Puzzlebot Gazebo Simulator

Instructions manual

{Learn, Create, Innovate};

Gazebo simulation

Initialising the simulation

- Download the ROS packages to your workspace source folder (e.g., `ros2_ws/src`)

```
puzzlebot_gazebo
puzzlebot_description
```

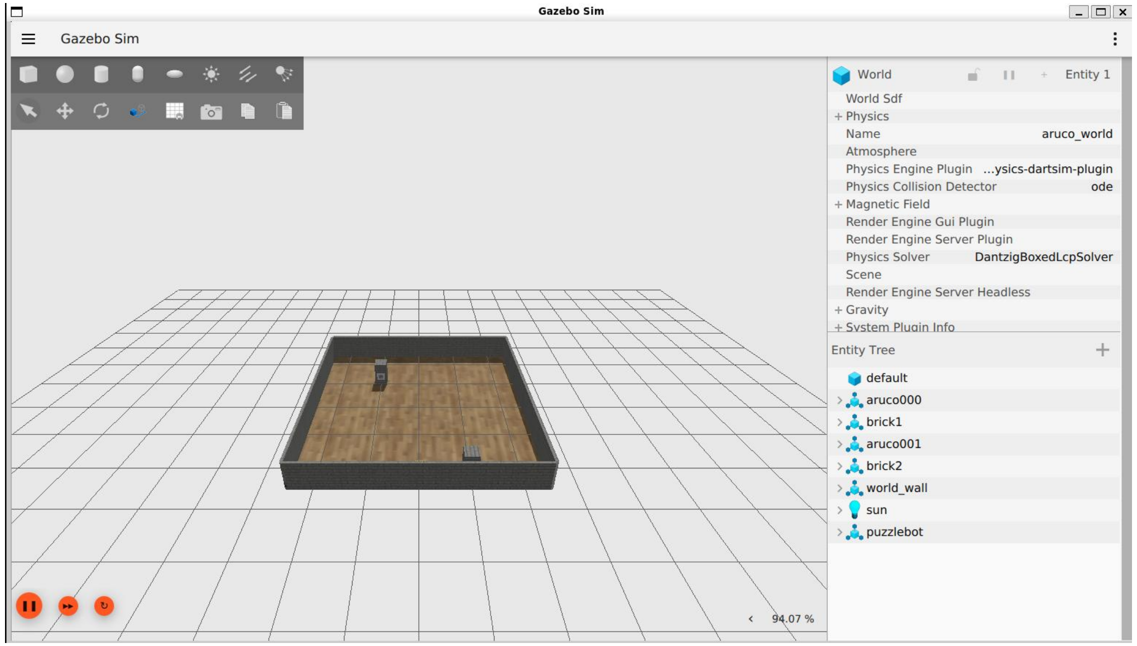
- Build and source the packages

```
$ cd ~/ros2_ws
$ colcon build
$ source install/setup.bash
```

- Run the Puzzlebot gazebo simulator using the following command

```
$ ros2 launch puzzlebot_gazebo bringup_simulation_launch.py
```

- The following screen should appear



- If having rendering issues using a VM or WSL type the following command. More information [here]

```
$ export LIBGL_ALWAYS_SOFTWARE=1
```

