

Project 1: Learning Robot Operating System (ROS)

Assigned: February 08, 2024

Deliverable 1 (Task 1) Due Date: February 14, 23:59:59

Deliverable 2 (Task 2) Due Date: February 23, 23:59:59

Note: Project 1 has a four-day late submission period for each deliverable.
Late submissions will lose 25% points per day of that specific deliverable.

In this project, the goal is to setup, learn, and test ROS, as well as write a ROS “Hello World” program. This project provides an opportunity to understand open-source packages for robotics development and become ready for future course projects.

Project 1 must be done individually, although group discussion is encouraged.

1 Task 1: Setup and Test Development Environment

1.1 Task 1.1: Setup ROS in Ubuntu

Task 1.1 focuses on setting up ROS Noetic as the development environment in Ubuntu 20.04 LTS, and learning ROS through its official tutorial.

Step 1: COMPSCI-603 course projects are required to be programmed using ROS Noetic in Ubuntu 20.04 LTS. You may follow the “Ubuntu 20.04 & ROS Noetic Installation Guide”: [ROS_installation_guide.pdf](#) (provided on the course website) to install the OS and setup the development environment.

Step 2: Go through ROS tutorial “Beginner Level”:

- <http://wiki.ros.org/ROS/Tutorials>

Then, use Turtlesim to understand how ROS works. Make sure you follow the Turtlesim tutorial under ROS Noetic:

- <http://wiki.ros.org/turtlesim>

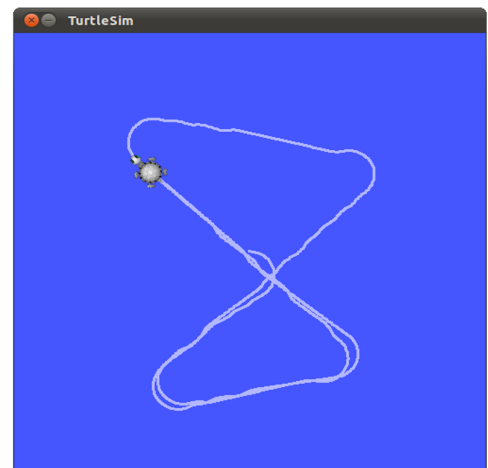


Figure 1: Screenshot of a working Turtlesim.

1.2 Task 1.2: Understand the Gazebo Simulator

Step 1: Go through Gazebo beginner tutorial 1-3 “Overview and Installation, Understanding the GUI, and Model Editor” to get familiar with Gazebo:

- https://classic.gazebosim.org/tutorials?cat=guided_b&tut=guided_b2

Step 2: First, install ROS packages for simulating TurtleBot3 in Gazebo, mapping, and planner:

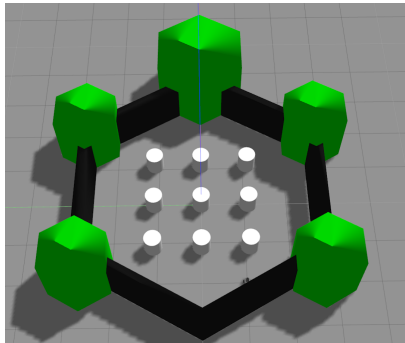
```
sudo apt-get install ros-noetic-turtlebot3*  
sudo apt-get install ros-noetic-slam-gmapping  
sudo apt-get install ros-noetic-dwa-local-planner
```

Then, go through TurtleBot3 Simulation tutorial from 6.1.1 “Install Simulation Package” to 6.3.4 “Set Navigation Goal”

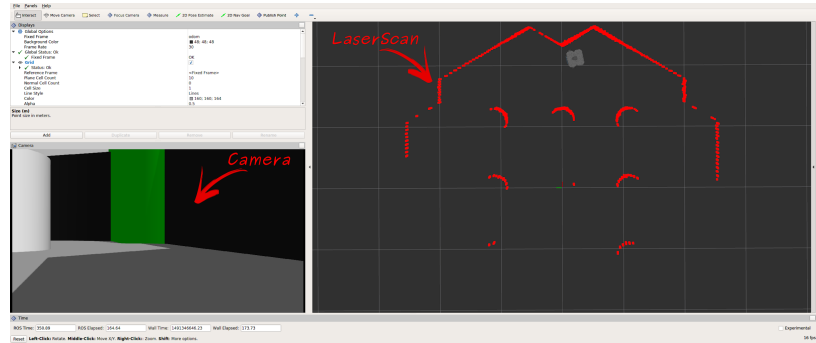
- <https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation>

Choose `waffle_pi` as `TURTLEBOT3_MODEL` because it is equipped with both a camera and a 360° LiDAR. Make sure to select `Noetic` at the top of the page.

