

robotics/catkin_ws/src

Robotics Course - ROS

Giovanni Pacherà

Aprile 2025

Contents

1	pub_sub	7
1.1	CMakeLists.txt	7
1.2	launch	7
1.2.1	multi_turtle.launch	7
1.2.2	simple_launch_namespace.launch	8
1.2.3	simple_launch.launch	8
1.3	package.xml	8
1.4	src	9
1.4.1	pub.cpp	9
1.4.2	sub.cpp	10
2	pub_sub_same	10
2.1	CMakeLists.txt	10
2.2	package.xml	11
2.3	src	13
2.3.1	test_pub.cpp	13
2.3.2	test_sub.cpp	14
3	service	16
3.1	CMakeLists.txt	16
3.2	package.xml	16
3.3	src	17
3.3.1	add_two_int.cpp	17
3.3.2	client.cpp	18
3.4	srv	19
3.4.1	AddTwoInts.srv	19
4	custom_messages	19
4.1	CMakeLists.txt	19
4.2	package.xml	20
4.3	msg	21
4.3.1	Num.msg	21
4.4	src	21
4.4.1	pub.cpp	21
4.4.2	sub.cpp	22

5	timer	22
5.1	CMakeLists.txt	22
5.2	package.xml	23
5.3	src	23
5.3.1	pub.cpp	23
6	parameter_test	24
6.1	CMakeLists.txt	24
6.2	package.xml	25
6.3	cfg	27
6.3.1	parameters.cfg	27
6.4	launch	27
6.4.1	param_set.launch	27
6.5	src	28
6.5.1	param_first.cpp	28
6.5.2	param_second.cpp	29
7	message_filters	30
7.1	CMakeLists.txt	30
7.2	package.xml	30
7.3	src	33
7.3.1	pub.cpp	33
7.3.2	sub_pol.cpp	34
7.3.3	sub.cpp	35
8	tf	36
8.1	CMakeLists.txt	36
8.2	package.xml	36
8.3	launch	38
8.3.1	turtle.launch	38
8.3.2	turtle.launch.old	39
8.4	src	39
8.4.1	get_tf.cpp	39
8.4.2	pub.cpp	41
9	fibonacci	42
9.1	CMakeLists.txt	42
9.2	package.xml	44
9.3	action	46
9.3.1	Fibonacci.action	46
9.4	launch	47

9.4.1	launcher.launch	47
9.4.2	launcher2.launch	47
9.5	src	47
9.5.1	fibonacci_client.cpp	47
9.5.2	fibonacci_client2.cpp	49
9.5.3	fibonacci_server.cpp	51
10	pub_latched	54
10.1	CMakeLists.txt	54
10.2	package.xml	54
10.3	src	56
10.3.1	pub.cpp	56
10.4	launch	57
10.4.1	launcher.launch	57
11	asynch	57
11.1	CMakeLists.txt	57
11.2	package.xml	58
11.3	src	60
11.3.1	pub.cpp	60
11.3.2	standard_sub.cpp	61
11.3.3	asynch_sub.cpp	62
12	stage	63
12.1	maze.png	63
12.2	maze.world	63
12.3	turtlebot.inc	64
13	demo_mapping	65
13.1	CMakeLists.txt	65
13.2	package.xml	66
13.3	launch	68
13.3.1	gmapping.launch	68
13.3.2	slam_toolbox.launch	68
14	nav2d_conf	68
14.1	CMakeLists.txt	68
14.2	package.xml	74
14.3	cfg	76
14.3.1	costmap_common_params.yaml	76
14.3.2	dwa_local_planner_params.yaml	77

14.3.3	global_costmap_params.yaml	79
14.3.4	local_costmap_params.yaml	80
14.3.5	move_base_params.yaml	80
14.4	maps	81
14.4.1	maze.png	81
14.4.2	maze.yaml	81
14.4.3	robopark_plan.yaml	81
14.4.4	robopark2.bmp	82
14.4.5	stage	82
14.5	rviz	85
14.5.1	robot_navigation.rviz	85
14.6	launch	95
14.6.1	amcl.launch.xml	95
14.6.2	gmapping.launch.xml	97
14.6.3	turtlebot_amcl.launch	98
14.6.4	turtlebot_gmapping.launch	101

1 pub_sub

1.1 CMakeLists.txt

```
1 cmake_minimum_required(VERSION 2.8.3)
2 project(pub_sub)
3
4 ## Find catkin and any catkin packages
5 find_package(catkin REQUIRED COMPONENTS roscpp std_msgs)
6
7 ## Declare a catkin package
8 catkin_package()
9
10 ## Build talker and listener
11 include_directories(include ${catkin_INCLUDE_DIRS})
12
13 add_executable(pub src/pub.cpp)
14 target_link_libraries(pub ${catkin_LIBRARIES})
15
16 add_executable(sub src/sub.cpp)
17 target_link_libraries(sub ${catkin_LIBRARIES})
```

1.2 launch

1.2.1 multi_turtle.launch

```
1 <launch>
2   <group ns="turtlesim1">
3     <node pkg="turtlesim" name="sim" type="turtlesim_node" />
4     <node pkg="turtlesim" name="teleop" type="turtle_teleop_key" />
5   </group>
6
7   <group ns="turtlesim2">
8     <node pkg="turtlesim" name="sim" type="turtlesim_node" />
9   </group>
10
11   <node pkg="turtlesim" name="mimic" type="mimic">
12     <remap from="input" to="turtlesim1/turtle1" />
13     <remap from="output" to="turtlesim2/turtle1" />
```

```
14   </node>
15 </launch>
```

1.2.2 simple_launch_namespace.launch

```
1 <launch>
2   <!-- Publisher node with namespace -->
3   <group ns="namespace">
4     <node pkg="pub_sub" type="pub" name="talker" output="
       screen" />
5   </group>
6 </launch>
```

1.2.3 simple_launch.launch

```
1 <launch>
2   <!-- Publisher node -->
3   <node pkg="pub_sub" type="pub" name="talker" output="
       screen" />
4   <!-- Subscriber node -->
5   <node pkg="pub_sub" type="sub" name="listener" output
       ="screen" />
6 </launch>
```

1.3 package.xml

```
1 <?xml version="1.0"?>
2 <package format="2">
3   <name>pub_sub</name>
4   <version>0.0.0</version>
5   <description>The pub_sub package</description>
6   <maintainer email="simone.mentasti@polimi.it">simone</
       maintainer>
7   <license>GPL</license>
8
9   <buildtool_depend>catkin</buildtool_depend>
10  <build_depend>roscpp</build_depend>
11  <build_depend>std_msgs</build_depend>
12
13  <build_export_depend>roscpp</build_export_depend>
```



```
14 <build_export_depend>std_msgs</build_export_depend>
15
16 <exec_depend>roscpp</exec_depend>
17 <exec_depend>std_msgs</exec_depend>
18
19 <export>
20   <!-- Other tools can request additional information
      be placed here -->
21 </export>
22 </package>
```

1.4 src

1.4.1 pub.cpp

```
1 #include "ros/ros.h"
2 #include "std_msgs/String.h"
3
4 int main(int argc, char **argv){
5     ros::init(argc, argv, "talker"); //, ros::
      init_options::AnonymousName);
6     ros::NodeHandle n;
7     ros::Publisher chatter_pub = n.advertise<std_msgs::
      String>("chatter", 1);
8
9     ros::Rate loop_rate(10);
10
11     while (ros::ok()){
12         std_msgs::String msg;
13         msg.data = "hello world!";
14         ROS_INFO("%s", msg.data.c_str());
15         chatter_pub.publish(msg);
16
17         ros::spinOnce();
18
19         loop_rate.sleep();
20     }
21
22     return 0;
23 }
```

1.4.2 sub.cpp

```
1 #include "ros/ros.h"
2 #include "std_msgs/String.h"
3
4 void chatterCallback(const std_msgs::String::ConstPtr&
   msg){
5     ROS_INFO("I heard: [%s]", msg->data.c_str());
6 }
7
8 int main(int argc, char **argv){
9     ros::init(argc, argv, "listener");
10    ros::NodeHandle n;
11    ros::Subscriber sub = n.subscribe("chatter", 1,
        chatterCallback);
12
13    ros::spin();
14
15    return 0;
16 }
```

2 pub_sub_same

2.1 CMakeLists.txt

```
1 cmake_minimum_required(VERSION 2.8.3)
2 project(pub_sub_same)
3
4 find_package(catkin REQUIRED)
5 find_package(catkin REQUIRED COMPONENTS
6     roscpp
7 )
8
9 catkin_package(
10 )
11
12 include_directories(
13     include ${catkin_INCLUDE_DIRS}
14 )
15
16 add_executable(test_pub
```

```
17   src/test_pub.cpp
18 )
19
20 add_dependencies(test_pub ${${PROJECT_NAME}
   _EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
21
22 target_link_libraries(test_pub
23   ${catkin_LIBRARIES}
24 )
25
26 add_executable(test_sub
27   src/test_sub.cpp
28 )
29
30 add_dependencies(test_sub ${${PROJECT_NAME}
   _EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
31
32 target_link_libraries(test_sub
33   ${catkin_LIBRARIES}
34 )
```

2.2 package.xml

```
1 <?xml version="1.0"?>
2 <package format="2">
3   <name>pub_sub_same</name>
4   <version>0.0.0</version>
5   <description>The test package</description>
6   <!-- One maintainer tag required, multiple allowed, one
   person per tag -->
7   <!-- Example: -->
8   <!-- <maintainer email="jane.doe@example.com">Jane Doe
   </maintainer> -->
9   <maintainer email="simone@todo.todo">simone</maintainer
   >
10
11   <!-- One license tag required, multiple allowed, one
   license per tag -->
12   <!-- Commonly used license strings: -->
13   <!-- BSD, MIT, Boost Software License, GPLv2, GPLv3,
   LGPLv2.1, LGPLv3 -->
14   <license>TODO</license>
```

```

15 <!-- Url tags are optional, but multiple are allowed,
    one per tag -->
16 <!-- Optional attribute type can be: website,
    bugtracker, or repository -->
17 <!-- Example: -->
18 <!-- <url type="website">http://wiki.ros.org/test</url>
    -->
19 <!-- Author tags are optional, multiple are allowed,
    one per tag -->
20 <!-- Authors do not have to be maintainers, but could
    be -->
21 <!-- Example: -->
22 <!-- <author email="jane.doe@example.com">Jane Doe</
    author> -->
23 <!-- The *depend tags are used to specify dependencies
    -->
24 <!-- Dependencies can be catkin packages or system
    dependencies -->
25 <!-- Examples: -->
26 <!-- Use depend as a shortcut for packages that are
    both build and exec dependencies -->
27 <!-- <depend>roscpp</depend> -->
28 <!-- Note that this is equivalent to the following:
    -->
29 <!-- <build_depend>roscpp</build_depend> -->
30 <!-- <exec_depend>roscpp</exec_depend> -->
31 <!-- Use build_depend for packages you need at compile
    time: -->
32 <!-- <build_depend>message_generation</build_depend>
    -->
33 <!-- Use build_export_depend for packages you need in
    order to build against this package: -->
34 <!-- <build_export_depend>message_generation</
    build_export_depend> -->
35 <!-- Use buildtool_depend for build tool packages: -->
36 <!-- <buildtool_depend>catkin</buildtool_depend> -->
37 <!-- Use exec_depend for packages you need at runtime:
    -->
38 <!-- <exec_depend>message_runtime</exec_depend> -->
39 <!-- Use test_depend for packages you need only for
    testing: -->
40 <!-- <test_depend>gtest</test_depend> -->

```

```
41 <!-- Use doc_depend for packages you need only for
    building documentation: -->
42 <!-- <doc_depend>doxygen</doc_depend> -->
43 <buildtool_depend>catkin</buildtool_depend>
44 <!-- The export tag contains other, unspecified, tags
    -->
45
46 <export>
47 <!-- Other tools can request additional information
    be placed here -->
48 </export>
49 </package>
```

2.3 src

2.3.1 test_pub.cpp

```
1 #include "ros/ros.h"
2 #include "std_msgs/String.h"
3 #include <stdlib.h>
4 #include <sstream>
5
6 int main(int argc, char *argv[])
7 {
8     ros::init(argc, argv, "publisher");
9     ros::NodeHandle n;
10    ros::Publisher chatter_pub = n.advertise<std_msgs::
        String>("chatter", 1000);
11    ros::Publisher chatter_pub2 = n.advertise<std_msgs::
        String>("chatter2", 1000);
12
13    ros::Rate loop_rate(10);
14    int count = 0;
15
16    while (ros::ok())
17    {
18        std_msgs::String msg;
19        std::stringstream ss;
20        ss << "hello world " << count;
21        msg.data = ss.str();
22        ROS_INFO("%s", msg.data.c_str());
23    }
```

```
24         if (rand() % 10 < 6){
25             chatter_pub.publish(msg);
26         }
27
28         if (rand() % 10 < 2){
29             chatter_pub2.publish (msg);
30         }
31
32         ros::spinOnce();
33         loop_rate.sleep();
34         ++count;
35     }
36
37     return 0;
38 }
```

