Assignment 1

The objective of this assignment is for you to become familiar with the Robot Operating System (ROS) software, which you will be using in future assignments to program various algorithms for robots.

1. Installing and learning ROS

ROS is officially supported on Ubuntu Linux (www.ubuntu.com), so if you would like to install ROS on your personal computer, we highly recommend that you install it alongside the version of Ubuntu 20.04.5 LTS (Focal Fossa). The following assume that you are installing ROS on Ubuntu.

Please follow these instructions to install ROS; you will want to install the ROS Noetic Ninjemys (ros noetic-desktop-full version).

The Ubuntu Linux computers in Swearinger should all have ROS noetic installed already, so you may choose to use those machines to solve your assignments. Please notify us if you need a particular ROS package that is not installed on these machines. If you plan to use these computers for your assignments, you may skip to part 2.
Please follow the following instructions to install ROS:

http://wiki.ros.org/noetic/Installation/Ubuntu

If you have sufficient hard drive space, ideally you should install ros-noetic-desktop-full version.

The previous ROS installation page is in fact the first item in a comprehensive tutorial on learning ROS. Please follow through all of the **Beginner level** tutorials on the http://www.ros.org/wiki/ROS/Tutorials page. You may skip items 12 and 15, as we will not be using Python in this course. Read through the Intermediate Level tutorials 1 till 5.

2. Installing and learning ROS stage package

In the first few assignments we will be working with a simulated robot equipped with a horizontal sweeping laser scanner. Therefore, in addition to installing the core ROS packages, you will need to install the http://www.ros.org/wiki/stage#http://www.ros.org/wiki/stage stage robot simulator. Depending on what version of ROS you've installed previously, you may already have this stage package already. To find out, type the following in a terminal:

```
> rosmake stage
```

Please, go through the stage tutorials:
http://wiki.ros.org/stage/Tutorials

If you have stage installed, it should report something along the lines of:

```
> [ rosmake ] Built 37 packages with 0 failures.
```

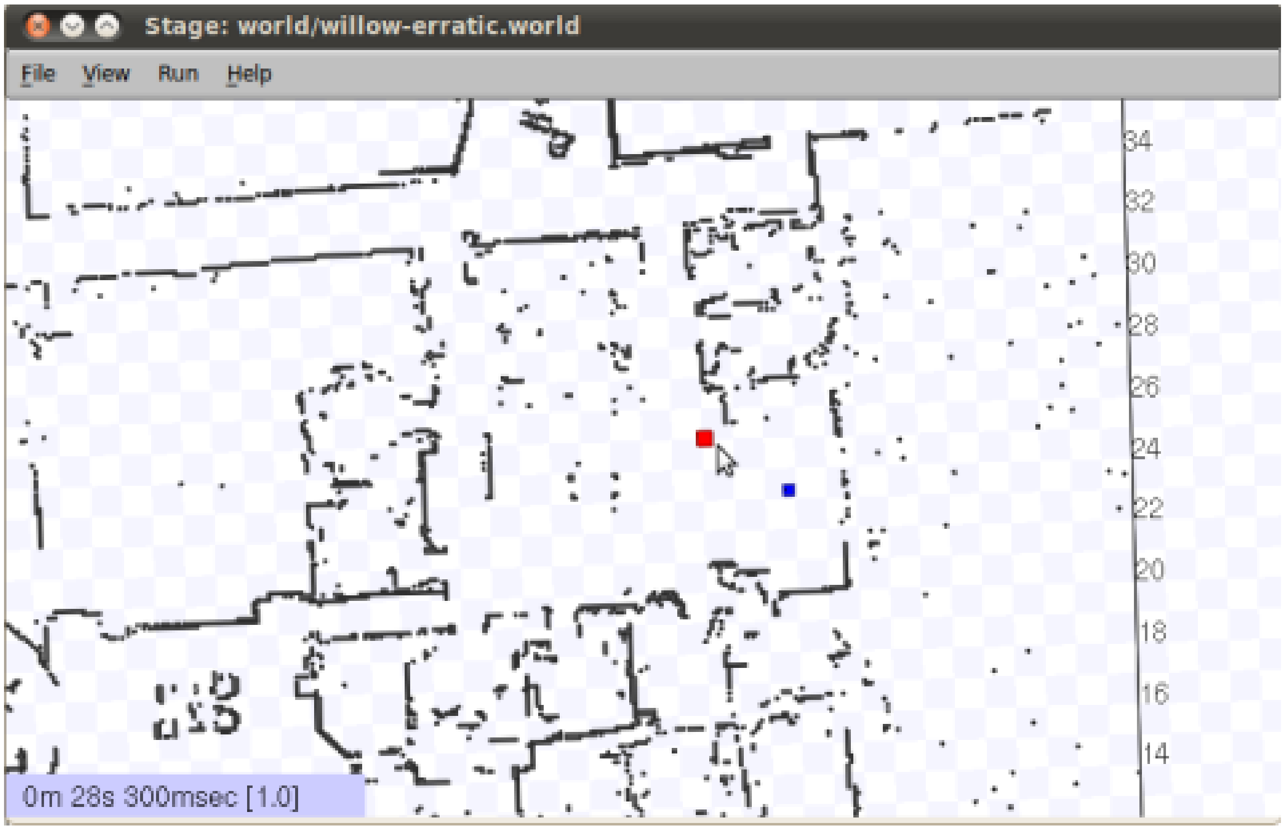
If you do not see this line, (the number of packages may vary) then you must install the stage package. If you have super-user permissions on your machine, type the following in a terminal:

```
sudo apt-get install ros-noetic-stage-ros
```

.

Now follow these http://www.ros.org/wiki/stage/Tutorials/SimulatingOneRobot instructions (from step 3 onwards) to run stage visualization and keyboard control. Keep in mind that you will need to execute roscore, stageros, teleop_base, and rviz simultaneously using separate terminals.

You will see two robots on the Stage window: a red square and a blue square. Your commands will affect the blue robot only.



3. Creating a random walk algorithm

Now that you are familiar with the fundamentals of ROS and the stage simulator, you will learn to create a simple ROS node that will drive the robot around, much like a Roomba robot vaccum cleaner. More specifically, the robot should move forward until it reaches an obstacle, then rotate in place for a random amount of time, then move forward again and repeat.

First type the following in a terminal: (for your own benefits, make sure you understand what each of these lines are doing)

```
> cd ~/catkin_ws/src
> catkin_create_pkg random_walk roscpp geometry_msgs sensor_msgs
> cd random_walk
```

Change the CMakeLists.txt by:

1- change the first line, so the cmake_minimum_required version is 2.8.3

2- adding these lines:

```
include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(random_walk src/random_walk.cpp)
target_link_libraries(random_walk ${catkin_LIBRARIES})
add_dependencies(random_walk beginner_tutorials_generate_messages_cpp) >
```

```
touch src/random_walk.cpp
```

Now go into the random_walk/src folder and paste the following code into random_walk.cpp:

<random_walk.cpp>

To build and run this code, type the following in a terminal:

```
> catkin_make random_walk
> rosrun random_walk random_walk
```

You should start seeing range values being printed back to you in the terminal. If you use your mouse to drag the blue square robot in the Stage window, you will see the range values change.

Your task is now to implement a simple random walk algorithm: if the robot is moving sufficiently close to an obstacle in front of it, then rotate in-place for a specific duration; otherwise move forward by default. Do this by filling in the 3 sections labeled TODO in random_walk.cpp.

What to Submit

Please, together with your code, provide a report discussing your implementation together with some illustrative examples (screen shots) **please enable the trace of the robot** so the trajectory can be seen.

Link to random_walk.cpp