



1.- Actualizar ordenador	2
2.- Instalar Gazebo	2
3.- Preparar entorno ROS2	3
4.- Lanzar Gazebo.....	3
5.- Dependencias y paquetes Gazebo/ROS2	4
6.- Modificar nuestro modelo.....	5
7.- Instalar dependencias con rosdep.....	7
8.- Primer lanzamiento del modelo.....	8
9.- Arreglar el modelo	9
10.- Finalizacion del modelo.....	10

1.- Actualizar ordenador

Obviamente, haces de hacer cualquier cosa, habra que actualizar por completo el ordenador. Para ello usaremos:

```
sudo apt update && sudo apt upgrade -y
```

```
iabd12@iabd12-VirtualBox:~$ sudo apt update && sudo apt upgrade -y
[sudo] password for iabd12:
Hit:1 http://es.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://es.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://security.ubuntu.com/ubuntu jammy-security InRelease
```

2.- Instalar Gazebo

Ahora, con el ordenador actualizado, hay que instalar gazebo. Para ello, puedes probar con:

```
sudo apt install gazebo11
```

Si este no funciona, simplemente con:

```
sudo apt install gazebo
```

```
iabd12@iabd12-VirtualBox:~$ sudo apt install gazebo11
[sudo] password for iabd12:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package gazebo11
iabd12@iabd12-VirtualBox:~$ sudo apt install gazebo
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and
linux-headers-6.8.0-49-generic linux-hwe-6.8-headers-
linux-hwe-6.8-tools-6.8.0-49 linux-image-6.8.0-49-gen
linux-modules-6.8.0-49-generic linux-modules-extra-6.
linux-tools-6.8.0-49-generic
```

3.- Preparar entorno ROS2

Ya con gazebo instalado, tendremos que configurar el entorno de ROS2 para que funcione:

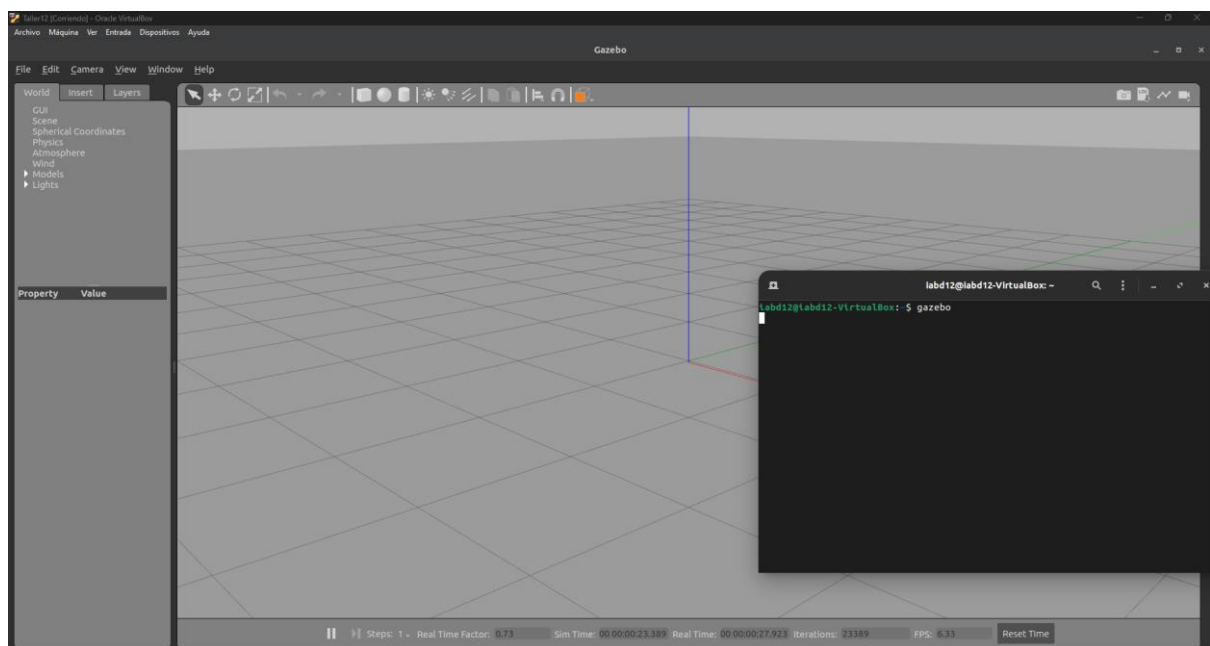
```
source /opt/ros/humble/setup.bash
```

```
iabd12@iabd12-VirtualBox:~$ source /opt/ros/humble/setup.bash  
iabd12@iabd12-VirtualBox:~$
```

