Introduction to Gazebo in ROS 2

Learning Objectives

By the end of this lab, you will be able to:

1. Understand the purpose and capabilities of **Gazebo** for robot simulation in ROS 2.
2. Gain an overview of **Gazebo versions** and how they interact with different ROS versions.
3. Set up a **Gazebo simulation** environment in ROS 2.
4. Launch your first **Gazebo simulation** in ROS 2 (Humble).

Introduction: Gazebo is a powerful robot simulation software that integrates with the **Robot Operating System (ROS)**. It allows you to simulate robots in a 3D environment, complete with realistic physics, lighting, and sensor data. Gazebo is used for testing and prototyping robot designs without needing physical hardware, which saves time and resources.

Key Features of Gazebo:

- **Realistic Physics Engine**: Simulates gravity, collisions, and inertial effects.
- **3D Visualization**: Provides a 3D interface for monitoring the simulation.
- **Sensor Integration**: Simulates a wide range of sensors like cameras, LIDAR, and IMUs.
- **Plugin Support**: Customize and extend Gazebo through various plugins for robotic tasks.
- **ROS 2 Integration**: Allows seamless interaction with ROS 2 to send commands, receive sensor data, and control robots.

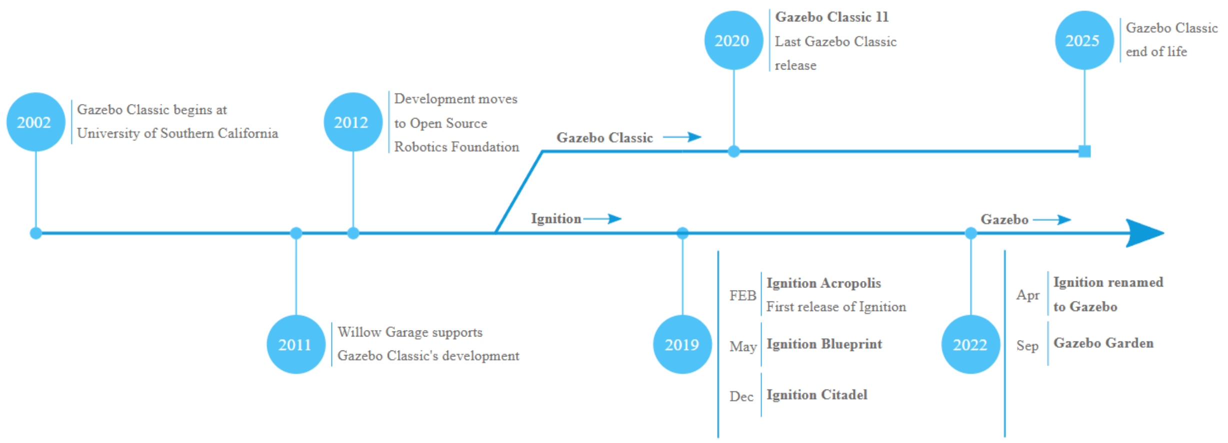
History of Gazebo from their Site.

Gazebo has gone through several major versions, with each one offering improvements in performance, integration with ROS, and support for new features. Since ROS 2 came into the picture, the **Gazebo 11** and **Ignition** series have become the most used versions.

Gazebo was initially launched in 2002, and after more than 15 years of continuous development, it became evident that a major upgrade and modernization were necessary. This upgrade provided an opportunity to transition from a monolithic architecture to a collection of loosely coupled libraries, leading to the creation of the new development initiative, **Ignition**. The purpose of this rebranding was to clearly distinguish the new development from the original **Gazebo**, now referred to as **Gazebo Classic**.

The transition to Ignition was highly successful, thanks in large part to the contributions and support from our global community of users and stakeholders.

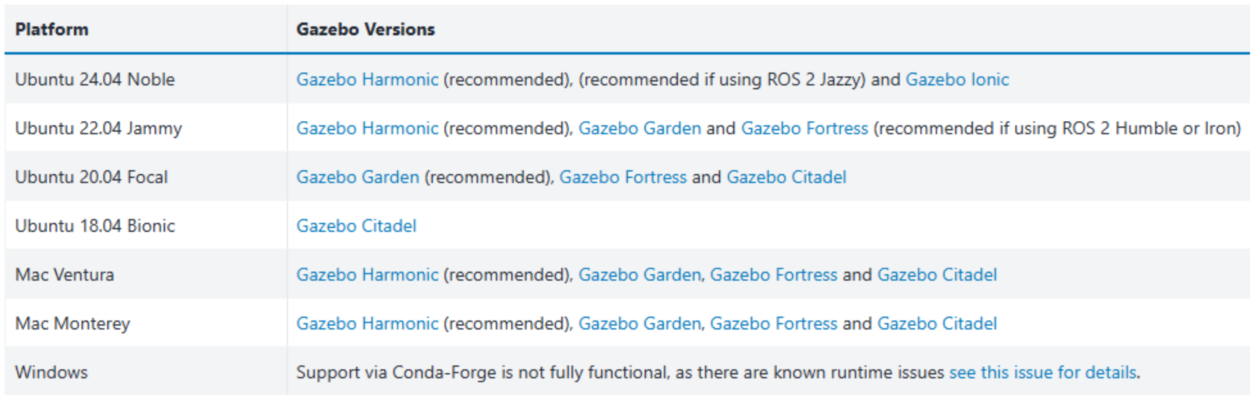
However, in 2022, a trademark issue arose concerning the use of the name "Ignition." We took this challenge as an opportunity to return to the widely recognized name of **Gazebo**. Going forward, the modern robotics software collection previously known as Ignition will now be branded as **Gazebo**. See the following diagram to have visualization of the



2002
Gazebo Classic begins at University of Southern California
2011
Willow Garage supports Gazebo Classic's development
2012
Development moves to Open Source Robotics Foundation
Gazebo Classic
Ignition
2019
FEB Ignition Acropolis First release of Ignition
May Ignition Blueprint
Dec Ignition Citadel
2020
Gazebo Classic 11 Last Gazebo Classic release
Gazebo
2022
Apr Ignition renamed to Gazebo
Sep Gazebo Garden
2025
Gazebo Classic end of life

Gazebo 11 is the latest stable version that works well with **ROS 2 Humble** (Garden). It uses the **Ignition Fortress** framework and supports the **gz** command-line interface.

