Introduction to Robotic Operating System (ROS)

by W.S. Ooi

Simultaneous Localization and Mapping (SLAM)

- Simultaneous localization and mapping (SLAM) is the process by which a mobile robot can construct a map of an unknown environment and simultaneously compute its location using the map.

Simultaneous Localization and Mapping (SLAM)

- **$ roscore**
- **$ roslaunch turtlebot3_gazebo_ros turtlebot3_world.launch**
- **$ roslaunch turtlebot3_slam turtlebot3_slam.launch**
- **$ rosrun rviz rviz -d `rospack find turtlebot3_slam`/rviz/turtlebot3_slam.rviz**
  - Add map and set the topic to /map
  - Add RobotModel
- **$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch**
  - Move the robot to generate the map
- **$ rosrun map_server map_saver –f ~/mymap**
  - Kill off slam, keyboard, and rviz
- **$ roslaunch turtlebot3_navigation turtlebot3_navigation.launch map_file:=$HOME/mymap.yaml**
- **$ rosrun rviz rviz –d ‘rospack find turtlebot3_navigation’/rviz/turtlebot3_nav.rviz**
  - Add map and set the topic to /map
  - Add RobotModel
  - Select 2D Pose Estimate
  - Select 2D Nav Goa

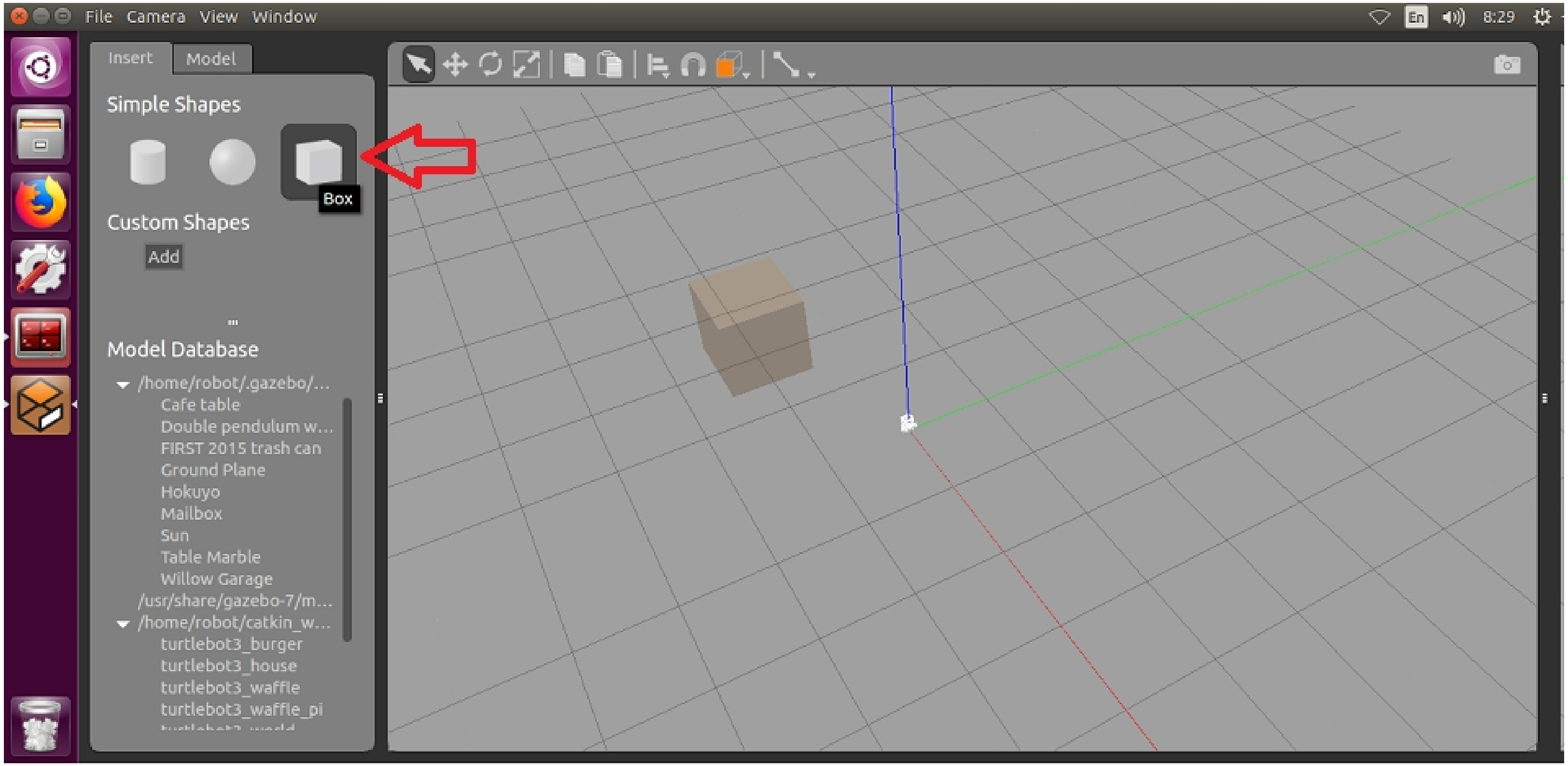
https://youtu.be/MAt-snbFj4EI

Create Empty World

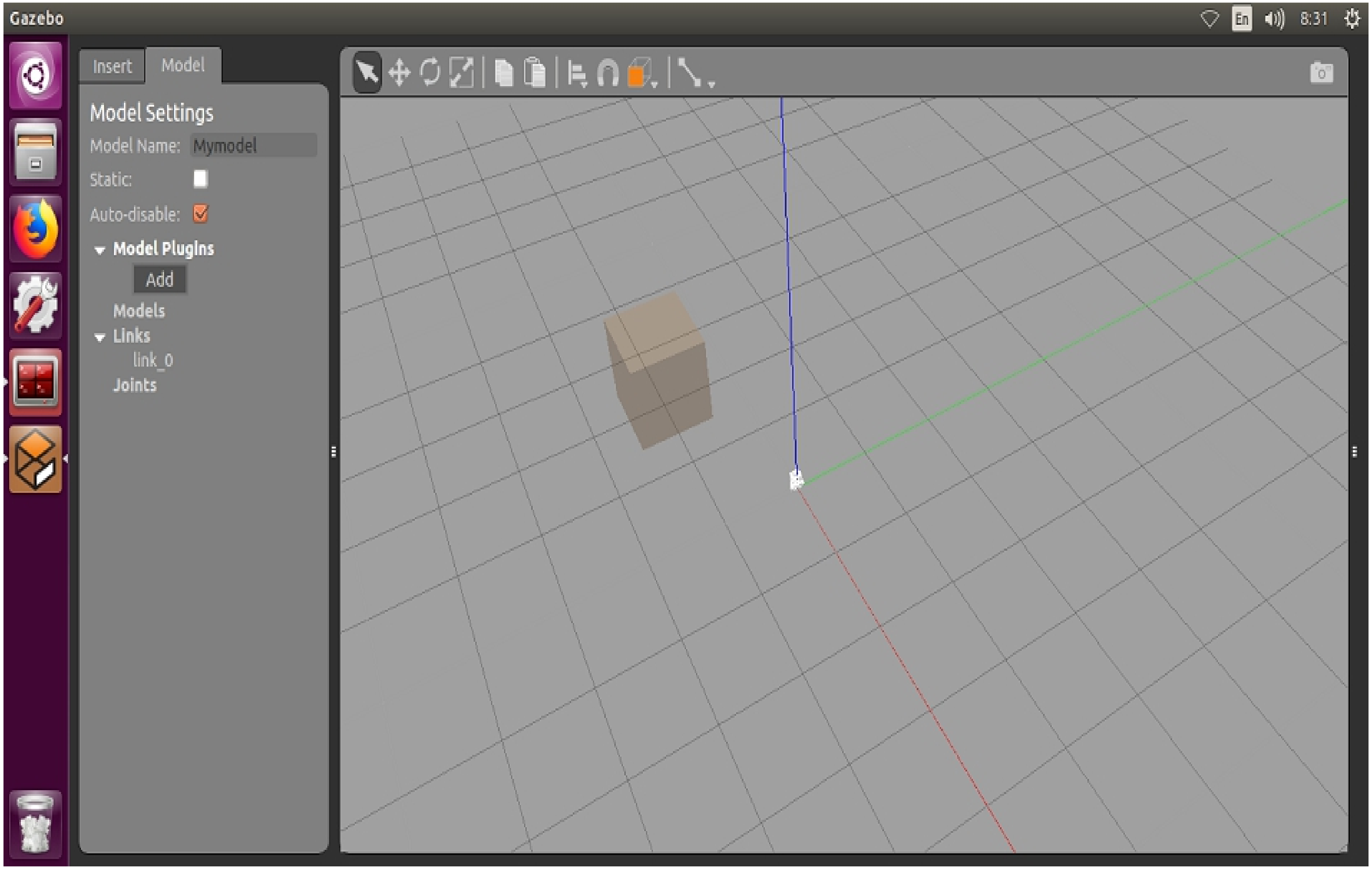
- **roslaunch turtlebot3_gazebo_ros turtlebot3_empty_world.launch**

