hình 34.

sensor_msgs/PointCloud2 Message

File: `sensor_msgs/PointCloud2.msg`

Raw Message Definition

```
# This message holds a collection of N-dimensional points, which may
# contain additional information such as normals, intensity, etc. The
# point data is stored as a binary blob, its layout described by the
# contents of the "fields" array.

# The point cloud data may be organized 2d (image-like) or 1d
# (unordered). Point clouds organized as 2d images may be produced by
# camera depth sensors such as stereo or time-of-flight.

# Time of sensor data acquisition, and the coordinate frame ID (for 3d
# points).
Header header

# 2D structure of the point cloud. If the cloud is unordered, height is
# 1 and width is the length of the point cloud.
uint32 height
uint32 width

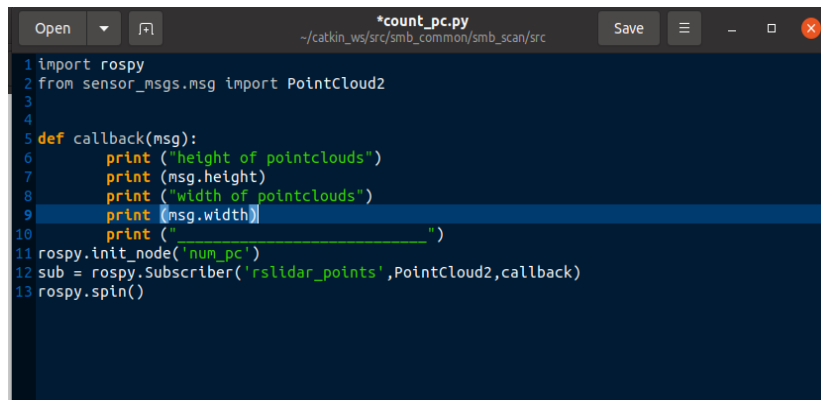
# Describes the channels and their layout in the binary data blob.
PointField[] fields

bool is_bigendian # Is this data bigendian?
uint32 point_step # Length of a point in bytes
uint32 row_step # Length of a row in bytes
uint8[] data # Actual point data, size is (row_step*height)

bool is_dense # True if there are no invalid points
```

Hình 34. Thông tin msg `sensor_msgs/PointCloud2`

Kích thước của pointcloud được thể hiện qua mảng với chiều cao là `height` và chiều rộng là `width`. Số lượng points sẽ bằng tích của `height` và `width`. Ta sẽ viết file code “`count_pc.py`” để đọc dữ liệu `height` và `width`.



```
*count_pc.py
~/catkin_ws/src/smb_common/smb_scan/src

1 import rospy
2 from sensor_msgs.msg import PointCloud2
3
4
5 def callback(msg):
6     print("height of pointclouds")
7     print(msg.height)
8     print("width of pointclouds")
9     print(msg.width)
10    print(" ")
11 rospy.init_node('num_pc')
12 sub = rospy.Subscriber('rslidar_points', PointCloud2, callback)
13 rospy.spin()
```

Hình 35. File `count_pc.py`

Sau đó, ta khởi tạo môi trường `smb_gazebo.launch`, điều khiển robot và xem số lượng point trong pointcloud sau mỗi lần quét. Hình 36 thể hiện pointcloud thay đổi trên đường đi với số lượng thay đổi 6972, 6970, 6963, ...