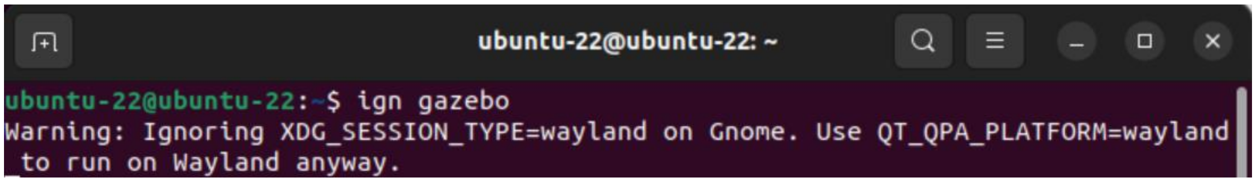
Ignition Fortress: We will use ignition fortress as it is recommended for operating system (Ubuntu 22.04) and the ROS version (Humble) as per their website.



Platform	Gazebo Versions
Ubuntu 24.04 Noble	Gazebo Harmonic (recommended), (recommended if using ROS 2 Jazzy) and Gazebo Ionic
Ubuntu 22.04 Jammy	Gazebo Harmonic (recommended), Gazebo Garden and Gazebo Fortress (recommended if using ROS 2 Humble or Iron)
Ubuntu 20.04 Focal	Gazebo Garden (recommended), Gazebo Fortress and Gazebo Citadel
Ubuntu 18.04 Bionic	Gazebo Citadel
Mac Ventura	Gazebo Harmonic (recommended), Gazebo Garden, Gazebo Fortress and Gazebo Citadel
Mac Monterey	Gazebo Harmonic (recommended), Gazebo Garden, Gazebo Fortress and Gazebo Citadel
Windows	Support via Conda-Forge is not fully functional, as there are known runtime issues see this issue for details.

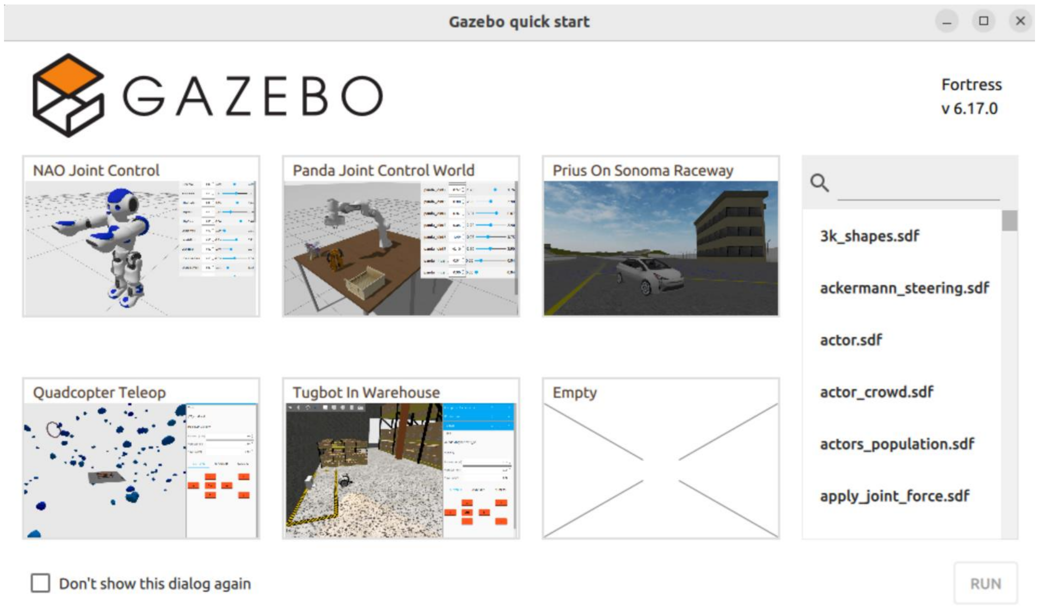
Installing ignition fortress: It can be installed by executing few commands given on https://gazebosim.org/docs/fortress/install_ubuntu/ link given on their official page. Navigate to right top right corner to ensure the fortress is selected to avoid any problem in the installation process.

Launching Gazebo: We can launch the gazebo quick start page by executing the following command in the terminal.

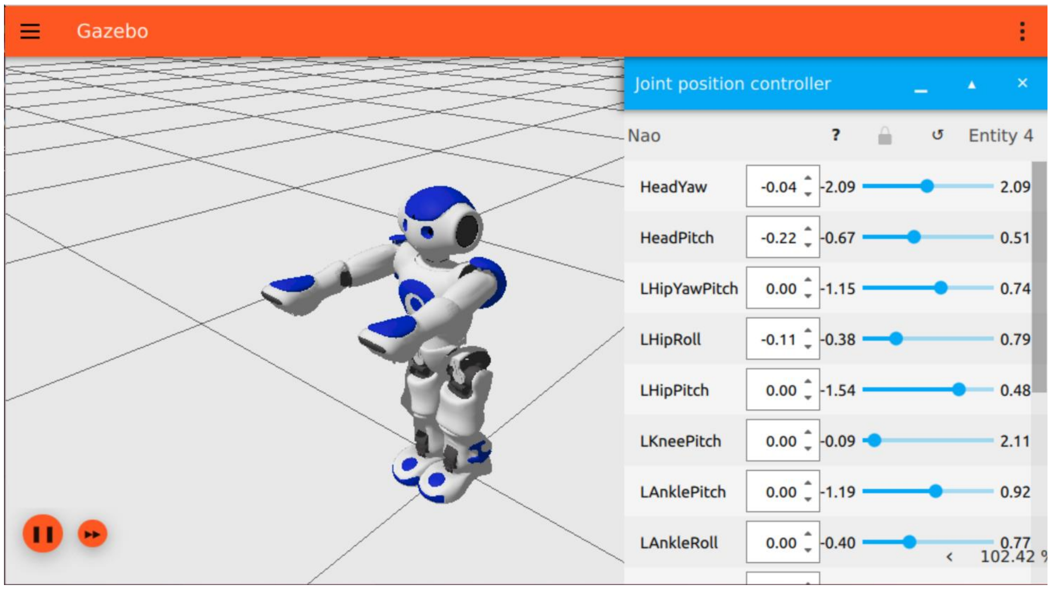


```
ubuntu-22@ubuntu-22:~$ ign gazebo
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland
 to run on Wayland anyway.
```

Output: After launching the gazebo quick start page, you will see the following window. Select the Nao joint control and RUN.