How to Save Custom Model?

- 1) Go to Model Editor. Insert the object from the shapes provided. Double click the object and change the size of the object.



2) Change the Model Name



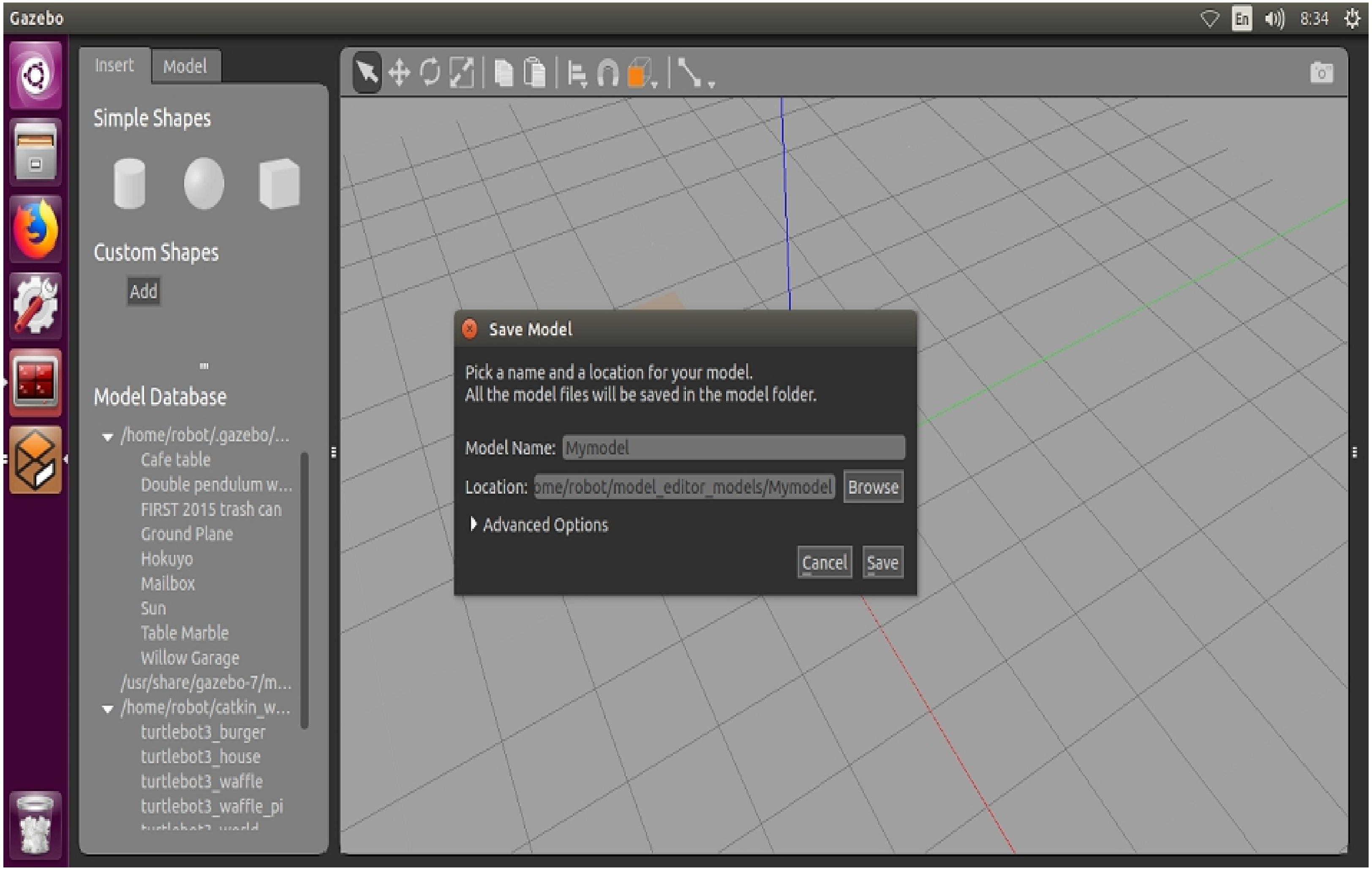
