MEMORIA DE PRÁCTICAS EXTERNAS

Alberto León Luengo

ASOCIACIÓN DE ROBÓTICA E INTELIGENCIA ARTIFICIAL JDEROBOT

Programación de Robots desde Navegador Web

18 de Septiembre de 2025 - 10 de Febrero de 2026

ÍNDICE

[] 1. INTRODUCCIÓN

En esta memoria explicaré detalladamente todo los conceptos, tanto teóricos como prácticos, que he aprendido durante la realización de mis Prácticas Externas con la empresa JdeRobot a lo largo del primer cuatrimestre del curso académico 2025-2026, concretamente entre el 18 de Septiembre de 2025 y el 10 de Febrero de 2026, y tutorizadas por **José María Cañas Plaza**, profesor que me ha cursado las asignaturas de Robótica Móvil y Robótica de Servicios dentro del Grado en Ingeniería de Robótica Software en el Campus de Fuenlabrada de la Universidad Rey Juan Carlos.

Mi tarea principal durante mi periodo de Prácticas Externas ha consistido en la migración de los ejercicios de la plataforma web Unibotics de Gazebo 11 / Classic a Gazebo Harmonic.

A continuación, explicaré detalladamente todas las tareas que he realizado, comenzando por la configuración del entorno de trabajo y la realización de pequeños ejercicios de testeo con el objetivo de familiarizarme con el software, para así posteriormente poder llevar a cabo la migración de Gazebo 11 / Classic a Gazebo Harmonic de los ejercicios de la plataforma web Unibotics, los cuales, la mayoría de ellos ya había realizado en las asignaturas de Robótica Móvil y Robótica de Servicios, y otros que no había visto en estas asignaturas, pero que también se encuentran dentro del campo de los robots móviles y la conducción autónoma, entre otros.

[] 2. INSTALACIÓN DE DOCKER EN UBUNTU 24.04

Mi primera semana la dediqué enteramente a la instalación del entorno de trabajo y de todo el software alojado en los repositorios de Github de JdeRobot, cuya ejecución la llevaría a cabo mediante el uso e instalación de contenedores **Docker** en mi máquina nativa, la cual tiene la versión 24.04 del sistema operativo Linux, concretamente de la distribución Ubuntu.

A continuación, voy a explicar paso a paso todo el proceso llevado a cabo para instalar Docker en mi máquina.

En primer lugar, se abre una terminal en el directorio **/home/username** (en mi caso, **/home/aalbeerto-02**), mediante la cual accedo al escritorio, clono el repositorio oficial de RoboticsAcademy y accedo a él.

```
cd Escritorio/
git clone --recurse-submodules https://github.com/JdeRobot/RoboticsAcademy.git
cd RoboticsAcademy/
```

Después, configuro el entorno de trabajo utilizando el script de preparación **./scripts/develop_academy.sh**. Sin embargo, al intentar ejecutar este script por primera vez, aparecen varios errores debido a que todas las dependencias de **yarn** y **docker** todavía no se han instalado.

Por ello, es necesario ejecutar una serie de comandos de configuración encargados de actualizar los repositorios de APT, instalar herramientas necesarias como curl y sus certificados, crear el directorio para las claves de Docker para así poder descargar su clave GPG a la cual hay que dar permisos de lectura, añadir el repositorio de Docker a APT y volver a actualizar esta lista de paquetes. Todo lo que acabo de mencionar se lleva a cabo ejecutando los siguientes comandos en una terminal:

```
# Add Docker's official GPG key
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources
echo \
 "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/ubuntu \
 $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
 sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Tras haber ejecutado todos los comandos de configuración mencionados anteriormente, llega la hora de instalar Docker, Docker Compose y todos sus componentes, y una vez finalizada la instalación, comprobar que Docker está activo de la siguiente forma:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
sudo systemctl status docker
sudo apt install docker-compose
```

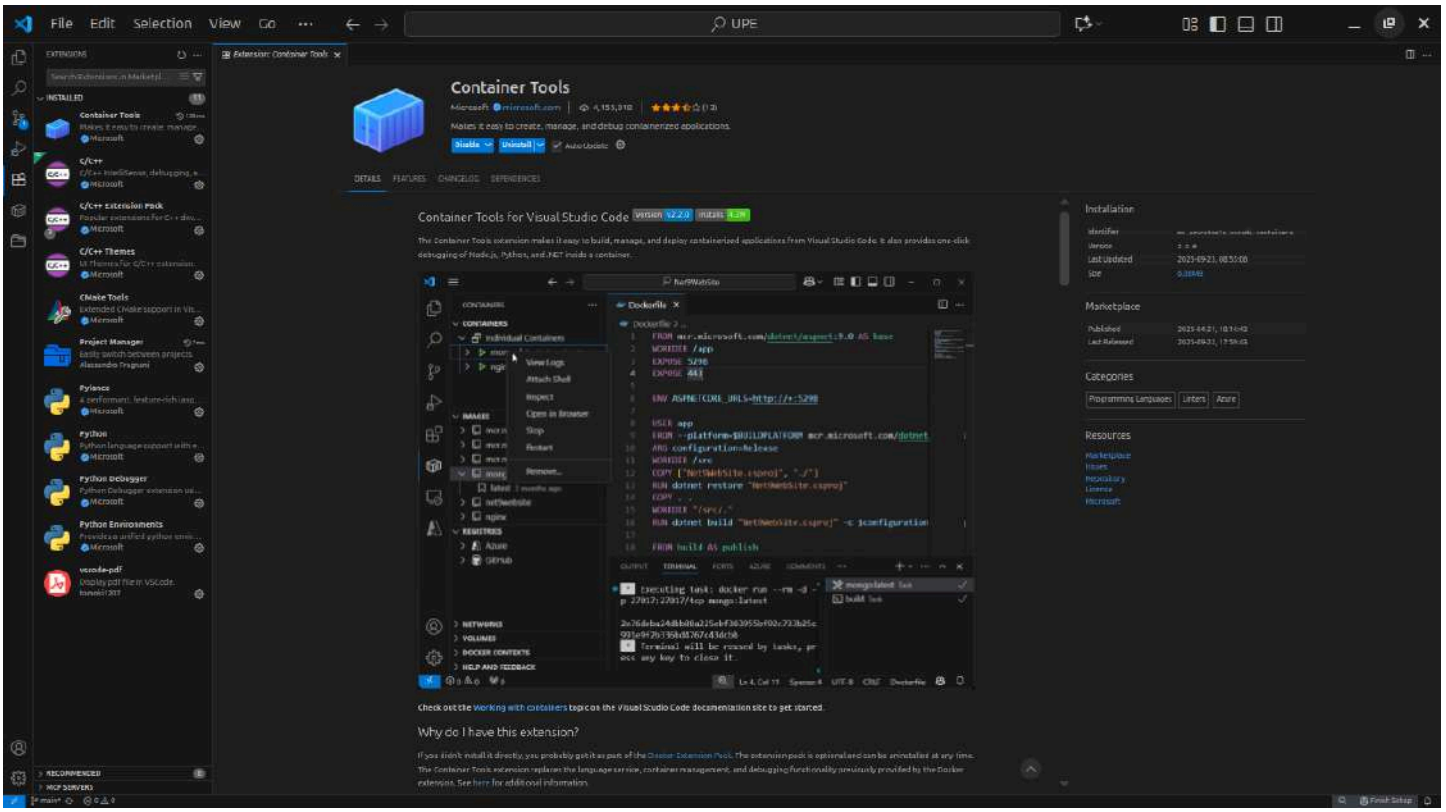
Como paso previo a la ejecución del script de preparación, hay que verificar la versión de Docker instalada, crear el grupo de usuarios de Docker y añadir el usuario actual al nuevo grupo Docker.