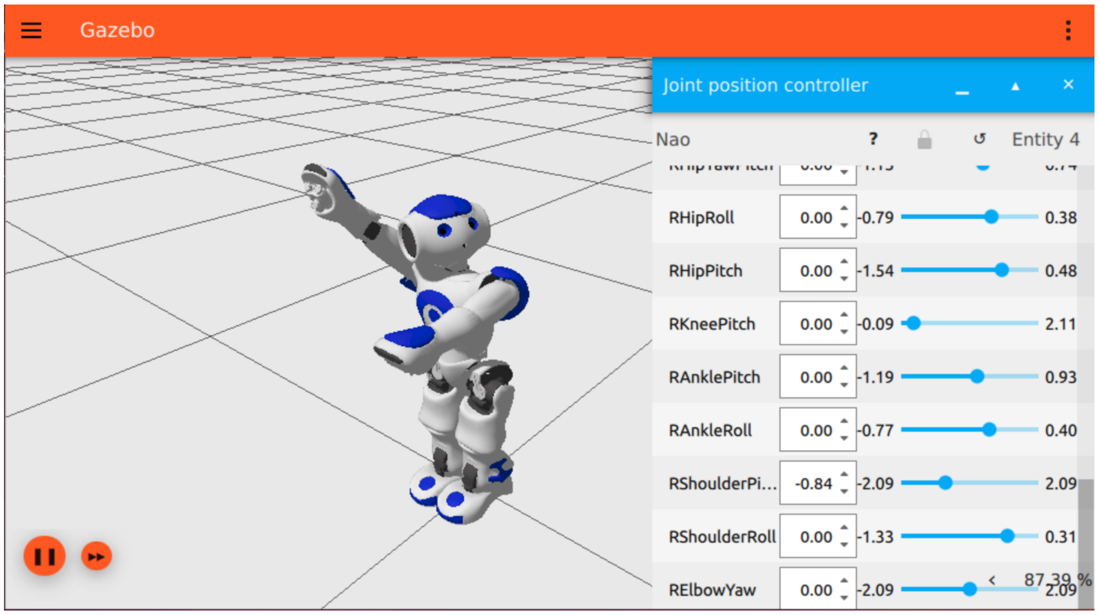


Nao Simulation in Gazebo: After running it, you will the Nao robot being in the Gazebo GUI window as given below. Click the play button to start the simulation.

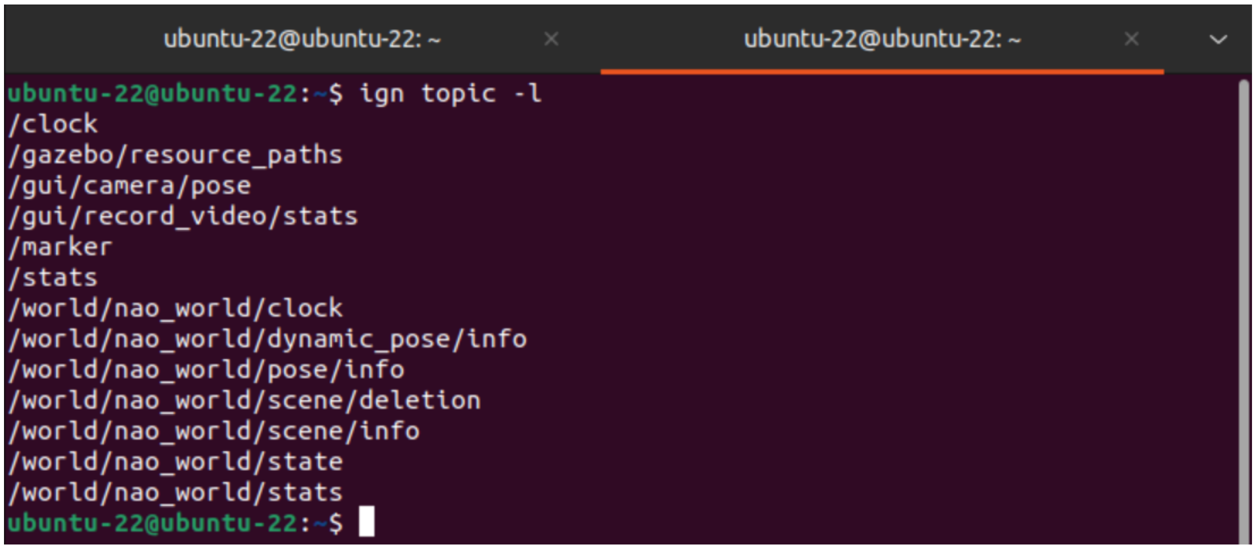


HeadYaw	-0.04	-2.09	2.09
HeadPitch	-0.22	-0.67	0.51
LHipYawPitch	0.00	-1.15	0.74
LHipRoll	-0.11	-0.38	0.79
LHipPitch	0.00	-1.54	0.48
LKneePitch	0.00	-0.09	2.11
LAnklePitch	0.00	-1.19	0.92
LAnkleRoll	0.00	-0.40	0.77

Joint Control: We can control all joints of Nao robot in the joint position controller given on the left side of the Gazebo GUI window as shown above and it will be simulated in real-time as shown below.

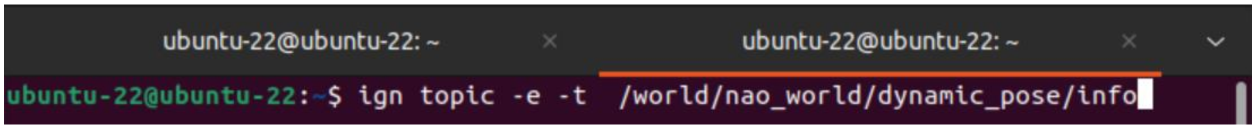

Gazebo
Joint position controller
Nao
Entity 4
RHipRoll 0.00 -0.79 0.38
RHipPitch 0.00 -1.54 0.48
RKneePitch 0.00 -0.09 2.11
RAnklePitch 0.00 -1.19 0.93
RAnkleRoll 0.00 -0.77 0.40
RShoulderPi... -0.84 -2.09 2.09
RShoulderRoll 0.00 -1.33 0.31
RElbowYaw 0.00 -2.09 2.09
87.39 %

Ignition Topics: We can also see the active topics in the current running simulation by using the following command in the terminal.