4.- Lanzar Gazebo

Con el entorno ya preparado y configurado. Probamos a lanzar gazebo:

```
gazebo
```



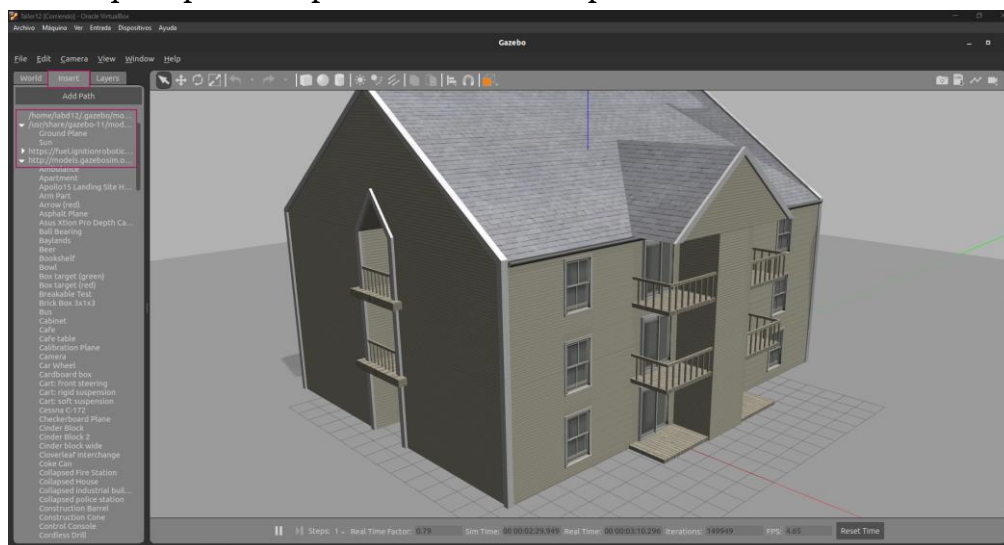
5.- Dependencias y paquetes Gazebo/ROS2

Si conseguimos lanzar gazebo sin problemas, podremos instalar las dependencias y paquetes necesarios para lo siguiente en la tarea. Para ello:

```
sudo apt install ros-humble-gazebo-ros-pkgs -y
```

```
labd12@labd12-VirtualBox:~$ sudo apt install ros-humble-gazebo-ros-pkgs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer req
  linux-headers-6.8.0-49-generic linux-hwe-6.8-headers-6.8.0-49
  linux-hwe-6.8-tools-6.8.0-49 linux-image-6.8.0-49-generic
  linux-modules-6.8.0-49-generic linux-modules-extra-6.8.0-49-generic
  linux-tools-6.8.0-49-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  comerr-dev freeglut3-dev krb5-multidev libavdevice-dev libavfilter-dev
```

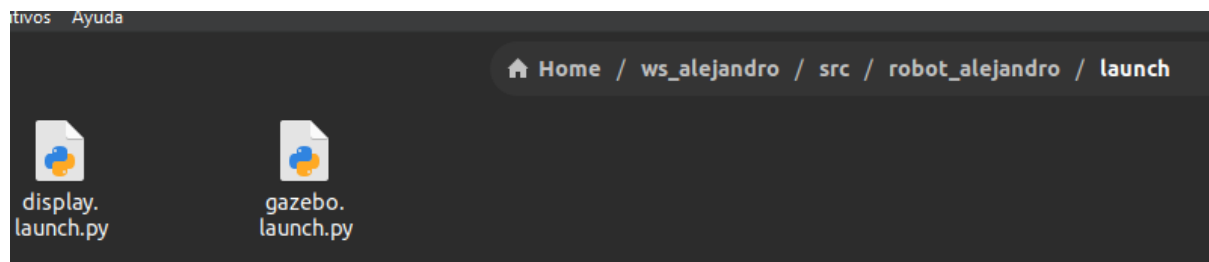
Después de instalarlo, si volvemos a entrar en Gazebo, podremos cargar modelos dados por la misma compañía. Entrando en “Insert” y eligiendo uno de los nombres que aparecen podremos hacerlo perfectamente.