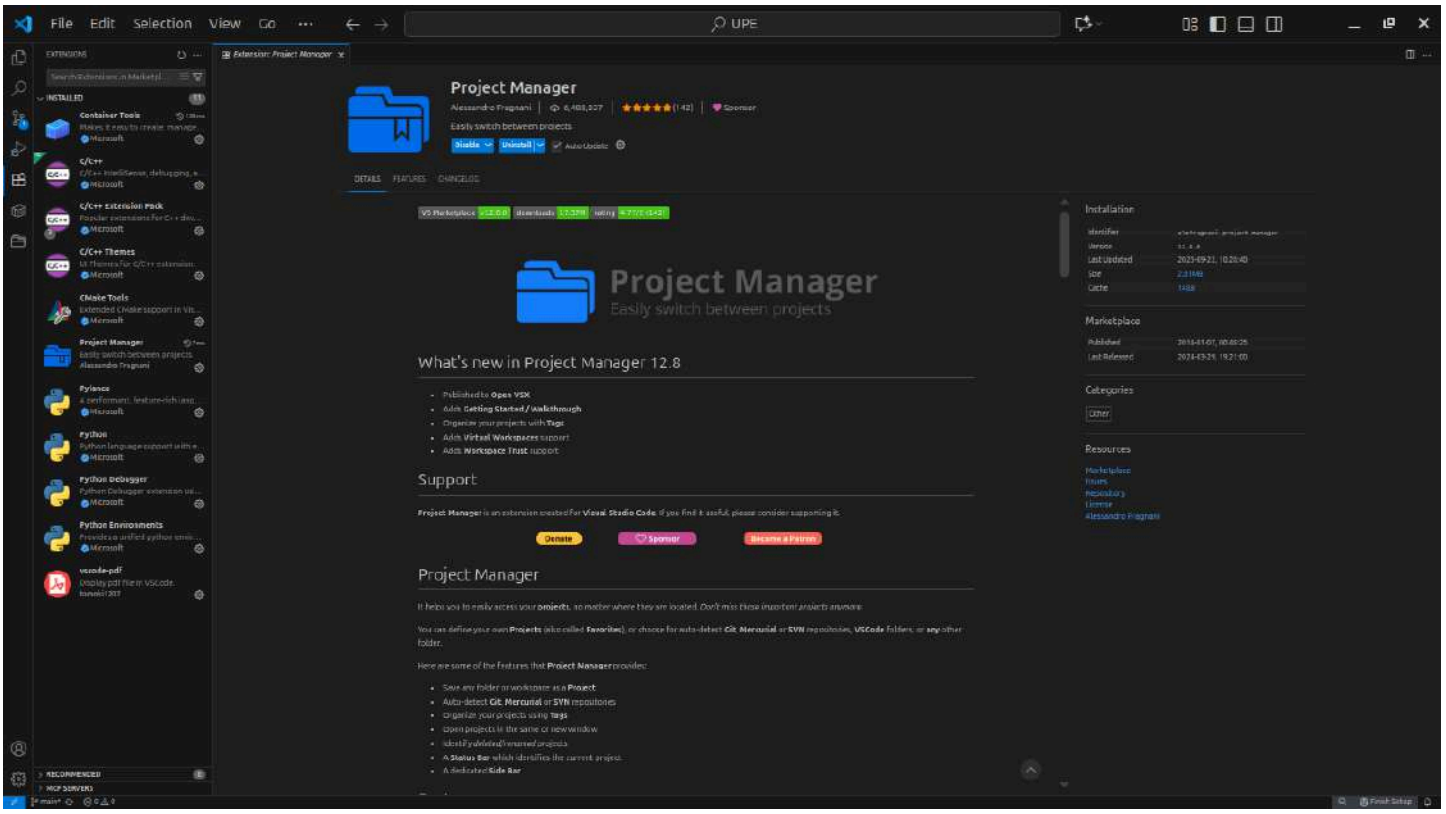
Y por último, ejecutar el script de preparación mencionado anteriormente (**./scripts/develop_academy.sh**).

```
docker --version
sudo groupadd docker
sudo usermod -aG docker $USER
newgrp docker
sudo ./scripts/develop_academy.sh
```

[] 3. CONFIGURACIÓN Y USO DE DOCKER EN VISUAL STUDIO CODE

Mi segunda semana la dediqué enteramente a configurar y aprender a usar Docker desde un editor. En mi caso, el editor elegido para trabajar dentro del repositorio de RoboticsAcademy ha sido **Visual Studio Code**, dentro del cual he instalado las extensiones **Container Tools** y **Project Manager**, encargadas de integrar y configurar Docker en Visual Studio Code de forma que siempre que se abra el editor, el repositorio de RoboticsAcademy esté configurado y listo para ser utilizado con Docker.





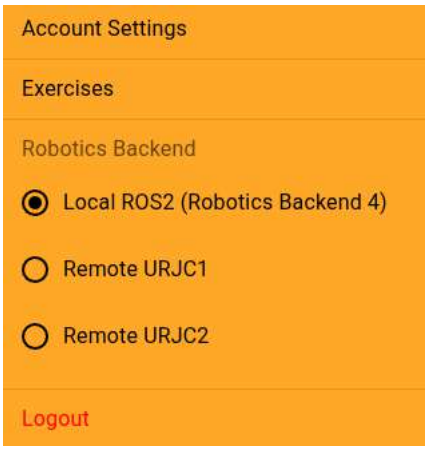
Y por último, una vez finalizada la instalación de ambas extensiones, es necesario reiniciar el equipo para verificar que tanto Docker como Visual Studio Code se han integrado y configurado correctamente.

[] 4. LANZAMIENTO DEL "ROBOTICSBACKEND" EN LOCAL

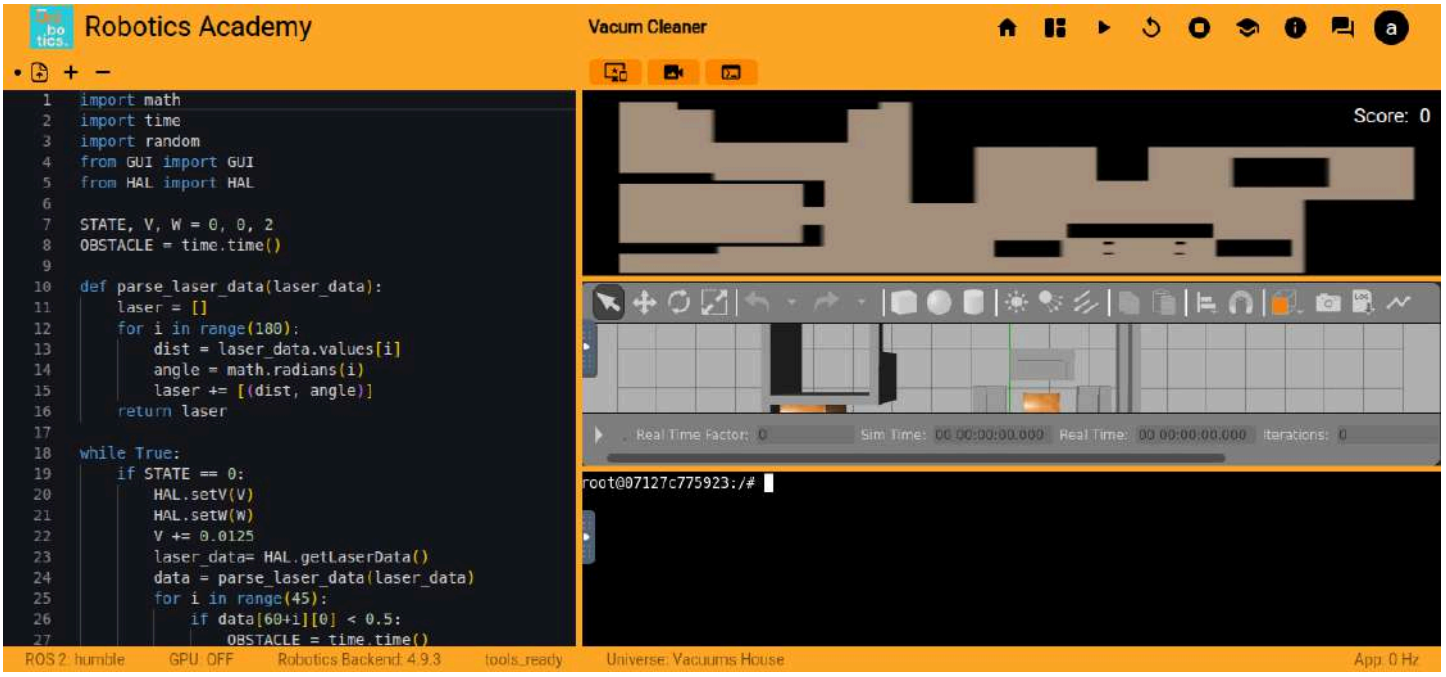
Mi tercera semana la dediqué enteramente a aprender a lanzar el **RoboticsBackend** desde mi máquina utilizando Docker, para así poder trabajar dentro de los ejercicios de la plataforma web Unibotics en local. Para ello, se debe dejar ejecutando el comando adjunto a continuación en una terminal a la vez que se accede a la [página web de Unibotics]():

```
docker run --rm -it --device /dev/dri -p 6080:6080 -p 1108:1108 -p 7163:7163
jderobot/robotics-backend:latest
```

