INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES
DE MONTERREY
CAMPUS GUADALAJARA

Curso de ROS
Apuntes de clase

Impartido por:
Dr. Eduardo de Jesús Dávila Meza

para el bloque integrador de:
Robótica y Sistemas Inteligentes

para estudiantes de:
Ingeniería en Robótica y Sistemas Digitales

de la Escuela de:
Ingeniería y Ciencias

Tecnológico de Monterrey, Campus Guadalajara. Zapopan, Jalisco. Marzo 2025.

Contents

CHAPTER 1

ROS 2 Environment Setup and Development Tools

Author: Dr. Eduardo de Jesús Dávila Meza.
EduardoDavila-AI-PhD

1.1 Introduction

1.1.1 Basic Concepts

The Robot Operating System (ROS) is not an actual operating system but a flexible *framework* for writing robot software. It provides a collection of software libraries, tools, and conventions to simplify the task of creating complex and robust robot applications. From drivers and state-of-the-art algorithms to powerful developer tools, ROS has the open source tools you need for your next robotics project. Some of its main advantages include:

- **Multi-language:** While ROS supports multiple programming languages, the primary supported languages are C++ and Python. Support for other languages like Java exists but is less common.
- **Free and open-source:** ROS 1 and ROS 2 are available on Ubuntu, with ROS 2 also supporting Windows and macOS. However, the availability and stability can vary across different versions and platforms.
- **Support:** Since ROS was started in 2007, a lot has changed in the robotics and ROS community. The ROS 2 project began in 2015 to address the evolving needs of the robotics community, building upon the strengths of ROS 1 and introducing improvements for better performance, security, and support for real-time systems.

1.1.2 Some ROS Distributions

ROS is released as distributions (or "distros"), with several versions supported concurrently. Some releases come with long-term support (LTS), meaning they are more stable and have undergone extensive testing, while other distributions are newer, have shorter lifetimes, and support more recent platforms and package versions. Generally, a new ROS distro is released every year on World Turtle Day, with LTS releases appearing in even-numbered years. ROS is available for different versions of Ubuntu and other operating systems. Some of the notable distributions include:

- **Indigo Igloo (ROS 1):** Ubuntu 14.04 (Trusty Tahr)
- **Kinetic Kame (ROS 1):** Ubuntu 16.04 (Xenial Xerus)
- **Melodic Morenia (ROS 1):** Ubuntu 18.04 (Bionic Beaver)
- **Noetic Ninjemys (ROS 1):** Ubuntu 20.04 (Focal Fossa)
- **Foxy Fitzroy (ROS 2):** Ubuntu 20.04 (Focal Fossa)
- **Humble Hawksbill (ROS 2):** Ubuntu 22.04 (Jammy Jellyfish)
- **Jazzy Jalisco (ROS 2):** Ubuntu 24.04 (Noble Numbat)

For more details, see the ROS Distributions list. Currently, the Noetic Ninjemys, Humble Hawksbill, and Jazzy Jalisco distributions are actively supported.

1.1.3 ROS Distribution for the Course

For this course, we will use **ROS 2 Humble Hawksbill**, a long-term support (LTS) release that offers enhanced performance, security, and real-time capabilities. Its logo, shown in Figure 1.1, reflects its distinctive branding. ROS 2 Humble Hawksbill is compatible with Ubuntu 22.04 (Jammy Jellyfish) and Windows 10, making it a versatile choice for both simulation and deployment on physical hardware. This distribution will serve as the foundation for all class exercises and, in particular, for project development.

Figure 1.1: ROS 2 Humble Hawksbill Logo.

1.2 Installation of Ubuntu 22 and ROS 2 Humble Hawksbill

1.2.1 Installing Ubuntu 22.04.5 LTS (Jammy Jellyfish)

1. **Download the Universal USB Installer**
   Visit the official Universal USB Installer website to download the tool. Follow the instructions provided on the website or refer to a YouTube tutorial for creating a bootable USB.
2. **Download the Ubuntu ISO**
   Download the Ubuntu 22.04.5 LTS (Jammy Jellyfish) ISO from the official Ubuntu releases page.
3. **Create a Bootable USB Drive**
   Use the Universal USB Installer tool with the downloaded ISO to create a bootable USB stick. Follow the on-screen prompts to properly set up the drive.
4. **Install Ubuntu on Your Machine**
   Boot your computer from the USB drive. Follow the Ubuntu installation wizard to install Ubuntu 22.04.5 LTS and configure your system settings as needed.