2.3.2 test_sub.cpp

```
1 #include "ros/ros.h"
2 #include "std_msgs/String.h"
3
4 class PubSubNode {
5 private:
6     // ROS node handle
7     ros::NodeHandle nh_;
8     // Subscribers for the "/chatter" and "/chatter2"
9     // topics
10    ros::Subscriber chatter_sub_;
11    ros::Subscriber chatter2_sub_;
12    // Publisher for the "/rechatter" topic
13    ros::Publisher rechatter_pub_;
14    // Timer for periodic publishing
15    ros::Timer timer_
16    // Latest received messages from the subscribed
17    // topics
18
19    std_msgs::String chatterMsg_;
20    std_msgs::String chatter2Msg_;
21 public:
22     // Constructor: sets up subscribers, publisher, and
23     // timer
24     PubSubNode() {
```

```
22     // Subscribe to topics with a queue size of 1
23     chatter_sub_ = nh_.subscribe("/chatter", 1, &
        PubSubNode::chatterCallback, this);
24     chatter2_sub_ = nh_.subscribe("/chatter2", 1, &
        PubSubNode::chatter2Callback, this);
25     // Advertise the publisher on "/rechatter" topic
26     rechatter_pub_ = nh_.advertise<std_msgs::String>(
        "/rechatter", 1);
27     // Create a timer that triggers every 1 second to
        publish messages
28     timer_ = nh_.createTimer(ros::Duration(1.0), &
        PubSubNode::timerCallback, this);
29 }
30
31 // Callback function for the "/chatter" topic
32 void chatterCallback(const std_msgs::String::ConstPtr
    & msg) {
33     chatterMsg_ = *msg;
34 }
35
36 // Callback function for the "/chatter2" topic
37 void chatter2Callback(const std_msgs::String::
    ConstPtr& msg) {
38     chatter2Msg_ = *msg;
39 }
40
41 // Timer callback function: publishes messages
    periodically
42 void timerCallback(const ros::TimerEvent&) {
43     // Publish the latest messages from both
        subscribers
44     rechatter_pub_.publish(chatterMsg_);
45     rechatter_pub_.publish(chatter2Msg_);
46     ROS_INFO("Timer callback executed: messages
        published.");
47 }
48 };
49 int main(int argc, char **argv) {
50     ros::init(argc, argv, "subscribe_and_publish");
51     PubSubNode node;
52
53     ros::spin();
54 }
```

```
55     return 0;
56 }
```

3 service

3.1 CMakeLists.txt

```
1 cmake_minimum_required(VERSION 2.8.3)
2 project(service)
3
4 ## Find catkin and any catkin packages
5 find_package(catkin REQUIRED COMPONENTS roscpp std_msgs
6         message_generation )
7
8 add_service_files(
9     FILES
10    AddTwoInts.srv
11)
12
13 generate_messages(
14     DEPENDENCIES
15     std_msgs
16 )
17
18 ## Declare a catkin package
19 catkin_package(CATKIN_DEPENDS message_runtime)
20
21 ## Build talker and listener
22 include_directories(include ${catkin_INCLUDE_DIRS})
23 add_executable(add_two_int src/add_two_int.cpp)
24 target_link_libraries(add_two_int ${catkin_LIBRARIES})
25 add_dependencies(add_two_int ${catkin_EXPORTED_TARGETS})
26 add_executable(client src/client.cpp)
27 target_link_libraries(client ${catkin_LIBRARIES})
28 add_dependencies(client ${catkin_EXPORTED_TARGETS})
```

3.2 package.xml

```
1 <?xml version="1.0"?>
2 <package format="2">
```



```
3 <name>service</name>
4 <version>0.0.0</version>
5 <description>The service package</description>
6 <maintainer email="simone.mentasti@polimi.it">simone</
  maintainer>
7 <license>GPL</license>
8
9 <buildtool_depend>catkin</buildtool_depend>
10 <build_depend>roscpp</build_depend>
11 <build_depend>std_msgs</build_depend>
12 <build_depend>message_generation</build_depend>
13
14 <build_export_depend>roscpp</build_export_depend>
15 <build_export_depend>std_msgs</build_export_depend>
16
17 <exec_depend>roscpp</exec_depend>
18 <exec_depend>std_msgs</exec_depend>
19 <exec_depend>message_runtime</exec_depend>
20
21 <!-- The export tag contains other, unspecified, tags
  -->
22 <export>
23   <!-- Other tools can request additional information
     be placed here -->
24 </export>
25 </package>
```

3.3 src

3.3.1 add_two_int.cpp

```
1 #include "ros/ros.h"
2 #include "service/AddTwoInts.h"
3
4 bool add(service::AddTwoInts::Request &req, service::
  AddTwoInts::Response &res)
5 {
6   res.sum = req.a + req.b;
7   ROS_INFO("request: x=%ld, y=%ld", (long int)req.a, (
     long int)req.b);
8   ROS_INFO("sending back response: [%ld]", (long int)res.
     sum);
```

```
9   return true;
10 }
11
12 int main(int argc, char **argv)
13 {
14     ros::init(argc, argv, "add_two_ints_server");
15     ros::NodeHandle n;
16     ros::ServiceServer service = n.advertiseService("
        add_two_ints", add);
17     ROS_INFO("Ready to add two ints.");
18
19     ros::spin();
20
21     return 0;
22 }
```

3.3.2 client.cpp

```
1 #include "ros/ros.h"
2 #include "service/AddTwoInts.h"
3
4 int main(int argc, char **argv)
5 {
6     ros::init(argc, argv, "add_two_ints_client");
7
8     if (argc != 3)
9     {
10         ROS_INFO("usage: add_two_ints_client X Y");
11         return 1;
12     }
13
14     ros::NodeHandle n;
15     ros::ServiceClient client = n.serviceClient<service::
        AddTwoInts>("add_two_ints");
16     service::AddTwoInts srv;
17     srv.request.a = atoll(argv[1]);
18     srv.request.b = atoll(argv[2]);
19
20     if (client.call(srv))
21     {
22         ROS_INFO("Sum: %ld", (long int)srv.response.sum);
23     }
```

```
24     else
25     {
26         ROS_ERROR("Failed to call service add_two_ints");
27
28         return 1;
29     }
30
31     return 0;
32 }
```

3.4 srv

3.4.1 AddTwoInts.srv

```
1 int64 a
2 int64 b
3 ---
4 int64 sum
```

4 custom_messages

4.1 CMakeLists.txt

```
1 cmake_minimum_required(VERSION 2.8.3)
2 project(custom_messages)
3
4 ## Find catkin and any catkin packages
5 find_package(catkin REQUIRED COMPONENTS roscpp std_msgs
6           message_generation)
7
8 add_message_files(
9     FILES
10    Num.msg
11)
12
13 generate_messages(
14     DEPENDENCIES
15     std_msgs
16 )
```

```
17 ## Declare a catkin package
18 catkin_package( CATKIN_DEPENDS message_runtime)
19
20 ## Build talker and listener
21 include_directories(include ${catkin_INCLUDE_DIRS})
22 add_executable(pub_custom src/pub.cpp)
23 add_dependencies(pub_custom
24     custom_messages_generate_messages_cpp)
25
26 target_link_libraries(pub_custom ${catkin_LIBRARIES})
27 add_executable(sub_custom src/sub.cpp)
28
29 target_link_libraries(sub_custom ${catkin_LIBRARIES})
30 add_dependencies(sub_custom
31     custom_messages_generate_messages_cpp)
```

4.2 package.xml

```
1 <?xml version="1.0"?>
2 <package format="2">
3   <name>custom_messages</name>
4   <version>0.0.0</version>
5   <description>The custom mkenvironment package</description>
6   <maintainer email="simone.mentasti@polimi.it">simone</
7     maintainer>
8   <license>GPL</license>
9
10   <buildtool_depend>catkin</buildtool_depend>
11   <build_depend>roscpp</build_depend>
12   <build_depend>std_msgs</build_depend>
13   <build_depend>message_generation</build_depend>
14
15   <build_export_depend>roscpp</build_export_depend>
16   <build_export_depend>std_msgs</build_export_depend>
17
18   <exec_depend>roscpp</exec_depend>
19   <exec_depend>std_msgs</exec_depend>
20   <exec_depend>message_runtime</exec_depend>
21
22   <export>
23     <!-- Other tools can request additional information
24        be placed here -->
```

```
23 </export>
24 </package>
```

4.3 msg

4.3.1 Num.msg

```
1 int64 num
```

4.4 src

4.4.1 pub.cpp

```
1 #include "ros/ros.h"
2 #include "std_msgs/String.h"
3 #include "custom_messages/Num.h"
4 #include <sstream>
5
6 int main(int argc, char **argv){
7     ros::init(argc, argv, "talker");
8     ros::NodeHandle n;
9     ros::Publisher chatter_pub = n.advertise<
        custom_messages::Num>("chatter", 1000);
10
11     ros::Rate loop_rate(10);
12     int count = 0;
13
14     while (ros::ok()){
15         static int i=0;
16         i=(i+1)%1000;
17
18         custom_messages::Num msg;
19         msg.num =i;
20
21         chatter_pub.publish (msg);
22     }
23
24     return 0;
25 }
```

4.4.2 sub.cpp

```
1 #include "ros/ros.h"
2 #include "std_msgs/String.h"
3 #include "custom_messages/Num.h"
4
5 void chatterCallback(const custom_messages::Num::ConstPtr
    & msg){
6     ROS_INFO("I heard: [%ld]", msg->num);
7 }
8
9 int main(int argc, char **argv){
10     ros::init(argc, argv, "listener");
11     ros::NodeHandle n;
12     ros::Subscriber sub = n.subscribe("chatter", 1000,
        chatterCallback);
13
14     ros::spin();
15
16     return 0;
17 }
```

5 timer

5.1 CMakeLists.txt

```
1 cmake_minimum_required(VERSION 2.8.3)
2 project(timer)
3
4 ## Find catkin and any catkin packages
5 find_package(catkin REQUIRED COMPONENTS roscpp std_msgs)
6
7 ## Declare a catkin package
8 catkin_package()
9
10 ## Build talker and listener
11 include_directories(include ${catkin_INCLUDE_DIRS})
12
13 add_executable(timed_pub src/pub.cpp)
14 target_link_libraries(timed_pub ${catkin_LIBRARIES})
```

5.2 package.xml

```
1 <?xml version="1.0"?>
2 <package format="2">
3   <name>timer</name>
4   <version>0.0.0</version>
5   <description>The timer package</description>
6   <maintainer email="simone.mentasti@polimi.it">simone</
   maintainer>
7   <license>GPL</license>
8
9   <buildtool_depend>catkin</buildtool_depend>
10  <build_depend>roscpp</build_depend>
11  <build_export_depend>std_msgs</build_export_depend>
12
13  <build_export_depend>roscpp</build_export_depend>
14  <build_export_depend>std_msgs</build_export_depend>
15
16  <exec_depend>roscpp</exec_depend>
17  <exec_depend>std_msgs</exec_depend>
18
19  <!-- The export tag contains other, unspecified, tags
   -->
20  <export>
21    <!-- Other tools can request additional information
       be placed here -->
22  </export>
23 </package>
```

5.3 src

5.3.1 pub.cpp

```
1 #include "ros/ros.h"
2 #include "std_msgs/String.h"
3 #include <time.h>
4
5 void timerCallback(const ros::TimerEvent& ev){
6     ROS_INFO_STREAM("Callback called at time: " << ros::
       Time::now());
7 }
8
```

```
9 int main(int argc, char **argv){
10     ros::init(argc, argv, "timed_talker");
11     ros::NodeHandle n;
12     ros::Timer timer = n.createTimer(ros::Duration(0.1),
13         timerCallback);
14     ros::spin();
15     return 0;
16 }
```

6 parameter_test

6.1 CMakeLists.txt

```
1 cmake_minimum_required(VERSION 2.8.3)
2 project(parameter_test)
3
4 ## Find catkin and any catkin packages
5 find_package(catkin REQUIRED COMPONENTS roscpp std_msgs
6     dynamic_reconfigure)
7
8 generate_dynamic_reconfigure_options(
9     cfg/parameters.cfg
10 )
11
12 ## Declare a catkin package
13 catkin_package()
14
15 ## Build talker and listener
16 include_directories(include ${catkin_INCLUDE_DIRS})
17
18 add_executable(param_first src/param_first.cpp)
19 target_link_libraries(param_first ${catkin_LIBRARIES})
20
21 add_executable(param_second src/param_second.cpp)
22 target_link_libraries(param_second ${catkin_LIBRARIES})
23
24 add_dependencies(param_second ${PROJECT_NAME}_gencfg)
25 target_link_libraries(param_second ${catkin_LIBRARIES})
```