Una vez dentro de Unibotics, se selecciona el ejercicio en el que se quiera trabajar. Para asegurarse de que el ejercicio carga correctamente en local, hay que seleccionar la opción **Local ROS2 (RoboticsBackend 4)** que aparece en la esquina superior derecha de la ventana al hacer clic en la foto de perfil, la cual puede aparecer como la inicial del nombre de usuario si aún no se ha añadido ninguna foto de perfil.



Y por último, una vez seleccionada esta opción, queda verificar que tanto la simulación en Gazebo como la terminal se han lanzado correctamente. Y si es así, ya se puede comenzar a realizar pruebas de código para hacer funcionar el ejercicio tal y como se muestra a continuación, en este caso, en el ejercicio **Basic Vacuum Cleaner**.



[] 5. LANZAMIENTO DE LA “ROBOTICSDATABASE” JUNTO A “ROBOTICSACADEMY” EN LOCAL

Mi cuarta semana la dediqué enteramente a aprender a lanzar la **RoboticsDatabase** junto a RoboticsAcademy desde mi máquina utilizando Docker. Para ello, se debe ejecutar el siguiente comando en una terminal:

```
docker run --hostname my-postgres --name academy_db -d\
   -e POSTGRES_DB=academy_db \
   -e POSTGRES_USER=user-dev \
   -e POSTGRES_PASSWORD=robotics-academy-dev \
   -e POSTGRES_PORT=5432 \
   -d -p 5432:5432 \
   jderobot/robotics-database:latest
```

Este comando se encarga de crear un nuevo contenedor para la RoboticsDatabase, al cual se llamará con el flag –link cada vez que se vaya a lanzar RoboticsAcademy, por lo que ya no es necesario volver a ejecutar este comando. A continuación se muestran tres formas diferentes de lanzar la RoboticsDatabase junto a RoboticsAcademy, en función de si la máquina en la que se está ejecutando tiene GPU con NVIDIA, GPU sin NVIDIA o CPU, respectivamente.

```
docker run --rm -it $(nvidia-smi >/dev/null 2>&1 && echo "--gpus all" || echo "") --device /dev/dri -p
6080:6080 -p 6081:6081 -p 1108:1108 -p 7163:7163 -p 7164:7164 --link academy_db
jderobot/robotics-academy:latest
```

```
docker run --rm -it --device /dev/dri -p 6080:6080 -p 6081:6081 -p 1108:1108 -p 7163:7163 -p 7164:7164
--link academy_db jderobot/robotics-academy:latest
```

```
docker run --rm -it -p 6080:6080 -p 6081:6081 -p 1108:1108 -p 7163:7163 -p 7164:7164 --link academy_db
jderobot/robotics-academy:latest
```

Y por último, para verificar que todos los contenedores que se vayan a utilizar se han descargado y configurado correctamente, se debe ejecutar el comando **sudo docker images** en una terminal, cuyo resultado debe ser el mostrado a continuación:

```
REPOSITORY                  TAG       IMAGE ID       CREATED       SIZE
jderobot/robotics-backend   latest    0e2a10ccfaf4   5 days ago    27.2GB
jderobot/robotics-academy   latest    1c67a50c79c2   13 days ago   28.5GB
jderobot/robotics-database  latest    3a1d0407347e   13 days ago   483MB
```

[] 6. EJERCICIO DE CALENTAMIENTO: CAMBIAR MUNDO

Mi quinta semana la dediqué enteramente a realizar un pequeño ejercicio de calentamiento para irme familiarizando con el software y el código fuente de **RoboticsInfrastructure**, un repositorio de Github alojado dentro del propio repositorio de RoboticsAcademy (**RoboticsAcademy/RoboticsInfrastructure**), y en el que he realizado gran parte de mi trabajo para así poder comenzar a migrar los ejercicios de Gazebo 11 / Classic a Gazebo Harmonic en un futuro próximo. Este ejercicio consistía básicamente en cambiar el universo de uno de los ejercicios ya migrados a Gazebo Harmonic.

Para ello, hay que acceder al fichero **RoboticsInfrastructure/database/universes.sql** para localizar los universos ya migrados a Gazebo Harmonic. La forma más sencilla de identificar estos universos es observando si en su sexta columna aparece escrito gazebo o gz. En caso de que aparezca escrito **gazebo**, significa que ese universo se encuentra en Gazebo 11 / Classic. Pero si lo que aparece escrito es **gz**, significa que ese universo se encuentra en Gazebo Harmonic.

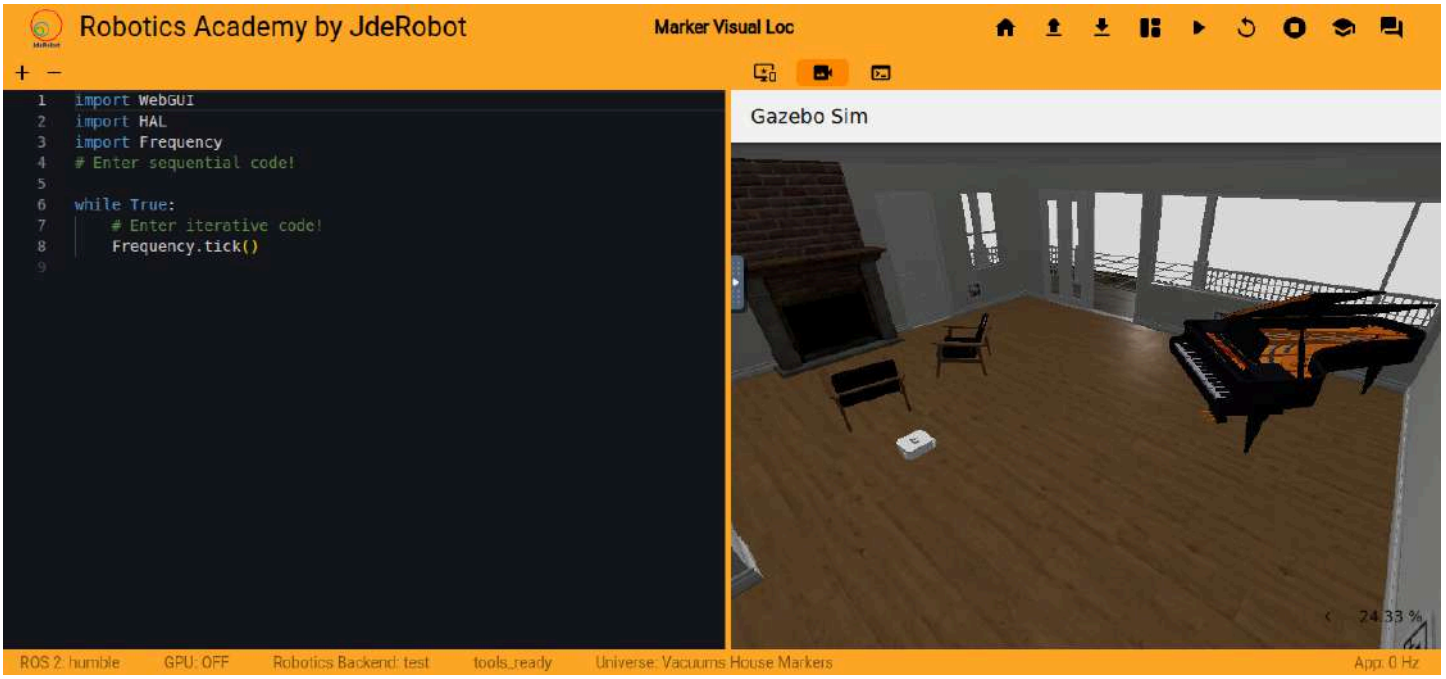
En mi caso, he seleccionado los universos correspondientes a los ejercicios **Laser Mapping** y **Marker Based Visual Loc**, en el que el universo de Laser Mapping es un almacén de Amazon, mientras que el de Marker Based Visual Loc es un chalet de dos plantas. El resultado final de este ejercicio visualiza el universo del ejercicio Laser Mapping en el visor de Gazebo del ejercicio Marker Based Visual Loc.