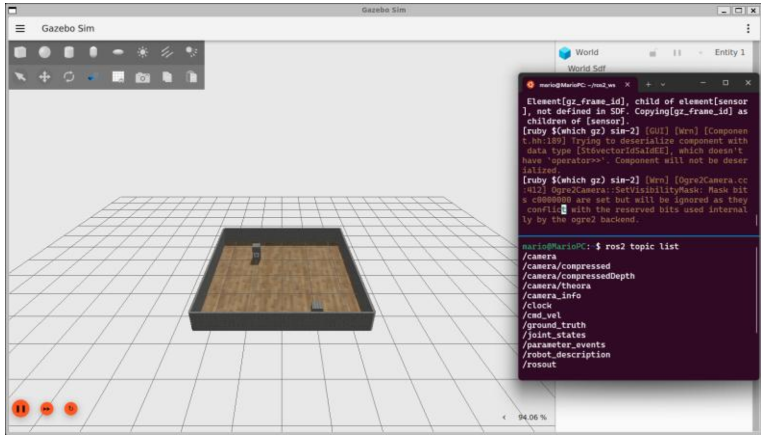
Gazebo simulation

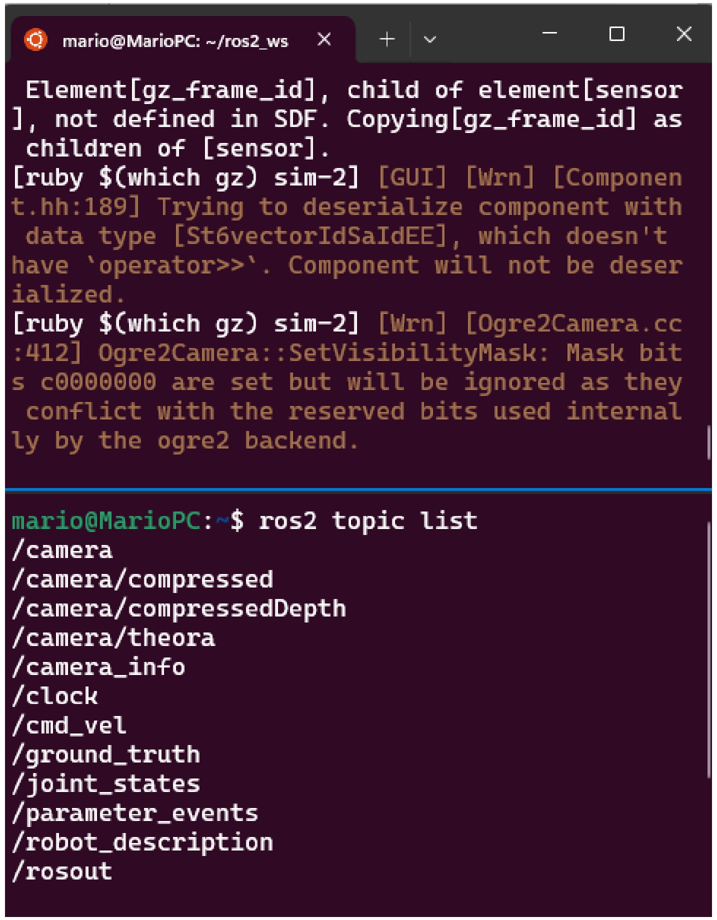
Initialising the simulation

- Open another terminal
- Verify the topics generated by the simulator using the following command

```
$ ros2 topic list
```

- The following list of topics should appear



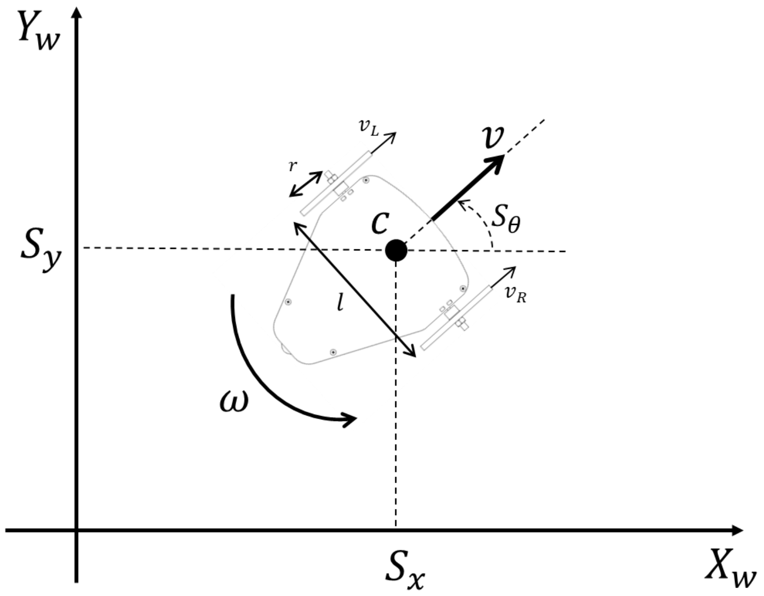


Gazebo simulation

Running the simulation

- Sending velocity commands to the robot is done using the following topic

```
/cmd_vel
```



- The message to be used is a geometry_msg/Twist (more information here)
- The robot's linear velocity v and angular velocity ω must be set in the *linear.x* and *angular.z* sections of the message (ROS convention for differential drive robots).