6.2 package.xml

```
1 <?xml version="1.0"?>
2 <package format="2">
3   <name>parameter_test</name>
4   <version>0.0.0</version>
5   <description>The pub_sub package</description>
6
7   <!-- One maintainer tag required, multiple allowed, one
      person per tag -->
8   <!-- Example: -->
9   <!-- <maintainer email="jane.doe@example.com">Jane Doe
      </maintainer> -->
10  <maintainer email="simone@todo.todo">simone</maintainer>
      >
11  <!-- One license tag required, multiple allowed, one
      license per tag -->
12  <!-- Commonly used license strings: -->
13  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3,
      LGPLv2.1, LGPLv3 -->
14  <license>TODO</license>
15  <!-- Url tags are optional, but multiple are allowed,
      one per tag -->
16  <!-- Optional attribute type can be: website,
      bugtracker, or repository -->
17  <!-- Example: -->
18  <!-- <url type="website">http://wiki.ros.org/pub_sub</
      url> -->
19  <!-- Author tags are optional, multiple are allowed,
      one per tag -->
20  <!-- Authors do not have to be maintainers, but could
      be -->
21  <!-- Example: -->
22  <!-- <author email="jane.doe@example.com">Jane Doe</
      author> -->
23  <!-- The *depend tags are used to specify dependencies
      -->
24  <!-- Dependencies can be catkin packages or system
      dependencies -->
25  <!-- Examples: -->
26  <!-- Use depend as a shortcut for packages that are
      both build and exec dependencies -->
27  <!--   <depend>roscpp</depend> -->
```

```

28 <!-- Note that this is equivalent to the following:
   -->
29 <!-- <build_depend>roscpp</build_depend> -->
30 <!-- <exec_depend>roscpp</exec_depend> -->
31 <!-- Use build_depend for packages you need at compile
   time: -->
32 <!-- <build_depend>message_generation</build_depend>
   -->
33 <!-- Use build_export_depend for packages you need in
   order to build against this package: -->
34 <!-- <build_export_depend>message_generation</
   build_export_depend> -->
35
36 <!-- Use buildtool_depend for build tool packages: -->
37 <!-- <buildtool_depend>catkin</buildtool_depend> -->
38 <!-- Use exec_depend for packages you need at runtime:
   -->
39 <!-- <exec_depend>message_runtime</exec_depend> -->
40 <!-- Use test_depend for packages you need only for
   testing: -->
41 <!-- <test_depend>gtest</test_depend> -->
42 <!-- Use doc_depend for packages you need only for
   building documentation: -->
43
44 <!-- <doc_depend>doxygen</doc_depend> -->
45 <buildtool_depend>catkin</buildtool_depend>
46 <build_depend>roscpp</build_depend>
47 <build_depend>std_msgs</build_depend>
48 <build_export_depend>roscpp</build_export_depend>
49 <build_export_depend>std_msgs</build_export_depend>
50 <exec_depend>roscpp</exec_depend>
51 <exec_depend>std_msgs</exec_depend>
52 <!-- The export tag contains other, unspecified, tags
   -->
53
54 <export>
55 <!-- Other tools can request additional information
   be placed here -->
56 </export>
57
58 </package>

```

6.3 cfg

6.3.1 parameters.cfg

```
1 #!/usr/bin/env python
2 PACKAGE = "parameter_test"
3 from dynamic_reconfigure.parameter_generator_catkin
4     import *
5
6 gen = ParameterGenerator()
7 gen.add("int_param",    int_t,    0, "An Integer
8     parameter", 50, 0, 100)
9 gen.add("double_param", double_t, 1, "A double parameter"
10     , .5, 0, 1)
11 gen.add("str_param",    str_t,    2, "A string parameter"
12     , "Hello World")
13 gen.add("bool_param",   bool_t,   3, "A Boolean parameter
14     ", True)
15 size_enum = gen.enum([ gen.const("Small",      int_t, 0,
16     "A small constant"),
17     gen.const("Medium",      int_t, 1,
18     "A medium constant"),
19     gen.const("Large",       int_t, 2,
20     "A large constant"),
21     gen.const("ExtraLarge",  int_t, 3,
22     "An extra large constant")],
23     "An enum to set size")
24 gen.add("size", int_t, 4, "A size parameter which is
25     edited via an enum", 1, 0, 3, edit_method=size_enum)
26
27 exit(gen.generate(PACKAGE, "param_second", "parameters"))
```

6.4 launch

6.4.1 param_set.launch

```
1 <launch>
2   <!-- Global parameter available to all nodes -->
3   <param name="global_param" value="global_value" />
4   <!-- Launch the node with a private parameter -->
5
6   <node pkg="parameter_test" type="param_first" name="
7       param_first" output="screen">
```

```
7     <!-- Private parameter for the node "my_node" -->
8     <param name="private_param" value="
        node_specific_value" />
9 </node>
10 </launch>
```

6.5 src

6.5.1 param_first.cpp

```
1 #include <ros/ros.h>
2 #include <string>
3
4 int main(int argc, char **argv)
5 {
6     // Initialize the ROS node with the name "my_node"
7     ros::init(argc, argv, "my_node");
8     // Create a NodeHandle for global parameters
9     ros::NodeHandle nh;
10    // Create a NodeHandle in the private namespace for
        node-specific parameters
11    ros::NodeHandle private_nh("~");
12
13    // Variables to hold the parameter values
14    std::string global_param_value;
15    std::string private_param_value;
16    // Retrieve a global parameter (set using an absolute
        name like "/global_param")
17
18    if (nh.getParam("/global_param", global_param_value))
19    {
20        ROS_INFO("Global parameter: %s", global_param_value.
            c_str());
21    }
22    else
23    {
24        ROS_WARN("Global parameter not found, using default
            value");
25        global_param_value = "default_global";
26    }
27
```

```
28 // Retrieve a private (node-specific) parameter (set
    using the private namespace, e.g., "~private_param")
29 if (private_nh.getParam("private_param",
    private_param_value))
30 {
31     ROS_INFO("Private parameter: %s", private_param_value
        .c_str());
32 }
33 else
34 {
35     ROS_WARN("Private parameter not found, using default
        value");
36     private_param_value = "default_private";
37 }
38
39 // The node can now use these parameters as needed
40 ros::spin();
41
42 return 0;
43 }
```

6.5.2 param_second.cpp

```
1 #include <ros/ros.h>
2 #include <dynamic_reconfigure/server.h>
3 #include <parameter_test/parametersConfig.h>
4
5 void callback(parameter_test::parametersConfig &config,
    uint32_t level) {
6     ROS_INFO("Reconfigure Request: %d %f %s %s %d",
7         config.int_param, config.double_param,
8         config.str_param.c_str(),
9         config.bool_param?"True":"False",
10        config.size);
11     ROS_INFO ("%d",level);
12 }
13
14 int main(int argc, char **argv) {
15     ros::init(argc, argv, "param_second");
16     dynamic_reconfigure::Server<parameter_test::
        parametersConfig> server;
```

```
17 dynamic_reconfigure::Server<parameter_test::  
    parametersConfig>::CallbackType f;  
18 f = boost::bind(&callback, _1, _2);  
19 server.setCallback(f);  
20 ROS_INFO("Spinning node");  
21  
22 ros::spin();  
23  
24 return 0;  
25 }
```

7 message_filters

7.1 CMakeLists.txt

```
1 cmake_minimum_required(VERSION 2.8.3)  
2 project(message_filters_example)  
3  
4 ## Find catkin and any catkin packages  
5 find_package(catkin REQUIRED COMPONENTS roscpp std_msgs  
    geometry_msgs message_filters)  
6  
7 ## Declare a catkin package  
8 catkin_package( CATKIN_DEPENDS geometry_msgs  
    message_filters)  
9  
10 ## Build talker and listener  
11 include_directories(include ${catkin_INCLUDE_DIRS})  
12 add_executable(multi_publisher src/pub.cpp)  
13 target_link_libraries(multi_publisher ${catkin_LIBRARIES}  
    )  
14 add_executable(filter_subscriber src/sub.cpp)  
15 target_link_libraries(filter_subscriber ${  
    catkin_LIBRARIES})  
16 add_executable(filter_subscriber_policy src/sub_pol.cpp)  
17 target_link_libraries(filter_subscriber_policy ${  
    catkin_LIBRARIES})
```

7.2 package.xml

```
1 <?xml version="1.0"?>
2 <package format="2">
3   <name>message_filters_example</name>
4   <version>0.0.0</version>
5   <description>The pub_sub package</description>
6
7   <!-- One maintainer tag required, multiple allowed, one
      person per tag -->
8   <!-- Example: -->
9   <!-- <maintainer email="jane.doe@example.com">Jane Doe
      </maintainer> -->
10  <maintainer email="simone@todo.todo">simone</maintainer
      >
11  <!-- One license tag required, multiple allowed, one
      license per tag -->
12  <!-- Commonly used license strings: -->
13  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3,
      LGPLv2.1, LGPLv3 -->
14  <license>TODO</license>
15  <!-- Url tags are optional, but multiple are allowed,
      one per tag -->
16  <!-- Optional attribute type can be: website,
      bugtracker, or repository -->
17  <!-- Example: -->
18  <!-- <url type="website">http://wiki.ros.org/pub_sub</
      url> -->
19  <!-- Author tags are optional, multiple are allowed,
      one per tag -->
20  <!-- Authors do not have to be maintainers, but could
      be -->
21  <!-- Example: -->
22  <!-- <author email="jane.doe@example.com">Jane Doe</
      author> -->
23  <!-- The *depend tags are used to specify dependencies
      -->
24  <!-- Dependencies can be catkin packages or system
      dependencies -->
25  <!-- Examples: -->
26  <!-- Use depend as a shortcut for packages that are
      both build and exec dependencies -->
27  <!--   <depend>roscpp</depend> -->
```

```

28 <!-- Note that this is equivalent to the following:
    -->
29 <!-- <build_depend>roscpp</build_depend> -->
30 <!-- <exec_depend>roscpp</exec_depend> -->
31 <!-- Use build_depend for packages you need at compile
    time: -->
32 <!-- <build_depend>message_generation</build_depend>
    -->
33 <!-- Use build_export_depend for packages you need in
    order to build against this package: -->
34 <!-- <build_export_depend>message_generation</
    build_export_depend> -->
35 <!-- Use buildtool_depend for build tool packages: -->
36 <!-- <buildtool_depend>catkin</buildtool_depend> -->
37 <!-- Use exec_depend for packages you need at runtime:
    -->
38 <!-- <exec_depend>message_runtime</exec_depend> -->
39 <!-- Use test_depend for packages you need only for
    testing: -->
40 <!-- <test_depend>gtest</test_depend> -->
41 <!-- Use doc_depend for packages you need only for
    building documentation: -->
42 <!-- <doc_depend>doxygen</doc_depend> -->
43
44 <buildtool_depend>catkin</buildtool_depend>
45 <build_depend>roscpp</build_depend>
46 <build_depend>std_msgs</build_depend>
47 <build_depend>geometry_msgs</build_depend>
48 <build_depend>message_filters</build_depend>
49
50 <build_export_depend>roscpp</build_export_depend>
51 <build_export_depend>std_msgs</build_export_depend>
52 <build_export_depend>message_filters</
    build_export_depend>
53
54 <exec_depend>roscpp</exec_depend>
55 <exec_depend>std_msgs</exec_depend>
56 <exec_depend>geometry_msgs</exec_depend>
57 <exec_depend>message_filters</exec_depend>
58
59 <!-- The export tag contains other, unspecified, tags
    -->
60 <export>

```



```
61     <!-- Other tools can request additional information
        be placed here -->
62 </export>
63 </package>
```

7.3 src

7.3.1 pub.cpp

```
1 #include "ros/ros.h"
2 #include "geometry_msgs/Vector3Stamped.h"
3 #include <sstream>
4
5 int main(int argc, char **argv){
6     ros::init(argc, argv, "publisher");
7     ros::NodeHandle n;
8     ros::Publisher pub1 = n.advertise<geometry_msgs::
        Vector3Stamped>("topic1", 1000);
9     ros::Publisher pub2 = n.advertise<geometry_msgs::
        Vector3Stamped>("topic2", 1000);
10
11     ros::Rate loop_rate(1);
12     int count = 0;
13
14     while (ros::ok()){
15         geometry_msgs::Vector3Stamped msg1;
16         geometry_msgs::Vector3Stamped msg2;
17
18         msg1.header.stamp = ros::Time::now();
19         msg1.header.frame_id = "f1";
20
21         msg1.vector.x = 1;
22         msg1.vector.y = 1;
23         msg1.vector.z = 1;
24
25         msg2.header.stamp = ros::Time::now();
26         msg2.header.frame_id = "f2";
27
28         msg2.vector.x = 2;
29         msg2.vector.y = 2;
30         msg2.vector.z = 2;
31
```

```
32     pub1.publish(msg1);
33     pub2.publish(msg2);
34
35     ROS_INFO ("Publishing message");
36
37     ros::spinOnce();
38     loop_rate.sleep();
39     ++count;
40 }
41
42 return 0;
43 }
```

7.3.2 sub_pol.cpp

```
1 #include "ros/ros.h"
2 #include "geometry_msgs/Vector3Stamped.h"
3 #include <message_filters/subscriber.h>
4 #include <message_filters/time_synchronizer.h>
5 #include <message_filters/sync_policies/exact_time.h>
6 #include <message_filters/sync_policies/approximate_time.
7     h>
8
9 void callback(const geometry_msgs::Vector3StampedConstPtr
10     & msg1, const geometry_msgs::Vector3StampedConstPtr&
11     msg2)
12 {
13     ROS_INFO ("Received two messages: (%f,%f,%f) and (%f,%f
14         ,%f)", msg1->vector.x, msg1->vector.y, msg1->vector.z,
15         msg2->vector.x, msg2->vector.y, msg2->vector.z);
16 }
17
18 int main(int argc, char** argv)
19 {
20     ros::init(argc, argv, "subscriber_sync");
21     ros::NodeHandle n;
22
23     message_filters::Subscriber<geometry_msgs::
24         Vector3Stamped> sub1(n, "topic1", 1);
25     message_filters::Subscriber<geometry_msgs::
26         Vector3Stamped> sub2(n, "topic2", 1);
27 }
```

```

21 //typedef message_filters::sync_policies::ExactTime<
    geometry_msgs::Vector3Stamped, geometry_msgs::
    Vector3Stamped> MySyncPolicy;
22 typedef message_filters::sync_policies::ApproximateTime
    <geometry_msgs::Vector3Stamped, geometry_msgs::
    Vector3Stamped> MySyncPolicy;
23 message_filters::Synchronizer<MySyncPolicy> sync(
    MySyncPolicy(10), sub1, sub2);
24 sync.registerCallback(boost::bind(&callback, _1, _2));
25 ros::spin();
26
27 return 0;
28 }

