Turtlebot 3 Linux Software Commands

Ubuntu	ISO download link: https://www.ubuntu-fr.org/download/ • **Software update:** *sudo apt update* • **Software upgrade**: *sudo apt upgrade* • **Install specific package:** *sudo apt-get install package_name* • **Uninstall specific program:** *sudo apt-get purge program_name**
Requirements	**Install Pyton 3 Colcon and Curl :** *sudo apt install python3-colcon-common-extensions* *sudo apt install curl*
ROS 2 Humble	**Link:** https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html **Installation:** • **Add the ROS 2 apt repository to your system:** *sudo apt install software-properties-common* *sudo add-apt-repository universe* • **Add the ROS 2 GPG key with apt:** *sudo apt update && sudo apt install curl -y* *sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg* • **Add the repository to your sources list:** *echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null* • **Install ROS 2 packages (ROS, Rviz, Demos, Tutorials):** *sudo apt update* *sudo apt upgrade* *sudo apt install ros-humble-desktop* • **ROS-Base install: Communication libraries, message packages, command line tools (No GUI tools):** *sudo apt install ros-humble-ros-base*

	• **Development tools: Compilers and other tools to build ROS packages** *sudo apt install ros-dev-tools* • **Environment setup** *source /opt/ros/humble/setup.bash* **Check installation:** • **In one terminal, source the setup file and then run a C++ talker:** *source /opt/ros/humble/setup.bash* *ros2 run demo_nodes_cpp talker* • **In another terminal source the setup file and then run a Python listener:** *source /opt/ros/humble/setup.bash* *ros2 run demo_nodes_py listener* **Uninstall ROS 2 Humble:** *sudo apt remove ~nros-humble-* && sudo apt autoremove* *sudo rm /etc/apt/sources.list.d/ros2.list* *sudo apt update* *sudo apt autoremove* *sudo apt upgrade*
Gazebo =	**Link:** https://gazebosim.org/docs/garden/install_ubuntu https://installati.one/install-gazebo-ubuntu-22-04/ **Installation:** • **Install some necessary tools:** *sudo apt-get update* *sudo apt-get install lsb-release wget gnupg* • **Install Gazebo for Ubuntu 22.04** *sudo apt install gazebo* *sudo apt install libgazebo-dev* **Uninstall Gazebo:** *sudo apt-get remove gazebo* *sudo apt-get -y autoremove gazebo* *sudo apt-get -y purge gazebo* *sudo apt-get -y autoremove*

Turtlebot3 Humble	**Link:** https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/ **Installation:** • **Install Simulation package:** *sudo apt install ros-humble-gazebo-** *sudo apt install ros-humble-cartographer* *sudo apt install ros-humble-cartographer-ros* *sudo apt install ros-humble-navigation2* *sudo apt install ros-humble-nav2-bringup* *sudo apt remove ros-humble-turtlebot3-msgs* *sudo apt remove ros-humble-turtlebot3* *mkdir -p ~/turtlebot3_ws/src* *cd ~/turtlebot3_ws/src/* *git clone -b humble-devel https://github.com/ROBOTIS-GIT/DynamixelSDK.git* *git clone -b humble-devel https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git* *git clone -b humble-devel https://github.com/ROBOTIS-GIT/turtlebot3.git* *cd ~/turtlebot3_ws* *colcon build* *echo 'source ~/turtlebot3_ws/install/setup.bash' >> ~/.bashrc* *source ~/.bashrc* *echo 'export TURTLEBOT3_MODEL=burger' >> ~/.bashrc* *source ~/.bashrc* *cd ~/turtlebot3_ws/src/* *git clone -b humble-devel https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git* *cd ~/turtlebot3_ws && colcon build* • **Launch Simulation World** ○ **Empty World** *ros2 launch turtlebot3_gazebo empty_world.launch.py* ○ **Turtlebot3 World** *ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py* • **Operate Turtlebot3** *ros2 run turtlebot3_teleop teleop_keyboard*

- **Cartographer SLAM**

  ros2 launch turtlebot3_cartographer cartographer.launch.py

- **Save Map**

  ros2 run nav2_map_server map_saver_cli -f ~/map

- **TurtleSim :** *ros2 run turtlesim turtlesim_node*

- **Teleop keyboard for TurtleSim :** *ros2 run turtlesim turtle_teleop_key*

SSH Connection:

- **Configure a Wifi hotspot from a computer:**

nmcli con add type wifi ifname <ip of hotspot wireless interface) con-name <name> autoconnect yes ssid <name SSID> ap ipv4.method shared

In our case: *nmcli con add type wifi ifname wlp2s0 con-name Co4Sys autoconnect yes ssid Co4Sys 802-11-wireless.mode ap ipv4.method shared*

→ Normaly, the hotspot is configured, you just have to check if the Wifi network is On and in hotspot mode

- **Connection to the hotspot created before:**

nmcli con up <name of the wireless network>

In our case *: nmcli con up Co4Sys*

- **Configure the SSH settings of the RaspberryPi to connect it to the Wifi hotspot :**

  Login: ubuntu
  Password: turtlebot

- **Get the Turtlebot3 address:** *ip n (must be reachable)*

- **Connect to Turtlebot3:**

ssh ubuntu@ip_address

In our case: *ssh ubuntu@10.42.0.216*

- **Launch the Turtlebot3 and then the keyboard on another terminal:**

ros2 launch turtlebot3_bringup robot.launch.py

	ros2 run turtlebot3_teleop teleop_keyboard
Others commands	**IP Information:** *ifconfig* **List of wireless interfaces:** *ip a* **Wireless scan:** *sudo arp-scan-localnet -I<wireless interface with ip a>* **Display your own IP Address:** *hostname -I* **Display all the hotspot connection:** *nmcli connection show*
ROS2 Node	**<u>Create a ROS2 Workspace and code your own node in Python:</u>** • **Install colcon extensions:** *sudo apt install python3-colcon-common-extensions* • **Autocompletion:** *gedit ~/.bashrc* **and add the following line at the end:** *source /usr/share/colcon_argcomplete/hook/ colcon-argcomplete.bash* • **Make a workspace directory:** *mkdir ros2_ws* • **Make a source directory inside the workspace:** *cd ros2_ws && mkdir src* • **Build the workspace:** *colcon build* • **Source the workspace:** *cd && source ~/ros2_ws/install/setup.bash && gedit ~/.bashrc* **and add in the file the following line at the end :** *source ~/ros2_ws/install/setup.bash* • **Create a ROS2 package into the workspace:** *cd ros2_ws/src/ && ros2 pkg create name_package –build-type ament_python –dependencies rclpy* • **Make sure Visual Studio Code is installed with Python and ROS2 extensions** • **Check installation:** *sudo snap install code --classic* • **Launch the workspace from the terminal:** *code .* • **Build the package with colcon:** *cd ros2_ws/ && colcon build* • **Fix "SetuptoolsDeprecationWarning: setup.py install is deprecated" in colcon build:** *sudo apt install python3-pip && pip3 install setuptools==58.2.0*

- **Create a python file to write a node and make it executable:** *cd ros2_ws/src/name_package/name_package/touch name_file_node.py && chmod +x name_file_node.py*

- **Execute the python node file created:** *cd ros2_ws/src/name_package/name_package/ && ./name_file_node.py*

Example of a Python node code:

→ my_first_node.py:

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node

class MyNode(Node):

    def __init__(self):
        super().__init__("first_node") #Initialization of
the Node
        self.get_logger().info("Hello from ROS2") #Display
logger into the terminal

def main(args=None):
    rclpy.init(args=args) #Initialization ROS2
communication
    node = MyNode() #Create a node from the MyNode class
    rclpy.spin(node) #Keep alive the node


    rclpy.shutdown() #Close the ROS2 communication

if __name__ == '__main__':
    main()
```

→ setup.py

```
entry_points={
        'console_scripts': [
            "test_node =
my_robot_controller.my_first_node:main"
        ],
    },
```

- **Auto-compilation without colcon build every time:**

cd ros2_ws/
colcon build –symlink-install
source ~/.bashrc
ros2 run my_robot_controller test_node

- **Useful commands:**

ros2 node list
ros2 node info /name_of_the_node

	ros2 topic list *ros2 topic info /name_of_the_topic* *rqt_graph* *ros2 interface show <type of ros2 topic info >*
CasADi	**Install CasADi for Python:** • Verify pip installation (should be ≥8.1) : *pip -V* • Install CasADi: *pip install casadi* • **Python** script to verify the installation: *from casadi import ** *x = MX.sym("x")* *print(jacobian(sin(x),x))* If CasADi is well installed, the script should display "*cos(x)*"