Model Settings

Model Name: Mymodel

Static: □

Auto-disable: ☑

3) Save the Model



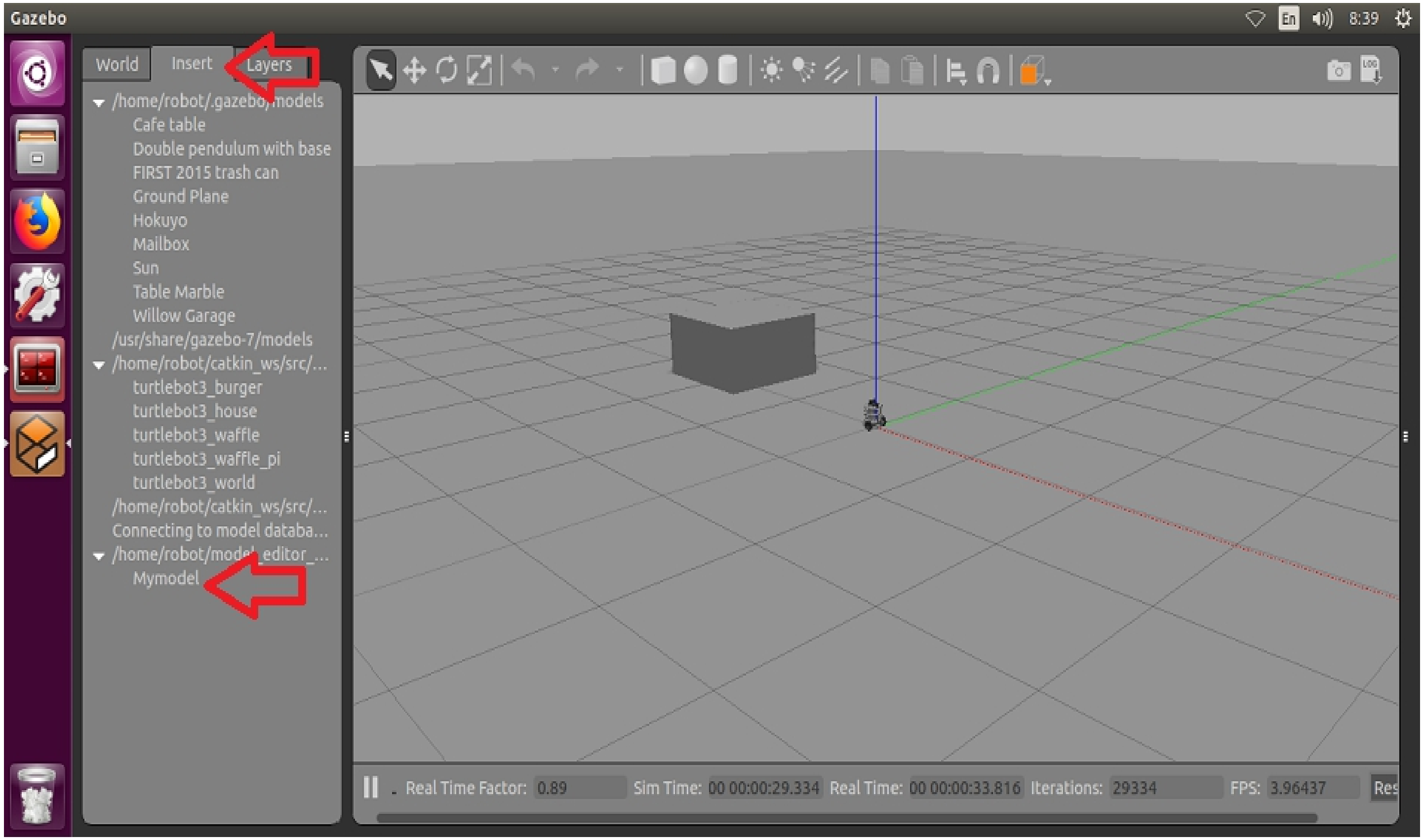
Save Model

Pick a name and a location for your model.
All the model files will be saved in the model folder.