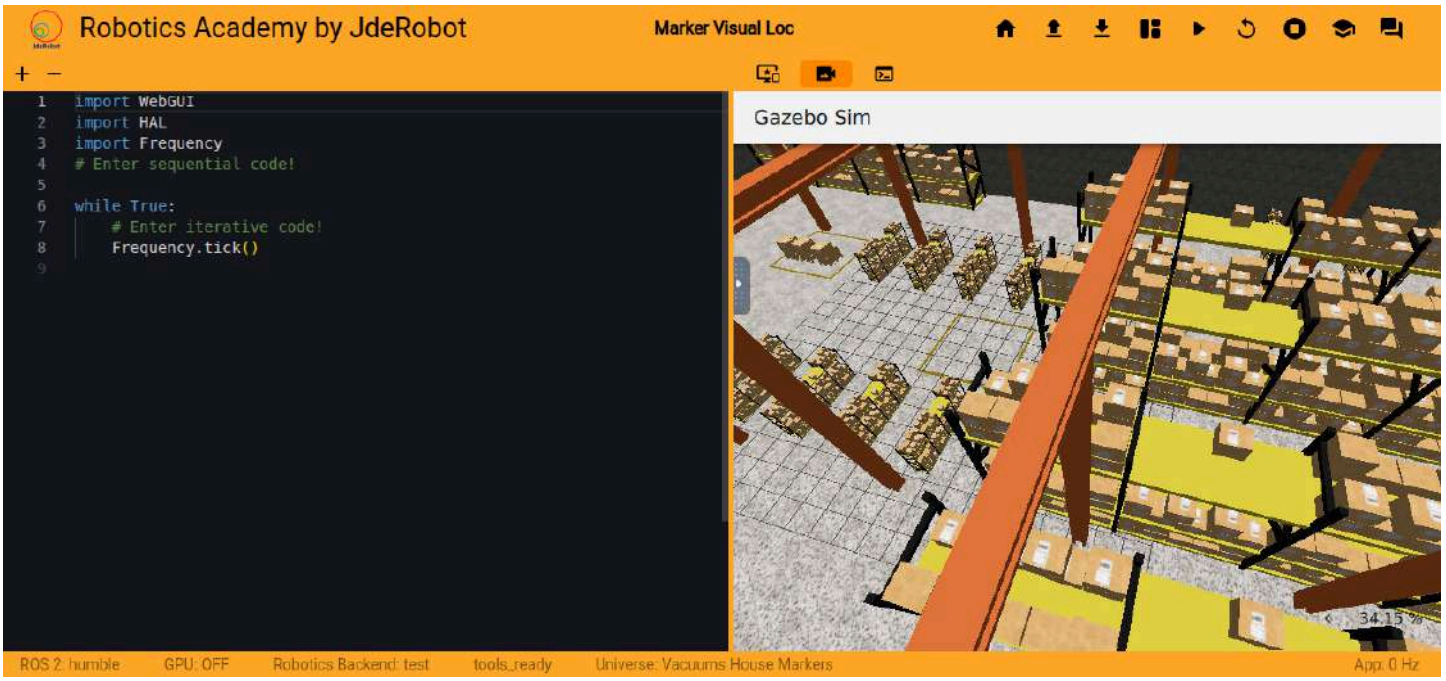
```
12      Laser Mapping Warehouse       /opt/jderobot/Launchers/laser_mapping.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/laser_mapping.config"}   ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}

25      Vacuums House Markers /opt/jderobot/Launchers/marker_visual_loc.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/marker_visual_loc.config"}      ROS2    gz
{1,-1.5,0.6,0,0,0}
```

La única modificación a realizar para resolver este ejercicio se lleva a cabo en el fichero **RoboticsInfrastructure/Launchers/marker_visual_loc.launch.py** en la línea en la que se declara la variable **world_file_name**:

```
# ANTES (UNIVERSO ORIGINAL)
world_file_name = "marker_visual_loc.world"

# DESPUÉS (UNIVERSO NUEVO)
world_file_name = "laser_mapping.world"
```

Y por último, se muestra la comparativa entre cómo se ve inicialmente el universo del ejercicio Marker Based Visual Loc, y cómo se ve tras haber realizado la modificación mencionada anteriormente:





[] 7. MIGRACIÓN DE EJERCICIOS DE GAZEBO 11 / CLASSIC A GAZEBO HARMONIC

} 7.1 INTRODUCCIÓN

Todos los días comprendidos entre mi sexta semana hasta la finalización de mis prácticas las dediqué enteramente a la migración completa de un total de nueve ejercicios, y dos más en los que avancé lo máximo posible y que no pude finalizar de forma completa, ya que ambos abarcan conceptos que, pese a haberlos visto de forma teórica en mi grado, como son los robots que utilizan imágenes de profundidad y otras aplicaciones avanzadas de Visión Artificial, y los robots industriales, requerí de la ayuda de otros compañeros de prácticas para que se encargaran de estas partes de la migración.