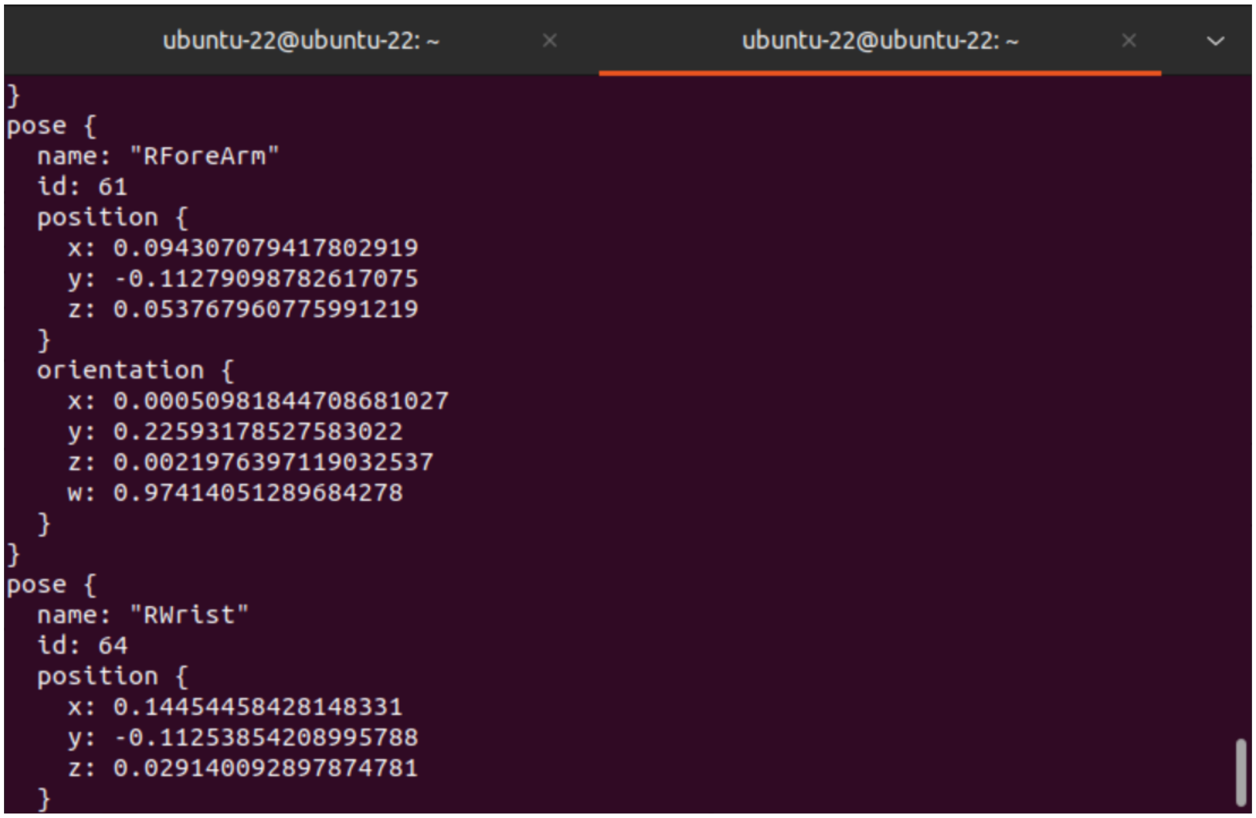
```
ubuntu-22@ubuntu-22:~$ ign topic -l
/clock
/gazebo/resource_paths
/gui/camera/pose
/gui/record_video/stats
/marker
/stats
/world/nao_world/clock
/world/nao_world/dynamic_pose/info
/world/nao_world/pose/info
/world/nao_world/scene/deletion
/world/nao_world/scene/info
/world/nao_world/state
/world/nao_world/stats
ubuntu-22@ubuntu-22:~$
```

Echoing the topic: We can see real-time change in the joints of the NAO robot by executing the following command in the terminal. It will echo the real-time simulated dynamic pose of NAO in the Gazebo world.



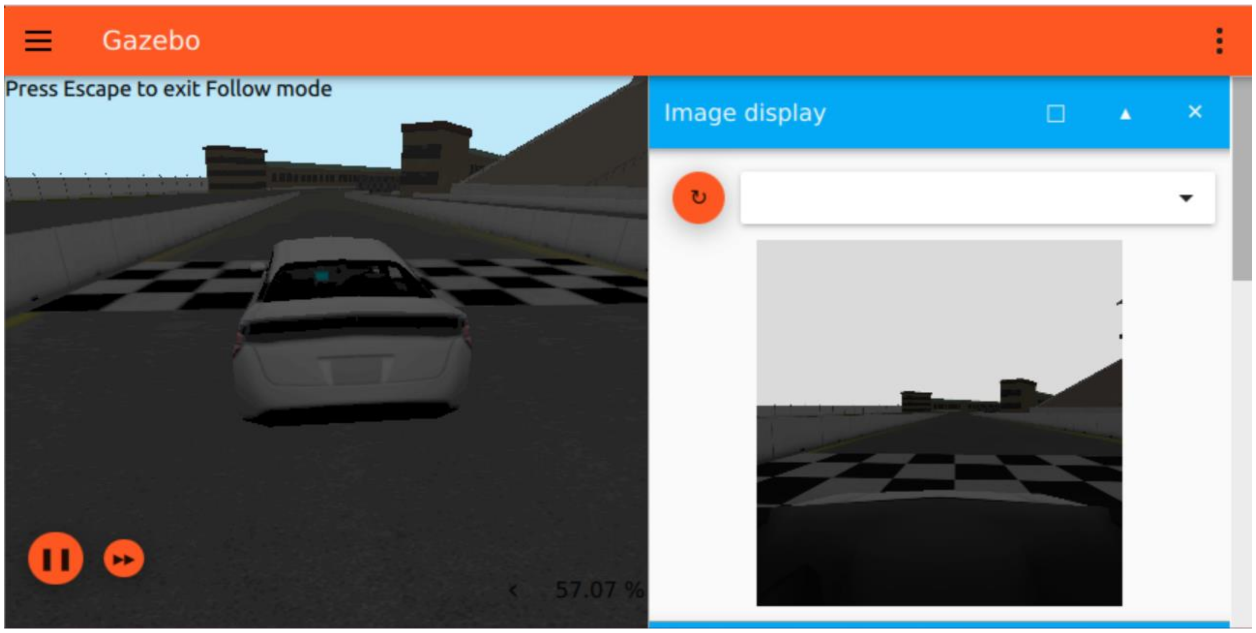
```
ubuntu-22@ubuntu-22:~$ ign topic -e -t  /world/nao_world/dynamic_pose/info
```