```

7.3.3 sub.cpp

```

1 #include "ros/ros.h"
2 #include "geometry_msgs/Vector3Stamped.h"
3 #include <message_filters/subscriber.h>
4 #include <message_filters/time_synchronizer.h>
5
6 void callback(const geometry_msgs::Vector3StampedConstPtr
    & msg1, const geometry_msgs::Vector3StampedConstPtr&
    msg2)
7 {
8     ROS_INFO ("Received two messages: (%f,%f,%f) and (%f,%f
    ,%f)", msg1->vector.x, msg1->vector.y, msg1->vector.z,
    msg2->vector.x, msg2->vector.y, msg2->vector.z);
9 }
10
11 int main(int argc, char** argv)
12 {
13     ros::init(argc, argv, "subscriber");
14     ros::NodeHandle n;
15     message_filters::Subscriber<geometry_msgs::
    Vector3Stamped> sub1(n, "topic1", 1);
16     message_filters::Subscriber<geometry_msgs::
    Vector3Stamped> sub2(n, "topic2", 1);
17     message_filters::TimeSynchronizer<geometry_msgs::
    Vector3Stamped, geometry_msgs::Vector3Stamped> sync(
    sub1, sub2, 10);
18     sync.registerCallback(boost::bind(&callback, _1, _2));

```

```
19
20   ros::spin();
21
22   return 0;
23 }
```

8 tf

8.1 CMakeLists.txt

```
1 cmake_minimum_required(VERSION 2.8.3)
2 project(tf_examples)
3
4 ## Find catkin and any catkin packages
5 find_package(catkin REQUIRED COMPONENTS roscpp std_msgs
6           tf)
7
8 ## Declare a catkin package
9 catkin_package()
10
11 ## Build talker and listener
12 include_directories(include ${catkin_INCLUDE_DIRS})
13 add_executable(tf_pub src/pub.cpp)
14 target_link_libraries(tf_pub ${catkin_LIBRARIES})
15 add_executable(get_tf src/get_tf.cpp)
16 target_link_libraries(get_tf ${catkin_LIBRARIES})
```

8.2 package.xml

```
1 <?xml version="1.0"?>
2 <package format="2">
3   <name>tf_examples</name>
4   <version>0.0.0</version>
5   <description>The pub_sub package</description>
6
7   <!-- One maintainer tag required, multiple allowed, one
8        person per tag -->
9   <!-- Example: -->
10  <!-- <maintainer email="jane.doe@example.com">Jane Doe
11      </maintainer> -->
```

```

10 <maintainer email="simone@todo.todo">simone</maintainer
    >
11 <!-- One license tag required, multiple allowed, one
    license per tag -->
12 <!-- Commonly used license strings: -->
13 <!-- BSD, MIT, Boost Software License, GPLv2, GPLv3,
    LGPLv2.1, LGPLv3 -->
14 <license>TODO</license>
15 <!-- Url tags are optional, but multiple are allowed,
    one per tag -->
16 <!-- Optional attribute type can be: website,
    bugtracker, or repository -->
17 <!-- Example: -->
18 <!-- <url type="website">http://wiki.ros.org/pub_sub</
    url> -->
19 <!-- Author tags are optional, multiple are allowed,
    one per tag -->
20 <!-- Authors do not have to be maintainers, but could
    be -->
21 <!-- Example: -->
22 <!-- <author email="jane.doe@example.com">Jane Doe</
    author> -->
23 <!-- The *depend tags are used to specify dependencies
    -->
24 <!-- Dependencies can be catkin packages or system
    dependencies -->
25 <!-- Examples: -->
26 <!-- Use depend as a shortcut for packages that are
    both build and exec dependencies -->
27 <!-- <depend>roscpp</depend> -->
28 <!-- Note that this is equivalent to the following:
    -->
29 <!-- <build_depend>roscpp</build_depend> -->
30 <!-- <exec_depend>roscpp</exec_depend> -->
31 <!-- Use build_depend for packages you need at compile
    time: -->
32 <!-- <build_depend>message_generation</build_depend>
    -->
33 <!-- Use build_export_depend for packages you need in
    order to build against this package: -->
34 <!-- <build_export_depend>message_generation</
    build_export_depend> -->
35 <!-- Use buildtool_depend for build tool packages: -->

```

```

36 <!-- <buildtool_depend>catkin</buildtool_depend> -->
37 <!-- Use exec_depend for packages you need at runtime:
    -->
38 <!-- <exec_depend>message_runtime</exec_depend> -->
39 <!-- Use test_depend for packages you need only for
    testing: -->
40 <!-- <test_depend>gtest</test_depend> -->
41 <!-- Use doc_depend for packages you need only for
    building documentation: -->
42 <!-- <doc_depend>doxygen</doc_depend> -->
43 <buildtool_depend>catkin</buildtool_depend>
44 <build_depend>roscpp</build_depend>
45 <build_export_depend>roscpp</build_export_depend>
46 <build_export_depend>std_msgs</build_export_depend>
47 <exec_depend>roscpp</exec_depend>
48 <exec_depend>std_msgs</exec_depend>
49 <!-- The export tag contains other, unspecified, tags
    -->
50
51 <export>
52 <!-- Other tools can request additional information
    be placed here -->
53
54 </export>
55 </package>

```

8.3 launch

8.3.1 turtle.launch

```

1 <launch>
2 <node pkg="tf_examples" type = "tf_pub" name = "tf_pub"/>
3 <node pkg="turtlesim" type = "turtlesim_node" name = "
  turtlesim_node"/>
4 <node pkg="turtlesim" type = "turtle_teleop_key" name = "
  turtle_teleop_key"/>
5 <node pkg="tf2_ros" type="static_transform_publisher"
  name="back_right" args="0.3 -0.3 0 0 0 0 1 turtle
  FRleg" />
6 <node pkg="tf2_ros" type="static_transform_publisher"
  name="front_right" args="0.3 0.3 0 0 0 0 1 turtle
  FLleg" />

```

```

7 <node pkg="tf2_ros" type="static_transform_publisher"
  name="front_left" args="-0.3 0.3 0 0 0 0 1 turtle
    BLleg" />
8 <node pkg="tf2_ros" type="static_transform_publisher"
  name="back_left" args="-0.3 -0.3 0 0 0 0 1 turtle
    BRleg" />
9 </launch>

```

8.3.2 turtle.launch.old

```

1 <launch>
2 <node pkg="tf_examples" type = "tf_pub" name = "tf_pub"/>
3 <node pkg="turtlesim" type = "turtlesim_node" name = "
  turtlesim_node"/>
4 <node pkg="turtlesim" type = "turtle_teleop_key" name = "
  turtle_teleop_key"/>
5 <node pkg="tf" type="static_transform_publisher" name="
  back_right" args="0.3 -0.3 0 0 0 0 1 turtle FRleg 100"
  />
6 <node pkg="tf" type="static_transform_publisher" name="
  front_right" args="0.3 0.3 0 0 0 0 1 turtle FLleg 100"
  />
7 <node pkg="tf" type="static_transform_publisher" name="
  front_left" args="-0.3 0.3 0 0 0 0 1 turtle BLleg 100"
  />
8 <node pkg="tf" type="static_transform_publisher" name="
  back_left" args="-0.3 -0.3 0 0 0 0 1 turtle BRleg 100"
  />
9 </launch>

```

8.4 src

8.4.1 get_tf.cpp

```

1 #include <ros/ros.h>
2 #include <tf/transform_listener.h>
3 #include <geometry_msgs/TransformStamped.h>
4
5 int main(int argc, char** argv){
6     // Initialize the ROS node
7     ros::init(argc, argv, "world_to_frleg_listener");

```

```
8 // Create a ROS node handle
9 ros::NodeHandle node;
10 // Create a TransformListener object that will listen
    to tf data
11 tf::TransformListener listener;
12
13 // Set the rate at which we want to check the
    transformation
14 ros::Rate rate(10.0);
15
16 while (node.ok()){
17     tf::StampedTransform transform;
18
19     try{
20         // Wait for up to 1 second for the transform to
            become available
21         listener.waitForTransform("/world", "/FRleg", ros::
            Time(0), ros::Duration(1.0));
22         // Look up the transformation from "world" to "
            FRleg"
23         listener.lookupTransform("/world", "/FRleg", ros::
            Time(0), transform);
24     }
25     catch (tf::TransformException &ex) {
26         // If there is an exception print the error message
27         ROS_ERROR("%s",ex.what());
28         ros::Duration(1.0).sleep();
29         continue;
30     }
31
32     // Print transformation (only pose, but you can get
        the orientation)
33     ROS_INFO("Translation: x=%f, y=%f, z=%f",
34             transform.getOrigin().x(),
35             transform.getOrigin().y(),
36             transform.getOrigin().z());
37
38     // Sleep
39     rate.sleep();
40 }
41
42 return 0;
43 }
```


8.4.2 pub.cpp

```
1 #include "ros/ros.h"
2 #include "turtlesim/Pose.h"
3 #include <tf/transform_broadcaster.h>
4
5 class TfSubPub {
6 public:
7     TfSubPub() {
8         // Subscribe to the topic and bind the callback
9         // method.
10        sub = n.subscribe("/turtle1/pose", 1000, &
11        TfSubPub::callback, this);
12    }
13
14    void callback(const turtlesim::Pose::ConstPtr& msg) {
15        // Update the transform's origin with the new
16        // pose
17        transform.setOrigin(tf::Vector3(msg->x, msg->y,
18        0));
19
20        // Update the quaternion based on the new theta
21        // value
22        q.setRPY(0, 0, msg->theta);
23        transform.setRotation(q);
24
25        // Publish the updated transform
26        br.sendTransform(tf::StampedTransform(transform,
27        ros::Time::now(), "world", "turtle"));
28    }
29
30 private:
31     ros::NodeHandle n;
32     tf::TransformBroadcaster br;
33     ros::Subscriber sub;
34     // Declare transform and quaternion as class members
35     // to reuse them in each callback
36     tf::Transform transform;
37     tf::Quaternion q;
38 };
39
40 int main(int argc, char **argv) {
41     ros::init(argc, argv, "subscribe_and_publish");
```

```
35     TfSubPub myTfSubPub;  
36     ros::spin();  
37  
38     return 0;  
39 }
```

9 fibonacci

9.1 CMakeLists.txt

```
1  cmake_minimum_required(VERSION 2.8.3)  
2  project(actionlib_tutorials)  
3  
4  ## Compile as C++11, supported in ROS Kinetic and newer  
5  add_compile_options(-std=c++11)  
6  
7  ## Find catkin macros and libraries  
8  ## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)  
9  ## is used, also find other catkin packages  
10 find_package(catkin REQUIRED COMPONENTS  
11   actionlib  
12   actionlib_msgs  
13   roscpp  
14 )  
15  
16  
17 ## Generate actions in the 'action' folder  
18 add_action_files(  
19   DIRECTORY action  
20   FILES Fibonacci.action  
21 )  
22  
23 ## Generate added messages and services with any  
24   dependencies listed here  
24 generate_messages(  
25   DEPENDENCIES actionlib_msgs std_msgs # Or other  
26     packages containing msgs  
27 )  
28 #
```

```
29 catkin_package(  
30     CATKIN_DEPENDS actionlib_msgs  
31 )  
32  
33 #####  
34 ## Build ##  
35 #####  
36  
37 ## Specify additional locations of header files  
38 ## Your package locations should be listed before other  
   locations  
39 include_directories(  
40     # include  
41     ${catkin_INCLUDE_DIRS}  
42 )  
43  
44  
45  
46 add_executable(fibonacci_server src/fibonacci_server.cpp)  
47  
48 target_link_libraries(  
49     fibonacci_server  
50     ${catkin_LIBRARIES}  
51 )  
52  
53 add_dependencies(  
54     fibonacci_server  
55     ${actionlib_tutorials_EXPORTED_TARGETS}  
56 )  
57  
58  
59  
60 add_executable(fibonacci_client src/fibonacci_client.cpp)  
61  
62 target_link_libraries(  
63     fibonacci_client  
64     ${catkin_LIBRARIES}  
65 )  
66  
67 add_dependencies(  
68     fibonacci_client  
69     ${actionlib_tutorials_EXPORTED_TARGETS}  
70 )
```

```
71
72 add_executable(fibonacci_client2 src/fibonacci_client2.
    cpp)
73
74 target_link_libraries(
75     fibonacci_client2
76     ${catkin_LIBRARIES}
77 )
78
79 add_dependencies(
80     fibonacci_client2
81     ${actionlib_tutorials_EXPORTED_TARGETS}
82 )
```

