

EJERCICIO
PRACTICO 3
ROS2 Y RVIZ

Antes de hacer cualquier cosa, lo primero sería actualizar por completo el ordenador.

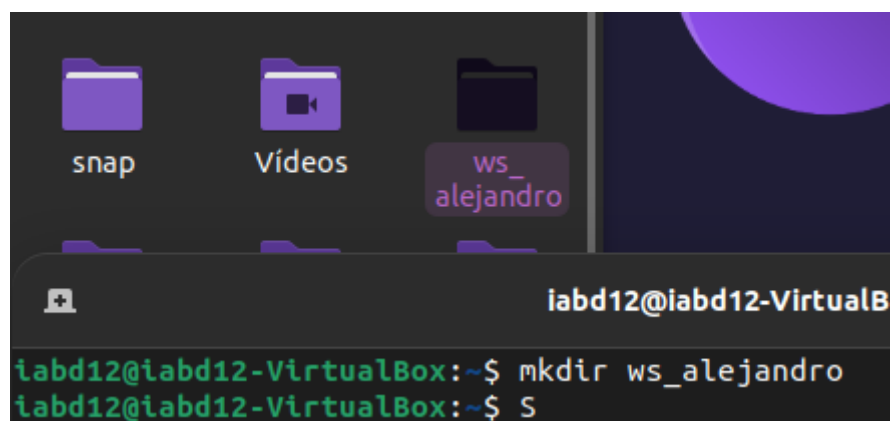
Simplemente con el siguiente comando lo hacemos todo:

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

```
iabd12@iabd12-VirtualBox: ~  
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://security.ubuntu.com/ubuntu jammy-security InRelease  
Hit:3 http://es.archive.ubuntu.com/ubuntu jammy-updates InRelease  
Hit:4 http://es.archive.ubuntu.com/ubuntu jammy-backports InRelease  
Hit:5 http://packages.ros.org/ros2/ubuntu jammy InRelease
```

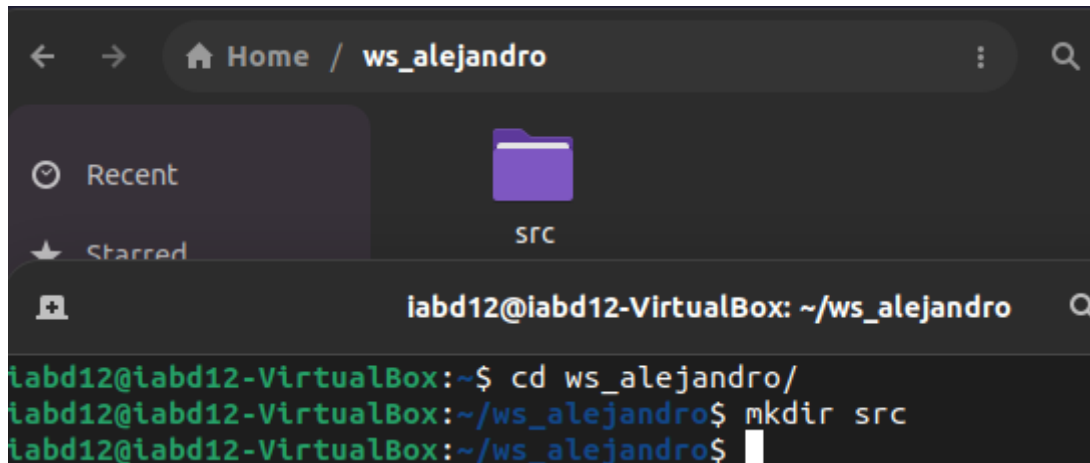
Ahora, tendremos que crearnos una carpeta en la cual vallamos a trabajar:

```
mkdir <nombre_proyecto>
```



Ahora dentro de la carpeta que hemos creado, vamos a crear otra la cual guardara nuestros proyectos:

```
cd <nombre_proyecto>  
mkdir src
```

