

---

# CSCE574 – Robotics

## Spring 2012 – Project 2

---

**Assigned:** February 9

**Due:** Demo completed by March 2, 5pm

The purpose of this assignment is to give you some familiarity with the TurtleBot robot platforms that we'll use in the second half of the class by interacting with the simulation of those robots provided by ROS.

*You should do this assignment in the groups you've been assigned.*

## Getting ready

On the website for this course, you will find a document titled "Notes on Simulating the TurtleBot," that explains how to initiate and interact with the simulation. This document also provides links to the relevant parts of the ROS documentation. Everyone should learn the material there, and preferably try each of the features described.

## Tasks

Create ROS package named `TBsim_x`, in which *x* is the single-letter name of your team. Within this package, create a launch file named `tbsim.launch` that...

- ...uses `sim` time for all of its nodes.
- ...starts a simulated Turtlebot in a world that you create.

**Details:** The course website has an example world file, along with a image used as a map by that world file. Your project should contain these files in the correct places. In addition, you should create a new map image in a shape of your choosing, and create a new world file (which will be very similar to the given one) to use that new map instead.

- ...starts an instance of RViz.

**Details:** Create a `vcg` file that includes displays for a grid, the robot model, a laser scan (`/scan`), a map (see below), and any other displays you like. Store this file inside your package. Use the `-d` argument in your launch file to load this configuration file when RViz starts up.

- ...starts a node named `rotate` that commands the robot to rotate in place.

**Details:** Create a new C++ program that publishes the correct messages.

- ...starts a node named `map` that uses the fake laser scans published on the topic `scan` to construct a **map** of the part of the environment seen as the robot moves.

**Details:** Create a new C++ program to construct a map of type `nav_msgs/OccupancyGrid`<sup>1</sup> at publish it on `/map` every second or so. You should assume that the odometry information is correct. That is, you should assume that the robot's pose in frame `/odom_combined` is

---

<sup>1</sup>[http://www.ros.org/doc/api/nav\\_msgs/html/msg/OccupancyGrid.html](http://www.ros.org/doc/api/nav_msgs/html/msg/OccupancyGrid.html)

---

always fully accurate.<sup>2</sup> Your program still work correctly if I teleoperate the robot instead of using your rotate node. (Do *not* assume that the robot will only rotate at the origin.) Your map's `frame_id` should be `/odom_combined`. Its `resolution` should be read from a parameter called `map_resolution`, which should be set to 0.1 in your launch file.

## What to Submit

There are three “deliverables”:

- **Hardcopy:** At least 24 hours before your demo, turn in:
  1. The cover sheet from the final page of this document.
  2. A short typewritten report (1-2 pages of complete sentences) describing your results. The report should contain two sections of roughly equal length.
    - (a) **Description:** How does your program work? When does it fail? Describe your method carefully.
    - (b) **Reflection:** What design decisions did you make? What difficulties or surprises did you encounter? What did you learn?
  3. A staple fastening your cover sheet to your report.
- **Dropbox:** At least 24 hours before your demo, submit a zip file or gzipped tar archive of your entire package directory to the CSE dropbox. Each team should submit in only one dropbox.
- **Demonstration:** Demonstrate the completed project to the instructor outside of class. It is important that everyone in the group be present for this demo, and for everyone to understand the programs you submit. This demonstration, though informal, will contribute significantly to the grading of the project.

---

<sup>2</sup>This assumption is certainly false, and as a result, the accuracy of your map update is likely to decrease as the robot continues to move. However, without this assumption, the assignment would be much, much, more difficult.

# CSCE574 – Project 2 Cover Sheet – Team \_\_\_\_\_

---

## General (10):

- Students are conversant in ROS concepts and operation from notes.
- Package contains required files.
- Launch file satisfies requirements.

## Gazebo and RViz (15):

- Turtlebot spawns in Gazebo.
- New map created in a usable format.
- World file loads map correctly.
- RViz loads custom configuration file.

## Rotate node (5):

- Rotation commands are published correctly.
- Node uses sim time correctly.

## Mapping node (60):

- Node uses scan data to determine obstacle positions in Kinect frame.
- Node correctly transforms obstacle positions into map's frame.
- Map correctly identifies observed obstacles.
- Map correctly identifies observed free space.
- Map correctly identifies cells that have not been observed.
- Map is published rapidly, near 1Hz or faster.
- Node publishes a well-formed map.

## Style and Documentation (10):

- Submitted file contains a complete well-formed ROS package.
- All files in correct locations.
- Report exists, has reasonable length, and is mostly typo-free.
- Report summarizes programs' operation succinctly and accurately.
- Report contains non-trivial reflections on the process of completing the tasks.

## Other comments:

## Total:

---