9.2 package.xml

```
1 <?xml version="1.0"?>
2 <package format="2">
3   <name>actionlib_tutorials</name>
4   <version>0.0.0</version>
5   <description>The fibonacci package</description>
6
7   <!-- One maintainer tag required, multiple allowed, one
      person per tag -->
8   <!-- Example:  -->
9   <!-- <maintainer email="jane.doe@example.com">Jane Doe
      </maintainer> -->
10  <maintainer email="simone@todo.todo">simone</maintainer
      >
11
12
13  <!-- One license tag required, multiple allowed, one
      license per tag -->
14  <!-- Commonly used license strings: -->
15  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3,
      LGPLv2.1, LGPLv3 -->
16  <license>TODO</license>
17
18
19  <!-- Url tags are optional, but multiple are allowed,
      one per tag -->
```

```
20 <!-- Optional attribute type can be: website,  
    bugtracker, or repository -->  
21 <!-- Example: -->  
22 <!-- <url type="website">http://wiki.ros.org/  
    fibonacci_server</url> -->  
23  
24  
25 <!-- Author tags are optional, multiple are allowed,  
    one per tag -->  
26 <!-- Authors do not have to be maintainers, but could  
    be -->  
27 <!-- Example: -->  
28 <!-- <author email="jane.doe@example.com">Jane Doe</  
    author> -->  
29  
30  
31 <!-- The *depend tags are used to specify dependencies  
    -->  
32 <!-- Dependencies can be catkin packages or system  
    dependencies -->  
33 <!-- Examples: -->  
34 <!-- Use depend as a shortcut for packages that are  
    both build and exec dependencies -->  
35 <!-- <depend>roscpp</depend> -->  
36 <!-- Note that this is equivalent to the following:  
    -->  
37 <!-- <build_depend>roscpp</build_depend> -->  
38 <!-- <exec_depend>roscpp</exec_depend> -->  
39 <!-- Use build_depend for packages you need at compile  
    time: -->  
40 <!-- <build_depend>message_generation</build_depend>  
    -->  
41 <!-- Use build_export_depend for packages you need in  
    order to build against this package: -->  
42 <!-- <build_export_depend>message_generation</  
    build_export_depend> -->  
43 <!-- Use buildtool_depend for build tool packages: -->  
44 <!-- <buildtool_depend>catkin</buildtool_depend> -->  
45 <!-- Use exec_depend for packages you need at runtime:  
    -->  
46 <!-- <exec_depend>message_runtime</exec_depend> -->  
47 <!-- Use test_depend for packages you need only for  
    testing: -->
```

```
48 <!-- <test_depend>gtest</test_depend> -->
49 <!-- Use doc_depend for packages you need only for
      building documentation: -->
50 <!-- <doc_depend>doxygen</doc_depend> -->
51 <buildtool_depend>catkin</buildtool_depend>
52 <build_depend>actionlib</build_depend>
53 <build_depend>actionlib_msgs</build_depend>
54 <build_depend>roscpp</build_depend>
55 <build_export_depend>actionlib</build_export_depend>
56 <build_export_depend>actionlib_msgs</
      build_export_depend>
57 <build_export_depend>roscpp</build_export_depend>
58 <exec_depend>actionlib</exec_depend>
59 <exec_depend>actionlib_msgs</exec_depend>
60 <exec_depend>roscpp</exec_depend>
61 <exec_depend>message_generation</exec_depend>
62
63
64 <!-- The export tag contains other, unspecified, tags
      -->
65 <export>
66   <!-- Other tools can request additional information
        be placed here -->
67
68 </export>
69 </package>
```

9.3 action

9.3.1 Fibonacci.action

```
1 #goal definition
2 int32 order
3 ---
4 #result definition
5 int32[] sequence
6 ---
7 #feedback
8 int32[] sequence
```

9.4 launch

9.4.1 launcher.launch

```
1 <launch>
2 <node name="fibonacci_client" pkg="actionlib_tutorials"
   type="fibonacci_client" output="screen"/>
3
4 <param name="order" value="10" type="int"/>
5 <param name="duration" value="15" type="double"/>
6 </launch>
```

9.4.2 launcher2.launch

```
1 <launch>
2 <node name="fibonacci_client" pkg="actionlib_tutorials"
   type="fibonacci_client2" output="screen"/>
3
4 <param name="order" value="10" type="int"/>
5 <param name="duration" value="10" type="double"/>
6 </launch>
```

9.5 src

9.5.1 fibonacci_client.cpp

```
1 #include <ros/ros.h>
2 #include <actionlib/client/simple_action_client.h>
3 #include <actionlib/client/terminal_state.h>
4 #include <actionlib_tutorials/FibonacciAction.h>
5
6 int main (int argc, char **argv)
7 {
8     ros::init(argc, argv, "test_fibonacci");
9
10    // create the action client
11    actionlib::SimpleActionClient<actionlib_tutorials::
        FibonacciAction> ac("fibonacci", true);
12
13    ROS_INFO("Waiting for action server to start.");
14    // wait for the action server to start
```

```
15  ac.waitForServer(); //will wait for infinite time
16
17  ROS_INFO("Action server started, sending goal.");
18  // send a goal to the action
19  actionlib_tutorials::FibonacciGoal goal;
20  int order =10;
21  double duration =1.0;
22  ros::param::get("order",order);
23  ros::param::get("duration",duration);
24  goal.order = order;
25  ac.sendGoal(goal);
26
27  //wait for the action to return
28  bool finished_before_timeout = ac.waitForResult(ros::
    Duration(duration));
29
30
31  if (finished_before_timeout)
32  {
33      actionlib::SimpleClientGoalState state = ac.getState
34          ();
35      ROS_INFO("Action finished: %s",state.toString().c_str
36          ());
37
38      actionlib_tutorials::FibonacciResultConstPtr result
39          = ac.getResult();
40      for (int i=0; i<result->sequence.size();i++){
41          ROS_INFO ("%d ", result->sequence[i]);
42      }
43  }
44  else{
45      ROS_INFO("Action did not finish before the time out."
46          );
47      //ac.cancelGoal ();
48  }
49
50  //exit
51  return 0;
52 }
```


9.5.2 fibonacci_client2.cpp

```
1 #include <ros/ros.h>
2 #include <actionlib/client/simple_action_client.h>
3 #include <actionlib/client/terminal_state.h>
4 #include <actionlib_tutorials/FibonacciAction.h>
5
6 typedef actionlib::SimpleActionClient<actionlib_tutorials
   ::FibonacciAction> Client;
7
8 void doneCb(const actionlib::SimpleClientGoalState& state
   ,
9             const actionlib_tutorials::
10              FibonacciResultConstPtr& result) {
11     ROS_INFO("Finished in state [%s]", state.toString().
12             c_str());
13     std::stringstream ss;
14     for (auto value : result->sequence) {
15         ss << value << " ";
16     }
17     ROS_INFO("Result: %s", ss.str().c_str());
18 }
19
20 void activeCb() {
21     ROS_INFO("Goal just went active");
22 }
23
24 void feedbackCb(const actionlib_tutorials::
25                 FibonacciFeedbackConstPtr& feedback) {
26     ROS_INFO("Got Feedback of length %lu", feedback->
27             sequence.size());
28 }
29
30 void preemptTimerCallback(const ros::TimerEvent&, Client*
31                           client) {
32     if (client->getState() == actionlib::
33         SimpleClientGoalState::ACTIVE ||
34         client->getState() == actionlib::
35         SimpleClientGoalState::PENDING) {
36         ROS_INFO("Preempting the current goal due to
37                 timeout.");
38         client->cancelGoal();
39     }
40 }
```

```
32 }
33
34 int main (int argc, char **argv) {
35     ros::init(argc, argv, "test_fibonacci");
36     ros::NodeHandle nh;
37
38     Client client("fibonacci", true);
39     ROS_INFO("Waiting for action server to start.");
40     client.waitForServer();
41
42     ROS_INFO("Action server started, sending goal.");
43     actionlib_tutorials::FibonacciGoal goal;
44     int order = 10; // Default order
45     double duration = 5.0; // Default duration in seconds
46     nh.param("order", order, 10); // Retrieve order if
        specified in parameters
47     nh.param("duration", duration, 5.0); // Retrieve
        duration if specified in parameters
48
49     goal.order = order;
50     client.sendGoal(goal, &doneCb, &activeCb, &feedbackCb
        );
51
52     // Setup a timer to preempt the goal after the
        specified duration
53     ros::Timer timer = nh.createTimer(ros::Duration(
        duration), boost::bind(preemptTimerCallback, _1, &
        client), true);
54
55     ros::Rate loop_rate(1);
56
57     while (ros::ok()){
58         ROS_INFO("doing other processing");
59         ros::spinOnce();
60
61         loop_rate.sleep();
62     }
63
64     return 0;
65 }
```

9.5.3 fibonacci_server.cpp

```
1 #include <ros/ros.h>
2 #include <actionlib/server/simple_action_server.h>
3 #include <actionlib_tutorials/FibonacciAction.h>
4
5 class FibonacciAction
6 {
7 private:
8     ros::NodeHandle nh_;
9     actionlib::SimpleActionServer<actionlib_tutorials::
10         FibonacciAction> as_;
11     std::string action_name_;
12     // create messages that are used to published feedback/
13     result
14     actionlib_tutorials::FibonacciFeedback feedback_;
15     actionlib_tutorials::FibonacciResult result_;
16
17 public:
18     FibonacciAction(std::string name) :
19         as_(nh_, name, boost::bind(&FibonacciAction::
20             executeCB, this, _1), false),
21         action_name_(name)
22     {
23         as_.start();
24     }
25
26     ~FibonacciAction(void)
27     {
28     }
29
30     void executeCB(const actionlib_tutorials::
31         FibonacciGoalConstPtr &goal)
32     {
33         // helper variables
34         ros::Rate r(1); //simulate compute time
35         bool success = true;
36
37         // clear and set first two values
38         feedback_.sequence.clear();
39         feedback_.sequence.push_back(0);
40         feedback_.sequence.push_back(1);
```

```
38
39 // publish info to the console for the user
40 ROS_INFO("%s: Executing, creating fibonacci sequence
    of order %i with seeds %i, %i", action_name_.c_str
    (), goal->order, feedback_.sequence[0], feedback_.
    sequence[1]);
41
42 // start executing the action
43 for(int i=1; i<=goal->order; i++)
44 {
45     // check that preempt has not been requested by the
        client
46     if (as_.isPreemptRequested() || !ros::ok())
47     {
48         ROS_INFO("%s: Preempted", action_name_.c_str());
49         // set the action state to preempted
50         as_.setPreempted();
51         success = false;
52         break;
53     }
54     feedback_.sequence.push_back(feedback_.sequence[i]
        + feedback_.sequence[i-1]);
55     // publish the feedback
56     as_.publishFeedback(feedback_);
57     // this sleep is not necessary, we simulate compute
        time
58     r.sleep();
59 }
60
61 if(success)
62 {
63     result_.sequence = feedback_.sequence;
64     ROS_INFO("%s: Succeeded", action_name_.c_str());
65     // set the action state to succeeded
66     as_.setSucceeded(result_);
67 }
68 }
69
70
71 };
72
73
74 int main(int argc, char** argv)
```

```
75 {  
76   ros::init(argc, argv, "fibonacci");  
77  
78   FibonacciAction fibonacci("fibonacci");  
79   ros::spin();  
80  
81   return 0;  
82 }
```

10 pub_latched

10.1 CMakeLists.txt

```
1 cmake_minimum_required(VERSION 2.8.3)
2 project(pub_latched)
3
4 ## Find catkin and any catkin packages
5 find_package(catkin REQUIRED COMPONENTS roscpp std_msgs)
6
7
8 ## Declare a catkin package
9 catkin_package()
10
11 ## Build talker and listener
12 include_directories(include ${catkin_INCLUDE_DIRS})
13
14 add_executable(pub_latched src/pub.cpp)
15 target_link_libraries(pub_latched ${catkin_LIBRARIES})
```

10.2 package.xml

```
1 <?xml version="1.0"?>
2 <package format="2">
3   <name>pub_latched</name>
4   <version>0.0.0</version>
5   <description>The latched package</description>
6
7   <!-- One maintainer tag required, multiple allowed, one
8        person per tag -->
9   <!-- Example:  -->
10  <!-- <maintainer email="jane.doe@example.com">Jane Doe
11        </maintainer> -->
12  <maintainer email="simone@todo.todo">simone</maintainer>
13
14  <!-- One license tag required, multiple allowed, one
15        license per tag -->
16  <!-- Commonly used license strings: -->
17  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3,
18        LGPLv2.1, LGPLv3 -->
```

```
15 <license>TODO</license>
16
17 <!-- Url tags are optional, but multiple are allowed,
    one per tag -->
18 <!-- Optional attribute type can be: website,
    bugtracker, or repository -->
19 <!-- Example: -->
20 <!-- <url type="website">http://wiki.ros.org/pub_sub</
    url> -->
21
22 <!-- Author tags are optional, multiple are allowed,
    one per tag -->
23 <!-- Authors do not have to be maintainers, but could
    be -->
24 <!-- Example: -->
25 <!-- <author email="jane.doe@example.com">Jane Doe</
    author> -->
26
27 <!-- The *depend tags are used to specify dependencies
    -->
28 <!-- Dependencies can be catkin packages or system
    dependencies -->
29 <!-- Examples: -->
30 <!-- Use depend as a shortcut for packages that are
    both build and exec dependencies -->
31 <!-- <depend>roscpp</depend> -->
32 <!-- Note that this is equivalent to the following:
    -->
33 <!-- <build_depend>roscpp</build_depend> -->
34 <!-- <exec_depend>roscpp</exec_depend> -->
35 <!-- Use build_depend for packages you need at compile
    time: -->
36 <!-- <build_depend>message_generation</build_depend>
    -->
37 <!-- Use build_export_depend for packages you need in
    order to build against this package: -->
38 <!-- <build_export_depend>message_generation</
    build_export_depend> -->
39 <!-- Use buildtool_depend for build tool packages: -->
40 <!-- <buildtool_depend>catkin</buildtool_depend> -->
41 <!-- Use exec_depend for packages you need at runtime:
    -->
42 <!-- <exec_depend>message_runtime</exec_depend> -->
```

```
43 <!-- Use test_depend for packages you need only for
    testing: -->
44 <!-- <test_depend>gtest</test_depend> -->
45 <!-- Use doc_depend for packages you need only for
    building documentation: -->
46 <!-- <doc_depend>doxygen</doc_depend> -->
47 <buildtool_depend>catkin</buildtool_depend>
48 <build_depend>roscpp</build_depend>
49 <build_export_depend>roscpp</build_export_depend>
50 <build_export_depend>std_msgs</build_export_depend>
51 <exec_depend>roscpp</exec_depend>
52 <exec_depend>std_msgs</exec_depend>
53
54
55
56 <!-- The export tag contains other, unspecified, tags
    -->
57 <export>
58   <!-- Other tools can request additional information
        be placed here -->
59
60 </export>
61 </package>
```

