ROS - Publishing and Subscribing to The Turtlebot Simulator

ME 4140 - Introduction to Robotics - Fall 2016

1. This tutorial assumes you have been following the course so far. TO begin create a new package with the name of your choosing.

```
$ catkin_create_pkg publish_goal std_msgs rospy roscpp
```

2. Open a new file in the proper src folder and insert the following code.

```
   #include "ros/ros.h"
#include "geometry_msgs/Twist.h"
#include "geometry_msgs/PoseStamped.h"
#include <sstream>
int main(int argc, char **argv)
{
   ros::init(argc, argv, "publish_goal");
   ros::NodeHandle n;
   ros::Publisher ttu_publisher =
   n.advertise<geometry_msgs::PoseStamped>("/move_base_simple/goal", 1000);
   ros::Rate loop_rate(10);

   geometry_msgs::PoseStamped msg;
   msg.header.stamp=ros::Time::now();
   msg.header.frame_id="map";

   int count = 0;
   while ((ros::ok())&&(count<5))
   {
      msg.pose.position.x = 3.0;
      msg.pose.position.y = 2.0;
      msg.pose.position.z = 0;

      msg.pose.orientation.w = 1.0;

      ttu_publisher.publish(msg);
      ros::spinOnce();
      loop_rate.sleep();
      ++count;
   }
}
```

3. Now you need to install the 'turtlebot' simulator into your ROS system.

```
$ sudo apt-get install ros-indigo-turtlebot-simulator
```

4. Now install the physical 'turtlebot' drivers into your ROS system. This step may only

be necessary if you are using a real robot. Link Here

```
$ sudo apt-get install ros-indigo-turtlebot ros-indigo-turtlebot-
apps ros-indigo-turtlebot-interactions ros-indigo-turtlebot-simulator
ros-indigo-kobuki-ftdi ros-indigo-rocon-remocon ros-indigo-rocon-
qt-library ros-indigo-ar-track-alvar-msgs
```

5. This simulates a physical robot in a 2D world. Next we neeed to setup the world. There are 3 important files that control the world. Your installation came with a demo world.
   - `/opt/ros/indigo/share/turtlebot_stage/maps/maze.png`
   - `/opt/ros/indigo/share/turtlebot_stage/maps/maze.yaml`
   - `/opt/ros/indigo/share/turtlebot_stage/maps/stage/maze.world`
6. First try the simulator in the demo world called *maze*. We will export the files as *environment variables*

```
$ export TURTLEBOT_STAGE_MAP_FILE=
"/opt/ros/indigo/share/turtlebot_stage/maps/maze.yaml"
```

```
$ export TURTLEBOT_STAGE_WORLD_FILE=
"/opt/ros/indigo/share/turtlebot_stage/maps/stage/maze.world"
```

7. Now use the launch file (available upon install) to start the simulator.
   `$ roslaunch turtlebot_stage turtlebot_in_stage.launch`
8. Now you can modify the world you have just simulated. To do this copy all three files and rename them something sensible. Open the *.png* file with any image editor, and draw on it and save. You also need to modify just a few lines in the *.yaml* file and the *.world* file. (Note: This step will be detailed in the next tutorial. Continue at your own risk or contact me for help.)