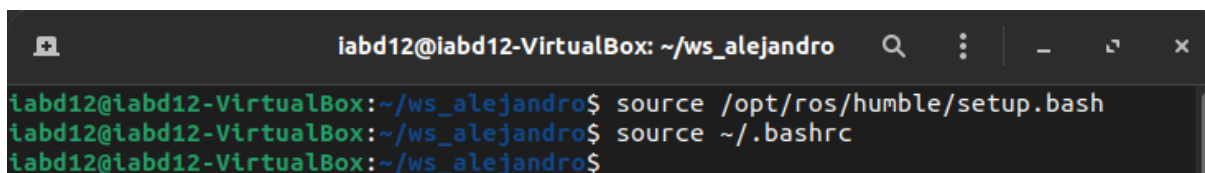


Ahora, indicamos que este directorio, va a trabajar con ROS, para ello, lanzaremos el siguiente comando:

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

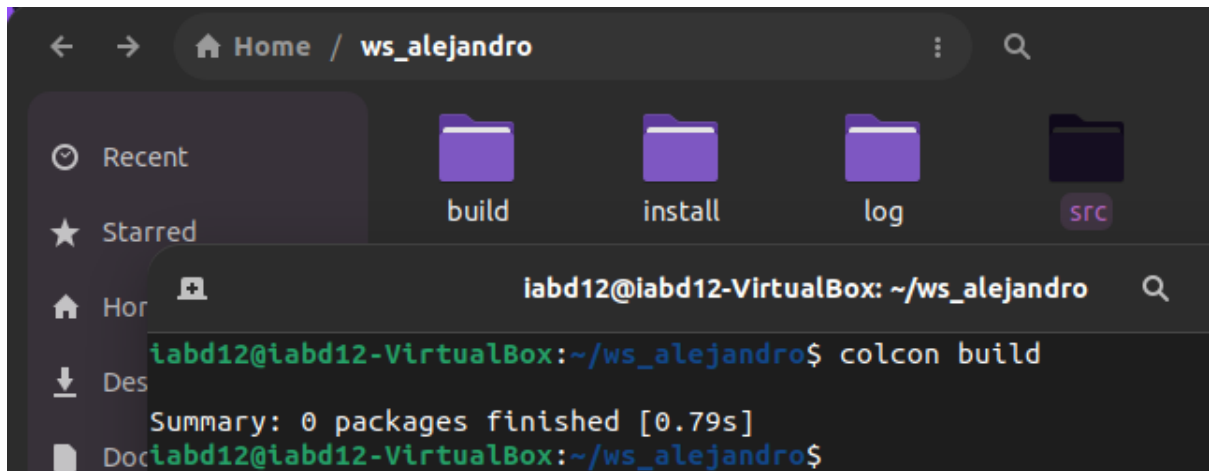
Y después, refrescamos el archivo, para que podamos utilizarlo:

```
source ~/.bashrc
```



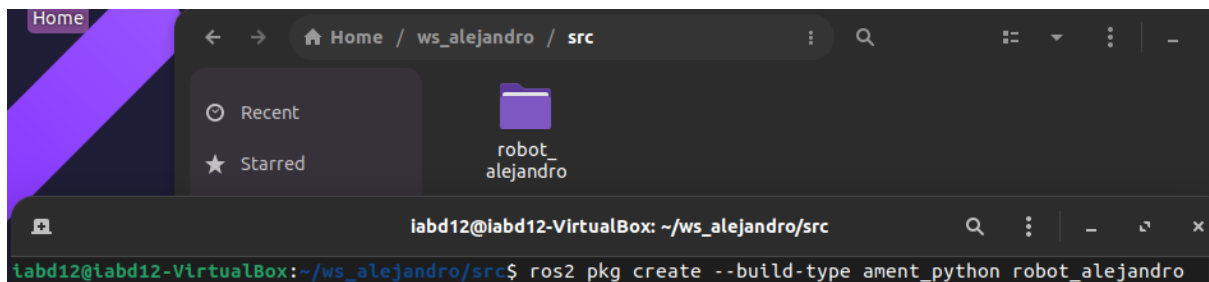
Ya con esto, hacemos un colcon build para preparar el entorno:

```
colcon build
```