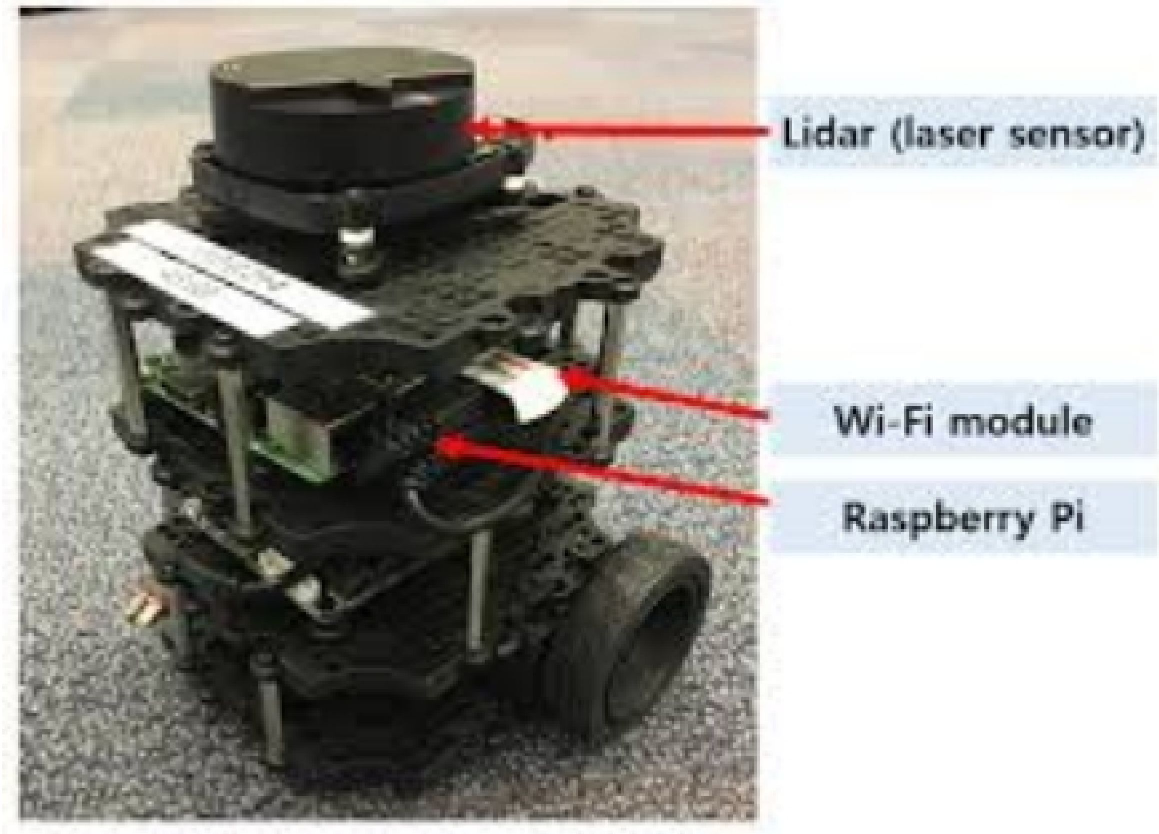
Model Name: Mymodel

Location: ome/robot/model_editor_models/Mymodel

- 4) Close Gazebo and run " roslaunch turtlebot3_gazebo_ros turtlebot3_empty_world.launch".
- Then, insert your model.



