METR4202

Robotics & Automation

Week 1: Practical - Intro to ROS

In Summary...

- ROS stands for **R**obot **O**perating **S**ystem
- It is a middleware for communicating between processes that independently perform tasks for your robot stack (e.g., sensing, decision-making, acting)
- As robot developers you will use ROS as a library inside your Python or C++ code
- It is recommended to get comfortable with BASH (terminal) to run ROS commands during development (there is a cheat sheet in the METR4202 repository for your convenience)

Using the ROS Command-line

Prepare your Virtual Machine

Exercise 1 (Together)

Create a publisher and echo the result in separate terminals.

Terminal 1

```
rostopic pub <topic> <msg> <data>
```

Terminal 2

```
rostopic echo <topic>
```

Exercise 2 (Your turn)

Run the following command and try to publish to the node in a separate terminal. There are several ways to do this.

Terminal 1

```
rosrun rospy_tutorials listener.py
```

Terminal 2

```
# Hint: Find out the topic name and message type the node subscribes to
```

Exercise 3 (Challenge)

Run the following command. This will open a simulation window of a turtle. In a separate terminal, publish to the appropriate topic to spin the turtle at 50 rad/s.

Terminal 1

```
rosrun turtlesim turtlesim_node
```

Terminal 2

```
# Hint: When entering the message data, you may use this format instead
rostopic pub /topic example_msgs/Example -- '[a, b, c]' '[...]'

# Each array will correspond to a component of the message type
# For example,
rostopic pub /topic geometry_msgs/Pose -- '[1, 2, 3]' '[4, 5, 6]'
# Where [1, 2, 3] are the positions x, y, z
# And [4, 5, 6] are the orientations x, y, z
# See `rosmsg show geometry_msgs/Pose`
```

Next Week

- Learn how to create your own ROS package
- Write simple ROS nodes (executables)
- Launch multiple nodes with launch files
- Learn about more advanced features of ROS
  - Parameters
  - Services
  - Actions