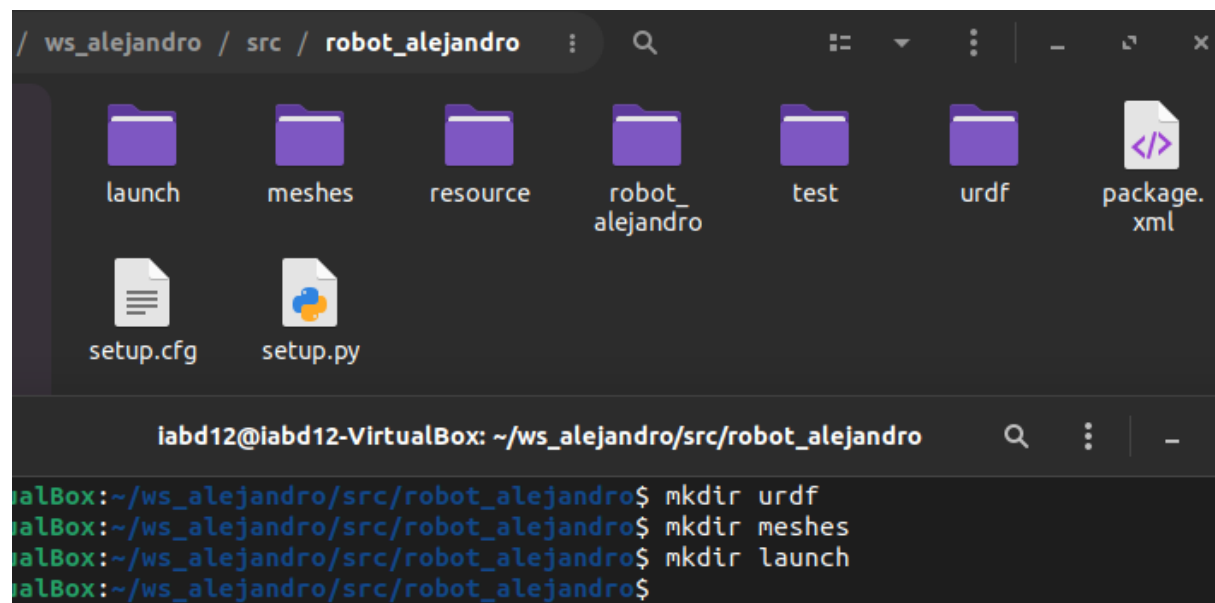
Y creamos el entorno, el cual guardara el modelo que vamos a visualizar:

```
ros2 pkg create --build-type ament_python <nombre_proyecto>
```



Ya tenemos el entorno, pero nos falta de crear los directorios (y archivos), los cuales son el modelo de las piezas del coche. Por ello, tendremos que entrar en nuestro proyecto y lanzamos lo siguiente:

```
cd robot_alejandro  
  
mkdir urdf  
mkdir meshes  
mkdir launch
```



Descargamos el modelo de practica dado, y copiamos en nuestras carpetas, los archivos que se encuentran dentro de estas mismas.

```
iabd12@iabd12-VirtualBox: ~/ws_alejandro/src/robot_alejandro

iabd12@iabd12-VirtualBox:~/ws_alejandro/src/robot_alejandro$ tree
.
├── launch
│   └── display.launch.py
├── meshes
│   ├── Base_castor_link.STL
│   ├── base_link.STL
│   ├── Rueda_castor_link.STL
│   ├── Rueda_derecha_link.STL
│   └── Rueda_izquierda_link.STL
├── package.xml
├── resource
│   └── robot_alejandro
├── robot_alejandro
│   └── __init__.py
├── setup.cfg
├── setup.py
├── test
│   ├── test_copyright.py
│   ├── test_flake8.py
│   └── test_pep257.py
└── urdf
    └── Seguidor_linea_robot.urdf
```

Ahora, nos instalaremos un paquete de ROS, el cual nos permitirá mover las articulaciones del modelo:

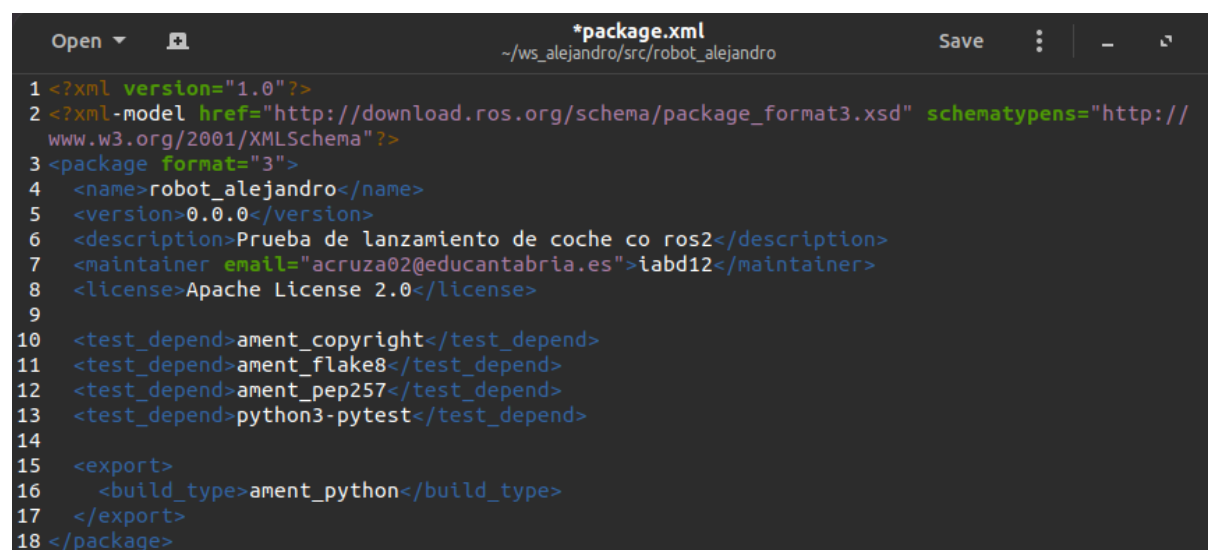
```
sudo apt install ros-humble-joint-state-publisher-gui
```

```
iabd12@iabd12-VirtualBox: ~/ws_alejandro/src/robot_alej...

iabd12@iabd12-VirtualBox:~/ws_alejandro/src/robot_alejandro$ sudo apt install
ros-humble-joint-state-publisher-gui
```

Ahora, dentro de los archivos que tenemos, tendremos que adaptarlos, ya que vienen por defecto y no tienen lo necesario. Empezaremos con el "package.xml", en el cual hay que cambiar estas líneas (por supuesto, con vuestras especificaciones):

```
<name>robot_alejandro</name>
<version>0.0.0</version>
<description>Prueba de lanzamiento de coche con ros2</description>
<maintainer email="acruza02@educantabria.es">Alejandro</maintainer>
<license>Apache License 2.0</license>
```