6.- Modificar nuestro modelo.

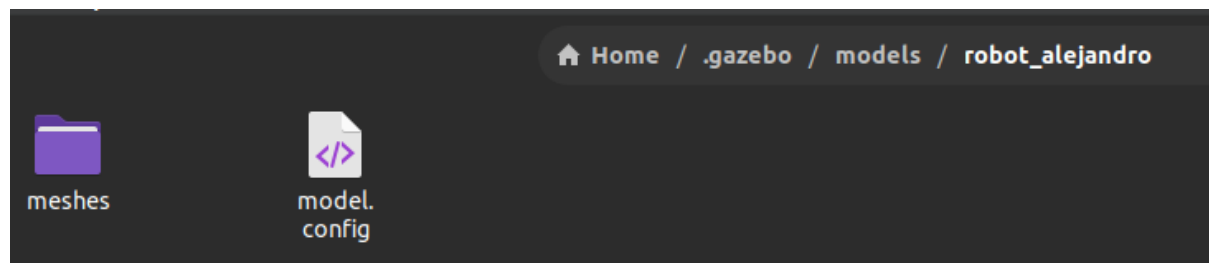
Ahora, tendremos que actualizar nuestro modelo personal, para poder visualizarlo con gazebo. Para ello, descargaremos el archivo “gazebo.launch.py” dado, y lo pondremos en la carpeta “launch” de nuestro modelo. La ruta debería ser así:

```
~/home/<carpeta_workspace>/src/<nombre_modelo>/launch
```



Aparte, en la nueva carpeta que se nos ha creado con gazebo, tendremos que crear una carpeta que se llame “models”, y dentro de este, otra que se llame como nuestro modelo. Dentro de este necesitaremos pegar la carpeta “meshes” de nuestro modelo de ROS2, y crear un archivo llamado “model.config” el cual tendrá nuestra configuración.

```
~/home/.gazebo/models/<nombre_modelo>/meshes
```



Cambios para model.config:

<name> “nombre_modelo” </name>

<author>

<name> “nombre_creador” </name>

<email> “email_creador” </email>

</author>

```
1 <?xml version="1.0"?>
2 <model>
3   <name>                </name>
4   <version>1.0</version>
5   <sdf version="1.6">Seguidor_linea_robot.urdf</sdf>
6   <author>
7     <name>              </name>
8     <email>              </email>
9   </author>
10  <description>
11    Muestra un robot en ros2
12  </description>
13 </model>
```

7.- Instalar dependencias con rosdep

Por primera vez, sino lo habias hecho ya antes (yo si), deberas lanzar:

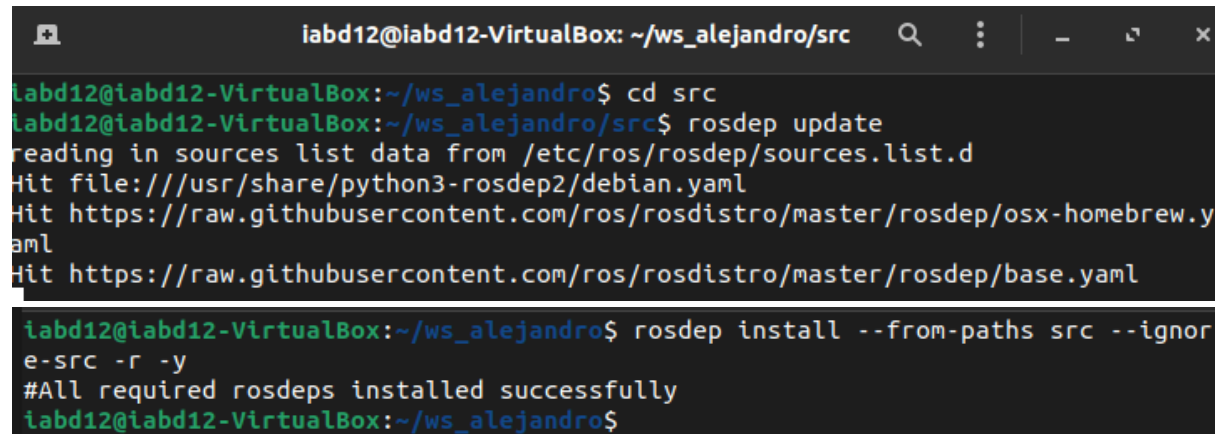
```
sudo rosdep init
```

Despues de lanzarlo o ya habiendolo lanzado:

```
rosdep update
```

Por ultimo, despues de actualizar:

```
rosdep install --from-paths src --ignore-src -r -y
```

A terminal window titled 'iabd12@iabd12-VirtualBox: ~/ws_alejandro/src'. The terminal shows the following commands and output:

```
iabd12@iabd12-VirtualBox:~/ws_alejandro$ cd src
iabd12@iabd12-VirtualBox:~/ws_alejandro/src$ rosdep update
Reading in sources list data from /etc/ros/rosdep/sources.list.d
Hit file:///usr/share/python3-rosdep2/debian.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/osx-homebrew.yaml
Hit https://raw.githubusercontent.com/ros/rosdistro/master/rosdep/base.yaml

iabd12@iabd12-VirtualBox:~/ws_alejandro$ rosdep install --from-paths src --ignore-src -r -y
#All required rosdeps installed successfully
iabd12@iabd12-VirtualBox:~/ws_alejandro$
```

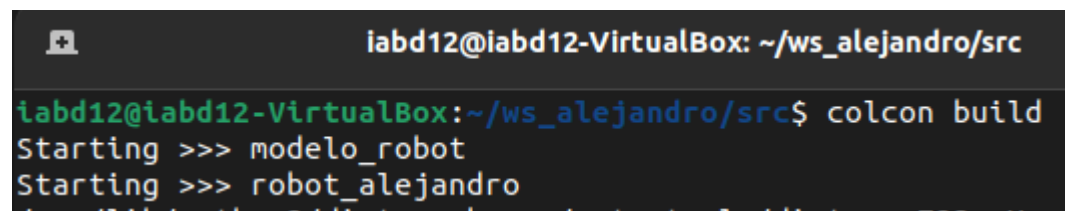
8.- Primer lanzamiento del modelo

Ya con todo cambiado, probamos a lanzarlo por primera vez en nuestro workspace:

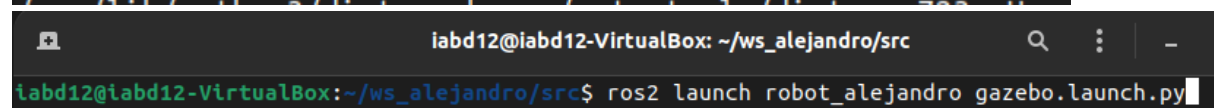
```
colcon build

source install/setup.bash

ros2 launch <nombre_modelo> <nombre_lanzador.py>
```



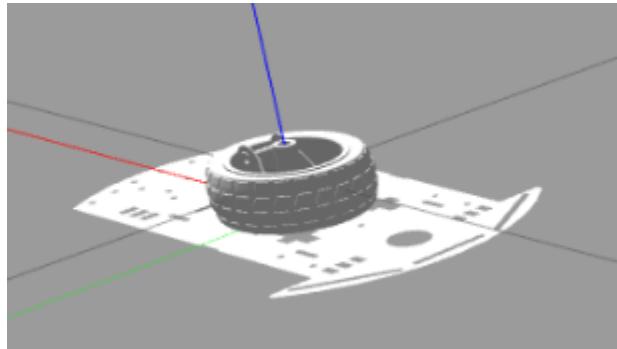
A terminal window titled 'iabd12@iabd12-VirtualBox: ~/ws_alejandro/src'. The prompt is 'iabd12@iabd12-VirtualBox:~/ws_alejandro/src\$'. The user enters 'colcon build'. The output shows 'Starting >>> modelo_robot' and 'Starting >>> robot_alejandro'.