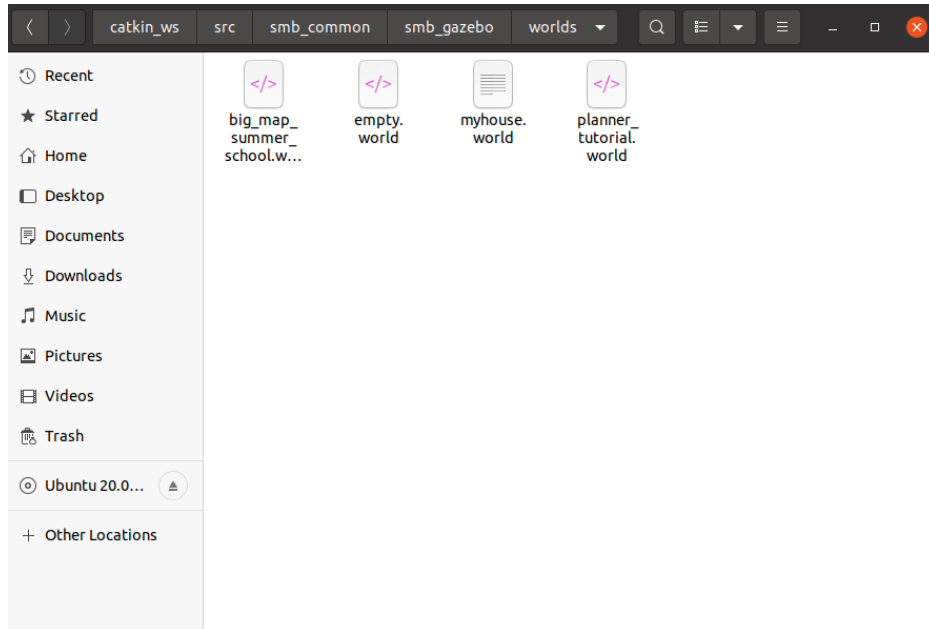
```
manh@manh: ~/catkin_ws
manh@manh: ~/catkin_ws 80x43
width of pointclouds
6972
-----
height of pointclouds
1
width of pointclouds
6972
-----
height of pointclouds
1
width of pointclouds
6970
-----
height of pointclouds
1
width of pointclouds
6968
-----
height of pointclouds
1
width of pointclouds
6968
-----
height of pointclouds
1
width of pointclouds
6963
-----
height of pointclouds
1
width of pointclouds
6959
-----
height of pointclouds
1
width of pointclouds
6954
-----
height of pointclouds
1
width of pointclouds
6945
-----
height of pointclouds
```

Hình 36. Số lượng point trong pointcloud thay đổi trên đường di chuyển.

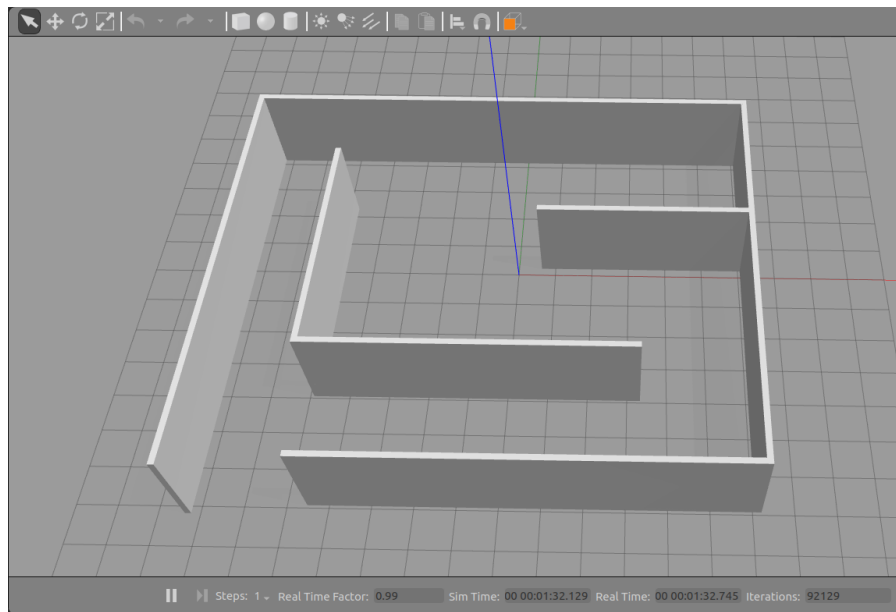
Note: Tạo một môi trường gazebo và chạy nó.

Tham khảo tại: <https://www.youtube.com/watch?v=S8pwfsK-F9w>

B1: Tham khảo video trên, tạo môi trường “myhouse.world” trong smb_gazebo/worlds. Môi trường myhouse.world được hiển thị trong *Hình b*.