Dicho esto, procedo a listar todos los ejercicios en los que he estado trabajando para llevar a cabo su migración de **Gazebo 11 / Classic** a **Gazebo Harmonic**, explicando brevemente en qué consisten y adjuntando el enlace a la documentación de cada uno de ellos:

- **Obstacle Avoidance**: Introduce de forma práctica la navegación local mediante el uso de campos de fuerza virtuales (VFF o Virtual Force Fields).
- **Global Navigation**: Introduce de forma práctica la navegación global mediante el uso e implementación de la lógica del algoritmo de planificación de ruta del gradiente (GPP o Gradient Path Planning).
- **Autoparking**: Consiste en la implementación de la lógica de un algoritmo de navegación en un vehículo autónomo que se encuentra buscando aparcamiento.
- **Follow Line**: Consiste en la implementación de un controlador PID reactivo capaz de seguir una línea roja en el suelo de un circuito de carreras.
- **Basic Vacuum Cleaner**: Consiste en la implementación de la lógica de un algoritmo de navegación para una aspiradora autónoma, con el objetivo de cubrir la mayor superficie posible de una casa.
- **Localized Vacuum Cleaner**: Consiste en implementar la lógica de un algoritmo de navegación para una aspiradora autónoma utilizando la ubicación del robot, equipado con un mapa y del cual se conoce su ubicación, con el objetivo de cubrir la mayor superficie posible de una casa.
- **Montecarlo Laser Localization**: Consiste en desarrollar un algoritmo de localización basado en el filtro de partículas utilizando el láser del robot.
- **Montecarlo Visual Localization**: Consiste en desarrollar un algoritmo de localización visual basado en el filtro de partículas.
- **3D Reconstruction**: Consiste en programar la lógica necesaria para permitir que un robot Kobuki genere una reconstrucción 3D de una escena que recibe a través de sus cámaras.
- **Follow Person**: Consiste en implementar la lógica de un algoritmo de navegación para seguir a una persona por un hospital utilizando una máscara con una Red Neuronal Convolucional basada en Regiones (R-CNN o Region-Based Convolutional Neural Network) llamada Detector de Disparo Único (SSD o Single Shot Detector).
- **Pick And Place**: Consiste en aprender la infraestructura subyacente de los ejercicios de robots industriales (ROS + MoveIt + nuestra API HAL de robótica industrial) y familiarizarse con los componentes clave necesarios para ejercicios más complejos al completar la tarea de elegir y colocar múltiples objetos y clasificarlos por color o forma.

A continuación, se encuentran todos los ficheros que se han modificado y/o creado (y de qué forma) para poder llevar a cabo la migración completa de todos estos ejercicios de Gazebo 11 / Classic a Gazebo Harmonic:

} 7.2 FICHERO MODEL.SDF

En este fichero, con ruta **RoboticsInfrastructure/CustomRobots/<robot_model_name>/models/<robot_model_name>/model.sdf**, sólo es necesario modificar el código correspondiente a las etiquetas **<plugin>** y **<sensor>**, donde el plugin utilizado por cada sensor pasa de estar declarado dentro de <sensor> a integrarse en él.

```
# GAZEBO 11 / CLASSIC
<sensor name='laser' type='ray'>
  <ray>
    <scan>
      <horizontal>
        ...
      </horizontal>
    </scan>
    <range>
      ...
    </range>
  </ray>
  <update_rate>20</update_rate>
  <always_on>1</always_on>
  <visualize>1</visualize>
  <plugin name="gazebo_ros_head_hokuyo_controller" filename="libgazebo_ros_laser.so">
    <topicName>/<robot_model_name>/laser/scan</topicName>
    <frameName><robot_model_name></frameName>
  </plugin>
</sensor>

# GAZEBO HARMONIC
<sensor name='laser' type='ray'>
  <ray>
    <scan>
      <horizontal>
        ...
      </horizontal>
    </scan>
    <range>
        ...
    </range>
  </ray>
  <update_rate>20</update_rate>
  <always_on>1</always_on>
  <visualize>1</visualize>
  <topic>/<robot_model_name>/laser/scan</topic>
  <gz_frame_id><robot_model_name></gz_frame_id>
</sensor>
```

Y por último, hay que añadir al final del fichero los plugins correspondientes al **velocity-control-system** y al **odometry-publisher-system**, entre otros.

```
# GAZEBO 11 / CLASSIC
<plugin name="object_controller" filename="libgazebo_ros_planar_move.so">
  <commandTopic>/<robot_model_name>/cmd_vel</commandTopic>
  <odometryTopic>/<robot_model_name>/odom</odometryTopic>
  <odometryFrame>odom</odometryFrame>
  <odometryRate>20</odometryRate>
  <robotBaseFrame><robot_model_name></robotBaseFrame>
</plugin>

# GAZEBO HARMONIC
<plugin filename="gz-sim-velocity-control-system" name="gz::sim::systems::VelocityControl">
  <topic>/<robot_model_name>/cmd_vel</topic>
</plugin>
<plugin filename="gz-sim-odometry-publisher-system" name="gz::sim::systems::OdometryPublisher">
  <odom_topic>/<robot_model_name>/odom</odom_topic>
  <robot_base_frame><robot_model_name></robot_base_frame>
  <odom_frame>odom</odom_frame>
  <odom_publish_frequency>20</odom_publish_frequency>
  <dimensions>3</dimensions>
</plugin>
```

} 7.3 FICHERO <ROBOT_MODEL_NAME>.YAML

Este fichero, con ruta **RoboticsInfrastructure/CustomRobots/<robot_model_name>/params/<robot_model_name>.yaml**, se crea de cero partiendo como base de cualquier otro fichero con el mismo formato de nombre correspondiente a cualquier modelo de robot que ya se encuentre en Gazebo Harmonic, y en el que se añaden todos los topics que el robot vaya a utilizar para suscribirse o publicar mensajes, como por ejemplo odometría, velocidad, láseres y cámaras, entre otros.

```
# ODOMETRY
- ros_topic_name: "odom"
  gz_topic_name: "/<robot_model_name>/odom"
  ros_type_name: "nav_msgs/msg/Odometry"
  gz_type_name: "gz.msgs.Odometry"
  direction: GZ_TO_ROS

# VELOCITY
- ros_topic_name: "cmd_vel"
  gz_topic_name: "/<robot_model_name>/cmd_vel"
  ros_type_name: "geometry_msgs/msg/Twist"
  gz_type_name: "gz.msgs.Twist"
  direction: ROS_TO_GZ

# LIDAR
- ros_topic_name: "/<robot_model_name>/laser/scan"
                  "/<robot_model_name>/scan_front"
                  "/<robot_model_name>/scan_side"
                  "/<robot_model_name>/scan_back"
  gz_topic_name: "/<robot_model_name>/laser/scan"
                 "/<robot_model_name>/scan_front"
                 "/<robot_model_name>/scan_side"
                 "/<robot_model_name>/scan_back"
  ros_type_name: "sensor_msgs/msg/LaserScan"
  gz_type_name: "gz.msgs.LaserScan"
  direction: GZ_TO_ROS

# CAMERA
- ros_topic_name: "/<robot_model_name>/camera_info"
  gz_topic_name: "/<robot_model_name>/camera_info"
  ros_type_name: "sensor_msgs/msg/CameraInfo"
  gz_type_name: "gz.msgs.CameraInfo"
  direction: GZ_TO_ROS

# BUMPER
- ros_topic_name: "/<robot_model_name>/events/center_bumper"
                  "/<robot_model_name>/events/right_bumper"
                  "/<robot_model_name>/events/left_bumper"
 gz_topic_name: "/<robot_model_name>/events/center_bumper"
                "/<robot_model_name>/events/right_bumper"
                "/<robot_model_name>/events/left_bumper"
 ros_type_name: "ros_gz_interfaces/msg/Contacts"
 gz_type_name: "gz.msgs.Contacts"
 direction: GZ_TO_ROS
```

} 7.4 FICHERO CMAKELISTS.TXT

En este fichero, con ruta **RoboticsInfrastructure/CustomRobots/CMakeLists.txt**, sólo es necesaria la adición de la línea **<robot_model_name>/params**, para que el fichero que se acaba de crear (**<robot_model_name>.yaml**) se tenga en cuenta a la hora de lanzar el docker de RoboticsAcademy.

```
install(
 DIRECTORY
   ...
   # <ROBOT_MODEL_NAME>
   <robot_model_name>/models
   <robot_model_name>/launch
   <robot_model_name>/worlds
   <robot_model_name>/params
   ...
 DESTINATION share/${PROJECT_NAME})
```