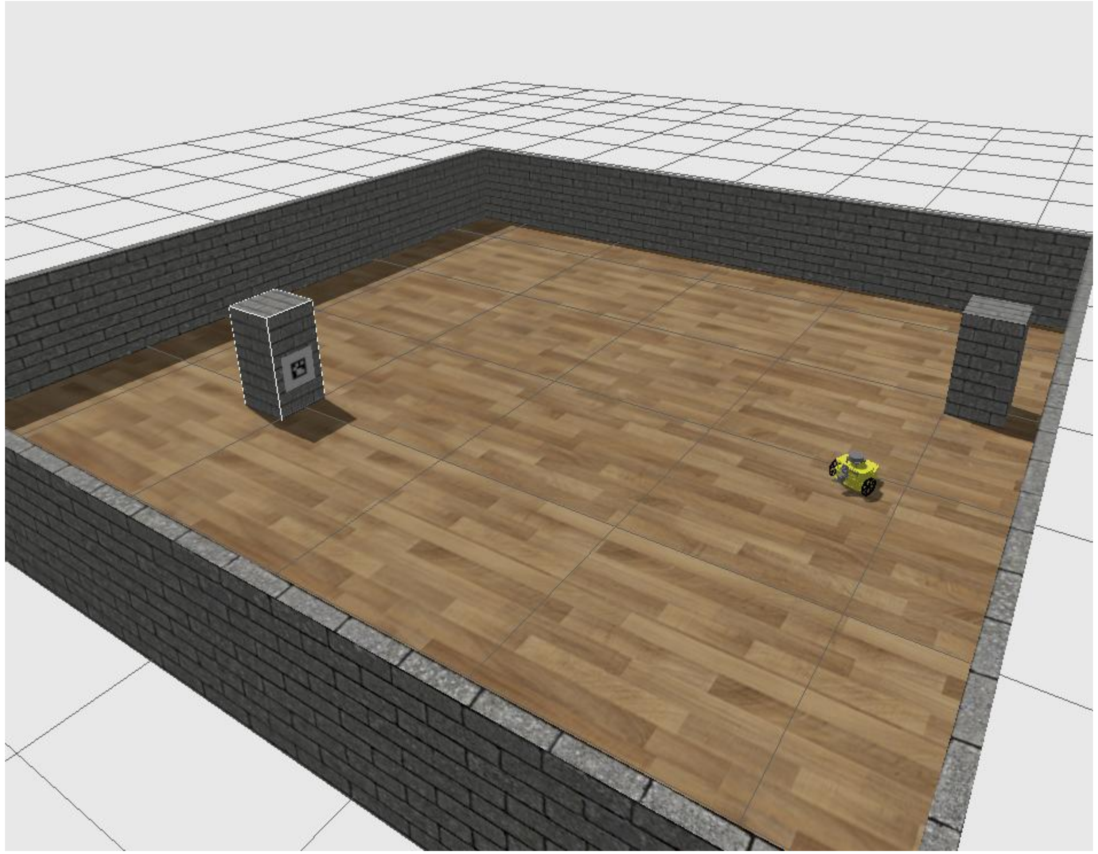
```
Topic: /cmd_vel
Message: geometry_msgs/Twist
     "linear:
       x: linear_velocity (v)
       y: 0.0
       z: 0.0
     angular:
       x: 0.0
       y: 0.0
       z: angular_velocity (omega)"
```

Gazebo simulation

Running the simulation

- To publish a message to the gazebo-simulated robot from the terminal can be done as follows.

```
$ ros2 topic pub /cmd_vel geometry_msgs/msg/Twist
"linear:
  x: 0.3
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.3"
```



- The previous command will make the robot to run in a circle.

