ROS - Publishing and Subscribing with Turtlebot

ME 4140 - Introduction to Robotics - Fall 2018

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

```
$ cd ~/<workspace_name>/src
$ catkin_create_pkg publish_goal std_msgs roscpp rospy
```

2. Now we are going to make a new node called **publish_goal**. Open a new file in the proper src folder and insert the following cpp code.

```
$ gedit ~/<workspace_name>/src/publish_goal/src/publish_goal.cpp
```

```
#include "ros/ros.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. Edit the *CMakeLists.txt* file for this specific node.

   ```
   $ gedit ~/<workspace_name>/src/publish_goal/CMakeLists.txt
   ```

   Add the following lines at the bottom of the file.

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

4. Compile before running. Turn Everything off before you do this.

   ```
   $ cd ~/<workspace_name>
   $ catkin_make
   ```

5. First start the turtlebot simulator then run the node you just made. You may have to repeat the export commands (or you can add them to the bottom of the .bashrc file.

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

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

   $ roslaunch turtlebot_stage turtlebot_in_stage.launch
   ```

6. Now open a new termimnal

   ```
   $ rosrun publish_goal publish_goal
   ```

   You should see your robot move to location programmed in your source file!

7. Now let us make a node called **subscribe_status** that can access information from the robot in the form of a topic. To do this we are going to make a new node that is part of the same package we just made/used. Open a new file in the proper src folder and insert the following cpp code.

```
$ gedit ~/<workspace_name>/src/publish_goal/src/subscribe_status.cpp
```

```
#include "ros/ros.h"
#include "std_msgs/String.h"
#include "actionlib_msgs/GoalStatusArray.h"

void statusCB(const actionlib_msgs::GoalStatusArray::ConstPtr& msg)
{
    ROS_INFO("Subscriber Callback Executed");
    if (!msg->status_list.empty())
    {
        actionlib_msgs::GoalStatus goalStatus = msg->status_list[0];
        ROS_INFO("Status Recieved: %i",goalStatus.status);
    }
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "subscribe_status");
    ros::NodeHandle n;
    ros::Subscriber sub = n.subscribe("/move_base/status", 1000, statusCB);
    ros::spin();
    return 0;
}
```

8. Edit the correct *CMakeLists.txt* file for this specific node too.

```
$ gedit ~/<workspace_name>/src/publish_goal/CMakeLists.txt
```

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

9. Compile before running. Turn Everything off before you do this.

```
$ cd ~/<workspace_name>
$ catkin_make
```

10. TEST YOUR NODE!