} 7.5 FICHERO SPAWN_ROBOT.LAUNCH.PY

Este fichero, con ruta **RoboticsInfrastructure/Launchers/<exercise_name>/spawn_robot.launch.py**, se crea de cero dentro de un nuevo directorio llamado **<exercise_name>/** partiendo como base de cualquier otro fichero con el mismo formato de nombre correspondiente a cualquier modelo de robot que ya se encuentre en Gazebo Harmonic, en el que hay que modificar el nombre del fichero que se le pasa como argumento a la variable **bridge_params**, y dependiendo de si el modelo de robot que se esté utilizando tenga o no cámara, se debe comentar todo el código relativo a las variables **start_gazebo_ros_image_bridge_cmd** y **start_gazebo_ros_depth_bridge_cmd**, en caso de que el robot no tenga cámara.

```
model_folder = "<robot_model_name>"

bridge_params = os.path.join(
    get_package_share_directory("custom_robots"), "params", "<robot_model_name>.yaml"
)

# UNCOMMENT / COMMENT IF THE ROBOT HAS / DOESN'T HAVE A CAMERA
# start_gazebo_ros_image_bridge_cmd = Node(
#     package="ros_gz_image",
#     executable="image_bridge",
#     arguments=["/<robot_model_name>/camera/image_raw"],
#     output="screen",
# )
# start_gazebo_ros_depth_bridge_cmd = Node(
#     package="ros_gz_image",
#     executable="image_bridge",
#     arguments=["/<robot_model_name>/camera/depth"],
#     output="screen",
# )
# ld.add_action(start_gazebo_ros_image_bridge_cmd)
# ld.add_action(start_gazebo_ros_depth_bridge_cmd)
```

} 7.6 FICHERO <WORLD_NAME>.LAUNCH.PY

Este fichero, con ruta **RoboticsInfrastructure/Launchers/<world_name>.launch.py**, se crea de cero partiendo como base de cualquier otro fichero con el mismo formato de nombre correspondiente a cualquier modelo de robot que ya se encuentre en Gazebo Harmonic, en el que hay que modificar las líneas correspondientes a las variables **robot_launch_dir** y **world_file_name**, que contienen las rutas al directorio que contiene el fichero spawn_robot.launch.py y el fichero <world_name>.world que utiliza el ejercicio, respectivamente.

Es importante mencionar que en caso de que no se haya creado el fichero **<exercise_name>/robot_state_publisher.launch.py**, es obligatorio comentar todo el código relacionado con la variable **robot_state_publisher_cmd**.

```
robot_launch_dir = "/opt/jderobot/Launchers/<exercise_name>"
world_file_name = "<world_name>.world"

# UNCOMMENT / COMMENT IF ROBOT_STATE_PUBLISHER.LAUNCH.PY HAS/HAS NOT BEEN CREATED
# robot_state_publisher_cmd = IncludeLaunchDescription(
#     PythonLaunchDescriptionSource(
#         os.path.join(robot_launch_dir, "robot_state_publisher.launch.py")
#     ),
#     launch_arguments={"use_sim_time": use_sim_time}.items(),
# )
# ld.add_action(robot_state_publisher_cmd)
```

} 7.7 FICHERO <WORLD_NAME>.WORLD

Este fichero, con ruta **RoboticsInfrastructure/Worlds/<world_name>.world**, se crea de cero partiendo como base de cualquier otro fichero con el mismo formato de nombre correspondiente a cualquier modelo de robot que ya se encuentre en Gazebo Harmonic, en el que únicamente hay que sustituir todos aquellos flags **<include>** de la versión antigua por aquellos que aparezcan en la versión nueva.

```
<?xml version="1.0" ?>
<sdf version="1.10">
<world name="default">

  <plugin name="gz::sim::systems::Physics" filename="gz-sim-physics-system"/>
  <plugin name="gz::sim::systems::SceneBroadcaster” filename="gz-sim-scene-broadcaster-system"/>
  <plugin name="gz::sim::systems::UserCommands" filename="gz-sim-user-commands-system">
  </plugin>
  <plugin name="gz::sim::systems::Sensors" filename="gz-sim-sensors-system">
    <render_engine>ogre2</render_engine>
  </plugin>

  <gravity>0 0 -9.8</gravity>
  <atmosphere type="adiabatic"/>
  <scene>
    <ambient>0.4 0.4 0.4 1</ambient>
    <background>0.7 0.7 0.7 1</background>
    <shadows>true</shadows>
    <grid>false</grid>
  </scene>

  <include>
    ...
  </include>

</world>
</sdf>
```

} 7.8 FICHERO <EXERCISE_NAME>.CONFIG

Este fichero, con ruta **RoboticsInfrastructure/Launchers/visualization/<exercise_name>.config**, se crea de cero partiendo como base de cualquier otro fichero con el mismo formato de nombre correspondiente a cualquier modelo de robot que ya se encuentre en Gazebo Harmonic.

Todos los ficheros **.config** que se encuentran dentro de este directorio son exactamente iguales en cuanto a contenido, debido a que todos tienen la misma funcionalidad: Configurar todas las herramientas, tanto visuales como en simulación que el ejercicio va a utilizar, y que se le pasa como argumento en su cuarta columna a todos los universos en Gazebo Harmonic que aparecen en la base de datos de RoboticsInfrastructure, la cual se explica más en detalle en el siguiente punto.

Un ejemplo genérico de este fichero puede encontrarse haciendo clic en el siguiente enlace.

} 7.9 FICHERO UNIVERSES.SQL

Este fichero, con ruta **RoboticsInfrastructure/database/universes.sql**, sólo necesita ser modificado en la cuarta y en la sexta columna de la lista de universos seleccionables, donde en la cuarta columna **tools_config** se sustituye su valor por defecto (**None**) por la ruta del fichero de configuración mencionado en el punto anterior (**{"gzsim":"/opt/jderobot/Launchers/visualization/<exercise_name>.config"}**), mientras que en la sexta columna **type** se debe modificar el valor por defecto (**gazebo**), el cual indica que el universo está utilizando Gazebo 11 / Classic, por **gz**, que indica que el universo ahora utiliza Gazebo Harmonic, sirviendo también como distintivo claro de la versión de Gazebo en la que se encuentra cada universo.