Gazebo simulation

Running the simulation

- To obtain the wheel speed (ω_r, ω_l) from the robot, the following topics must be used

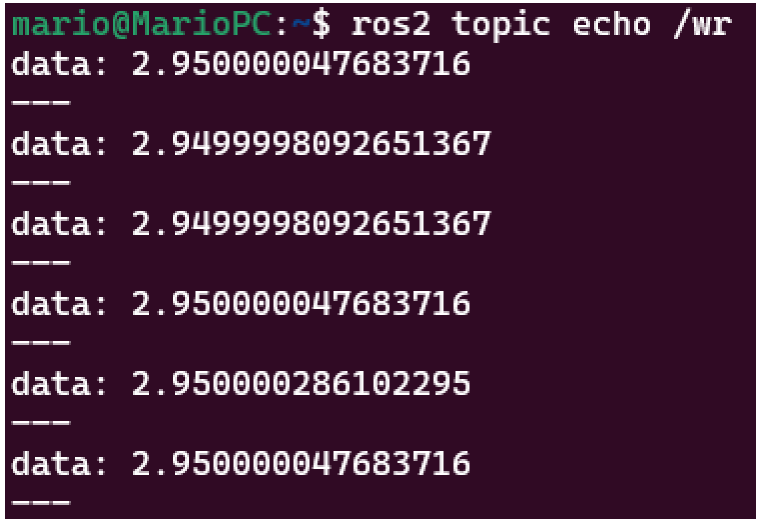
```
/VelocityEncR
/VelocityEncL
```

- Getting the wheel velocities from the robot is done using the following command

```
$ ros2 topic echo /VelocityEncR
```

- The topic uses a std_msg/Float32 message.

```
Topic: /VelocityEncR
Message: std_msgs/Float32
```



Robot Teleoperation

- Install the teleoperation package

```
sudo apt-get install ros-humble-teleop-twist-keyboard
```

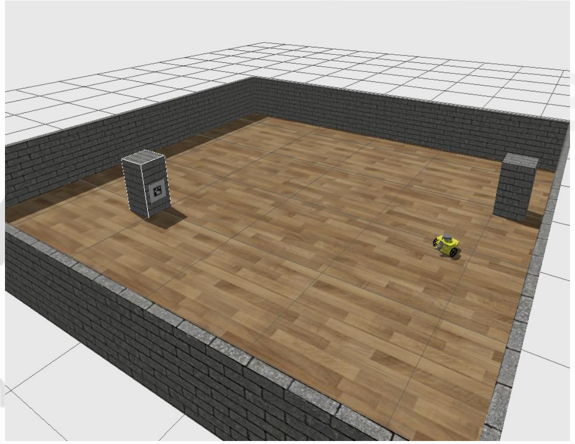
- Run the gazebo simulator as described before

```
$ ros2 launch puzzlebot_gazebo bringup_simulation_launch.py
```

- In another terminal, run the teleoperation node as follows

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

- Control the robot using the keyboard (i, j, l, k) as described on the terminal.



Puzzlebot Gazebo Simulator

Advanced Features

{Learn, Create, Innovate};

Simulator Configuration

Simulation Configuration

- The simulator can be configured from the file "bringup_simulation_launch.py"
- The configuration is defined in the section "SIMULATION CONFIGURATION"
- In this section, the user can configure the world to be used, the robot type and the position of the robot, as well as the frames for each sensor (if applicable).

```
# ---------------------------------------------------------------------------
#                          SIMULATION CONFIGURATION
# ---------------------------------------------------------------------------

# Name of the Gazebo world to load
world = 'obstacle_avoidance_4.world'

# General Gazebo settings
pause = 'true'            # Start Gazebo in paused state
verbosity = '4'           # Gazebo log verbosity level
use_sim_time = 'true'     # Enable use of simulated clock (for ROS time sync)

# Robot configurations (can be extended or loaded from a JSON file in future)
robot_config_list = [
    {
        'name': '',
        'type': 'puzzlebot_jetson_lidar_ed',
        'x': 0.0, 'y': 0.0, 'yaw': 0.0,
        'lidar_frame': 'laser_frame',
        'camera_frame': 'camera_link_optical',
        'tof_frame': 'tof_link'
    }
]
```

