Αυτόνομοι Πράκτορες : ΗΜΜΥ 189

Πολυτεχνείο Κρήτης
Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Γιάννης Περίδης 2018030069

Αναφορά 4ου Εργαστηρίου:

Εισαγωγή:

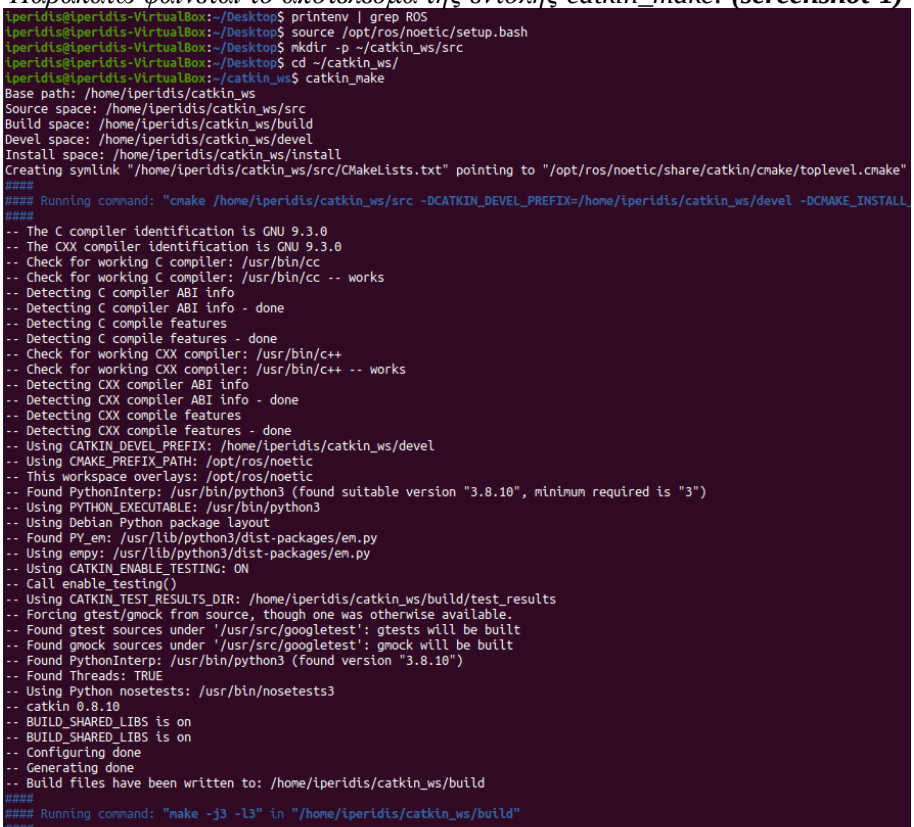
Σκοπός της παρούσας εργαστηριακής άσκησης ήταν η περαιτέρω εξοικείωση με τον προσομοιωτή Gazebo και με το R.O.S.(Robot Operating System). Το ROS ,είναι ένα πλαίσιο ανοιχτού κώδικα που βοηθά να δημιουργηθεί και να επαναχρησιμοποιηθεί κώδικας μεταξύ εφαρμογών ρομποτικής. Ακόμη, θα δούμε πως μπορούμε να ελέγξουμε ένα ρομποτικό μοντέλο σε μια προσομοίωση. Συγκεκριμένα θα χειριστούμε εργαλεία διεπαφής και αλληλεπίδρασης μεταξύ των ενεργών κινητών μερών του ρομπότ και του χρήστη, όπως επίσης θα γίνει επεξεργασία μετρήσεων των αισθητήρων αντίληψης ενός μοντέλου.

Πειραματισμός και Εγκατάσταση R.O.S. :

Αρχικά, ακολουθήθηκαν τα tutorials “Model Plugins” και “Elevators” με σκοπό την κατανόηση , δημιουργία και ενσωμάτωση των Gazebo plugins σε κάποιο ρομποτικό μοντέλο. Στην συνέχεια, εγκαταστάθηκε η έκδοση ROS Noetic Ninjemys για το σύστημα Ubuntu 20.04 , ακολουθώντας το αντίστοιχο tutorial.Ύστερα, έγινε η δημιουργία του directory catkin_ws και λήφθηκαν εκεί τα απαραίτητα πακέτα και εφαρμογές του ROS. Τέλος, εκτελέστηκε η εντολή catkin_make η οποία έκανε Build το directory , χωρίς να συνδέει με κάποιο πακέτο το gazebo και το ROS.