A continuación se adjunta una lista de todos los universos que he migrado enteramente de Gazebo 11 / Classic a Gazebo Harmonic, además de una comparación de cómo se veían en este fichero antes y después de ser migrados:

```
# GAZEBO 11 / CLASSIC
1       3d Reconstruction       /opt/jderobot/Launchers/3d_reconstruction.launch.py         None    ROS2
gazebo  {0.0,0.0,0.0,0.0,0.0,0.0}
8       City Large      /opt/jderobot/Launchers/taxi_navigator.launch.py    None    ROS2    gazebo
{0.0,0.0,0.0,0.0,0.0,0.0}
13      Montmelo Ackermann Circuit      /opt/jderobot/Launchers/montmelo_circuit_ackermann.launch.py        None
        ROS2    gazebo  {0.0,0.0,0.0,0.0,0.0,0.0}
14      Montmelo Circuit        /opt/jderobot/Launchers/montmelo_circuit.launch.py None    ROS2    gazebo
{0.0,0.0,0.0,0.0,0.0,0.0}
15      Montreal Ackermann Circuit      /opt/jderobot/Launchers/montreal_circuit_ackermann.launch.py        None
        ROS2    gazebo  {0.0,0.0,0.0,0.0,0.0,0.0}
16      Montreal Circuit        /opt/jderobot/Launchers/montreal_circuit.launch.py None    ROS2    gazebo
{0.0,0.0,0.0,0.0,0.0,0.0}
17      Nurburgring Ackermann Circuit/opt/jderobot/Launchers/nurburgring_circuit_ackermann.launch.py     None
        ROS2    gazebo  {0.0,0.0,0.0,0.0,0.0,0.0}
18      Nurburgring Circuit     /opt/jderobot/Launchers/nurburgring_circuit.launch.py       None    ROS2
gazebo  {0.0,0.0,0.0,0.0,0.0,0.0}
19      Obstacle Avoidance Default
/opt/jderobot/Launchers/simple_circuit_obstacles_followingcam.launch.py  None    ROS2    gazebo
{0.0,0.0,0.0,0.0,0.0,0.0}
24      Vacuums House   /opt/jderobot/Launchers/vacuum_cleaner.launch.py     None    ROS2    gazebo
{0.0,0.0,0.0,0.0,0.0,0.0}
26      Vacuums House Roof      /opt/jderobot/Launchers/montecarlo_visual_loc.launch.py     None    ROS2
gazebo  {0.0,0.0,0.0,0.0,0.0,0.0}
39      Autopark_line   /opt/jderobot/Launchers/autopark_line.launch.py      None    ROS2    gazebo
{0.0,0.0,0.0,0.0,0.0,0.0}
40      Autopark_battery        /opt/jderobot/Launchers/autopark_battery.launch.py None    ROS2    gazebo
{0.0,0.0,0.0,0.0,0.0,0.0}
41      Autopark_sideways       /opt/jderobot/Launchers/autopark_sideways.launch.py         None    ROS2
gazebo  {0.0,0.0,0.0,0.0,0.0,0.0}
```

```
# GAZEBO HARMONIC
1       3d Reconstruction     /opt/jderobot/Launchers/3d_reconstruction.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/3d_reconstruction.config"}      ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
8       City Large     /opt/jderobot/Launchers/taxi_navigator.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/global_nav.config"}      ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
13      Montmelo Ackermann Circuit    /opt/jderobot/Launchers/montmelo_circuit_ackermann.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/follow_line.config"}     ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
14      Montmelo Circuit      /opt/jderobot/Launchers/montmelo_circuit.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/follow_line.config"}     ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
15      Montreal Ackermann Circuit    /opt/jderobot/Launchers/montreal_circuit_ackermann.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/follow_line.config"}     ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
16      Montreal Circuit      /opt/jderobot/Launchers/montreal_circuit.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/follow_line.config"}     ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
17      Nurburgring Ackermann Circuit/opt/jderobot/Launchers/nurburgring_circuit_ackermann.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/follow_line.config"}     ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
18      Nurburgring Circuit   /opt/jderobot/Launchers/nurburgring_circuit.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/follow_line.config"}     ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
19      Obstacle Avoidance Default
/opt/jderobot/Launchers/simple_circuit_obstacles_followingcam.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/obstacle_avoidance.config"}     ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
22      Simple Ackermann Circuit      /opt/jderobot/Launchers/simple_circuit_ackermann.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/follow_line.config"}     ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
23      Simple Circuit /opt/jderobot/Launchers/simple_circuit.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/follow_line.config"}     ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
24      Vacuums House  /opt/jderobot/Launchers/vacuum_cleaner.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/vacuum_house.config"}    ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
26      Vacuums House Roof    /opt/jderobot/Launchers/montecarlo_visual_loc.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/montecarlo_visual_localization.config"} ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
39      Autopark_line  /opt/jderobot/Launchers/autopark_line.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/autoparking.config"}     ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
40      Autopark_battery      /opt/jderobot/Launchers/autopark_battery.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/autoparking.config"}     ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
41      Autopark_sideways     /opt/jderobot/Launchers/autopark_sideways.launch.py
{"gzsim":"/opt/jderobot/Launchers/visualization/autoparking.config"}     ROS2    gz
{0.0,0.0,0.0,0.0,0.0,0.0}
```

[] 8. COMPILACIÓN Y GENERACIÓN DE UN NUEVO RADI PARA PROBAR LOS CAMBIOS

Con todos estos cambios realizados en los ficheros mencionados anteriormente, hay que hacer commit y push de la nueva rama creada y publicada antes de comenzar a migrar cada ejercicio, y que debe incluir únicamente todos estos cambios recién mencionados. Todas estas ramas siguen el formato **<exercise-name>-harmonic** (por ejemplo, en el caso del ejercicio Follow Person, la rama se llama follow-person-harmonic). Es importante hacer esto siempre para evitar así cualquier tipo de conflicto con la rama principal, en este caso, **humble-devel**.

Como todos estos cambios se han realizado en ficheros que se encuentran dentro del repositorio RoboticsInfrastructure, es obligatorio compilar un nuevo **RADI (Robotics Academy Docker Image)** accediendo a la carpeta **scripts/RADI/** de RoboticsAcademy y ejecutando el script **build.sh**, especificando como único parámetro la rama que se quiera lanzar.

```
cd scripts/RADI/
./build.sh -i <exercise-name>-harmonic
```

Es importante mencionar que, cada vez que se compila un nuevo RADI no se borrará el anterior, lo que provocará que el espacio disponible en disco se vaya llenando y llenando hasta que no quede espacio disponible en disco. Por tanto, para evitar que esto ocurra, se debe ejecutar en una terminal el comando **docker system prune -af**, el cual se encargará de borrar todo el espacio de memoria ocupado por cada RADI nuevo que se haya compilado.

Una vez finalizada la ejecución del script build.sh, hay que regresar al repositorio de RoboticsAcademy (**cd ../..**). Pero antes de ejecutar el script develop_academy.sh, es obligatorio modificar la siguiente línea del fichero **RoboticsAcademy/compose_cfg/dev_humble_cpu.yaml**:

```
# ANTES
 robotics-academy:
   image: jderobot/robotics-academy:latest

# DESPUÉS
 robotics-academy:
   image: jderobot/robotics-academy:test
```

Con este cambio realizado, ya se puede lanzar el script **develop_academy.sh**.

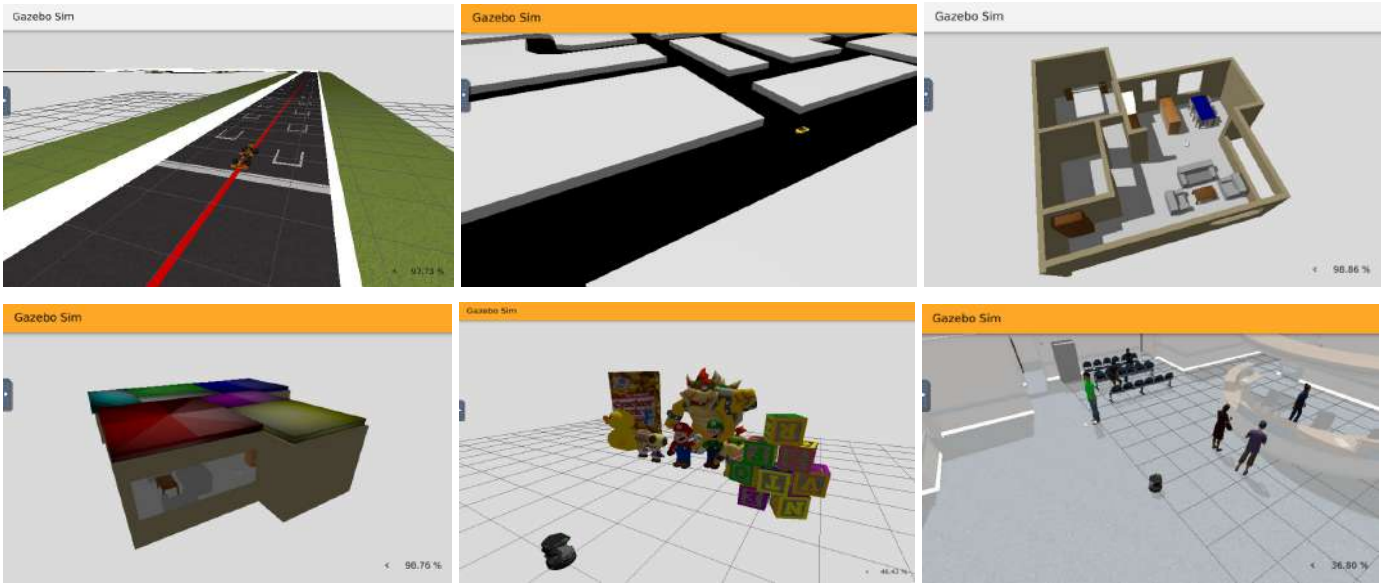
Y por último, solo queda acceder a la dirección web **http://0.0.0.0:7164/** que aparece en la terminal al ejecutar este script, enlace por el cual se accede a Unibotics lanzado en local para verificar que todos los cambios mencionados anteriormente se han realizado correctamente.

```
./scripts/develop_academy.sh
```