After echoing, change the joints of the NAO from the joint control panel given on the Gazebo GUI. It will cause the real-time change in Nao joint values in respective interconnected and interrelated joints as can be seen in

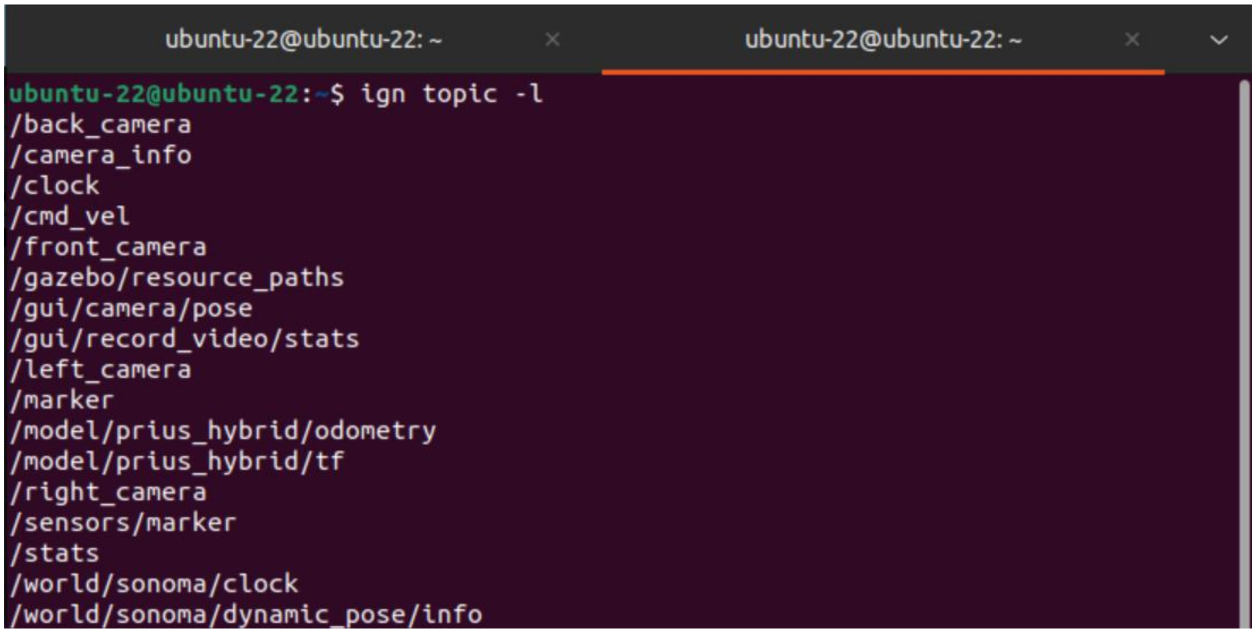


```
}
pose {
  name: "RForeArm"
  id: 61
  position {
    x: 0.094307079417802919
    y: -0.11279098782617075
    z: 0.053767960775991219
  }
  orientation {
    x: 0.00050981844708681027
    y: 0.22593178527583022
    z: 0.0021976397119032537
    w: 0.97414051289684278
  }
}
pose {
  name: "RWrist"
  id: 64
  position {
    x: 0.14454458428148331
    y: -0.11253854208995788
    z: 0.029140092897874781
  }
```

Prius Hybrid Model: Now to understand other components of Gazebo and robotics let's launch prius hybrid model in Gazebo simulation, after launching you will see the following window.

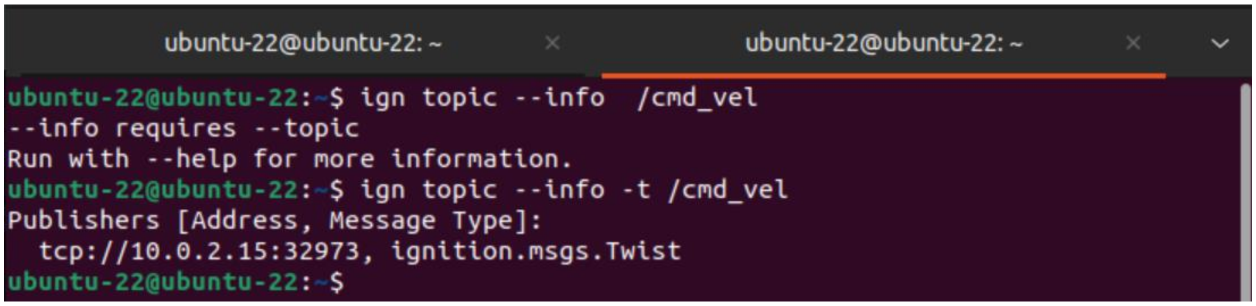


Printing the topics



```
ubuntu-22@ubuntu-22:~$ ign topic -l
/back_camera
/camera_info
/clock
/cmd_vel
/front_camera
/gazebo/resource_paths
/gui/camera/pose
/gui/record_video/stats
/left_camera
/marker
/model/prius_hybrid/odometry
/model/prius_hybrid/tf
/right_camera
/sensors/marker
/stats
/world/sonoma/clock
/world/sonoma/dynamic_pose/info
```