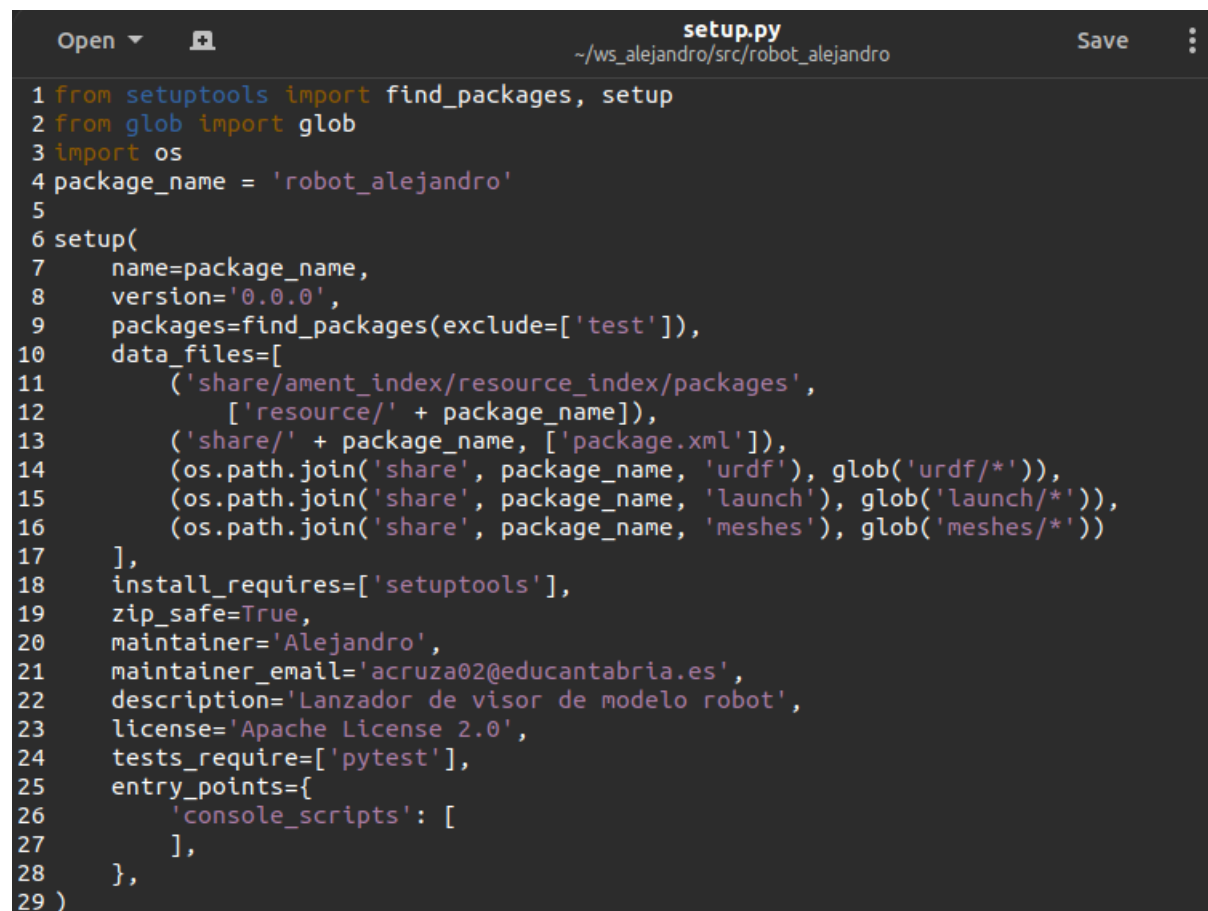
The screenshot shows a code editor window titled "*package.xml" with the file path "~/ws_alejandro/src/robot_alejandro". The editor contains the following XML code:

```
1 <?xml version="1.0"?>
2 <?xml-model href="http://download.ros.org/schema/package_format3.xsd" schematypens="http://
  www.w3.org/2001/XMLSchema"?>
3 <package format="3">
4   <name>robot_alejandro</name>
5   <version>0.0.0</version>
6   <description>Prueba de lanzamiento de coche co ros2</description>
7   <maintainer email="acruza02@educantabria.es">iabd12</maintainer>
8   <license>Apache License 2.0</license>
9
10  <test_depend>ament_copyright</test_depend>
11  <test_depend>ament_flake8</test_depend>
12  <test_depend>ament_pep257</test_depend>
13  <test_depend>python3-pytest</test_depend>
14
15  <export>
16    <build_type>ament_python</build_type>
17  </export>
18 </package>
```

También hay que cambiar el "setup.py", para que haga uso de las mallas, el lanzador y el organizador (urdf).

Entonces, tendremos que importar glob, y os. Y dentro de "data_files", añadir al final las siguientes líneas:

```
(os.path.join('share', package_name, 'urdf'), glob('urdf/*')),  
(os.path.join('share', package_name, 'launch'), glob('launch/*')),  
(os.path.join('share', package_name, 'meshes'), glob('meshes/*')),
```