Read Data from Laser



Read Data from Laser

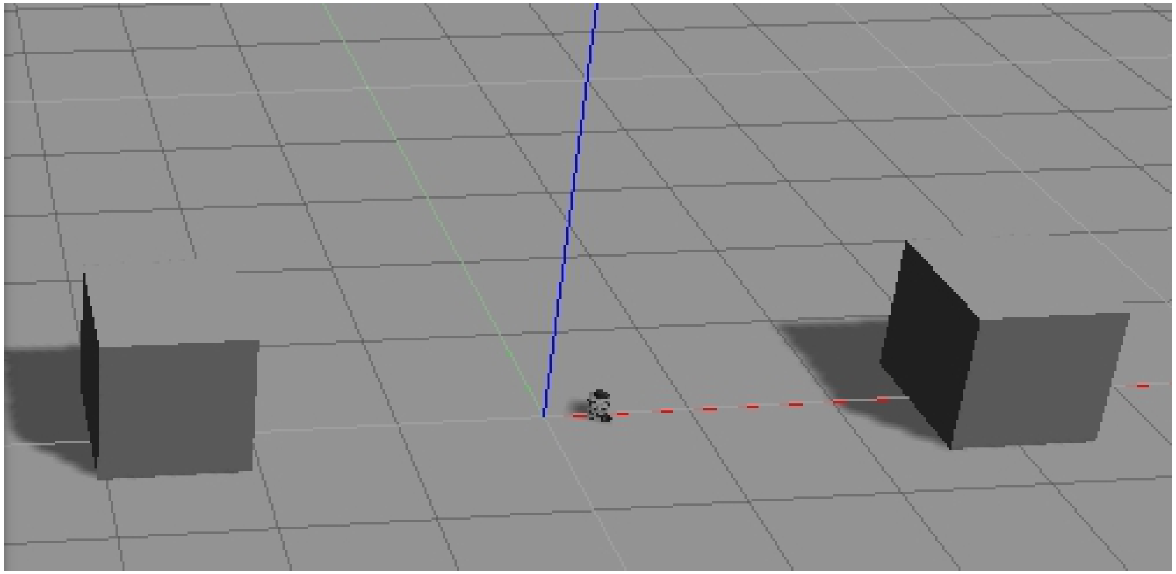
- Create an empty document and name it “mylaser1”. Then, write:

```
#!/usr/bin/env python
import rospy
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan
rospy.init_node('laser_readings')
rate=rospy.Rate(1)
while not rospy.is_shutdown():
    msg=rospy.wait_for_message("scan",LaserScan)
    print len(msg.ranges)
    print msg.ranges
    rate.sleep()
```

Read Data from Laser

-
  ```
  robot@vm:~$ roscore
  ```

-
  ```
  robot@vm:~$ roslaunch turtlebot3_gazebo_ros turtlebot3_empty_world.launch
  ```
- **Place 2 objects:**



