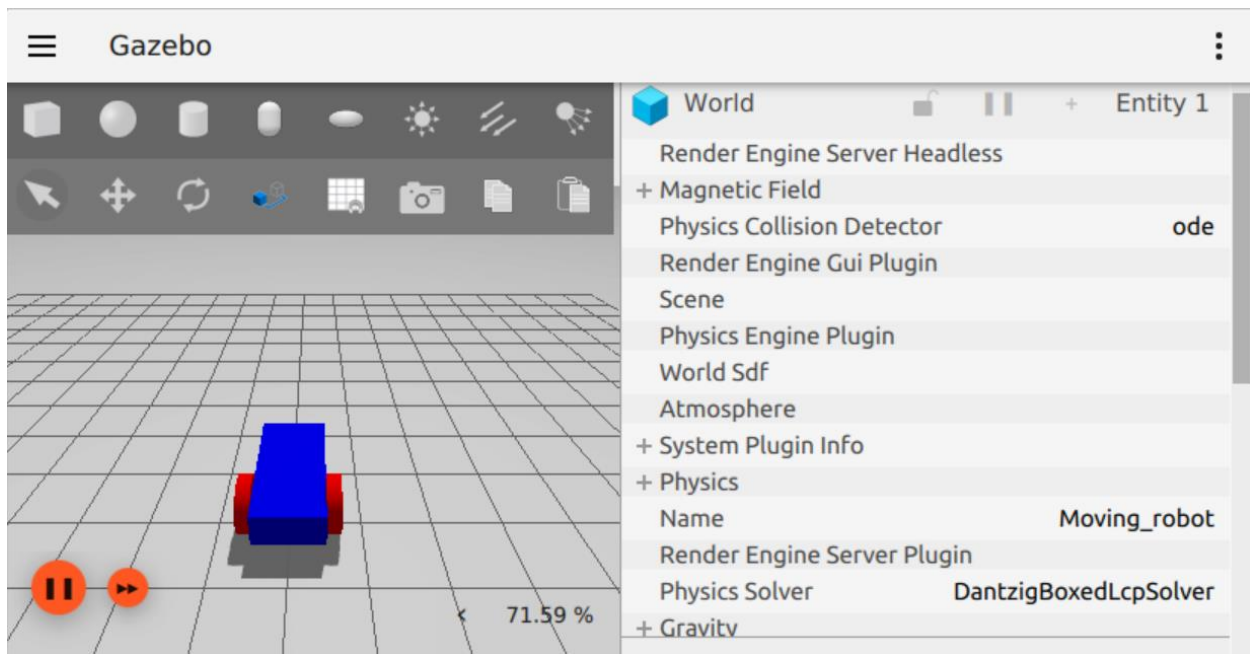


# Moving the robot in Gazebo Simulation

Our robot is simply a two-wheeled differential drive robot as shown in the following figure. It can be moved using simple command velocities in linear  $x$  (forward and backward movement) and angular  $z$  for (left and right turn).



To move the robot, we need to add the following gazebo plugins in our SDF file.

**DiffDrive:** This plugin enables the differential drive robot to receive command velocities and moves the left and right wheel for desired movement. It subscribes to 'cmd\_vel' topic to receive any velocity.

```
<plugin
  filename="libignition-gazebo-diff-drive-system.so"
  name="ignition::gazebo::systems::DiffDrive">
  <left_joint>left_wheel_joint</left_joint>
  <right_joint>right_wheel_joint</right_joint>
  <wheel_separation>1.2</wheel_separation>
  <wheel_radius>0.4</wheel_radius>
  <odom_publish_frequency>1</odom_publish_frequency>
  <topic>cmd_vel</topic>
</plugin>
```

**TriggeredPublisher:** This plugin is used to publish the command velocities on 'cmd\_vel' topic by using two values; linear x and angular z.