10.3 src

10.3.1 pub.cpp

```
1 #include "ros/ros.h"
2 #include "std_msgs/String.h"
3 #include <sstream>
4
5 int main(int argc, char **argv){
6
7     ros::init(argc, argv, "talker");
8     ros::NodeHandle n;
9
10    ros::Publisher chatter_pub = n.advertise<std_msgs::
        String>("chatter", 1, true);
11
12    ros::Rate loop_rate(0.05);
13
```



```
14     int count = 0;
15
16     while (ros::ok()){
17
18         std_msgs::String msg;
19
20         std::stringstream ss;
21         ss << "hello world " << count;
22         msg.data = ss.str();
23
24         ROS_INFO("%s", msg.data.c_str());
25
26         chatter_pub.publish(msg);
27
28         ros::spinOnce();
29
30         loop_rate.sleep();
31         ++count;
32     }
33
34     return 0;
35 }
```

10.4 launch

10.4.1 launcher.launch

```
1 <launch>
2   <node pkg="tf2_ros" type="static_transform_publisher"
3     name="example_tf" args="1 1 1 0 0 0 1 odom base_link" />
4 </launch>
```

11 asynch

11.1 CMakeLists.txt

```
1 cmake_minimum_required(VERSION 2.8.3)
2 project(sub_asynch)
3
```

```
4 ## Find catkin and any catkin packages
5 find_package(catkin REQUIRED COMPONENTS roscpp std_msgs)
6
7 ## Declare a catkin package
8 catkin_package()
9
10 ## Build talker and listener
11 include_directories(include ${catkin_INCLUDE_DIRS})
12
13 add_executable(standard_pub src/pub.cpp)
14 target_link_libraries(standard_pub ${catkin_LIBRARIES})
15
16 add_executable(standard_sub src/standard_sub.cpp)
17 target_link_libraries(standard_sub ${catkin_LIBRARIES})
18
19 add_executable(asynch_sub src/asynch_sub.cpp)
20 target_link_libraries(asynch_sub ${catkin_LIBRARIES})
```

11.2 package.xml

```
1 <?xml version="1.0"?>
2 <package format="2">
3   <name>sub_asynch</name>
4   <version>0.0.0</version>
5   <description>The asynch package</description>
6
7   <!-- One maintainer tag required, multiple allowed, one
8        person per tag -->
9   <!-- Example:  -->
10  <!-- <maintainer email="jane.doe@example.com">Jane Doe
11        </maintainer> -->
12  <maintainer email="simone@todo.todo">simone</maintainer>
13
14  <!-- One license tag required, multiple allowed, one
15        license per tag -->
16  <!-- Commonly used license strings: -->
17  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3,
18        LGPLv2.1, LGPLv3 -->
19  <license>TODO</license>
```

```
17 <!-- Url tags are optional, but multiple are allowed,  
    one per tag -->  
18 <!-- Optional attribute type can be: website,  
    bugtracker, or repository -->  
19 <!-- Example: -->  
20 <!-- <url type="website">http://wiki.ros.org/pub_sub</  
    url> -->  
21  
22  
23 <!-- Author tags are optional, multiple are allowed,  
    one per tag -->  
24 <!-- Authors do not have to be maintainers, but could  
    be -->  
25 <!-- Example: -->  
26 <!-- <author email="jane.doe@example.com">Jane Doe</  
    author> -->  
27  
28 <!-- The *depend tags are used to specify dependencies  
    -->  
29 <!-- Dependencies can be catkin packages or system  
    dependencies -->  
30 <!-- Examples: -->  
31 <!-- Use depend as a shortcut for packages that are  
    both build and exec dependencies -->  
32 <!-- <depend>roscpp</depend> -->  
33 <!-- Note that this is equivalent to the following:  
    -->  
34 <!-- <build_depend>roscpp</build_depend> -->  
35 <!-- <exec_depend>roscpp</exec_depend> -->  
36 <!-- Use build_depend for packages you need at compile  
    time: -->  
37 <!-- <build_depend>message_generation</build_depend>  
    -->  
38 <!-- Use build_export_depend for packages you need in  
    order to build against this package: -->  
39 <!-- <build_export_depend>message_generation</  
    build_export_depend> -->  
40 <!-- Use buildtool_depend for build tool packages: -->  
41 <!-- <buildtool_depend>catkin</buildtool_depend> -->  
42 <!-- Use exec_depend for packages you need at runtime:  
    -->  
43 <!-- <exec_depend>message_runtime</exec_depend> -->
```

```

44 <!-- Use test_depend for packages you need only for
    testing: -->
45 <!-- <test_depend>gtest</test_depend> -->
46 <!-- Use doc_depend for packages you need only for
    building documentation: -->
47 <!-- <doc_depend>doxygen</doc_depend> -->
48 <buildtool_depend>catkin</buildtool_depend>
49 <build_depend>roscpp</build_depend>
50 <build_depend>std_msgs</build_depend>
51 <build_export_depend>roscpp</build_export_depend>
52 <build_export_depend>std_msgs</build_export_depend>
53 <exec_depend>roscpp</exec_depend>
54 <exec_depend>std_msgs</exec_depend>
55
56
57 <!-- The export tag contains other, unspecified, tags
    -->
58 <export>
59 <!-- Other tools can request additional information
    be placed here -->
60
61 </export>
62 </package>

```

11.3 src

11.3.1 pub.cpp

```

1 #include "ros/ros.h"
2 #include "std_msgs/String.h"
3 #include <sstream>
4
5 int main(int argc, char **argv){
6     ros::init(argc, argv, "talker");
7     ros::NodeHandle n;
8     ros::Publisher chatter_pub1 = n.advertise<std_msgs::
    String>("talker1", 1);
9     ros::Publisher chatter_pub2 = n.advertise<std_msgs::
    String>("talker2", 1);
10    ros::Rate loop_rate(3.0);
11
12    std_msgs::String msg1;

```

```
13     std_msgs::String msg2;
14
15     int count = 0;
16
17     while (ros::ok()){
18
19         std::stringstream ss1;
20
21         std::stringstream ss2;
22
23         ss1 << "Hey 1:" << count;
24         msg1.data = ss1.str();
25
26         ss2 << "Hey 2:" << count;
27         msg2.data = ss2.str();
28
29         count++;
30
31         chatter_pub1.publish(msg1);
32         chatter_pub2.publish(msg2);
33
34         ros::spinOnce();
35
36         loop_rate.sleep();
37     }
38
39     return 0;
40 }
```

11.3.2 standard_sub.cpp

```
1 #include <ros/ros.h>
2 #include <std_msgs/String.h>
3 void callbackTalker1(const std_msgs::String::ConstPtr &
4     msg)
5 {
6     ROS_INFO_STREAM("Message from callback 1:" );
7     ros::Duration(2.0).sleep();
8     ROS_INFO("%s", msg->data.c_str());
9 }
```

```
10 void callbackTalker2(const std_msgs::String::ConstPtr &
    msg)
11 {
12     ROS_INFO_STREAM("Message from callback 2:");
13     ROS_INFO("%s", msg->data.c_str());
14 }
15
16 int main(int argc, char **argv)
17 {
18     ros::init(argc, argv, "talker_subscribers");
19     ros::NodeHandle nh;
20     ros::Subscriber counter1_sub = nh.subscribe("talker1"
        , 1, callbackTalker1);
21     ros::Subscriber counter2_sub = nh.subscribe("talker2"
        , 1, callbackTalker2);
22     ros::spin();
23 }
```

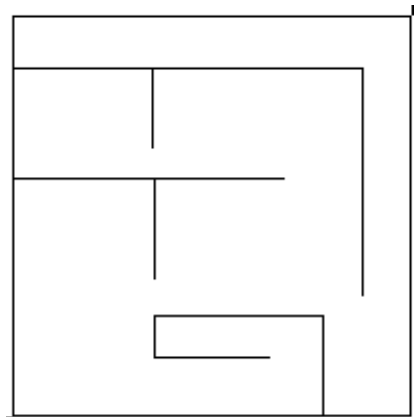
11.3.3 asynch_sub.cpp

```
1 #include <ros/ros.h>
2 #include <std_msgs/String.h>
3
4 void callbackTalker1(const std_msgs::String::ConstPtr &
    msg)
5 {
6     ROS_INFO_STREAM("Message from callback 1:" );
7     ros::Duration(2.0).sleep();
8     ROS_INFO("%s", msg->data.c_str());
9 }
10
11 void callbackTalker2(const std_msgs::String::ConstPtr &
    msg)
12 {
13     ROS_INFO_STREAM("Message from callback 2:");
14     ROS_INFO("%s", msg->data.c_str());
15 }
16
17 int main(int argc, char **argv)
18 {
19     ros::init(argc, argv, "talker_subscribers_asynch");
20     ros::NodeHandle nh;
```

```
21
22     ros::AsyncSpinner spinner(0); // num of thread
23     spinner.start();
24
25     ros::Subscriber counter1_sub = nh.subscribe("talker1"
26         , 1, callbackTalker1);
27     ros::Subscriber counter2_sub = nh.subscribe("talker2"
28         , 1, callbackTalker2);
29     ros::waitForShutdown();
30 }
```

12 stage

12.1 maze.png



12.2 maze.world

```
1 include "turtlebot.inc"
2
3 define floorplan model
4 (
5     # sombre, sensible, artistic
6     color "gray30"
7
8     # most maps will need a bounding box
9     boundary 1
10
11     gui_nose 0
```

```
12  gui_grid 0
13  gui_outline 0
14  gripper_return 0
15  fiducial_return 0
16  ranger_return 1
17 )
18
19 resolution 0.02
20 interval_sim 100 # simulation timestep in milliseconds
21
22 window
23 (
24   size [ 600.0 700.0 ]
25   center [ 0.0 0.0 ]
26   rotate [ 0.0 0.0 ]
27   scale 60
28 )
29
30 floorplan
31 (
32   name "maze"
33   bitmap "maze.png"
34   size [ 10.0 10.0 2.0 ]
35   pose [ 5.0 5.0 0.0 0.0 ]
36 )
37
38 # throw in a robot
39 turtlebot
40 (
41   pose [ 2.0 2.0 0.0 0.0 ]
42   name "turtlebot"
43   color "black"
44 )
```

12.3 turtlebot.inc

```
1 define kinect ranger
2 (
3   sensor
4   (
5     pose [ -0.1 0.0 -0.11 0.0 ]
6     size [0.1 0.1 0.1 ]
```



```
7     range [0 6.5]
8     fov 120.0
9     samples 640
10 )
11 # generic model properties
12 color "black"
13 size [ 0.06 0.15 0.03 ]
14 )
15
16 define turtlebot position
17 (
18     pose [ 0.0 0.0 0.0 0.0 ]
19
20     localization "odom"
21
22
23     odom_error [0.2 0.2 0.0 0.3 ]
24
25     size [ 0.2552 0.2552 0.40 ]
26     origin [ 0.0 0.0 0.0 0.0 ]
27     gui_nose 1
28     drive "diff"
29     color "grey"
30
31     kinect()
32 )
```

13 demo_mapping

13.1 CMakeLists.txt

```
1 cmake_minimum_required(VERSION 2.8.3)
2 project(demo_mapping)
3
4 ## Find catkin and any catkin packages
5 find_package(catkin REQUIRED COMPONENTS roscpp std_msgs)
6
7 ## Declare a catkin package
8 catkin_package()
```