Changing the environment

Puzzlebot Environments

- MCR2 provide different environments for the Puzzlebot to interact with.
- The available environments are in the package "*puzzlebot_gazebo*" in the folder "*worlds*".
- The world description files are the ones ending in ".world" or ".sdf".

```
• empty.world
• obstacle_avoidance_1.world
• obstacle_avoidance_2.world
• obstacle_avoidance_3.world
• obstacle_avoidance_4.world
• office.world
• puzzlebot_arena.world
• puzzlebot_arena_markers.world
• puzzlebot_world.world
```

Changing the environment

- Open the file "*bringup_simulation_launch.py*" in the "*puzzlebot_gazebo*" package.

```
~/puzzlebot_gazebo/launch/bringup_simulation_launch.py
```

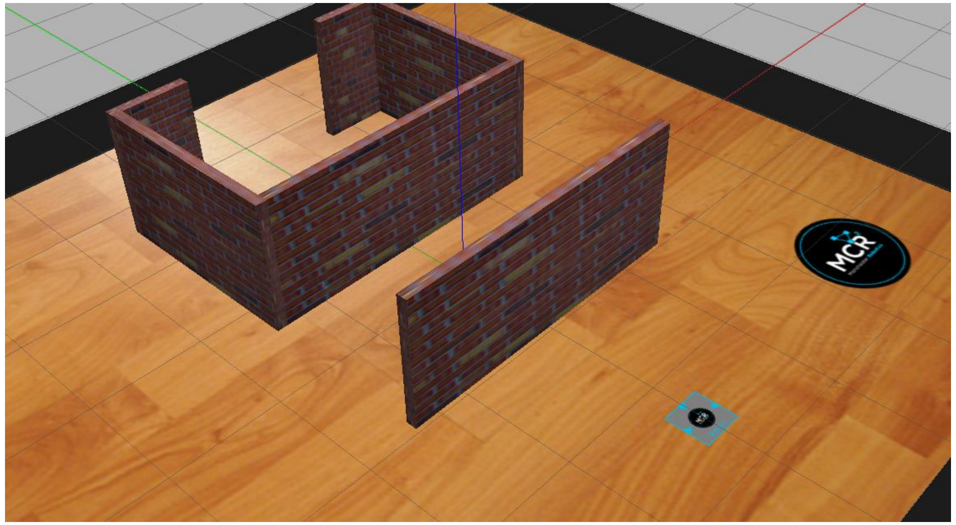
- Line 15 controls the environment to be used.