1.2.2 Installing ROS 2 Humble Hawksbill

Recommended Method: Using Debian Packages

The official ROS 2 Humble Hawksbill documentation recommends installing from deb packages for a stable and straightforward setup. Follow the installation instructions on the official ROS 2 Humble Hawksbill Installation Guide. This method installs pre-built binaries and generally covers all core dependencies needed for running ROS 2.

Instructions:

Set locale
Make sure you have a locale which supports UTF-8. The following settings are tested, although any UTF-8 supported locale should work.

```
$ locale  # check for UTF-8

$ sudo apt update && sudo apt install locales
$ sudo locale-gen en_US en_US.UTF-8
$ sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
$ export LANG=en_US.UTF-8

$ locale  # verify settings
```

Setup Sources
You will need to add the ROS 2 apt repository to your system. First, ensure that the Ubuntu Universe repository is enabled.

```
$ sudo apt install software-properties-common
$ sudo add-apt-repository universe
```

Now, add the ROS 2 GPG key with apt:

```
$ sudo apt update && sudo apt install curl -y
$ sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o
↪  /usr/share/keyrings/ros-archive-keyring.gpg
```

Then, add the repository to your sources list:

```
$ echo "deb [arch=$(dpkg --print-architecture)
↪  signed-by=/usr/share/keyrings/ros-archive-keyring.gpg]
↪  http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME)
↪  main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
```

Install ROS 2 Packages

Update your apt repository caches after setting up the repositories:

```
$ sudo apt update
```

ROS 2 packages are built on frequently updated Ubuntu systems. It is always recommended to ensure your system is up to date before installing new packages:

```
$ sudo apt upgrade
```

⚠ Warning:

Due to early updates in Ubuntu 22.04, it is important that `systemd` and `udev`-related packages are updated before installing ROS 2. Installing ROS 2's dependencies on a freshly installed system without upgrading can trigger the removal of critical system packages. Please refer to ros2/ros2#1272 and Launchpad #1974196 for more information.

Desktop Install (Recommended): This includes ROS, RViz, demos, and tutorials.

```
$ sudo apt install ros-humble-desktop
```

Development Tools: Compilers and other tools to build ROS packages.

```
$ sudo apt install ros-dev-tools
```

Environment Setup – Sourcing the Setup Script

Set up your environment by sourcing the following file (replace `.bash` with your shell if not using bash):

```
$ source /opt/ros/humble/setup.bash
```

Alternative: Building from Source

If you require customizations or intend to develop complex packages, you can also install ROS 2 Humble Hawksbill from source. Note that building from source may require additional dependency installations (similar to ROS1). For most beginners and for the purposes of this course, the deb package installation is recommended.

Sourcing the Setup Script

After installing ROS 2, source the setup script to ensure your environment is correctly configured:

```
$ source /opt/ros/humble/setup.bash
```

Optionally, add the sourcing command to your shell's initialization file (e.g., `.bashrc`) so that the ROS environment variables are automatically loaded every time a new terminal session is opened (replace `.bash` with your shell if not using bash).

```
$ echo "source /opt/ros/humble/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

1.3 Try Some Examples

1.3.1 Talker-Listener

If you installed `ros-humble-desktop`, try some examples. In one terminal, source the setup file and run a C++ talker:

```
$ source /opt/ros/humble/setup.bash
$ ros2 run demo_nodes_cpp talker
```

In another terminal, source the setup file and run a Python listener:

```
$ source /opt/ros/humble/setup.bash
$ ros2 run demo_nodes_py listener
```

You should see the talker publishing messages and the listener confirming reception, verifying that both the C++ and Python APIs are working properly.

1.3.2 Turtlesim: Publish and Move a Turtle

Another example you can try is the `turtlesim` package, which demonstrates basic ROS 2 communication using a simulated turtle.

First, ensure that the package is installed:

```
$ sudo apt install ros-humble-turtlesim
```

Then, in a new terminal, launch the turtlesim node:

```
$ ros2 run turtlesim turtlesim_node
```

Next, open another terminal, use the `teleop` node to control the turtle with your keyboard:

```
$ ros2 run turtlesim turtle_teleop_key
```

Now, pressing the arrow keys will move the turtle in the corresponding direction. This example demonstrates publishing velocity commands to a topic that the turtlesim node subscribes to, allowing interaction with the simulated environment.

1.4 Configuring the Development Environment

1.4.1 Installing Visual Studio Code

Visual Studio Code (VS Code) is a versatile code editor that supports various programming languages and development environments. To install it on Ubuntu 22.04, follow these steps:

1. **Using the Ubuntu Software Center**
   (a) Open the *Ubuntu Software* application from the applications menu.
   (b) In the search bar, type *Visual Studio Code*.
   (c) Locate *Visual Studio Code* in the search results and click *Install*.
2. **Using the Official .deb Package**
   (a) Visit the official VS Code download page: https://code.visualstudio.com/.
   (b) Click on the *.deb* package suitable for Debian/Ubuntu.
   (c) Once downloaded, open a terminal and navigate to the directory containing the downloaded file.
   (d) Install the package using:

   ```
   $ sudo apt install ./<file>.deb
   ```

   Replace <file> with the actual filename. This method ensures that VS Code is added to your system repositories and receives updates automatically.

1.4.2 Installing Recommended VS Code Extensions

Enhance your development experience by installing the following VS Code extensions:

- **Python** - Provides rich support for Python, including features such as IntelliSense, linting, and debugging.
- **C++** - Offers C++ IntelliSense, debugging, and code browsing.
- **CMake** - Simplifies working with CMake projects.
- **CMake Tools** - Provides CMake project integration.
- **ROS** - Adds support for Robot Operating System (ROS) development.
- **ROS2** - Facilitates ROS 2 development with code snippets and other utilities.
- **XML** - Enhances XML editing capabilities.
- **XML Tools** - Offers additional functionalities for XML files.
- **Indent-Rainbow** - Highlights indentation levels with different colors.
- **vscode-icons** - Adds file icons for better visual identification.
- **Error Lens** - Displays inline error messages in the code editor.

To install these extensions:

1. Open VS Code.
2. Navigate to the *Extensions* view by clicking on the square icon in the sidebar or pressing Ctrl+Shift+X.
3. Search for each extension by name and click *Install*.

1.4.3 Installing Terminator for Multiple Terminals

Terminator is a terminal emulator that allows splitting the window into multiple terminals, facilitating simultaneous operations. To install Terminator:

1. Open a terminal.
2. Update the package list:
   ```
   $ sudo apt update
   ```
3. Install Terminator:
   ```
   $ sudo apt install terminator
   ```
4. Launch Terminator from the applications menu, by typing `terminator` in the terminal, or by pressing `Ctrl+Alt+T`.

With these tools and extensions, your development environment will be well-equipped for ROS 2 projects.

1.5 `colcon` Configuration

`colcon` is the command-line tool used in ROS 2 to build sets of packages in a workspace. It automatically detects packages (via `package.xml`), resolves dependencies, and builds them (often in parallel) to simplify the development process. `colcon` also supports options like `-symlink-install` for faster iterative development.

1.5.1 Installing `colcon`

Update your package list and ensure that the common `colcon` extensions are installed. Although these packages are usually installed as part of the `ros-humble-desktop` and `ros-dev-tools` installations, it is good practice to verify their presence by running the following commands:

```
$ sudo apt update
$ sudo apt install python3-colcon-common-extensions
```

1.5.2 Enabling `colcon` Argument Completion

To simplify command usage with `colcon`, add the argument completion script to your shell initialization file (e.g., `.bashrc`). Execute the following commands:

```
$ echo "source /usr/share/colcon_argcomplete/hook/colcon-argcomplete.bash" >> ~/.bashrc
$ source ~/.bashrc
```

You can verify that the sourcing command was added by viewing your `.bashrc` file:

```
$ cat ~/.bashrc
```

1.6 Practice Assignment

In this assignment, you will run the ROS 2 examples presented in Section 1.3 on your computer and capture evidence of successful execution using a screenshot. Submit your evidence in the designated assignment area on Canvas (within the ROS 2 module) as a PNG file. The filename must follow this format:

`FirstnameLastname_evidence_wXsY.png`,

where `wX` and `sY` indicate the week and session numbers respectively, for example:

`EduardoDavila_evidence_w1s1.png`.

1.6.1 Examples to Try

Talker-Listener

After installing `ros-humble-desktop` (and optionally `Terminator`), try the following:

- In Terminal 1, source the setup file and run a C++ talker:

```
$ source /opt/ros/humble/setup.bash
$ ros2 run demo_nodes_cpp talker
```

- In Terminal 2, source the setup file and run a Python listener:

```
$ source /opt/ros/humble/setup.bash
$ ros2 run demo_nodes_py listener
```

You should observe that the talker publishes messages and the listener confirms their reception.

Turtlesim: Publish and Move a Turtle

Another example is provided by the `turtlesim` package:

- In Terminal 3, launch the turtlesim node:

```
$ ros2 run turtlesim turtlesim_node
```

- In Terminal 4, run the teleoperation node to control the turtle:

```
$ ros2 run turtlesim turtle_teleop_key
```

Use the arrow keys to move the turtle. This example demonstrates publishing velocity commands to control the simulated turtle.

1.6.2 Submission Instructions

1. Run the examples as described above.
2. Capture a **screenshot showing the terminal output** where the examples are running successfully (see Figure 1.2 for an example).
3. Save the screenshot in PNG format.

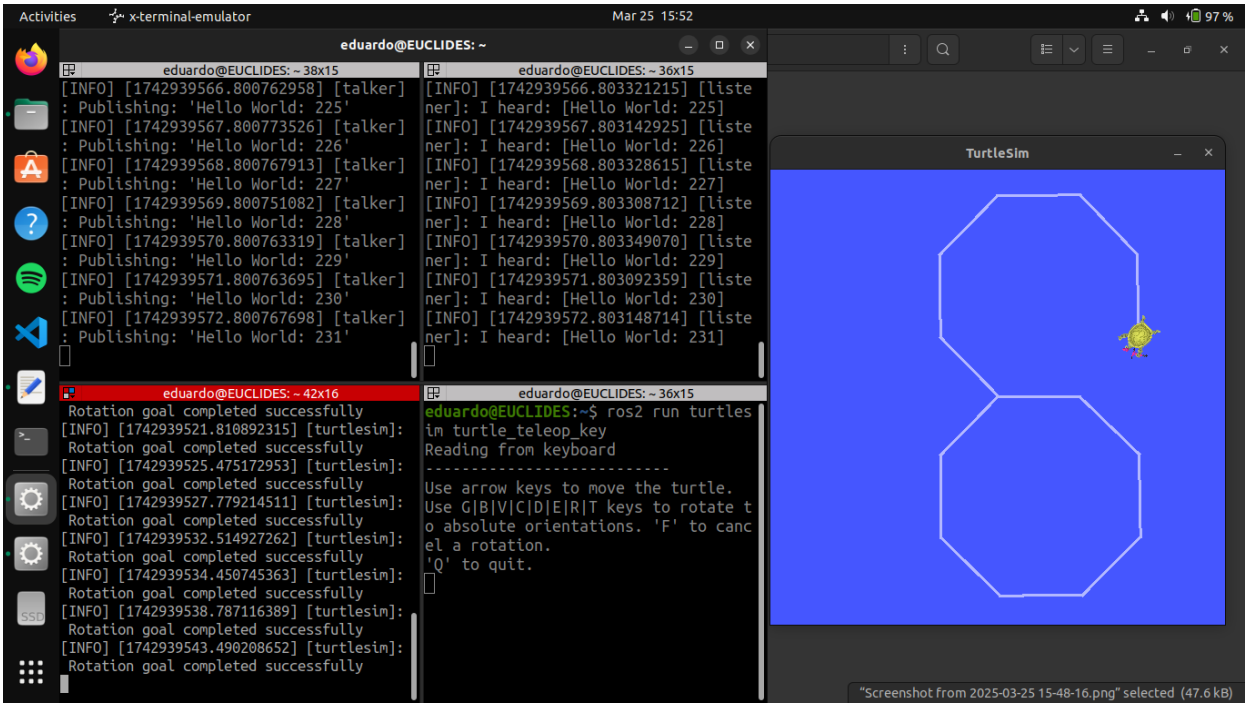

Figure 1.2: Example screenshot showing terminal outputs from both the Talker-Listener example and the Turtlesim node example in action.

4. Name the file according to the format: `FirstnameLastname_evidence_wXsY.png` (replace `X` with the week number and `Y` with the session number).
5. Submit your screenshot file as instructed by the course guidelines.

Note: Your machine's username and device name must be visible in the screenshot to verify the authenticity of the submission, as shown in Figure 1.2. Evidence that appears copied, unclear, or altered will be considered invalid and may result in a score of 0 for this assignment.

CHAPTER 2

ROS 2 Workspace, Package, and Node Development