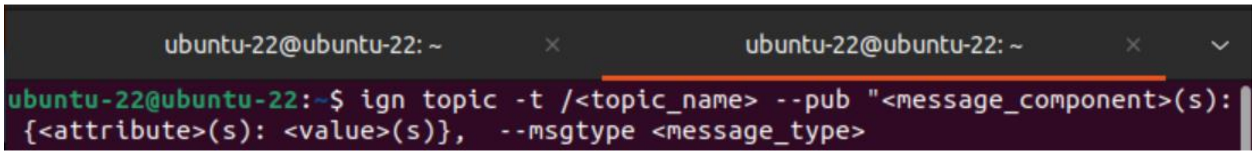
Command Velocities: We can move the car using the /cmd_vel topic, which stands for command velocities. This prius model is differential drive type mobile robot meaning it can accept linear velocities on x-axis (forward and backward) and angular velocities around z-axis (left and right). Let's investigate its message type to publish the respective data using the info option.



```
ubuntu-22@ubuntu-22:~$ ign topic --info  /cmd_vel
--info requires --topic
Run with --help for more information.
ubuntu-22@ubuntu-22:~$ ign topic --info -t /cmd_vel
Publishers [Address, Message Type]:
  tcp://10.0.2.15:32973, ignition.msgs.Twist
ubuntu-22@ubuntu-22:~$
```

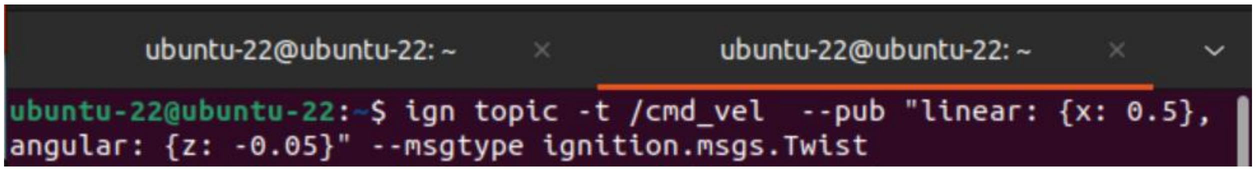
Note: We will have to provide -t (topic) option along with info as it's syntax works this way as shown above. Also we found that it receives the ignition.msgs.Twist message

Publishing on /cmd_vel: We can use option -p or –pub (publish) along with -m or –msgtype option with the following general syntax.



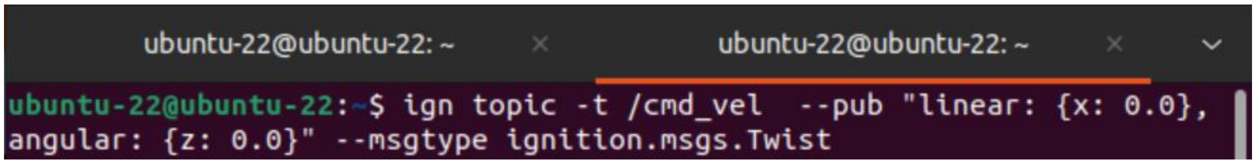
```
ubuntu-22@ubuntu-22:~$ ign topic -t /<topic_name> --pub "<message_component>(s):
 {<attribute>(s): <value>(s)},  --msgtype <message_type>
```

This case: Here we have two message components; **linear: (axis: velocity)** and **angular: (axis: velocity).** By publishing some data by using the following command, the car should be moving accordingly in the Gazebo simulation in real-time.



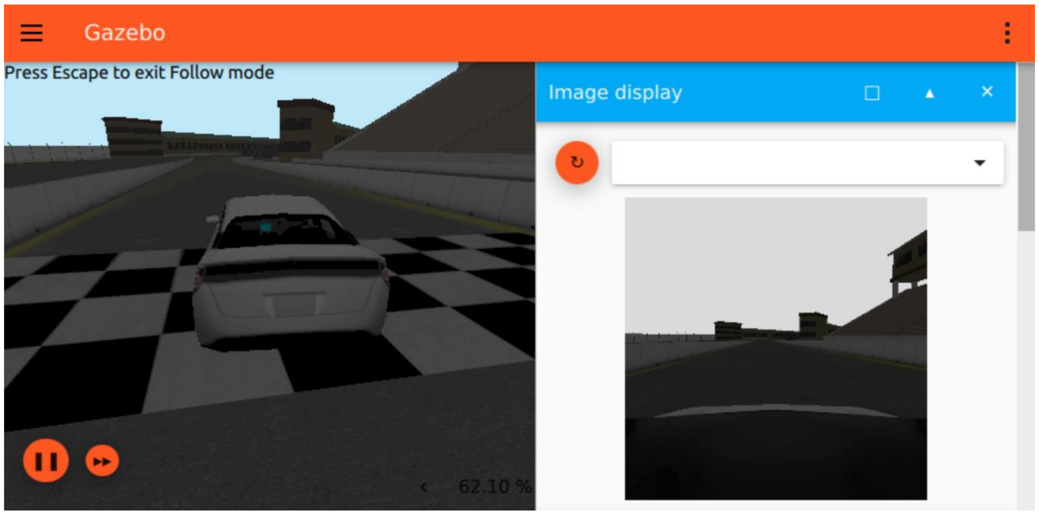
```
ubuntu-22@ubuntu-22:~$ ign topic -t /cmd_vel  --pub "linear: {x: 0.5},
angular: {z: -0.05}" --msgtype ignition.msgs.Twist
```

Stopping the Robot: We can stop the robot by publishing zero on both linear and angular velocities using the same command as shown below.

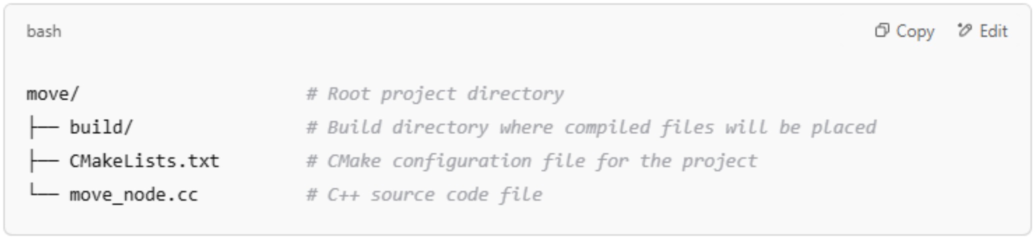


```
ubuntu-22@ubuntu-22:~$ ign topic -t /cmd_vel  --pub "linear: {x: 0.0},
angular: {z: 0.0}" --msgtype ignition.msgs.Twist
```

Movement in Robot: Below we can see the robot just moved some distance from the original position in the real-time Gazebo simulation.