A terminal window titled 'iabd12@iabd12-VirtualBox: ~/ws_alejandro/src'. The prompt is 'iabd12@iabd12-VirtualBox:~/ws_alejandro/src\$'. The user enters 'ros2 launch robot_alejandro gazebo.launch.py'.

9.- Arreglar el modelo

Si todo ha ido como esperábamos, nos deberá aparecer esto:



Para ello, en el archivo “Seguidor_linea_robot.urdf” dentro de la carpeta “urdf” de nuestro modelo de ros2 (workspace). Tendremos que añadir lo siguiente en cada etiqueta que se llame “<inertial>”:

```
<inertial>
  <inertia
    ixx="0.0001"
    ixy="0"
    ixz="0"
    iyy="0.0003"
    iyz="0"
    izz="0.0002" />
</inertial>
```

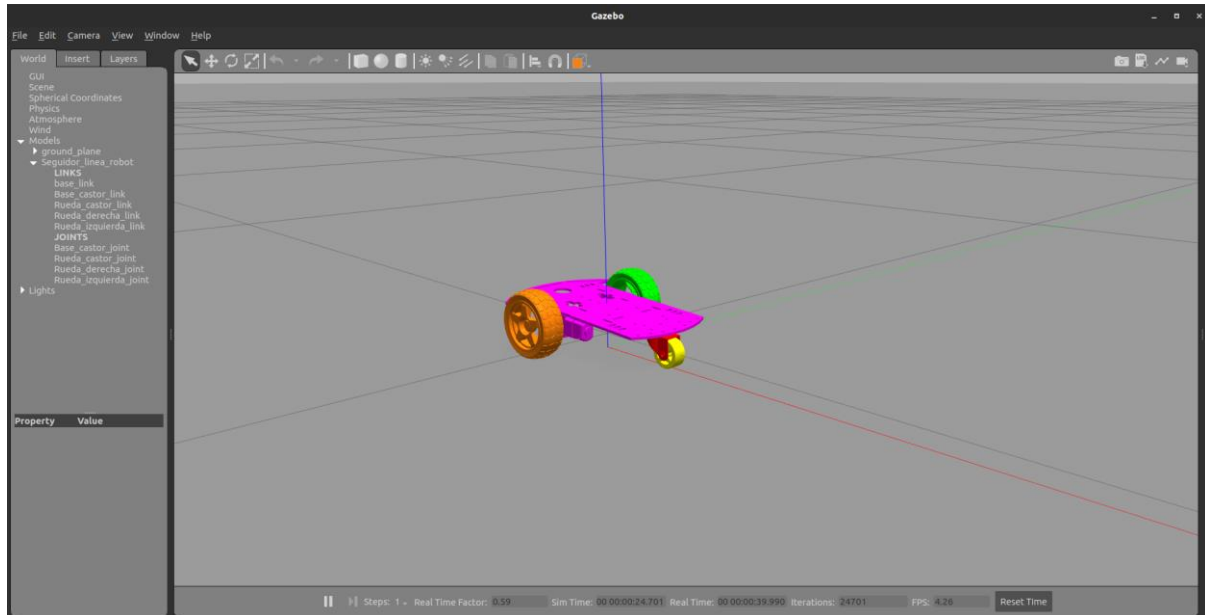
```
<link
  name="base_link">
  <inertial>
    <origin
      xyz="0.010085 -7.8763E-06 -0.0066048"
      rpy="0 0 0" />
    <mass
      value="0.12859" />
    <inertia
      ixx="0.0001"
      ixy="0"
      ixz="0"
      iyy="0.0003"
      iyz="0"
      izz="0.0002" />
    </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://robot_alejandro/meshes/base_link.STL" />
      </mesh>
    </geometry>
  </visual>
</link>
```

10.- Finalización del modelo

Cuando cambiemos el archivo por completo. Volveremos a hacer:

```
colcon build
```

Y lo volveremos a lanzar.



Ahora, podremos aplicar fuerzas al modelo para fuerza y torque y jugar un poco con ello.