[] 9. CONTENIDO MULTIMEDIA

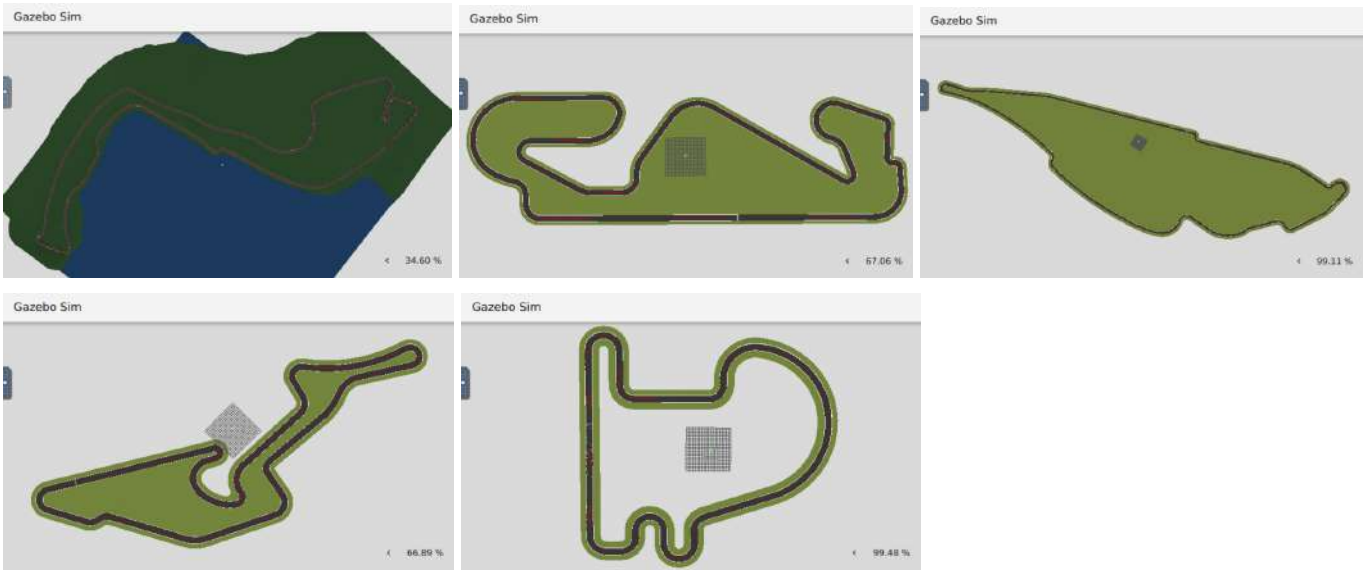
A continuación adjunto una serie de imágenes y vídeos de cómo se ven todos y cada uno de los ejercicios que he migrado de Gazebo 11 / Classic a Gazebo Harmonic al lanzar el docker de RoboticsAcademy con todos los cambios mencionados anteriormente.

- **Ejercicios Obstacle Avoidance, Global Navigation, Basic Vacuum Cleaner / Localized Vacuum Cleaner / Montecarlo Laser Localization (mismo universo), Montecarlo Visual Localization, 3D Reconstruction y Follow Person**



- **Ejercicio Autoparking (parking en línea, en batería y de lado)**

- **Ejercicio Follow Line (circuito de Mónaco, Montmeló, Montreal, Nurburgring y Simple)**



[] 10. CONCLUSIONES Y REFERENCIAS

La realización de estas Prácticas Externas en JdeRobot han supuesto una experiencia bastante formativa y enriquecedora en cuanto a nuevos conocimientos aprendidos, tanto a nivel técnico como personal. A lo largo de todos estos meses he podido trabajar de manera continuada en un entorno real de desarrollo software aplicado a la robótica, donde he podido enfrentarme a problemas reales y aportar soluciones que han tenido un impacto directo y positivo.

El objetivo principal de mis prácticas, centrado en la migración de ejercicios de Gazebo 11 / Classic a Gazebo Harmonic, me ha permitido profundizar de forma más práctica en el funcionamiento interno de los simuladores robóticos, así como en la infraestructura que los conecta con ROS2 y con los servicios web que ofrece Unibotics. Este proceso de migración no se ha limitado a cambios superficiales, sino que me ha servido para comprender en detalle la estructura de modelos SDF, la configuración de sensores y plugins, la creación de ficheros de parametrización para el bridge ROS / Gazebo, la modificación de launchers y mundos, y la integración correcta de todos estos elementos dentro del ecosistema de Docker y RoboticsAcademy.

Uno de los aspectos más relevantes de estas prácticas ha sido el aprendizaje progresivo y estructurado del entorno de trabajo. Haber podido comenzar desde la instalación y configuración completa de Docker, su integración con Visual Studio Code y el despliegue de distintos contenedores (RoboticsBackend, RoboticsDatabase y RoboticsAcademy) me ha permitido adquirir una visión global del flujo de trabajo y una mayor autonomía técnica, lo que me ha sido clave para poder depurar errores, reproducir entornos y validar correctamente cada ejercicio migrado.

Desde el punto de vista técnico, considero especialmente valioso el haber trabajado con una gran variedad de ejercicios, que abarcan campos como la navegación autónoma, la localización, la visión artificial, la reconstrucción 3D y la robótica industrial. Esta diversidad de temas me ha permitido consolidar conocimientos adquiridos previamente en asignaturas del grado, como han sido Robótica Móvil y Robótica de Servicios en mi caso, para así poder ampliarlos con nuevos conceptos que no había tratado en profundidad hasta el momento. Además, el hecho de migrar ejercicios que ya conocía desde el punto de vista del usuario me ha ayudado a comprender mejor la infraestructura subyacente que los hace funcionar.

También debo destacar la importancia del trabajo colectivo y del uso de buenas prácticas de desarrollo software, como el uso de ramas específicas para cada ejercicio, la compilación de nuevos RADI para validar cambios, y la documentación constante de todo el proceso llevado a cabo, lo que me ha ayudado a reforzar mi capacidad para trabajar en proyectos grandes, con código compartido y con un impacto directo en otros usuarios.

A nivel personal, estas prácticas me han permitido ganar confianza en mis capacidades como ingeniero de robótica software, mejorar mi autonomía a la hora de afrontar problemas complejos y reforzar mi interés por el desarrollo de herramientas educativas y plataformas de simulación. Considero que esta experiencia ha sido un paso muy importante en mi formación y que me ha preparado de manera sólida para futuros retos profesionales dentro del ámbito de la robótica y el software.

En conclusión, valoro muy positivamente el periodo de prácticas realizado en JdeRobot, tanto por los conocimientos técnicos adquiridos como por la experiencia real de trabajo en un proyecto de investigación y desarrollo. Sin duda, ha sido una experiencia clave en mi trayectoria académica y profesional.

- **Mi Blog de Prácticas**: https://theroboticsclub.github.io/2025-upe-alberto-leon/
- **Repositorio Blog de Prácticas**: https://github.com/TheRoboticsClub/2025-upe-alberto-leon
- **Repositorio RoboticsAcademy**: https://github.com/JdeRobot/RoboticsAcademy
- **Repositorio RoboticsInfrastructure**: https://github.com/JdeRobot/RoboticsInfrastructure
- **Repositorio IndustrialRobots**: https://github.com/JdeRobot/IndustrialRobots