# Lab03

Bringup and control a real robot

# Objectives

- Start Turtlebot3 driver and communicate with robot and sensors
- Drive a real robot with velocity commands
- Drive a real robot with the **Bump&Go controller** developed in simulation

# Requirements

**Create a workspace for your group in the home folder.** Do not try to modify files inside **`~/tb3_ws`**, this is a **read-only** folder that contains the main packages for running Turtlebot3 and its simulation. Create a copy in your own workspace if you need a modified version of these packages.

## Bringup Turlebot3

In robotics, "**bringup**" refers to the initial process of setting up and configuring a robot's hardware and software components connecting sensors, actuators, and controllers, as well as loading the necessary firmware and software stacks.
In this section there is a description of the main checks and commands to bring up the Turtlebot3.

### Check the `~/.bashrc` content

Since the robot will be used by different groups during the weeks, it is suggested as **best practice** to open the **`~/.bashrc`** file and **check its content**.

- [ ] Verify the exported **environment variables**. Here it is reported a list of the necessary ones.
    - [ ] **`TURTLEBOT3_MODEL=burger`**
    - [ ] **`LDS_MODEL=LDS-02`**
    - [ ] **`ROS_DOMAIN_ID=X`**, where X is the number of the robot. In this case do not put the 0 in front if it has only one digit.
    - [ ] **`CAMERA_MODEL=realsense`** (or **`oakd`**). Only if the lab requires the use of a camera, otherwise do not set it.

- ☐ Verify the **sourced workspaces**. Here are the minimum checks that you shall perform when starting a new lab.
    - ☐ Source **`/opt/ros/humble/setup.bash`**
    - ☐ Source **`/opt/ros/tb3_ws/install/setup.bash`**
    - ☐ Put a **source** command for the **workspace** of **your group**
    - ☐ **Comment** with an **#** at the start of the line every other source to **workspaces** of **other groups**

## Start the Turtlebot3 nodes

All the software necessary to communicate with the robot and the sensors is started with **a single launch**, that is reported below:

```
ros2 launch turtlebot3_bringup robot.launch.py
```

You can **check** that the system is running by searching for these lines in the logs

```
[ld08_driver-2] LDS-02 started successfully
...
...
[turtlebot3_ros-3] [INFO] [1730373263.920085681] [turtlebot3_node]: Run!
[turtlebot3_ros-3] [INFO] [1730373263.923334200]
[diff_drive_controller]: Init Odometry
[turtlebot3_ros-3] [INFO] [1730373263.928420518]
[diff_drive_controller]: Run!
```

Now, if you run the command **`ros2 topic list`** you should see a list of topics similar to the one you saw in simulations.

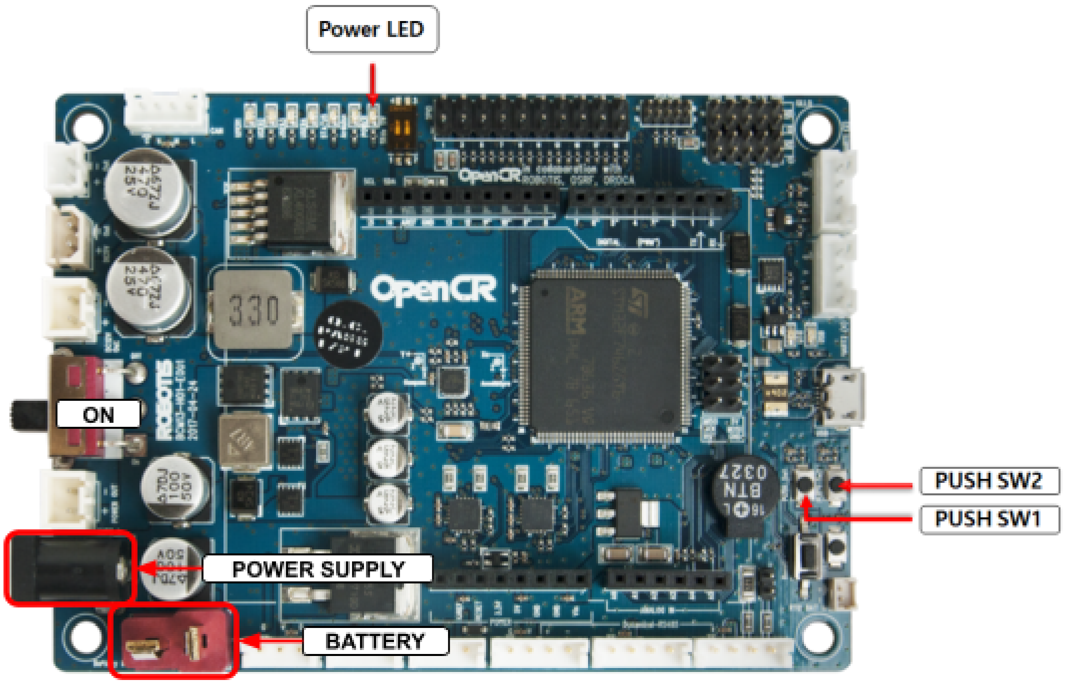
**Troubleshooting:** if you get an error in the connection with the port **`/dev/ttyACM0`**, you shall run the command **`sudo usermod -aG dialout students`** and reboot the PC.

