# INSTALACIÓN DE ROS

Para instalar ROS es necesario tener Ubuntu 20.04 LTS en wsl. Se puede encontrar fácilmente en la Microsoft Store
Posteriormente e la instalación, ejecutar los siguientes comandos:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sou
curl -sSL 'http://keyserver.ubuntu.com/pks/lookup?op=get&search=0xC1CF6E31E6BADE8868B172B4F42ED6
sudo apt update
sudo apt install -y ros-noetic-desktop python3-rosdep
sudo rosdep init
rosdep update
```

Una vez actualizado ROS ejecutar estos comandos para añadirlo al path de Ubuntu y añadir una herramienta para instalar dependencias

```
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
sudo apt install python3-rosdep
```

Para iniciar rosdep ejecutar:

```
sudo rosdep init
rosdep update
```

Iniciar roscore

```
roscore
```

# INSTALACIÓN DE TurtleBot

Instalar los paquetes necesarios para turtlebot:

```
sudo apt install ros-noetic-turtlebot3 ros-noetic-turtlebot3-bringup
```

Configurar y elegir el modelo de TurtleBot:

```
echo "export TURTLEBOT3_MODEL=burger" >> ~/.bashrc
source ~/.bashrc
```

Los modelos disponibles son burguer/waffle/waffle_pi

# INSTALACIÓN DE Gazebo

Para la simulacion del robot es necesario Gazebo

```
sudo apt install ros-noetic-turtlebot3-gazebo
```

Se puede poner en marcha la simulacion con

```
roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
```

# ESPACIO DE TRABAJO

Para poder ejecutar el fichero python debemos crear un espacio de trabajo y añadirlo a ROS
Antes de eso debemos instalar un compilador de C++ pues la mayoria de paquetes ROS estan en ese lenguaje y algunas dependencias adicionales

```
sudo apt install build-essential
sudo apt install ros-noetic-desktop-full
```

Creamos la carpeta

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/
catkin_make
```

Agregar la carpeta al bashrc en otra terminal

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

Creamos el paquete Turtlebot_control con las dependencias rospy y geometry_msgs

```
catkin_create_pkg turtlebot_control rospy std_msgs geometry_msgs
```

Crear carpeta scripts en src y añadir el codigo python

```
mkdir /src/scripts
nano ros_MartinezLineros_Alvaro.py
```

Ahora para poder ejecutarlo tenemos que darle permisos

```
chmod +x ros_MartinezLineros_Alvaro.py
```

Finalmente ejecutamos el fichero python con

```
rosrun turtlebot_control ros_NombreAlumno.py
```

# NAVEGACIÓN AUTÓNOMA

Dependencias necesarias para la navegacion autonoma

```
sudo apt-get install ros-noetic-dwa-local-planner
```

Crear el modelo en Gazebo en un mundo con obstaculos

```
roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

Para interactuar con el robot es necesario inciar RViz en una nueva terminal

```
roslaunch turtlebot3_navigation turtlebot3_navigation.launch
```

Crear un fichero python en el paquete turtlebot_control/scripts con el codigo de sim_MartinezLineros_Alvaro.py
Ejecutar el fichero

```
chmod +x sim_MartinezLineros_Alvaro.py
rosrun turtlebot_control sim_MartinezLineros_Alvaro.py
```

En la ventana RViz podremos visualizar la ruta que va a generar el robot, asi como establecer manualmente puntos finales de la trayectoria para ver como navega de manera autónoma [2D Nav Goal]