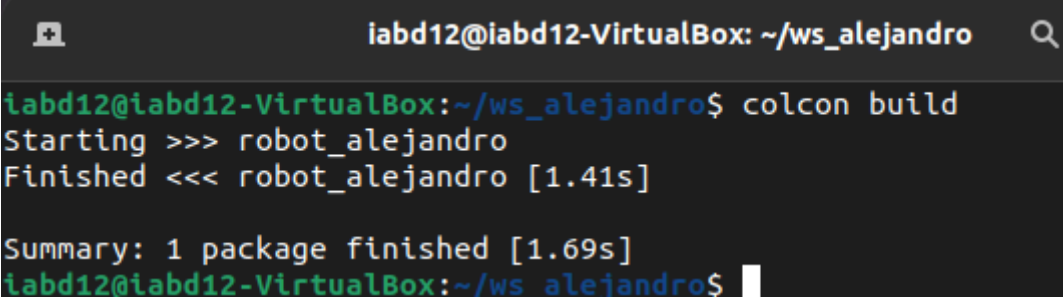


The screenshot shows a code editor window titled "setup.py" with the file path "~/ws_alejandro/src/robot_alejandro". The editor contains the following Python code:

```
1 from setuptools import find_packages, setup  
2 from glob import glob  
3 import os  
4 package_name = 'robot_alejandro'  
5  
6 setup(  
7     name=package_name,  
8     version='0.0.0',  
9     packages=find_packages(exclude=['test']),  
10    data_files=[  
11        ('share/ament_index/resource_index/packages',  
12         ['resource/' + package_name]),  
13        ('share/' + package_name, ['package.xml']),  
14        (os.path.join('share', package_name, 'urdf'), glob('urdf/*')),  
15        (os.path.join('share', package_name, 'launch'), glob('launch/*')),  
16        (os.path.join('share', package_name, 'meshes'), glob('meshes/*'))  
17    ],  
18    install_requires=['setuptools'],  
19    zip_safe=True,  
20    maintainer='Alejandro',  
21    maintainer_email='acruza02@educantabria.es',  
22    description='Lanzador de visor de modelo robot',  
23    license='Apache License 2.0',  
24    tests_require=['pytest'],  
25    entry_points={  
26        'console_scripts': [  
27            ],  
28    },  
29 )
```


Volvemos a la carpeta raíz, y volvemos a lanzar el “colcon build”:

```
colcon build
```

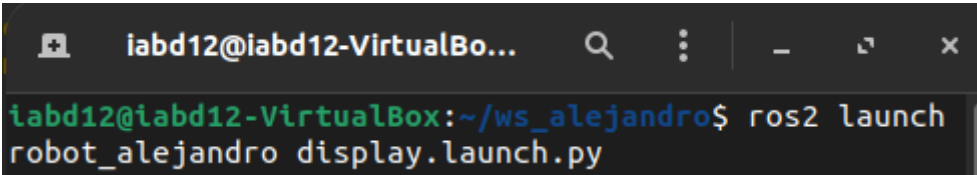
A terminal window titled 'iabd12@iabd12-VirtualBox: ~/ws_alejandro' showing the output of the 'colcon build' command. The output indicates that the 'robot_alejandro' package was built successfully in 1.41 seconds. A summary shows that 1 package was finished in 1.69 seconds. The prompt returns to the user's shell.

```
iabd12@iabd12-VirtualBox: ~/ws_alejandro
iabd12@iabd12-VirtualBox:~/ws_alejandro$ colcon build
Starting >>> robot_alejandro
Finished <<< robot_alejandro [1.41s]

Summary: 1 package finished [1.69s]
iabd12@iabd12-VirtualBox:~/ws_alejandro$
```

Y con lo siguiente, debería funcionar:

```
ros2 launch <nombre_proyecto> <nombre_python_launcher>
```

A terminal window titled 'iabd12@iabd12-VirtualBo...' showing the command 'ros2 launch robot_alejandro display.launch.py' being entered at the prompt. The window has standard Ubuntu window controls (minimize, maximize, close) and a search icon.

```
iabd12@iabd12-VirtualBo...
iabd12@iabd12-VirtualBox:~/ws_alejandro$ ros2 launch
robot_alejandro display.launch.py
```

Por último, aunque se haya lanzado la aplicación, puede que no se vea el modelo. Para ello, asegúrate de que, dentro del visualizador, en la zona "Global Options" => "Fixed Frame", sea la opción "base_link".

Aparte, debes añadir "RobotModel", el cual, puedes hacerlo fácilmente en la zona inferior izquierda con el botón "Add".

Por último, dentro de "RobotModel" que acabamos de añadir, en "Description Topic", debes seleccionar "/robot_description".