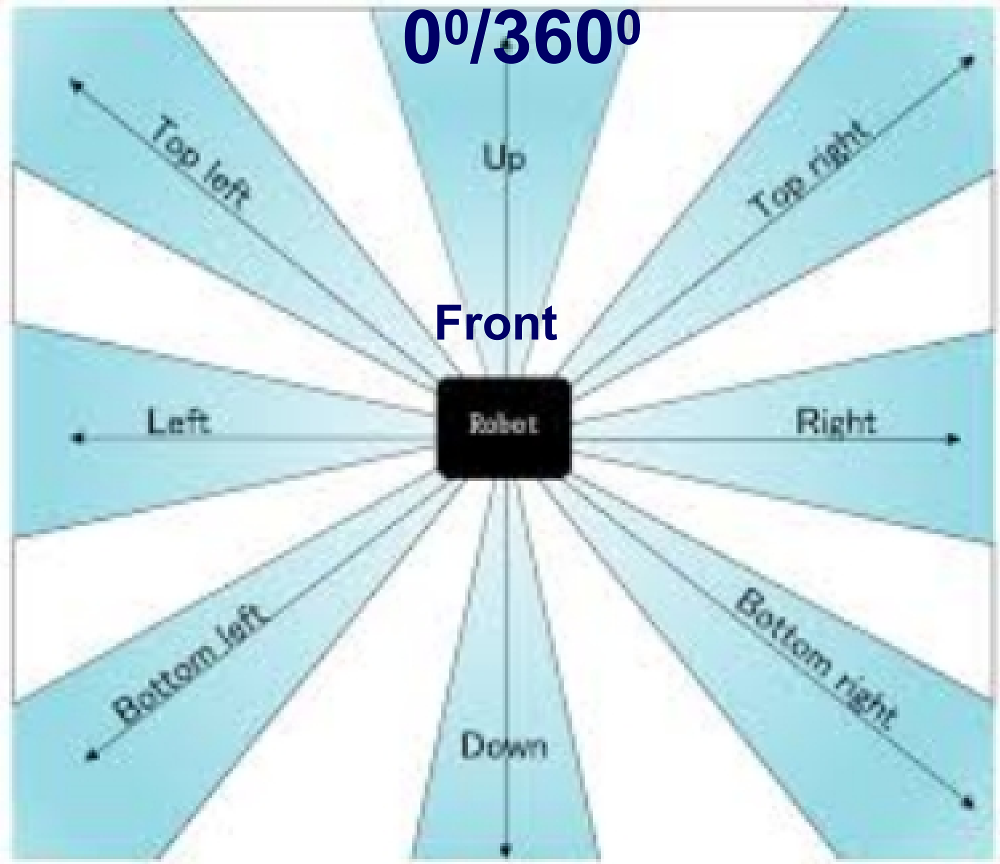
Read Data from Laser

```
robot@vm:~$ chmod +x /home/robot/mylaser
robot@vm:~$ /home/robot/mylaser
robot@vm:~$ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
```

https://youtu.be/zs1kpGcL9Ls

Read Data from Laser

- To obtain the middle value from laser
  - **print msg.ranges[1]**



More advanced coding

```
#!/usr/bin/env python
import rospy
from geometry_msgs.msg import Twist
from sensor_msgs.msg import LaserScan

def move():
    rospy.init_node('robot_cleaner', anonymous=True) # Starts new node
    velocity_publisher = rospy.Publisher('cmd_vel', Twist, queue_size=10)
    vel_msg = Twist()

    print("Let's move your robot")   #Receiveing the user's input
    speed = 0.2
    distance = 0.4
    isForward = input("Foward?: ")  #True or False
    if(isForward):
        vel_msg.linear.x = abs(speed)
    else:
        vel_msg.linear.x = -abs(speed)
```

Continue..

```
#Since we are moving just in x-axis
    vel_msg.linear.y = 0
    vel_msg.linear.z = 0
    vel_msg.angular.x = 0
    vel_msg.angular.y = 0
    vel_msg.angular.z = 0

    while not rospy.is_shutdown():
        t0 = rospy.Time.now().to_sec() #Set current time for dist calc
        current_distance = 0

        #Loop to move the turtle in an specified distance
        while(current_distance < distance):
            velocity_publisher.publish(vel_msg)  #Publish the velocity
            t1=rospy.Time.now().to_sec()  #Take actual time to vel calc
            current_distance= speed*(t1-t0)  #Calculates distance
            scan_msg=rospy.wait_for_message("scan",LaserScan)
            a=scan_msg.ranges
            print a
```

Continue…

```
#After the loop, stops the robot
        vel_msg.linear.x = 0
        #Force the robot to stop
        velocity_publisher.publish(vel_msg)
        # If we press control + C, the node will stop.
        rospy.spin()

if __name__ == '__main__':
    try:
        #Testing our function
        move()
    except rospy.ROSInterruptException: pass
```

To connect RVIZ to Gazebo

- roslaunch turtlebot3_gazebo_ros turtlebot3_gazebo_rviz.launch