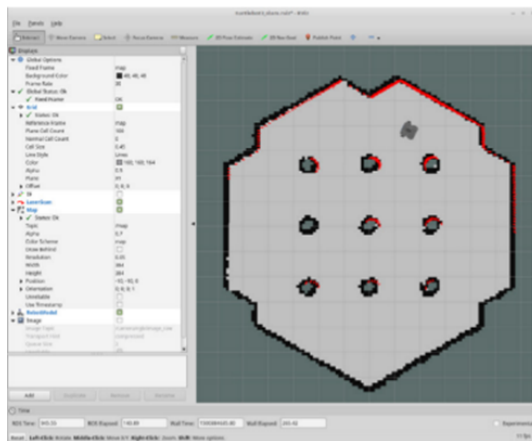
To complete this task, you will need to (1) load a pre-defined simulated world, (2) execute RViz to visualize the robot sensing, (3) build a map of your simulated world, and (4) use RViz to navigate through the world.



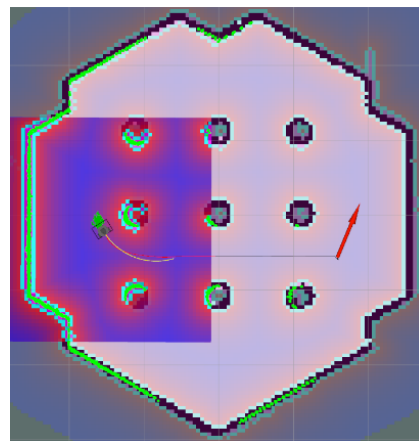
(a) Simulation World



(b) Visualized camera and LiDAR sensing



(c) Mapping result



(d) Robot navigation

Figure 2: Examples of the required files that must be submitted to deliverable 2.

1.3 What to Submit

From Task 1.1: You are required to submit a screenshot of a working turtlesim window for grading, similar to Figure 1 that shows a turtle's trajectory.

From Task 1.2: You are required to submit snapshots of (Examples are shown in Figure 2):

1. screenshot of the simulation world
2. visualization of the simulated robot sensing
3. visualization of your mapping result
4. visualization of navigation to goal behavior

Include all files from both Task 1.1 and Task 1.2 in a single tar or zip file named `P1D1_firstname_lastname.tar` (or zip) and submit to the Canvas portal named `P1-D1`.

2 Task 2: Write a ROS “Hello World”

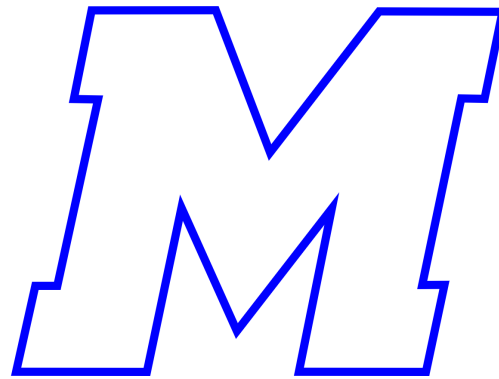
Before starting this task, make sure you understand ROS Tutorial 1.1.12 and 1.1.13. Python 3 is required to complete this task. Although C++ is a working solution (that we also discussed in the lecture to show ROS programming components), most 3rd-party ROS packages are implemented using Python, and documentation of ROS packages in C++ is also relatively sparse (e.g., for Gazebo).

Write a ROS node that uses the turtlesim simulator to draw the “M” shape within the UMass Amherst logo, as shown by the blue curve in Figure 3. Your code must implement a close-loop control, which must include a subscriber to receive turtle’s pose data and use it to improve navigation accuracy. In addition:

1. You must create a new package called P1D2_firstname_lastname (e.g., P1D2_zhang_hao). Your package should contain an executable with the same name as the package.
2. Your program should assume that roscore and turtlesim_node have been started independently.
3. The drawing should not take longer than two minutes to complete.



(a) UMass logo



(b) The “M” shape that must be drawn by the turtle

Figure 3: Logo of UMass Amherst. The **blue curve** is the trajectory the turtle needs to draw.

What to submit: You are required to submit (1) a complete ROS package (e.g., not a Python script only), (2) a README to detail how to compile and run your ROS package (and other information), and (3) a snapshot of the result. All the above three items must be included in a tar or zip file named P1D2_firstname_lastname.tar (or zip) and submit to the Canvas portal named P1-D2.

3 Project Submission and Grading

Project 1 has two deliverables, and each deliverable has its own due dates. Each deliverable has a four-day late submission period. Late submissions will lose 25% points per day of that specific deliverable. Project deliverables must be submitted through Canvas, although you are encouraged to use version control systems (GitHub or GitLab) to manage your code.

Project 1 will be graded as follows (for a total of 10 points):

- Deliverable 1: 5/10
 - Task 1.1: 2 points
 - Task 1.2: 3 points
- Deliverable 2: 5/10.