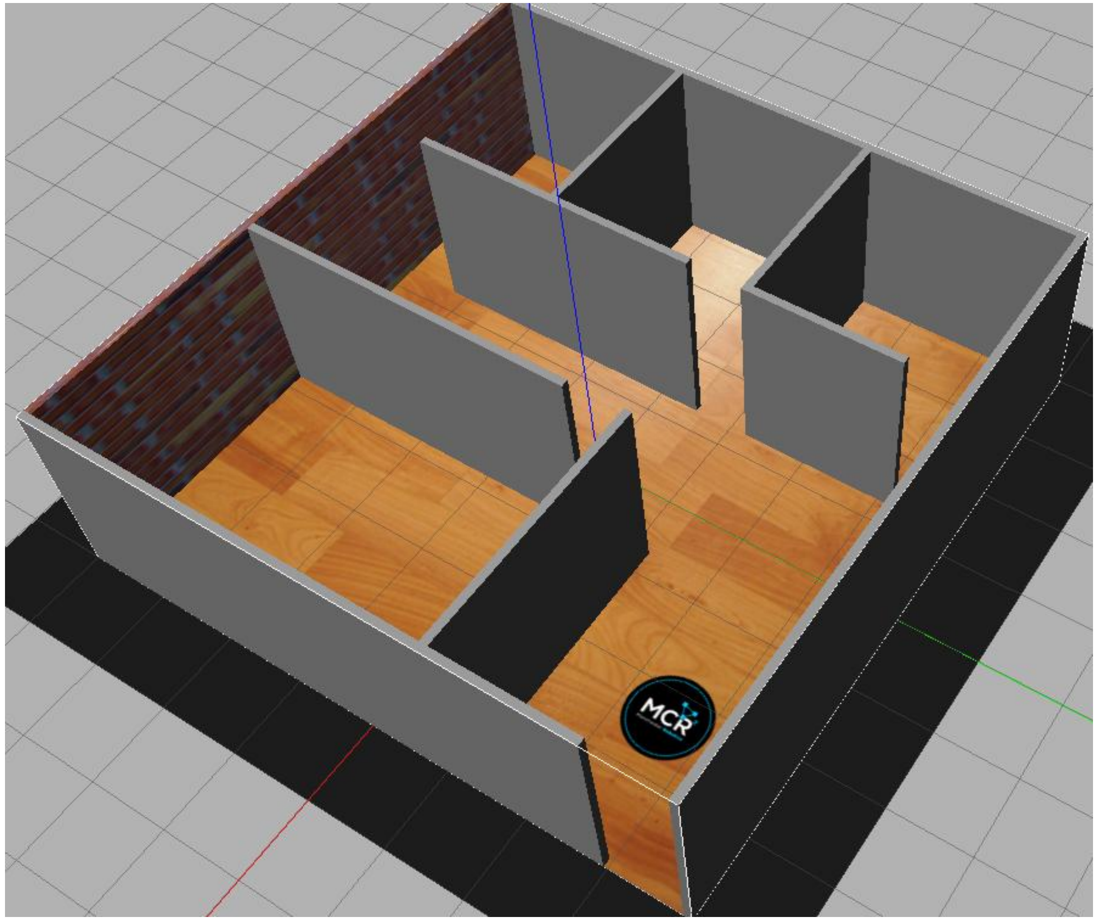
```
world = 'world_aruco_mip.sdf'
```

- Change the value of the name in the default value by the name of the world you want to use.

Changing the environment

- Save the file
- Follow the instructions in the previous section to initialise the Gazebo simulator.
- The following models should appear on the screen.





Simulator Configuration

General Gazebo Settings

- The Gazebo general settings can be also be configured.
- Open the file "*bringup_simulation_launch.py*" in the "*puzzlebot_gazebo*" package.

```
~/puzzlebot_gazebo/launch/bringup_simulation_launch.py
```

- Lines 17-20 control some basic settings such as pause, verbosity and sim_time.

```
# General Gazebo settings
pause = 'true'              # Start Gazebo in paused state
verbosity = '4'             # Gazebo log verbosity level
use_sim_time = 'true'       # Enable use of simulated clock
```

- **Pause** (true/false): sets the simulation to start paused or running from the start.
- **Verbosity** [0-4]: Sets the simulator's verbosity level (log description) 4 prints more information about the simulator on the terminal.
- **use_sim_time** (true/false): Enables the use of Gazebo clock (simulated clock). The clock is published on the topic "/clock"
  - If using it, ensure the parameter "use_sim_time" of each node that uses time from Gazebo is enabled.
  - Set to false if using Gazebo and a real robot simultaneously.

Changing Puzzlebot model

General Gazebo Settings

- The Puzzlebot models/entities are configured from a list that contains all the parameters required for each robot.

```
robot_config_list = [
    {
        'name': 'robot1',
        'type': 'puzzlebot_jetson_lidar_ed',
        'x': 0.0, 'y': 0.0, 'yaw': 0.0,
        'lidar_frame': 'laser_frame',
        'camera_frame': 'camera_link_optical',
        'tof_frame': 'tof_link'
    }
]
```

Configuration List

- Name: Sets the gazebo name of the robot (can be left blank). This name will be used to set the robot namespaces and prefixes of the topics and transforms (useful when simulating multiple robots).
- Type: Puzzlebot Selection