Το ρομπότ που θα χρησιμοποιηθεί για την προσομοίωση είναι το Husky.

Παρακάτω φαίνεται το αποτέλεσμα της εντολής catkin_make: ***(screenshot-1)***



Μετά την εντολή αυτή, λήφθηκαν τα απαραίτητα πακέτα χειρισμού του Husky και έγινε ξανά build,αυτή τη φορά με ενσωματωμένο το ROS στο gazebo. Επίσης, μέσω της ενεργοποίησης της μεταβλητής *HUSKY LMS1XX ENABLED*, ενεργοποιήθηκε ο αισθητήρας LiDAR του Husky , με σκοπό την απεικόνιση των δεδομένων που προέρχονται από το ROS. Η παραπάνω διαδικασία εκ κινήθηκε με την εντολή *$ roslaunch husky_gazebo husky_empty_world.launch.*

Ύστερα, μετά την εκκίνηση , εκτελέστηκαν σε ξεχωριστά terminals οι εντολές *$ rostopic list* και *$ rosservice list* οι οποίες είναι υπεύθυνες για να μεταφέρουν τις πληροφορίες των αισθητήρων, δηλαδή την απεικόνιση σε εμάς όλων των ROS topics και για την υλοποίηση των διεπαφών ανταλλαγής δεδομένων , δηλαδή τα services αντίστοιχα.