**Moving Forward:** For example below plugin is used to move the robot in forward direction by when "w" (87 in ASCII decimal) key is pressed. After key is pressed it publishes an ignition twist message on /cmd\_vel topic on linear x.

```
<!-- Moving Forward -->
<plugin filename="libignition-gazebo-triggered-publisher-system.so"
  name="ignition::gazebo::systems::TriggeredPublisher">
  <input type="ignition.msgs.Int32" topic="/keyboard/keypress">
    <match field="data">87</match>
  </input>
  <output type="ignition.msgs.Twist" topic="/cmd_vel">
    linear: {x: 0.5}, angular: {z: 0.0}
  </output>
</plugin>
```

**Moving Backward:** To move backward we need to publish on same linear x but with negative sign as shown below. We can publish it using 'x' key (88 in ASCII decimal).

```
<!-- Moving Backward -->
<plugin filename="libignition-gazebo-triggered-publisher-system.so"
  name="ignition::gazebo::systems::TriggeredPublisher">
  <input type="ignition.msgs.Int32" topic="/keyboard/keypress">
    <match field="data">88</match>
  </input>
  <output type="ignition.msgs.Twist" topic="/cmd_vel">
    linear: {x: -0.5}, angular: {z: 0.0}
  </output>
</plugin>
```

**Turning Right:** To turn the robot, we need to publish on angular z, positive and negative velocity will make it turn left and right respectively. Linear x velocity will be kept zero here as we will take a turn only. We can publish it using 'a' key (68 in ASCII decimal).

```
<!-- Turning Right -->
<plugin filename="libignition-gazebo-triggered-publisher-system.so"
  name="ignition::gazebo::systems::TriggeredPublisher">
  <input type="ignition.msgs.Int32" topic="/keyboard/keypress">
    <match field="data">68</match>
  </input>
  <output type="ignition.msgs.Twist" topic="/cmd_vel">
    linear: {x: 0.0}, angular: {z: -0.5}
  </output>
</plugin>
```

```
</output>
</plugin>
```

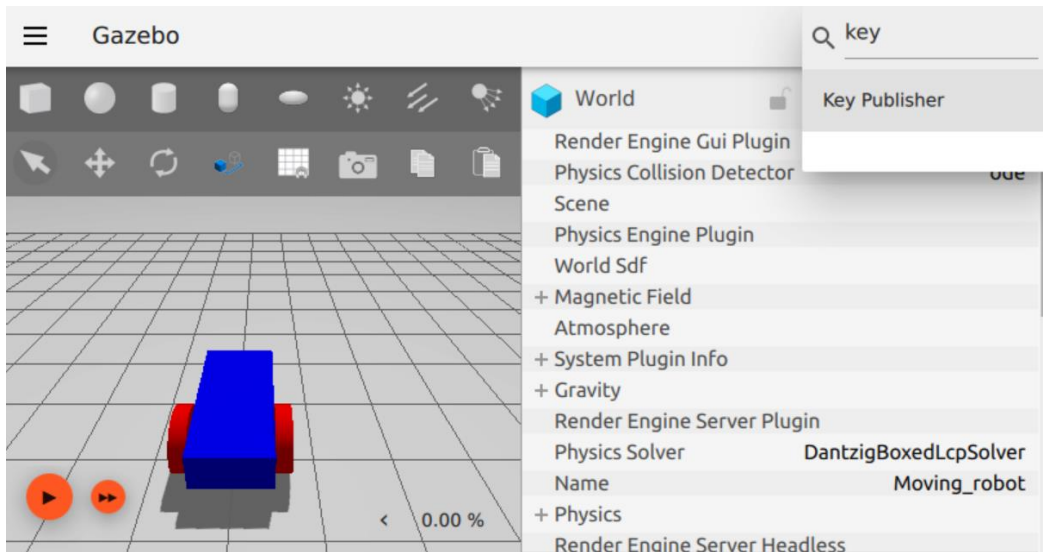
**Turning Left:** Here we will publish positive velocity on angular z. We can publish it using 'd' key (65 in ASCII decimal).

```
<!-- Turning Left -->
<plugin filename="libignition-gazebo-triggered-publisher-system.so"
  name="ignition::gazebo::systems::TriggeredPublisher">
  <input type="ignition.msgs.Int32" topic="/keyboard/keypress">
    <match field="data">65</match>
  </input>
  <output type="ignition.msgs.Twist" topic="/cmd_vel">
    linear: {x: 0.0}, angular: {z: 0.5}
  </output>
</plugin>
```

