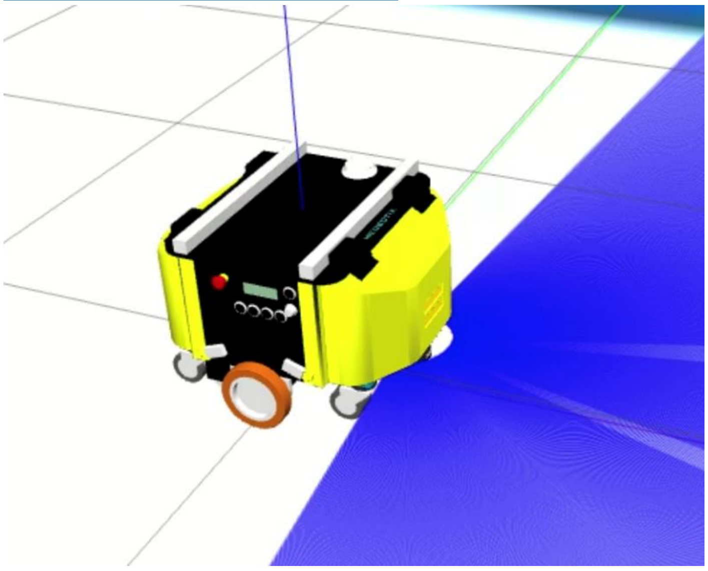
ROS 2 Basic Concepts

Estimated time to completion: **6 minutes**

2.2 Move a Robot with ROS2

On the right corner of the screen, you have your first simulated robot: the Neobotix MP-400. **Move that robot now!**
[Official Web: Neobotix MP-400]



How do you move the MP-400?

The easiest method is to control the robot using an existing ROS2 program. The executables created during the compilation of a ROS2 program are used to run it. Later in this guide, you will learn more about compilation.

Since it already exists in this workspace, you will launch a previously-made ROS2 program (executable) that allows you to move the robot using the keyboard.

- Example 2.1 -

So, get on with it. Execute the following command in **Web Shell #1**:

Execute in Shell #1

In [ ]:

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

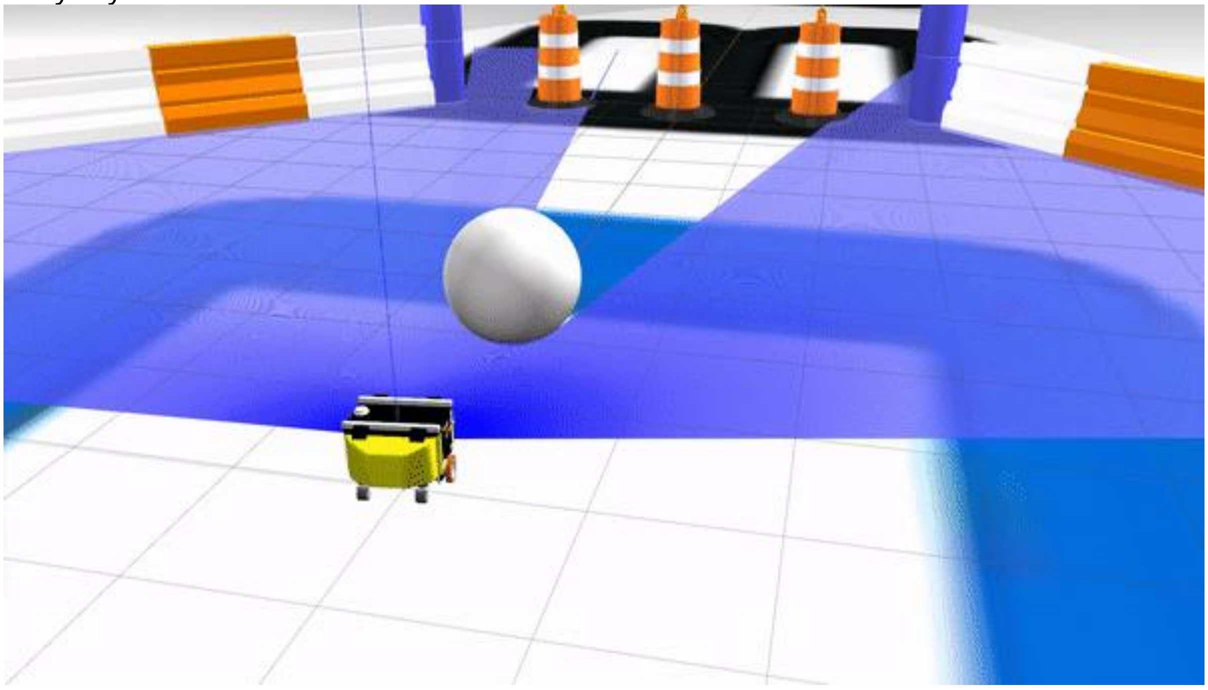
CONGRATULATIONS! You launched your first ROS2 program! In this terminal, you should have received a message similar to the following:

Shell #1 Output

```
This node takes keypresses from the keyboard and publishes them

as Twist messages. It works best with a US keyboard layout.

---------------------------

Moving around:

   u    i    o

   j    k    l

   m    ,    .
```

```
For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit
```

Now, you can use the keys indicated in the web shell output to move the robot around.
Okay. Try to move the robot around now!



REMEMBER to focus on the terminal (shell) where you launched the program for the keys to take effect. You know you have correctly focused when the cursor starts blinking.
When you are tired of playing with the robot, press **Ctrl+c** to stop the program's execution (remember to have the focus). If you want, you can close Web Shell #1 and Web Shell #2 to continue the tutorial.
Look back at what you have learned so far. The **ros2** keyword is used for the ROS2 commands. Therefore, to launch programs, you have two options:

- Launch the ROS2 program by directly running the **executable file**.

- Launch the ROS2 program by starting a **launch file**.

It may seem easier to use the run command to launch executables, but you will later understand why the launch command is also useful.

For now, you can directly run the executable file. The structure of the command is as follows:

In [ ]:

```
ros2 run <package_name> <executable_file>
```

The command has two parameters: the first parameter is **the name of the package** that contains the executable file. The second parameter is **the name of the executable file** (which is stored inside the package). To use a launch file, the structure of the command goes as follows:

In [ ]:

```
ros2 launch <package_name> <launch_file>
```

This command also has two parameters: the first parameter is **the name of the package** that contains the launch file. The second parameter is **the name of the launch file** (stored in the package).

- End of Example 2.1 -

ROS 2 Basic Concepts

Estimated time to completion: **2 minutes**

2.3 What is a Package?

ROS2 uses **packages** to organize its programs. Think of a package as **all the files that a specific ROS program contains**; all its CPP files, Python files, configuration files, compilation files, launch files, and parameter files. Also, organizing your ROS2 programs in packages makes sharing them with other developers/users much easier.

In ROS2, you can create two types of packages: CMake (C++) packages and Python packages. For this course, though, you will focus on the first ones.

Every CMake package will have the following structure of files and folders:

- `launch` folder: Contains launch files
- `src` folder: Contains source files (CPP, Python)
- `CMakeLists.txt`: List of Cmake rules for compilation
- `package.xml`: Package metadata and dependencies

They are essential, so remember the following:

- Every ROS2 program that you want to execute is organized in a package.
- Every ROS2 program that you create must be organized in a package.
- Packages are the primary organization system for ROS2 programs.