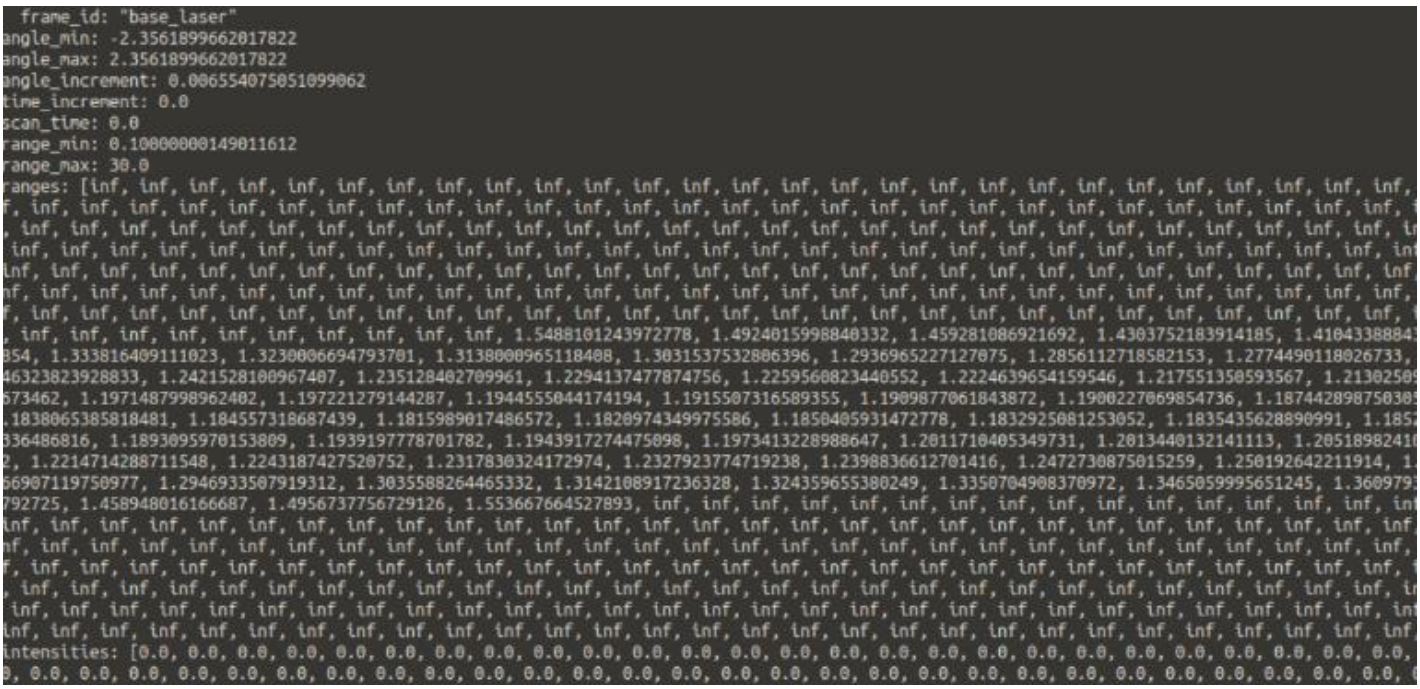
Παρακάτω φαίνονται τα αποτελέσματα των εντολών rostopic list και rosservice list:
(screenshots 2-3)

```
iperidis@iperidis-VirtualBox:~/catkin_ws$ rostopic list
/clock
/cmd_vel
/diagnostics
/e_stop
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/performance_metrics
/gazebo/set_link_state
/gazebo/set_model_state
/husky_velocity_controller/cmd_vel
/husky_velocity_controller/odom
/husky_velocity_controller/parameter_descriptions
/husky_velocity_controller/parameter_updates
/imu/data
/imu/data/accel/parameter_descriptions
/imu/data/accel/parameter_updates
/imu/data/bias
/imu/data/rate/parameter_descriptions
/imu/data/rate/parameter_updates
/imu/data/yaw/parameter_descriptions
/imu/data/yaw/parameter_updates
/joint_states
/joy_teleop/cmd_vel
/joy_teleop/joy
/joy_teleop/joy/set_feedback
/navsat/fix
/navsat/fix/position/parameter_descriptions
/navsat/fix/position/parameter_updates
/navsat/fix/status/parameter_descriptions
/navsat/fix/status/parameter_updates
/navsat/fix/velocity/parameter_descriptions
/navsat/fix/velocity/parameter_updates
/navsat/vel
/odometry/filtered
/rosout
/rosout_agg
/set_pose
/tf
/tf_static
/twist_marker_server/cmd_vel
/twist_marker_server/feedback
/twist_marker_server/update
/twist_marker_server/update_full
```

```
iperidis@iperidis-VirtualBox:~/catkin_ws$ rosservice list
/base_controller_spawner/get_loggers
/base_controller_spawner/set_logger_level
/controller_manager/list_controller_types
/controller_manager/list_controllers
/controller_manager/load_controller
/controller_manager/reload_controller_libraries
/controller_manager/switch_controller
/controller_manager/unload_controller
/ekf_localization/enable
/ekf_localization/get_loggers
/ekf_localization/set_logger_level
/ekf_localization/toggle
/gazebo/apply_body_wrench
/gazebo/apply_joint_effort
/gazebo/clear_body_wrenches
/gazebo/clear_joint_forces
/gazebo/delete_light
/gazebo/delete_model
/gazebo/get_joint_properties
/gazebo/get_light_properties
/gazebo/get_link_properties
/gazebo/get_link_state
/gazebo/get_loggers
/gazebo/get_model_properties
/gazebo/get_model_state
/gazebo/get_physics_properties
/gazebo/get_world_properties
/gazebo/pause_physics
/gazebo/reset_simulation
/gazebo/reset_world
/gazebo/set_joint_properties
/gazebo/set_light_properties
/gazebo/set_link_properties
/gazebo/set_link_state
/gazebo/set_logger_level
/gazebo/set_model_configuration
/gazebo/set_model_state
/gazebo/set_parameters
/gazebo/set_physics_properties
/gazebo/spawn_sdf_model
/gazebo/spawn_urdf_model
/gazebo/unpause_physics
/gazebo_gui/get_loggers
/gazebo_gui/set_logger_level
/husky_velocity_controller/set_parameters
/imu/data/accel/set_parameters
/imu/data/calibrate
/imu/data/rate/set_parameters
/imu/data/set_accel_bias
/imu/data/set_rate_bias
/imu/data/yaw/set_parameters
/joy_teleop/joy_node/get_loggers
/joy_teleop/joy_node/set_logger_level
/joy_teleop/teleop_twist_joy/get_loggers
/joy_teleop/teleop_twist_joy/set_logger_level
/navsat/fix/position/set_parameters
/navsat/fix/set_reference_geopose
/navsat/fix/status/set_parameters
/navsat/fix/velocity/set_parameters
/robot_state_publisher/get_loggers
/robot_state_publisher/set_logger_level
/rosout/get_loggers
/rosout/set_logger_level
/set_pose
/twist_marker_server/get_loggers
```