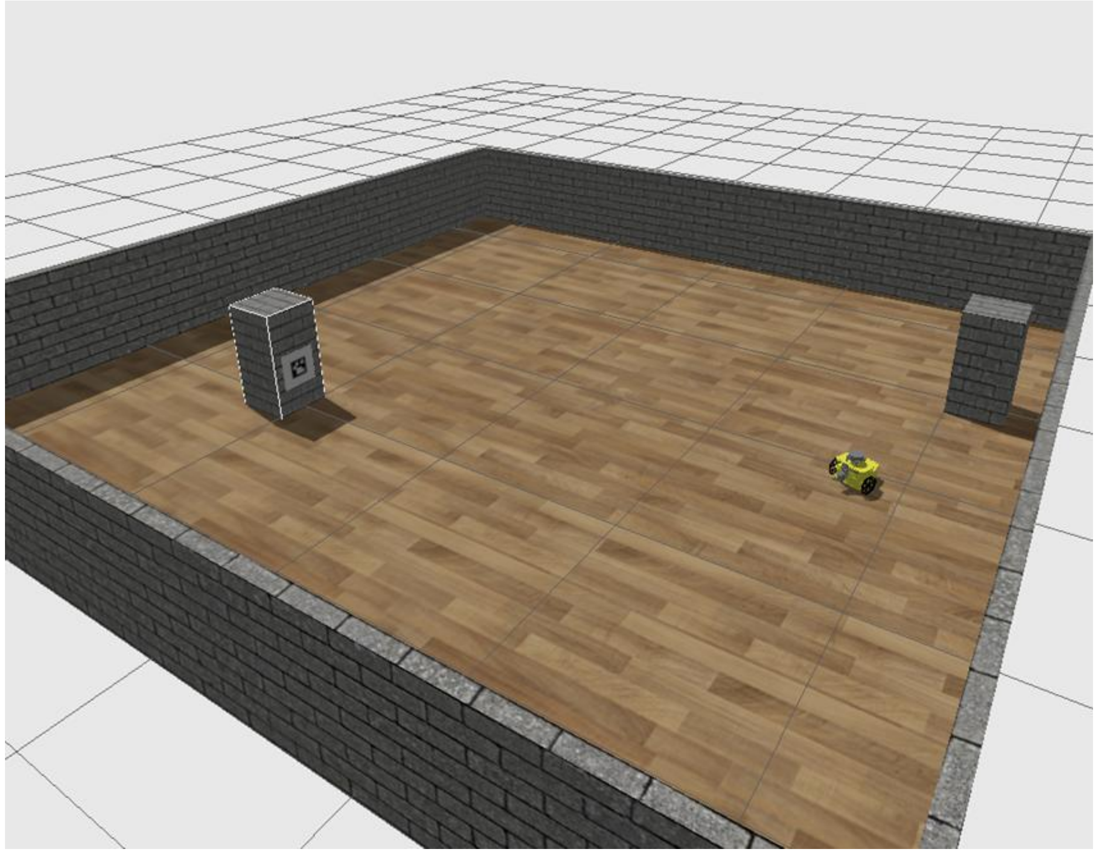
```
puzzlebot_hacker_ed
puzzlebot_jetson_ed
puzzlebot_jetson_lidar_ed
```

- x, y, yaw: Robot Initial position $[m, m, rad]$
- lidar_frame, camera_frame, tof_frame: Output Message Header frame ID for each sensor. Frame the data is associated with. The frames: laser_frame, camera_link_optical, and tof_frame are gazebo frames only used as an example; the user must use their own frames.

```
header:
  stamp:
    sec: 316
    nanosec: 400000000
  frame_id: camera_link_optical_2
```

Changing Puzzlebot model

- Save the file
- Follow the instructions in the previous section to initialise the Gazebo simulator.
- The following models should appear on the screen.



Changing Puzzlebot model

- Each robot contains different topics according to the sensors and the information published.
- Use a "*ros2 topic list*" command to view the topics of each robot

```
mario@MarioPC:~$ ros2 topic list
/camera
/camera/compressed
/camera/compressedDepth
/camera/theora
/camera_info
/clock
/cmd_vel
/ground_truth
/joint_states
/parameter_events
/robot_description
/rosout
```

Gazebo simulation

Running the simulation

- The simulator outputs a transform tree for the robot that can be used for visualisation in RVIZ.
- The transform tree root is the “world” frame.
- The robot frames will change depending on the robot type, i.e., Puzzlebot_hacker_ed, Puzzlebot_jetson_edition, Puzzlebot_jetson_lidar_ed, due to the frames of the sensors.