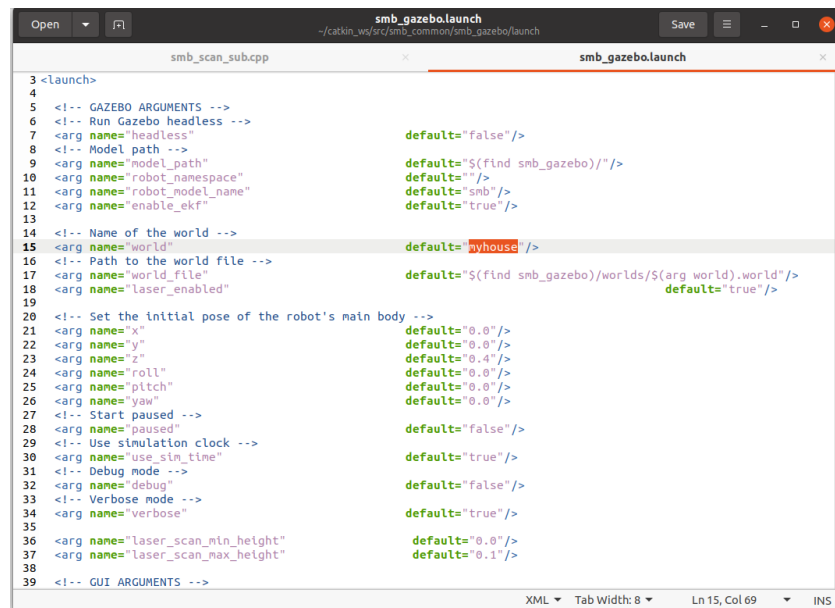


Hình a .Các môi trường trong smb_gazebo/world



Hình b. Môi trường myhouse.world

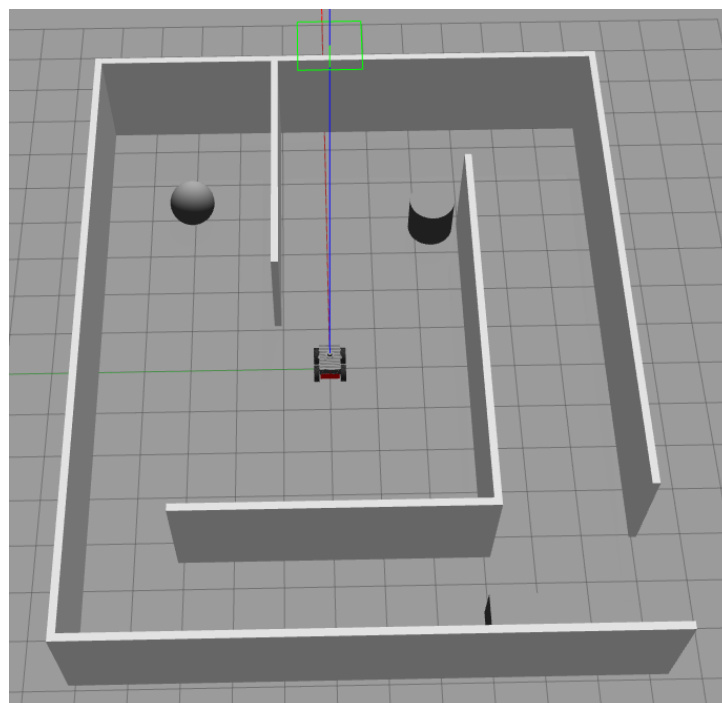
B2: Tại file `smb_gazebo.launch`, ta sẽ sửa tên môi trường ta gọi như *Hình c*, có thể thay thế bằng `empty` hay `big_map_summer_school` tùy thuộc vào máy tính.



```
3 <launch>
4
5 <!-- GAZEBO ARGUMENTS -->
6 <!-- Run Gazebo headless -->
7 <arg name="headless" default="false"/>
8 <!-- Model path -->
9 <arg name="model_path" default="$(find smb_gazebo)"/>
10 <arg name="robot_namespace" default="" />
11 <arg name="robot_model_name" default="smb"/>
12 <arg name="enable_ekf" default="true"/>
13
14 <!-- Name of the world -->
15 <arg name="world" default="myhouse" />
16 <!-- Path to the world file -->
17 <arg name="world_file" default="$(find smb_gazebo)/worlds/${arg world}.world"/>
18 <arg name="laser_enabled" default="true" />
19
20 <!-- Set the initial pose of the robot's main body -->
21 <arg name="x" default="0.0"/>
22 <arg name="y" default="0.0"/>
23 <arg name="z" default="0.4"/>
24 <arg name="roll" default="0.0"/>
25 <arg name="pitch" default="0.0"/>
26 <arg name="yaw" default="0.0"/>
27 <!-- Start paused -->
28 <arg name="paused" default="false"/>
29 <!-- Use simulation clock -->
30 <arg name="use_sim_time" default="true"/>
31 <!-- Debug mode -->
32 <arg name="debug" default="false"/>
33 <!-- Verbose mode -->
34 <arg name="verbose" default="true"/>
35
36 <arg name="laser_scan_min_height" default="0.0"/>
37 <arg name="laser_scan_max_height" default="0.1"/>
38
39 <!-- GUI ARGUMENTS -->
```

Hình c. Chỉnh sửa tại file `smb_gazebo.launch`

B3: Khởi chạy file `smb_gazebo.launch` để thực hiện các thao tác tại các bài tập 6-11.



Hình d. Robot trong môi trường `myhouse.world`