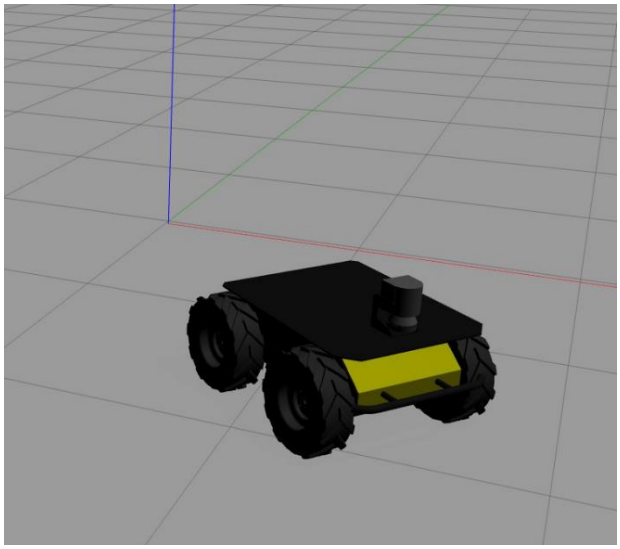
Έπειτα, για να δοθούν σε πραγματικό χρόνο τα αληθινά δεδομένα του αισθητήρα LiDAR που ενεργοποιήθηκε , εκτελέστηκε η εντολή *$ rostopic echo /scan* .Παρατηρήθηκε πως όλες οι τιμές του διανύσματος των ranges έγραφαν inf, δηλαδή άπειρο, γεγονός που το περιμέναμε, εφόσον το Husky βρίσκεται μόνο του στον κόσμο της προσομοίωσης και δεν έχει κάποιο εμπόδιο να ανιχνεύσει στον χώρο.

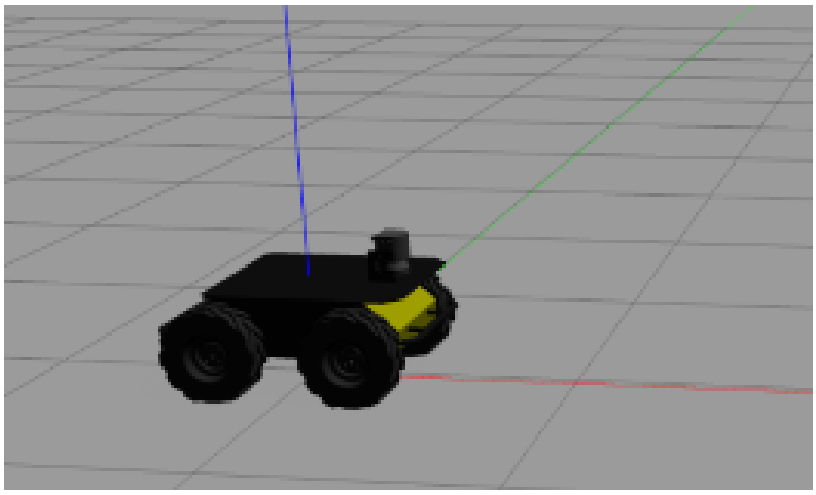
Παρακάτω φαίνονται τα αποτελέσματα της εντολής rostopic echo/scan: ***(screenshot -4)***



Εκτελέστηκε ακόμα η δεύτερη εντολή πάνω στα services η *$ rosservice call /gazebo/reset_world* η οποία επαναφέρει το ρομπότ Husky στην αρχική του θέση στην προσομοίωση. Για να φανούν τα αποτελέσματα της εντολής αυτή, μετακινήθηκε χειροκίνητα το ρομπότ σε μια άκυρη θέση και ύστερα , μετά την εκτέλεση της εντολής επέστρεψε στην αρχική θέση (0,0,0)

Παρακάτω φαίνονται τα αποτελέσματα πριν και μετά από την εκτέλεση της παραπάνω εντολής: ***(screenshots-5,6)***