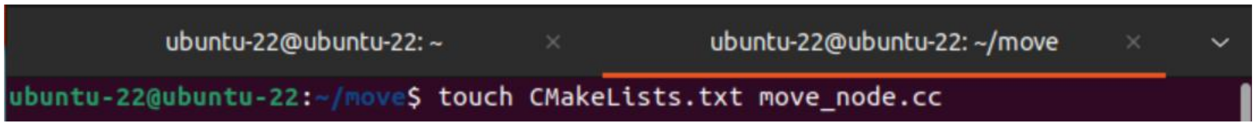


Controlling robot with script: We can write a C++ script to automate and customize the things according to us. For that we need to create some workspace to integrate the scripts or nodes with the gazebo simulation. We will have to create a folder with the name "move" or you can choose your own name. Create move_node.cc, CMakeLists.txt and build/ folder in it as shown below.



```
move/                 # Root project directory
├── build/            # Build directory where compiled files will be placed
├── CMakeLists.txt    # CMake configuration file for the project
└── move_node.cc      # C++ source code file
```

We can use touch command to create and nano or geany to edit the file



```
ubuntu-22@ubuntu-22:~/move$ touch CMakeLists.txt move_node.cc
```

Move Node Script: Put the following C++ code in the file.

```
#include <iostream>
#include <ignition/msgs/twist.pb.h>
#include <ignition/transport/Node.hh>
#include <ncurses.h>

ignition::transport::Node node;
auto pub = node.Advertise<ignition::msgs::Twist>("/cmd_vel");

void sendCommand(double linear_x, double angular_z) {
   ignition::msgs::Twist data;
   data.mutable_linear()->set_x(linear_x);
   data.mutable_angular()->set_z(angular_z);
   pub.Publish(data);
}

int main(int argc, char **argv) {
   initscr();
   timeout(0); // Set non-blocking input

   std::cout << "Control the robot:\n"
             "w - Move Forward\n"
             "a - Turn Left\n"
             "d - Turn Right\n"
             "x - Move Backward\n"
             "s - Stop\n"
             "q - Quit\n";

   while (true) {
      int ch = getch(); // Get character without waiting
      if (ch != ERR) { // If a key was pressed
         switch (ch) {
            case 'w': // Move Forward
               sendCommand(2, 0.0);
               break;
            case 'x': // Move Backward
               sendCommand(-2, 0.0);
               break;
            case 'a': // Turn Left
               sendCommand(0.0, 2);
               break;
            case 'd': // Turn Right
               sendCommand(0.0, -2);
               break;
            case 's': // Stop
               sendCommand(0.0, 0.0);
               break;
            case 'q': // Quit
               endwin(); // End curses mode
               return 0;
            default:
               break;
         }
      }
      // Your main loop logic here }}
```

Explanation: move_node.cc controls a robot by publishing movement commands to the /cmd_vel topic. It uses Ignition Transport to send velocity messages and ncurses for interactive keyboard input. The program listens for key presses (w, a, d, x, s, q) and sends corresponding movement commands (forward, backward, turn, stop, quit) to control the robot in real-time.

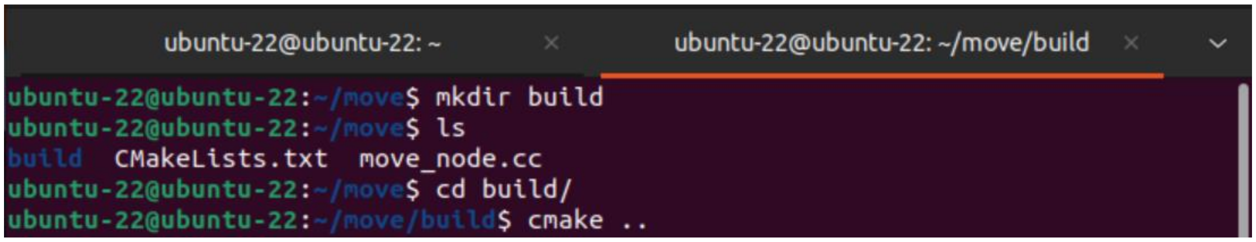
CMakeFile: Put the following configuration code in the CMake file to link ignition and ncurses libraries with the node. Fortress supports ignition transport 11 package.

```
cmake_minimum_required(VERSION 3.10.2 FATAL_ERROR)

project(move_robot)

# Find the Ignition_Transport library
find_package(ignition-transport11 QUIET REQUIRED OPTIONAL_COMPONENTS log)
set(IGN_TRANSPORT_VER ${ignition-transport11_VERSION_MAJOR})

# Find the ncurses library
find_package(Curses REQUIRED)

include_directories(${CMAKE_BINARY_DIR})

if (EXISTS "${CMAKE_SOURCE_DIR}/move_node.cc")
  add_executable(move_node move_node.cc)
  target_link_libraries(move_node
    ignition-transport${IGN_TRANSPORT_VER}::core
    ${CURSES_LIBRARIES}  # Link ncurses
  )
endif()
```

Brief Explanation:

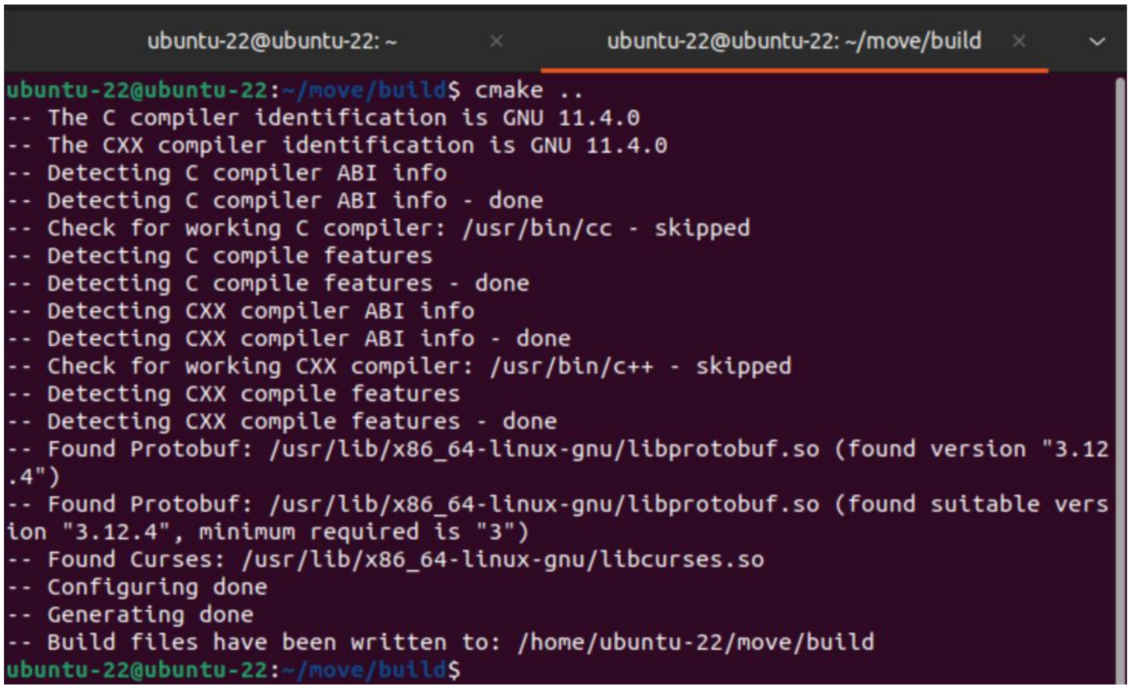
- **CMake Minimum Version:** Specifies the minimum CMake version required (3.10.2).
- Project Name: Defines the project name (move_robot).
- **Find Ignition Transport:** Searches for and ensures the Ignition Transport library is available.
- **Find Curses (ncurses):** Ensures the ncurses library is available for terminal control.
- **Create Executable:** Builds the move_node executable from move_node.cc.
- **Link Libraries:** Links the Ignition Transport and ncurses libraries to the move_node executable.

Running Cmake: Now we will make a folder with name build, navigate to run cmake .. to configure the build system by generating necessary files based on the project's CMakeLists.txt.



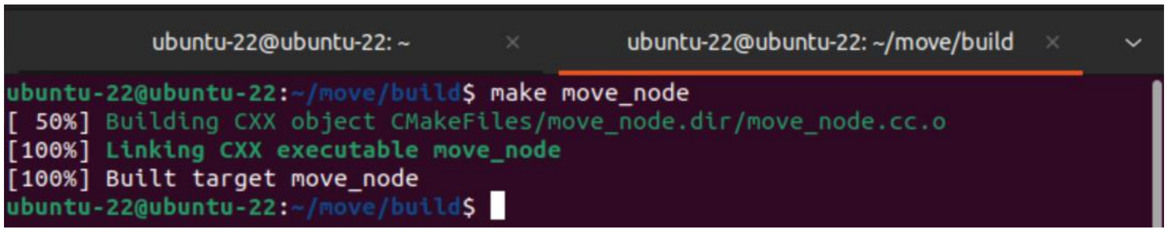
```
ubuntu-22@ubuntu-22:~/move$ mkdir build
ubuntu-22@ubuntu-22:~/move$ ls
build  CMakeLists.txt  move_node.cc
ubuntu-22@ubuntu-22:~/move$ cd build/
ubuntu-22@ubuntu-22:~/move/build$ cmake ..
```

Output: If everything worked fine, you would see the following messages after running the cmake.It will generate build files (e.g., Makefiles or project files) based on the configuration in CMakeLists.txt



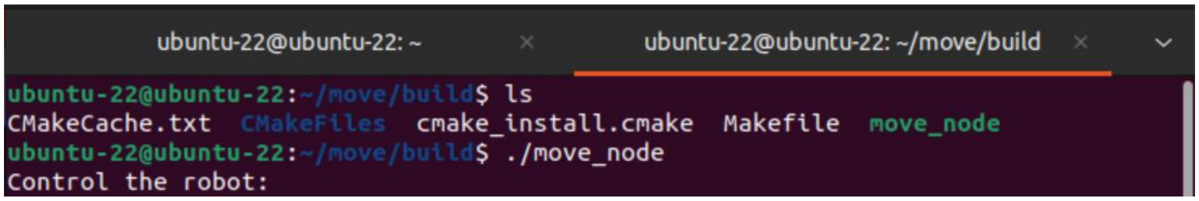
```
ubuntu-22@ubuntu-22:~/move/build$ cmake ..
-- The C compiler identification is GNU 11.4.0
-- The CXX compiler identification is GNU 11.4.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Protobuf: /usr/lib/x86_64-linux-gnu/libprotobuf.so (found version "3.12
.4")
-- Found Protobuf: /usr/lib/x86_64-linux-gnu/libprotobuf.so (found suitable vers
ion "3.12.4", minimum required is "3")
-- Found Curses: /usr/lib/x86_64-linux-gnu/libcurses.so
-- Configuring done
-- Generating done
-- Build files have been written to: /home/ubuntu-22/move/build
ubuntu-22@ubuntu-22:~/move/build$
```

Make Node: Now we will compile the source code and links libraries according to the build files generated by CMake as shown below to produce the final executable for move node.



```
ubuntu-22@ubuntu-22:~/move/build$ make move_node
[ 50%] Building CXX object CMakeFiles/move_node.dir/move_node.cc.o
[100%] Linking CXX executable move_node
[100%] Built target move_node
ubuntu-22@ubuntu-22:~/move/build$
```

Running the node: Now finally we can run the executable move node to control the robot using the terminal as below shown.



```
ubuntu-22@ubuntu-22:~/move/build$ ls
CMakeCache.txt  CMakeFiles  cmake_install.cmake  Makefile  move_node
ubuntu-22@ubuntu-22:~/move/build$ ./move_node
Control the robot:
```

Gazebo Compatibility Details: https://gazebosim.org/docs/latest/getstarted/

Ignition Fortress Installation Link: https://gazebosim.org/docs/fortress/install_ubuntu/

Learn More: https://gazebosim.org/docs/fortress/tutorials/

ROS 2 Integration: https://docs.ros.org/en/humble/Tutorials/Advanced/Simulators/Gazebo/Gazebo.html