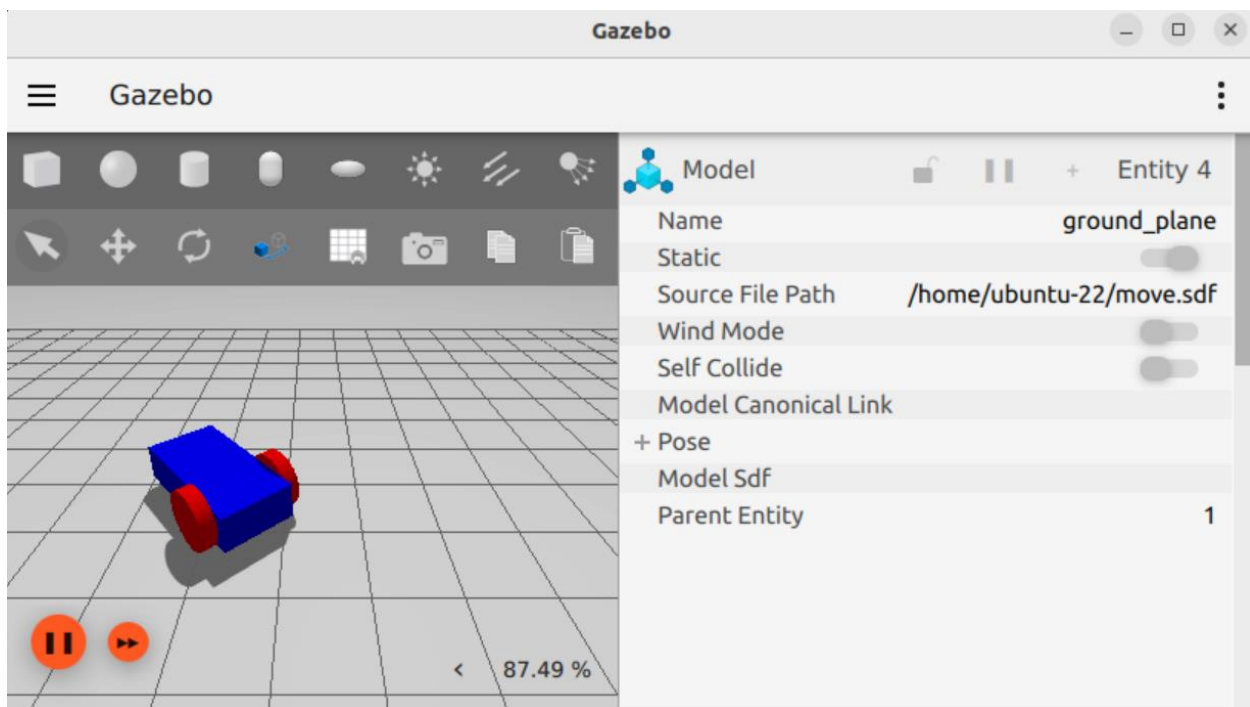
**Stopping Robot:** To stop the robot, we need to publish zero velocities on both components; linear x and angular z. We can publish it using 's' key (83 in ASCII decimal)

```
<!-- Stopping -->
<plugin filename="libignition-gazebo-triggered-publisher-system.so"
  name="ignition::gazebo::systems::TriggeredPublisher">
  <input type="ignition.msgs.Int32" topic="/keyboard/keypress">
    <match field="data">83</match>
  </input>
  <output type="ignition.msgs.Twist" topic="/cmd_vel">
    linear: {x: 0.0}, angular: {z: 0.0}
  </output>
</plugin>
```

**Setting Things Up:** To enable movement, we need to do 2 things; **add key publisher** as shown below in gazebo and **run the simulation using play button** on left down corner.



**Moving the robot:** Use 'w', 'x', 'a', and 'd' keys to in gazebo window to move robot in forward, backward, right and left direction respectively. Robot can be stopped by pressing 's' key or by stopping simulation.