13.2 package.xml

```
1 <?xml version="1.0"?>
2 <package format="2">
3   <name>demo_mapping</name>
4   <version>0.0.0</version>
5   <description>The demo_mapping package</description>
6
7   <!-- One maintainer tag required, multiple allowed, one
      person per tag -->
8   <!-- Example: -->
9   <!-- <maintainer email="jane.doe@example.com">Jane Doe
      </maintainer> -->
10  <maintainer email="simone@todo.todo">simone</maintainer>
      >
11
12
13  <!-- One license tag required, multiple allowed, one
      license per tag -->
14  <!-- Commonly used license strings: -->
15  <!-- BSD, MIT, Boost Software License, GPLv2, GPLv3,
      LGPLv2.1, LGPLv3 -->
16  <license>TODO</license>
17
18
19  <!-- Url tags are optional, but multiple are allowed,
      one per tag -->
20  <!-- Optional attribute type can be: website,
      bugtracker, or repository -->
21  <!-- Example: -->
22  <!-- <url type="website">http://wiki.ros.org/pub_sub</
      url> -->
23
24
25  <!-- Author tags are optional, multiple are allowed,
      one per tag -->
26  <!-- Authors do not have to be maintainers, but could
      be -->
27  <!-- Example: -->
28  <!-- <author email="jane.doe@example.com">Jane Doe</
      author> -->
29
30
```

```
31 <!-- The *depend tags are used to specify dependencies
    -->
32 <!-- Dependencies can be catkin packages or system
    dependencies -->
33 <!-- Examples: -->
34 <!-- Use depend as a shortcut for packages that are
    both build and exec dependencies -->
35 <!-- <depend>roscpp</depend> -->
36 <!-- Note that this is equivalent to the following:
    -->
37 <!-- <build_depend>roscpp</build_depend> -->
38 <!-- <exec_depend>roscpp</exec_depend> -->
39 <!-- Use build_depend for packages you need at compile
    time: -->
40 <!-- <build_depend>message_generation</build_depend>
    -->
41 <!-- Use build_export_depend for packages you need in
    order to build against this package: -->
42 <!-- <build_export_depend>message_generation</
    build_export_depend> -->
43 <!-- Use buildtool_depend for build tool packages: -->
44 <!-- <buildtool_depend>catkin</buildtool_depend> -->
45 <!-- Use exec_depend for packages you need at runtime:
    -->
46 <!-- <exec_depend>message_runtime</exec_depend> -->
47 <!-- Use test_depend for packages you need only for
    testing: -->
48 <!-- <test_depend>gtest</test_depend> -->
49 <!-- Use doc_depend for packages you need only for
    building documentation: -->
50 <!-- <doc_depend>doxygen</doc_depend> -->
51 <buildtool_depend>catkin</buildtool_depend>
52 <build_depend>roscpp</build_depend>
53 <build_depend>std_msgs</build_depend>
54 <build_export_depend>roscpp</build_export_depend>
55 <build_export_depend>std_msgs</build_export_depend>
56 <exec_depend>roscpp</exec_depend>
57 <exec_depend>std_msgs</exec_depend>
58
59
60 <!-- The export tag contains other, unspecified, tags
    -->
61 <export>
```

```
62     <!-- Other tools can request additional information
        be placed here -->
63
64     </export>
65 </package>
```

13.3 launch

13.3.1 gmapping.launch

```
1 <launch>
2   <node pkg="gmapping" type="slam_gmapping" name="
      gmapping" output="screen">
3     <remap from="scan" to="base_scan" />
4   </node>
5 </launch>
```

13.3.2 slam_toolbox.launch

```
1 <launch>
2   <node pkg="slam_toolbox" type="async_slam_toolbox_node"
      name="slam_toolbox" output="screen">
3     <rosparam command="load" file="$(find slam_toolbox)/
      config/mapper_params_online_async.yaml" />
4     <remap from="scan" to="base_scan" />
5   </node>
6 </launch>
```

14 nav2d_conf

14.1 CMakeLists.txt

```
1 cmake_minimum_required(VERSION 2.8.3)
2 project(nav2d_conf)
3
4 ## Compile as C++11, supported in ROS Kinetic and newer
5 # add_compile_options(-std=c++11)
6
7 ## Find catkin macros and libraries
```

```
8 ## if COMPONENTS list like find_package(catkin REQUIRED
   COMPONENTS xyz)
9 ## is used, also find other catkin packages
10 find_package(catkin REQUIRED COMPONENTS
11   move_base
12   std_msgs
13 )
14
15 ## System dependencies are found with CMake's conventions
16 # find_package(Boost REQUIRED COMPONENTS system)
17
18
19 ## Uncomment this if the package has a setup.py. This
   macro ensures
20 ## modules and global scripts declared therein get
   installed
21 ## See http://ros.org/doc/api/catkin/html/user\_guide/
   setup\_dot\_py.html
22 # catkin_python_setup()
23
24 #####
25 ## Declare ROS messages, services and actions ##
26 #####
27
28 ## To declare and build messages, services or actions
   from within this
29 ## package, follow these steps:
30 ## * Let MSG_DEP_SET be the set of packages whose message
   types you use in
31 ##   your messages/services/actions (e.g. std_msgs,
   actionlib_msgs, ...).
32 ## * In the file package.xml:
33 ##   * add a build_depend tag for "message_generation"
34 ##   * add a build_depend and a run_depend tag for each
   package in MSG_DEP_SET
35 ##   * If MSG_DEP_SET isn't empty the following
   dependency has been pulled in
36 ##     but can be declared for certainty nonetheless:
37 ##     * add a run_depend tag for "message_runtime"
38 ## * In this file (CMakeLists.txt):
39 ##   * add "message_generation" and every package in
   MSG_DEP_SET to
40 ##     find_package(catkin REQUIRED COMPONENTS ...)
```

```
41 ## * add "message_runtime" and every package in
    MSG_DEP_SET to
42 ##     catkin_package(CATKIN_DEPENDS ...)
43 ## * uncomment the add*_files sections below as needed
44 ##     and list every .msg/.srv/.action file to be
        processed
45 ## * uncomment the generate_messages entry below
46 ## * add every package in MSG_DEP_SET to
        generate_messages(DEPENDENCIES ...)
47
48 ## Generate messages in the 'msg' folder
49 # add_message_files(
50 #     FILES
51 #     Message1.msg
52 #     Message2.msg
53 # )
54
55 ## Generate services in the 'srv' folder
56 # add_service_files(
57 #     FILES
58 #     Service1.srv
59 #     Service2.srv
60 # )
61
62 ## Generate actions in the 'action' folder
63 # add_action_files(
64 #     FILES
65 #     Action1.action
66 #     Action2.action
67 # )
68
69 ## Generate added messages and services with any
        dependencies listed here
70 # generate_messages(
71 #     DEPENDENCIES
72 #     std_msgs
73 # )
74
75 #####
76 ## Declare ROS dynamic reconfigure parameters ##
77 #####
78
```

```

79 ## To declare and build dynamic reconfigure parameters
    within this
80 ## package, follow these steps:
81 ## * In the file package.xml:
82 ## * add a build_depend and a run_depend tag for "
    dynamic_reconfigure"
83 ## * In this file (CMakeLists.txt):
84 ## * add "dynamic_reconfigure" to
85 ##     find_package(catkin REQUIRED COMPONENTS ...)
86 ## * uncomment the "
    generate_dynamic_reconfigure_options" section below
87 ##     and list every .cfg file to be processed
88
89 ## Generate dynamic reconfigure parameters in the 'cfg'
    folder
90 # generate_dynamic_reconfigure_options(
91 #     cfg/DynReconf1.cfg
92 #     cfg/DynReconf2.cfg
93 # )
94
95 #####
96 ## catkin specific configuration ##
97 #####
98 ## The catkin_package macro generates cmake config files
    for your package
99 ## Declare things to be passed to dependent projects
100 ## INCLUDE_DIRS: uncomment this if your package contains
    header files
101 ## LIBRARIES: libraries you create in this project that
    dependent projects also need
102 ## CATKIN_DEPENDS: catkin_packages dependent projects
    also need
103 ## DEPENDS: system dependencies of this project that
    dependent projects also need
104 catkin_package(
105 #     INCLUDE_DIRS include
106 #     LIBRARIES 2dnav_conf
107 #     CATKIN_DEPENDS move_base std_msgs
108 #     DEPENDS system_lib
109 )
110
111 #####
112 ## Build ##

```

```
113 #####
114
115 ## Specify additional locations of header files
116 ## Your package locations should be listed before other
    locations
117 include_directories(
118 # include
119   ${catkin_INCLUDE_DIRS}
120 )
121
122 ## Declare a C++ library
123 # add_library(${PROJECT_NAME}
124 #   src/${PROJECT_NAME}/2dnav_conf.cpp
125 # )
126
127 ## Add cmake target dependencies of the library
128 ## as an example, code may need to be generated before
    libraries
129 ## either from message generation or dynamic reconfigure
130 # add_dependencies(${PROJECT_NAME} ${${PROJECT_NAME}
    _EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
131
132 ## Declare a C++ executable
133 ## With catkin_make all packages are built within a
    single CMake context
134 ## The recommended prefix ensures that target names
    across packages don't collide
135 # add_executable(${PROJECT_NAME}_node src/2dnav_conf_node
    .cpp)
136
137 ## Rename C++ executable without prefix
138 ## The above recommended prefix causes long target names,
    the following renames the
139 ## target back to the shorter version for ease of user
    use
140 ## e.g. "roslaunch someones_pkg node" instead of "roslaunch
    someones_pkg someones_pkg_node"
141 # set_target_properties(${PROJECT_NAME}_node PROPERTIES
    OUTPUT_NAME node PREFIX "")
142
143 ## Add cmake target dependencies of the executable
144 ## same as for the library above
```



```
145 # add_dependencies(${PROJECT_NAME}_node ${${PROJECT_NAME}
    _EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
146
147 ## Specify libraries to link a library or executable
    target against
148 # target_link_libraries(${PROJECT_NAME}_node
149 #   ${catkin_LIBRARIES}
150 # )
151
152 #####
153 ## Install ##
154 #####
155
156 # all install targets should use catkin DESTINATION
    variables
157 # See http://ros.org/doc/api/catkin/html/adv\_user\_guide/
    variables.html
158
159 ## Mark executable scripts (Python etc.) for installation
160 ## in contrast to setup.py, you can choose the
    destination
161 # install(PROGRAMS
162 #   scripts/my_python_script
163 #   DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
164 # )
165
166 ## Mark executables and/or libraries for installation
167 # install(TARGETS ${PROJECT_NAME} ${PROJECT_NAME}_node
168 #   ARCHIVE DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
169 #   LIBRARY DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
170 #   RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
171 # )
172
173 ## Mark cpp header files for installation
174 # install(DIRECTORY include/${PROJECT_NAME}/
175 #   DESTINATION ${CATKIN_PACKAGE_INCLUDE_DESTINATION}
176 #   FILES_MATCHING PATTERN "*.h"
177 #   PATTERN ".svn" EXCLUDE
178 # )
179
180 ## Mark other files for installation (e.g. launch and bag
    files, etc.)
181 # install(FILES
```

```

182 #   # myfile1
183 #   # myfile2
184 #   DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}
185 # )
186
187 #####
188 ## Testing ##
189 #####
190
191 ## Add gtest based cpp test target and link libraries
192 # catkin_add_gtest(${PROJECT_NAME}-test test/
193   test_2dnav_conf.cpp)
194 # if(TARGET ${PROJECT_NAME}-test)
195 #   target_link_libraries(${PROJECT_NAME}-test ${
196   PROJECT_NAME})
197 # endif()
198
199 ## Add folders to be run by python nosetests
200 # catkin_add_nosetests(test)

```

14.2 package.xml

```

1 <?xml version="1.0"?>
2 <package format="2">
3   <name>nav2d_conf</name>
4   <version>0.0.0</version>
5   <description>The 2dnav_conf package</description>
6
7   <!-- One maintainer tag required, multiple allowed, one
8     person per tag -->
9   <!-- Example:  -->
10  <!-- <maintainer email="jane.doe@example.com">Jane Doe
11    </maintainer> -->
12  <maintainer email="alessandro@todo.todo">alessandro</
13    maintainer>
14
15  <!-- One license tag required, multiple allowed, one
16    license per tag -->
17  <!-- Commonly used license strings: -->
18  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3,
19    LGPLv2.1, LGPLv3 -->
20  <license>TODO</license>

```

```
16
17 <!-- Url tags are optional, but multiple are allowed,
    one per tag -->
18 <!-- Optional attribute type can be: website,
    bugtracker, or repository -->
19 <!-- Example: -->
20 <!-- <url type="website">http://wiki.ros.org/2dnav_conf
    </url> -->
21
22
23 <!-- Author tags are optional, multiple are allowed,
    one per tag -->
24 <!-- Authors do not have to be maintainers, but could
    be -->
25 <!-- Example: -->
26 <!-- <author email="jane.doe@example.com">Jane Doe</
    author> -->
27
28 <!-- The *depend tags are used to specify dependencies
    -->
29 <!-- Dependencies can be catkin packages or system
    dependencies -->
30 <!-- Examples: -->
31 <!-- Use depend as a shortcut for packages that are
    both build and exec dependencies -->
32 <!-- <depend>roscpp</depend> -->
33 <!-- Note that this is equivalent to the following:
    -->
34 <!-- <build_depend>roscpp</build_depend> -->
35 <!-- <exec_depend>roscpp</exec_depend> -->
36 <!-- Use build_depend for packages you need at compile
    time: -->
37 <!-- <build_depend>message_generation</build_depend>
    -->
38 <!-- Use build_export_depend for packages you need in
    order to build against this package: -->
39 <!-- <build_export_depend>message_generation</
    build_export_depend> -->
40 <!-- Use buildtool_depend for build tool packages: -->
41 <!-- <buildtool_depend>catkin</buildtool_depend> -->
42 <!-- Use exec_depend for packages you need at runtime:
    -->
43 <!-- <exec_depend>message_runtime</exec_depend> -->
```

```

44 <!-- Use test_depend for packages you need only for
    testing: -->
45 <!-- <test_depend>gtest</test_depend> -->
46 <!-- Use doc_depend for packages you need only for
    building documentation: -->
47 <!-- <doc_depend>doxygen</doc_depend> -->
48 <buildtool_depend>catkin</buildtool_depend>
49 <build_depend>move_base</build_depend>
50 <build_depend>std_msgs</build_depend>
51 <build_export_depend>move_base</build_export_depend>
52 <build_export_depend>std_msgs</build_export_depend>
53 <exec_depend>move_base</exec_depend>
54 <exec_depend>std_msgs</exec_depend>
55
56
57 <!-- The export tag contains other, unspecified, tags
    -->
58 <export>
59 <!-- Other tools can request additional information
    be placed here -->
60
61 </export>
62 </package>

