**Steps for Testing Use Cases**

Version 0.0.01

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 08-05-2017 | 0.0.01 | Steps to test AMCL switching, parameter reconfiguration, and planner switching | Anindya Gautam |

**Formal Approval**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Issues** | **Approved by** |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Introduction 4](#_Toc489803966)

[1.1 Purpose 4](#_Toc489803967)

[1.2 Definitions, Acronyms, Abbreviations 4](#_Toc489803968)

[2. AMCL reconfiguration steps 5](#_Toc489803969)

[3. Steps to change local planner 5](#_Toc489803970)

[4. Parameter reconfiguration steps 6](#_Toc489803971)

# **Introduction**

The Turtlebot project aims at building a framework to support configuration/reconfiguration for mobile robots at run time.

## Purpose

The purpose of this document is to list down the steps to be followed to test each use-case for the reconfiguration framework.

## Definitions, Acronyms, Abbreviations

|  |  |
| --- | --- |
| **DAA** | **Term** |
| ROS | Robot Operating System |
| Bot | Robot |
| TBD | To Be Discussed |
| SCN | System Control Node |
| API | Application Programing Interface |

# **AMCL reconfiguration steps**

Ensure that the project is built correctly and contains the devel, build, include and src folders. The list of commands to test AMCL reconfiguration are as follows:

* source devel/setup.bash
* roscore
* rosrun reconfigure reconfigure\_node
* roslaunch turtlebot\_bringup minimal.launch
* roslaunch turtlebot\_navigation amcl\_demo.launch map\_file:=/home/turtlebot/maps/4207.pgm.yaml

Once all the nodes are running,

* roslaunch turtlebot\_rviz\_launchers view\_navigation.launch --screen

In rviz, set the 2D Pose Estimate and the 2D Nav Goal. Once the bot starts moving, we can switch amcl with amcl using the following command:

* rosservice call /userInterfaceService "{reconType: 2, preserveState: false, oldNode: 'amcl', oldNodePackage: ‘amcl’, newNode: 'amcl', newNodePackage: 'amcl'}"

A successful reconfiguration shall return a result of 0. This can be verified in the logs as the previous amcl node will be killed and a new amcl node shall be started. The bot shall halt for a small amount of time while the reconfiguration is ongoing and then continue with its movement.

# **Steps to change local planner**

Ensure that the project is built correctly and contains the devel, build, include and src folders. As the planners are static components of move\_base, move\_base needs to be recompiled every time a planner is changed. We make the assumption here that move\_base has been compiled twice – once with the original planner (E.g. Trajectory Planner) and another time with the new planner (E.g. DWA Planner).

The list of commands to test a change in local planner of movebase are as follows:

* source devel/setup.bash
* roscore
* rosrun reconfigure reconfigure\_node
* roslaunch turtlebot\_bringup minimal.launch
* roslaunch turtlebot\_navigation amcl\_demo.launch map\_file:=/home/turtlebot/maps/4207.pgm.yaml

After this, we recompile move\_base with the new planner. Once all the nodes are running,

* roslaunch turtlebot\_rviz\_launchers view\_navigation.launch --screen

In rviz, set the 2D Pose Estimate and the 2D Nav Goal. Once the bot starts moving, we can switch move\_base having the old planner with move\_base having the new planner using the following command:

* rosservice call /userInterfaceService "{reconType: 2, preserveState: false, oldNode: 'move\_base', , oldNodePackage: ‘move\_base’, newNode: 'move\_base', newNodePackage: 'move\_base'}"

A successful reconfiguration shall return a result of 0. This can be verified in the logs as the previous move\_base node will be killed and a new move\_base node shall be started. The bot shall halt after reconfiguration as its goals need to be set again for the new move\_base.

# **Parameter reconfiguration steps**

To change parameters such as minimum and maximum velocity, the bot does not require to enter reconfiguration mode as such parameters are dynamic. Such dynamic parameters can be modified using the existing dynamic reconfigure APIs already provided by ROS, using the command:

* rosrun dynamic\_reconfigure dynparam set /node parameter\_name value