**Ignition Gazebo Topics:** When we launch SDF file which has various plugins, it publishes and subscribes to its respective topics. They can be seen by executing the following command in the terminal.

```
ubuntu-22@ubuntu-22: ~  
ubuntu-22@ubuntu-22: ~  
ubuntu-22@ubuntu-22:~$ ign topic -l  
/clock  
/cmd_vel  
/gazebo/resource_paths  
/gui/camera/pose  
/keyboard/keypress  
/model/vehicle_blue/odometry  
/model/vehicle_blue/tf  
/stats  
/world/Moving_robot/clock  
/world/Moving_robot/dynamic_pose/info  
/world/Moving_robot/pose/info  
/world/Moving_robot/scene/deletion  
/world/Moving_robot/scene/info  
/world/Moving_robot/state  
/world/Moving_robot/stats  
ubuntu-22@ubuntu-22:~$
```

**Moving robot using script:** We can use scripting to communicate with Gazebo simulation for more fine control and customization. We will be using C++ for communication with Gazebo here.

**Moving Node:** First, let's have a look at the script (**move\_node.cc**) we will be using.

### Importing Necessary Libraries

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

### Initializing and advertising node:

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

### Creating a function for sending (publishing) velocities

```
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);  
}
```

Defining main function with some user information:

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

Taking continues and checking using switch case:

```
while (true) {
    int ch = getch(); // Get character without waiting
    if (ch != ERR) { // If a key was pressed
        switch (ch) {
            case 'w': // Move Forward
                sendCommand(1, 0.0);
                break;
            case 'x': // Move Backward
                sendCommand(-1, 0.0);
            case 'a': // Turn Left
                sendCommand(0.0, -1);
                break;
            case 'd': // Turn Right
                sendCommand(0.0, 1);
                break;
            case 's': // Stop
                sendCommand(0.0, 0.0);
                break;
            case 'q': // Quit
                endwin(); // End curses mode
                return 0;
            default:
                break;
        }
    }
}
```

**Defining CMakeList:** Save below script in CMakeList.txt to build our project, linking dependencies, and the source code.

```
cmake_minimum_required(VERSION 3.10.2 FATAL_ERROR)

project(move_node)

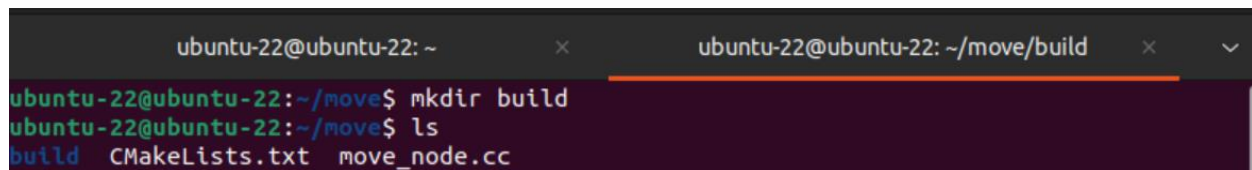
# Find the Ignition_Transport library
find_package(ignition-transport11 QUIET REQUIRED OPTIONAL_COMPONENTS log)
set(IGN_TRANSPORT_VER ${ignition-transport11_VERSION_MAJOR})

# 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()
```

**Setting up workspace:** Create a folder for your respective node and place these two files in that folder. Also make a folder named **build** in that folder as shown below.



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

**Installing curses package:** We need to install the curses package to build this node as it requires this library for the user input. We can install it using the following command.

```
sudo apt install libcurses*
```

**Note:** Your workspace directory should look like this.

```
move/
├── build/           # Compiled build files (typically created by CMake)
├── CMakeLists.txt   # CMake build configuration file
└── move_node.cc     # C++ source file (your main ROS/robotics node)
```



**Running CMake in build folder:** If everything worked fine you should see the following output.

```
ubuntu-22@ubuntu-22: ~  
ubuntu-22@ubuntu-22: ~/move/build  
ubuntu-22@ubuntu-22:~/move$ cd build/  
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 version "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$
```

**Making Node:** Now make the node (create node executable) using make command with its respective script name without its file extension as shown below. Executable is generated and can be seen in the green color inside the build folder.

```
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$ ls  
CMakeCache.txt  CMakeFiles  cmake_install.cmake  Makefile  move_node
```

**Running the executable:** Execute the following command to execute the move node and use the respective keys to control the robot movement as shown below.

```
ubuntu-22@ubuntu-22: ~  
ubuntu-22@ubuntu-22: ~/move/build  
ubuntu-22@ubuntu-22:~/move/build$ ./move_node  
Control the robot:  
w - Move Forward  
a - Turn Left  
d - Turn Right  
x - Move Backward  
s - Stop  
q - Quit
```