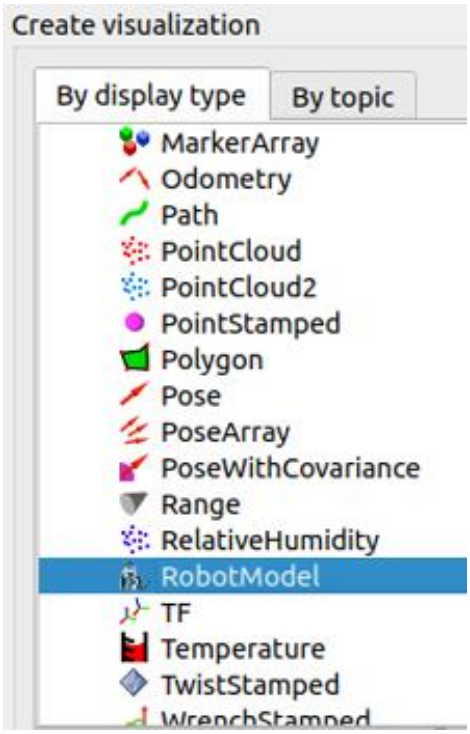


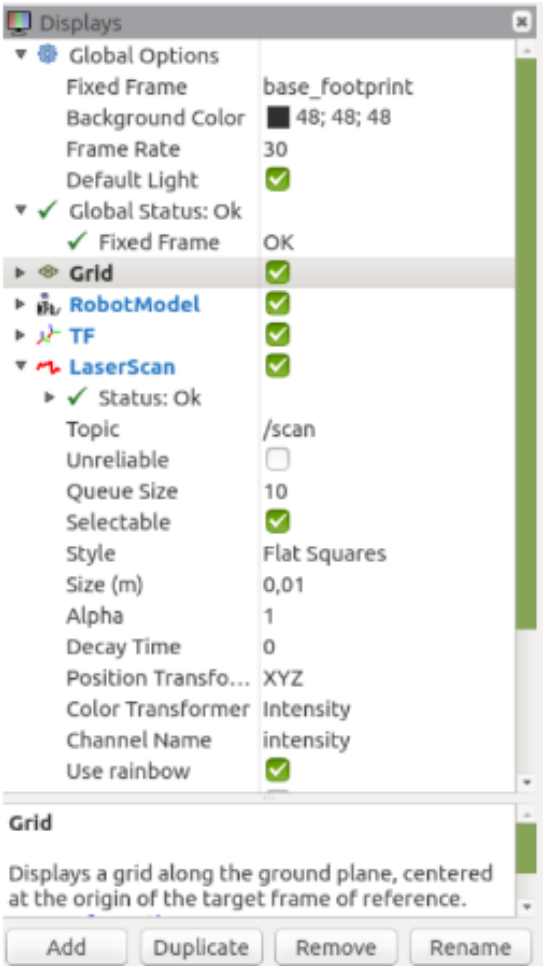


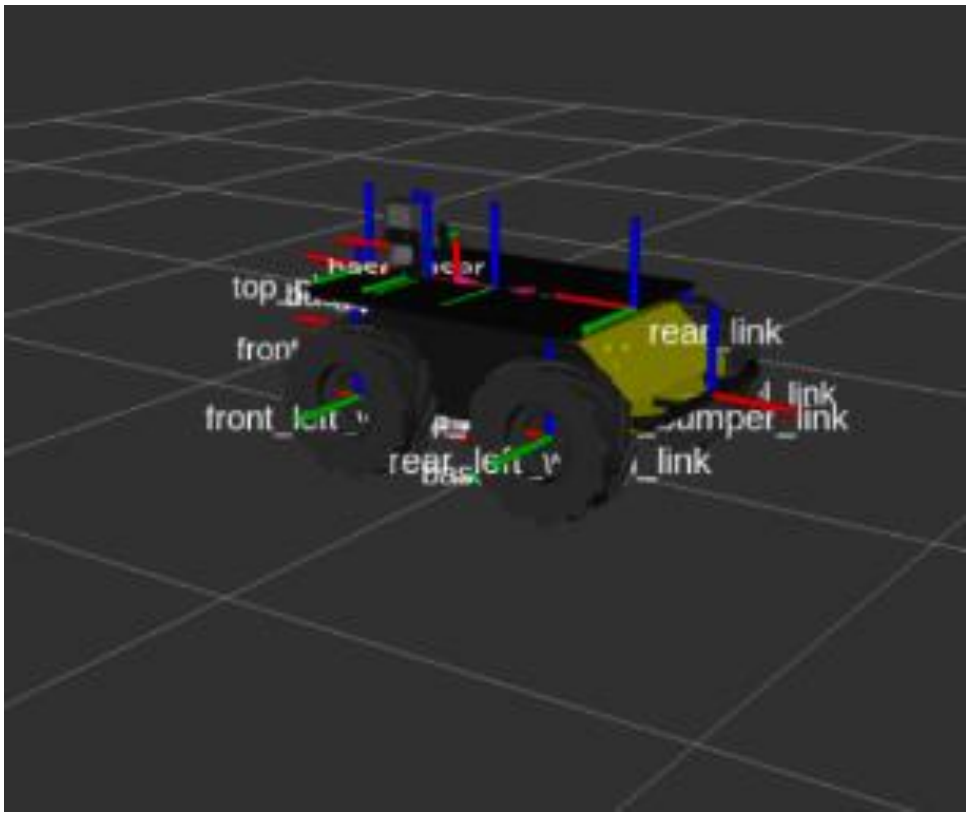
Εγκατάσταση του RViZ:

Στο σημείο αυτό, έγινε εγκατάσταση του προγράμματος απεικόνισης που παρέχει το ROS, το RViZ. Το RViZ δείχνει την κατάσταση των υπαρκτών ρομπότ με τις μεταξύ χωρικές συσχετίσεις, όπως και τα δεδομένα που λαμβάνουν οι αισθητήρες τους σε πραγματικό χρόνο. Στο περιβάλλον προσομοίωσης του rviz έγινε με την χρήση της εντολής add η προσθήκη RobotModel του Husky. Ύστερα, προστέθηκαν επίσης το δέντρο συστημάτων συντεταγμένων TF και το LaserScan. Τέλος, χρησιμοποιήθηκαν οι βασικές συντεταγμένες base footprint.

Παρακάτω φαίνεται το Display των επιλογών που έγιναν add και το παράθυρο προσομοίωσης του RViZ: ***(screenshots-7,8,9)***

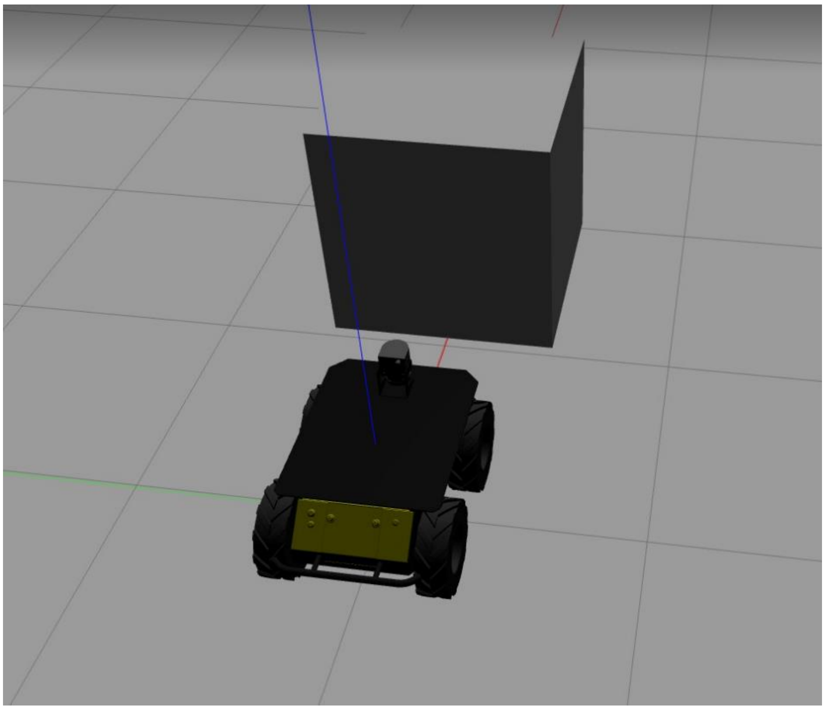






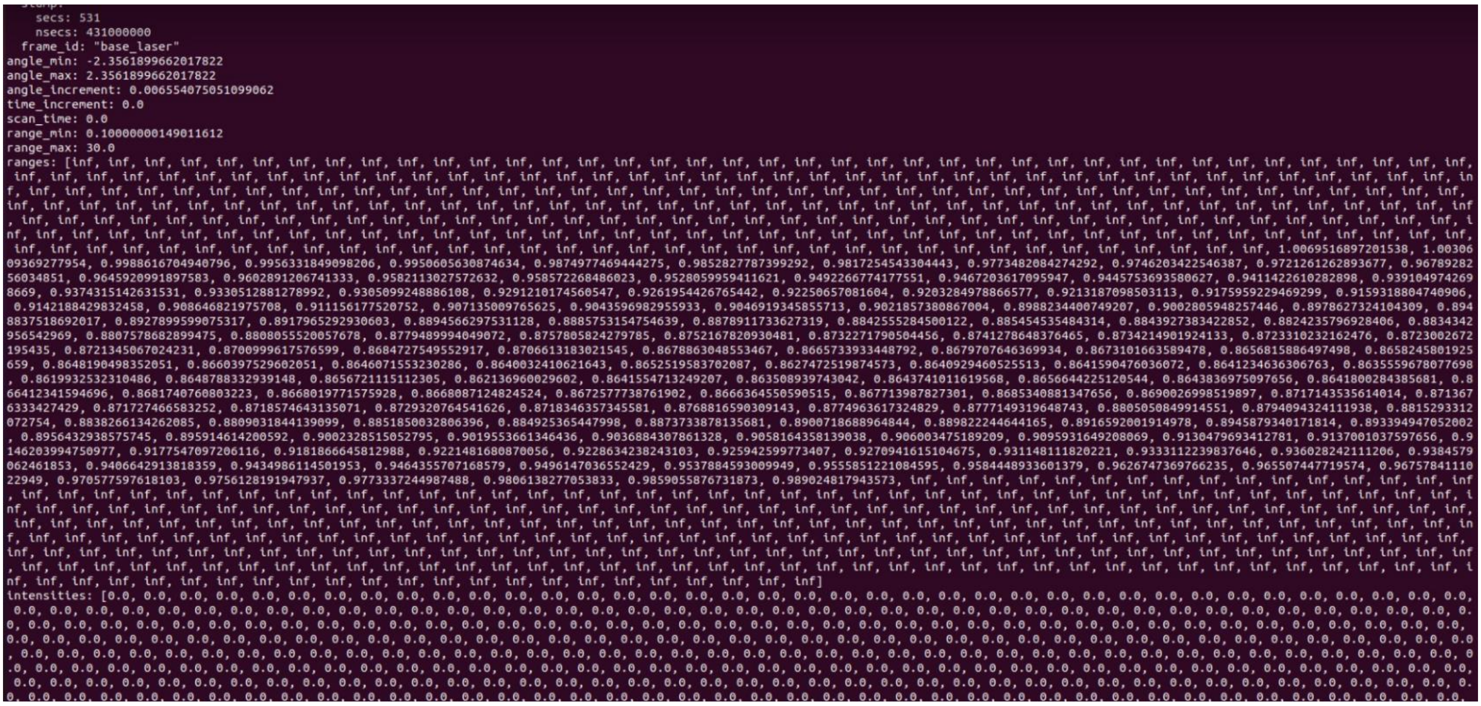
Συνεχίζοντας , τοποθετήθηκε ένα τετράγωνο εμπόδιο μπροστά στο ρομπότ στην προσομοίωση του gazebo. Αυτό είχε ως αποτέλεσμα να εμφανιστεί επίσης και στο rviz ένα περίγραμμα του εμποδίου που έπιανε ο αισθητήρας ,με μια κόκκινη διακεκομμένη γραμμή.

Παρακάτω φαίνεται η προσθήκη εμποδίου στο gazebo : ***(screenshots 10)***



Ξανά εκτελέστηκε ύστερα η εντολή *$ rostopic echo /scan* , αλλά αυτήν τη φορά αντίθετα με πριν , εφόσον ο αισθητήρας LiDAR έβρισκε το τετράγωνο εμπόδιο, ορισμένες τιμές του range ήταν διάφορες του 0 και αντιστοιχούσαν στο σχήμα που ανίχνευε ο αισθητήρας.

Παρακάτω φαίνεται το αποτέλεσμα της εντολής rostopic echo /scan με προσθήκη εμποδίου: ***(screeshot-11)***



Τέλος, εκτελέστηκαν οι εντολές *$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py* και *$ rostopic echo /husky_velocity_controller/cmd_vel* με σκοπό να γίνει ο ταυτόχρονος τηλεχειρισμός του ρομπότ από το πληκτρολόγιό μας και η παρακολούθηση των αποτελεσμάτων των εντολών σε πραγματικό χρόνο.

Παρακάτω φαίνονται τα αποτελέσματα των παραπάνω εντολών: ***(screenshot- 12)***