## Power the motors

**Power supply: 12V DC, or Lithium battery 3S (11.1 V)**

- **AT YOUR DESK:**
    - Make sure the **robot is raised** from the table and the wheels are not in contact with the ground
    - **Plug** the **12V** power supply
    - Move the switch to the **ON position**, the Power LED should turn green and a music should play after some seconds
    - To **verify** that everything is working properly you can keep pressing SW1 or SW2 for some seconds, the wheels should move
    - If the bringup is started, now you can command the robot from ROS 2 topics.
- **ON TEST FIELD:**
    - **Plug** the Lithium battery

- Move the switch to the **ON position**, the Power LED should turn green and a music should play after some seconds.
- If the bringup is started, now you can command the robot from ROS 2 topics.



## Visualize robot state and sensors with Rviz2

While the bringup command is running, open Rviz2. Now, you should see the laser scan points and the robot model.

```
ros2 launch turtlebot3_bringup rviz2.launch.py
```

## Teleoperation nodes

While the bringup is running, you can also send a velocity command to the robot to make it move. In the following sections, three different methods to send velocity commands are shown.

### ROS 2 cli

One simple way of sending a velocity command is publishing a velocity on the **`/cmd_vel`** topic through the command line interface. The **`-r`** option is used to control the publication frequency in Hz.

```
ros2 topic pub /cmd_vel geometry_msgs/msg/Twist -r 1 "linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0"
```

### Keyboard teleoperation

Another interface that can be used from a terminal is the teleop_keyboard node of the turtlebot3_teleop package. This node allows you to change the velocity in real time using your keyboard. Read the instructions on the terminal after starting the node.

```
ros2 run turtlebot3_teleop teleop_keyboard
```

### rqt Robot Steer

This is a graphical interface to send velocity commands. Set the topic name in the top text box. Then set the maximum and minimum velocities in the text boxes, according to the robot limits. Now, you can use the sliders to set the desired velocity.

## Exercise

**Modify** the controller you developed in Lab2 to make it work on the real robot:

- Change the Quality of Service (QoS) of the subscription to the topic **`/scan`**

```
# from this
self.create_subscription(LaserScan, "scan", self.laser_callback, 10)

# to this
from rclpy.qos import qos_profile_sensor_data
self.create_subscription(LaserScan, "scan", self.laser_callback,
qos_profile_sensor_data)
```

- Change the way in which you select the points from the array of ranges:
  - On the real sensor, due to hardware specifications, it is not guaranteed that you will have 360 values in the ranges field. Use the fields **`angle_min`**, **`angle_max`** and **`angle_increment`** to select the ranges in the correct direction.

**Try** the controller you developed in Lab2 on the real Turtlebot3.

**Record** a rosbag while the robot is moving and try to **plot** the data as suggested in Lab2. Does the odometry resemble the real path of the robot? Could you say why?

# Robot checklist

You can print this checklist and bring it with you to laboratories.

- [ ] Verify the exported **environment variables**. Here it is reported a list of the necessary ones.
    - [ ] **`TURTLEBOT3_MODEL=burger`**
    - [ ] **`LDS_MODEL=LDS-02`**
    - [ ] **`ROS_DOMAIN_ID=X`**, where X is the number of the robot. In this case do not put the 0 in front if it has only one digit.
    - [ ] **`CAMERA_MODEL=realsense`** (or **`oakd`**). Only if the lab requires the use of a camera, otherwise do not set it.

- [ ] Verify the **sourced workspaces**. Here are the minimum checks that you shall perform when starting a new lab.
    - [ ] Source **`/opt/ros/humble/setup.bash`**
    - [ ] Source **`/opt/ros/tb3_ws/install/setup.bash`**
    - [ ] Put a **source** command for the **workspace** of **your group**
    - [ ] **Comment** with an **`#`** at the start of the line every other source to **workspaces** of **other groups**