```

14.3 cfg

14.3.1 costmap_common_params.yaml

```

1 # Obstacle Cost Shaping (http://wiki.ros.org/costmap\_2d/hydro/inflation)
2 robot_radius: 0.20 # distance a circular robot should be
    clear of the obstacle (kobuki: 0.18)
3 # footprint: [[x0, y0], [x1, y1], ... [xn, yn]] # if the
    robot is not circular
4
5 map_type: costmap_2d
6
7 obstacle_layer:
8   enabled: true
9   unknown_threshold: 15
10  mark_threshold: 0
11  combination_method: 1

```

```
12 track_unknown_space: true      #true needed for
    disabling global path planning through unknown space
13 obstacle_range: 2.5 # maximum range in meters at which
    to insert obstacles into the costmap using sensor
    data
14 raytrace_range: 3.0 # maximum range in meters at which
    to raytrace out obstacles from the map using sensor
    data
15 observation_sources: scan
16 scan:
17   data_type: LaserScan
18   topic: scan
19   marking: true
20   clearing: true
21   min_obstacle_height: 0.25
22   max_obstacle_height: 0.35
23
24
25 #cost_scaling_factor and inflation_radius were now moved
    to the inflation_layer ns
26 inflation_layer:
27   enabled: true
28   cost_scaling_factor: 5.0 # exponential rate at which
    the obstacle cost drops off (default: 10)
29   inflation_radius: 0.6 # max. distance from an
    obstacle at which costs are incurred for planning
    paths.
30
31 static_layer:
32   enabled: true
```

14.3.2 dwa_local_planner_params.yaml

```
1 DWAPlanerROS:
2
3 # Robot Configuration Parameters - Kobuki
4 max_vel_x: 0.5 # 0.55
5 min_vel_x: 0.0
6
7 max_vel_y: 0.0 # diff drive robot
8 min_vel_y: 0.0 # diff drive robot
9
```

```
10 max_trans_vel: 0.4 # choose slightly less than the base
    's capability
11 min_trans_vel: 0.1 # this is the min trans velocity
    when there is negligible rotational velocity
12 trans_stopped_vel: 0.1
13
14 # Warning!
15 # do not set min_trans_vel to 0.0 otherwise dwa will
    always think translational velocities
16 # are non-negligible and small in place rotational
    velocities will be created.
17
18 max_rot_vel: 3.0 # choose slightly less than the base'
    s capability
19 min_rot_vel: 0.4 # this is the min angular velocity
    when there is negligible translational velocity
20 rot_stopped_vel: 0.4
21
22 acc_lim_x: 0.5 # maximum is theoretically 2.0, but we
    don't want to crash/do strange stuff/overshoot
23 acc_lim_theta: 1.0 #rad
24 acc_lim_y: 0.0 # diff drive robot
25
26 # Goal Tolerance Parameters
27 yaw_goal_tolerance: 0.3 # 0.05
28 xy_goal_tolerance: 0.10 # 0.10
29
30
31 # Forward Simulation Parameters
32 sim_time: 2.0 # 1.7 The amount of time to forward
    -simulate trajectories in seconds
33 vx_samples: 3 # 3 The number of samples to use
    when exploring the x velocity space
34 vy_samples: 10 # diff drive robot
35 vtheta_samples: 20 # 20
36
37 # Trajectory Scoring Parameters
38 #cost = path_distance_bias * (distance to path from the
    endpoint of the trajectory in meters) +
    goal_distance_bias * (distance to local goal from the
    endpoint of the trajectory in meters) +
    occdist_scale * (maximum obstacle cost along the
    trajectory in obstacle cost (0-254))
```

```

39 path_distance_bias: 10.0      # 32.0    - weighting for
    how much it should stick to the global path plan
40 goal_distance_bias: 24.0      # 24.0    - weighting for
    how much it should attempt to reach its goal
41 occdist_scale: 0.2           # 0.01    - weighting for
    how much the controller should avoid obstacles
42 forward_point_distance: 0.25 # 0.325   - how far along
    to place an additional scoring point
43 stop_time_buffer: 0.2        # 0.2     - amount of time
    a robot must stop in before colliding for a valid
    traj.
44 scaling_speed: 0.25          # 0.25    - absolute
    velocity at which to start scaling the robot's
    footprint
45 max_scaling_factor: 0.2      # 0.2     - how much to
    scale the robot's footprint when at speed.
46
47 # Oscillation Prevention Parameters
48 oscillation_reset_dist: 0.15 # 0.05    - how far to
    travel before resetting oscillation flags
49
50 # Debugging
51 publish_traj_pc : true
52 publish_cost_grid_pc: true
53 publish_cost_grid: true
54 global_frame_id: odom

```

14.3.3 global_costmap_params.yaml

```

1 global_costmap:
2
3   global_frame: map
4   robot_base_frame: base_footprint
5   update_frequency: 1.0
6   publish_frequency: 0.5
7   static_map: true
8   transform_tolerance: 0.5
9   plugins:
10     - {name: static_layer,          type: "costmap_2d
      ::StaticLayer"}
11     - {name: inflation_layer,      type: "costmap_2d
      ::InflationLayer"}

```

14.3.4 local_costmap_params.yaml

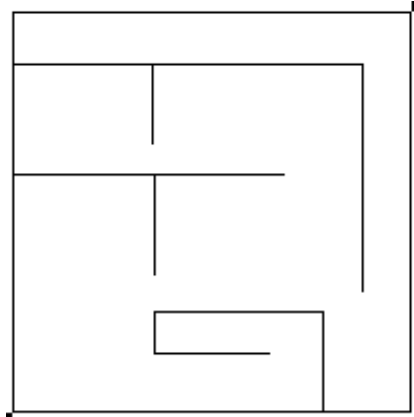
```
1 local_costmap:
2   global_frame: odom
3   robot_base_frame: base_footprint
4   update_frequency: 5.0
5   publish_frequency: 2.0
6   static_map: false
7   rolling_window: true
8   width: 6.0
9   height: 6.0
10  resolution: 0.05
11  transform_tolerance: 0.5 #maximum amount of latency
    allowed between tf
12  plugins:
13    - {name: obstacle_layer,      type: "costmap_2d::
      VoxelLayer"}
14    - {name: inflation_layer,    type: "costmap_2d::
      InflationLayer"}
```

14.3.5 move_base_params.yaml

```
1 shutdown_costmaps: false
2
3 controller_frequency: 5.0 #5
4 controller_patience: 3.0 #3
5
6 planner_frequency: 1.0
7 planner_patience: 5.0 #5
8
9 oscillation_timeout: 10.0
10 oscillation_distance: 0.2
11
12 # local planner - default is trajectory rollout
13 base_local_planner: "dwa_local_planner/DWAPlannerROS"
14
15 base_global_planner: "navfn/NavfnROS" #alternatives:
    global_planner/GlobalPlanner, carrot_planner/
    CarrotPlanner navfn/NavfnROS
```


14.4 maps

14.4.1 maze.png



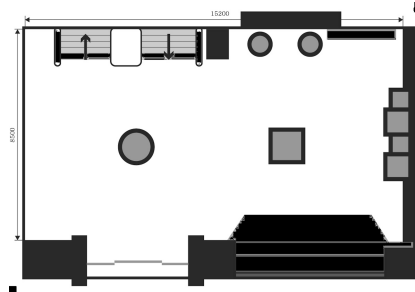
14.4.2 maze.yaml

```
1 image: maze.png
2 resolution: 0.05
3 origin: [0.0, 0.0, 0.0]
4 negate: 0
5 occupied_thresh: 0.65
6 free_thresh: 0.196
```

14.4.3 robopark_plan.yaml

```
1 image: robopark2.bmp
2 resolution: 0.014
3 origin: [-0.6, -2.24, 0.0] #The 2-D pose of the lower-
   left pixel in the map, as (x, y, yaw)
4 negate: 0
5 occupied_thresh: 0.65
6 free_thresh: 0.196
7
8 # 2150x700 pix
9 # 491 pix -> 13.900 m
10 # res = 0.0283 pix/m
11 # 1012 >>> 28.6396
12 # 340 >>> 9.622
```

14.4.4 robopark2.bmp



14.4.5 stage

maze.world

```

1 include "turtlebot.inc"
2
3 define floorplan model
4 (
5     # sombre, sensible, artistic
6     color "gray30"
7
8     # most maps will need a bounding box
9     boundary 1
10
11     gui_nose 0
12     gui_grid 0
13     gui_outline 0
14     gripper_return 0
15     fiducial_return 0
16     laser_return 1
17 )
18
19 resolution 0.02
20 interval_sim 100 # simulation timestep in milliseconds
21
22 window
23 (
24     size [ 600.0 700.0 ]
25     center [ 0.0 0.0 ]
26     rotate [ 0.0 0.0 ]
27     scale 60
28 )

```

```
29
30 floorplan
31 (
32     name "maze"
33     bitmap "../maze.png"
34     size [ 10.0 10.0 2.0 ]
35     pose [ 5.0 5.0 0.0 0.0 ]
36 )
37
38 # throw in a robot
39 turtlebot
40 (
41     pose [ 2.0 2.0 0.0 0.0 ]
42     name "turtlebot"
43     color "black"
44 )
```

robopark_plan.world

```
1 include "turtlebot.inc"
2
3 # Definition for an obstacle placed on the map.
4 define block model
5 (
6     size [0.500 0.500 1.500]
7     gui_nose 0
8 )
9
10 # throw in a robot
11 turtlebot
12 (
13     pose [ 2.000 2.000 0.000 0.000 ]
14     name "turtlebot1"
15     color "gray"
16     gui_nose 1
17 )
18
19 # throw in an obstacle
20 block( pose [ 4.000 4.000 0.000 0.000 ] color "red")
21
22
23 define floorplan model
```

```
24 (
25   # sombre, sensible, artistic
26   color "gray30"
27
28   # most maps will need a bounding box
29   boundary 1
30
31   gui_nose 0
32   gui_grid 0
33
34   gui_outline 0
35   gripper_return 0
36   fiducial_return 0
37   laser_return 1
38 )
39
40 # set the resolution of the underlying raytrace model in
41   meters
42 resolution 0.01
43
44 interval_sim 100 # simulation timestep in milliseconds
45
46 window
47 (
48   size [ 600 424 ]
49   rotate [ 0.000 0.000 ]
50 )
51
52 # load an environment bitmap
53 floorplan
54 (
55   name "Robo Park floor"
56   bitmap "../robopark2.bmp"
57   size [16.800 11.870 1.000]
58
59   pose [ 0.000 0.000 0.000 0.000 ]
60   #for test
61   origin [ 7.800 3.700 0.000 0.000] # specify the
62     position of the object's center, relative to its
63     pose
64 )
```

turtlebot.inc

```
1 define kinect ranger
2 (
3   sensor
4   (
5     range_max 8.0
6     fov 360.0
7     samples 640
8   )
9   # generic model properties
10  color "black"
11  size [ 0.06 0.15 0.03 ]
12 )
13
14 define turtlebot position
15 (
16   pose [ 0.0 0.0 0.0 0.0 ]
17
18   odom_error [0.01 0.01 999999 999999 999999 0.01]
19
20   size [ 0.2552 0.2552 0.40 ]
21   origin [ 0.0 0.0 0.0 0.0 ]
22   gui_nose 1
23   drive "diff"
24   color "gray"
25
26   kinect(pose [ -0.1 0.0 -0.11 0.0 ])
27 )
```

14.5 rviz**14.5.1 robot_navigation.rviz**

```
1 Panels:
2   - Class: rviz/Displays
3     Help Height: 78
4     Name: Displays
5     Property Tree Widget:
6       Expanded:
7         - /TF1/Frames1
8         - /TF1/Tree1
```

```
9      Splitter Ratio: 0.5
10     Tree Height: 786
11     - Class: rviz/Selection
12       Name: Selection
13     - Class: rviz/Tool Properties
14       Expanded:
15         - /2D Pose Estimate1
16         - /2D Nav Goal1
17       Name: Tool Properties
18       Splitter Ratio: 0.5886790156364441
19     - Class: rviz/Views
20       Expanded:
21         - /Current View1
22       Name: Views
23       Splitter Ratio: 0.5
24     - Class: rviz/Time
25       Name: Time
26       SyncMode: 0
27       SyncSource: LaserScan (kinect)
28 Preferences:
29   PromptSaveOnExit: true
30 Toolbars:
31   toolButtonStyle: 2
32 Visualization Manager:
33   Class: ""
34   Displays:
35     - Alpha: 0.5
36       Cell Size: 1
37       Class: rviz/Grid
38       Color: 160; 160; 164
39       Enabled: true
40       Line Style:
41         Line Width: 0.029999999329447746
42         Value: Lines
43       Name: Grid
44       Normal Cell Count: 0
45       Offset:
46         X: 0
47         Y: 0
48         Z: 0
49       Plane: XY
50       Plane Cell Count: 10
51       Reference Frame: <Fixed Frame>
```

```
52     Value: true
53   - Class: rviz/TF
54     Enabled: false
55     Filter (blacklist): ""
56     Filter (whitelist): ""
57     Frame Timeout: 15
58     Frames:
59       All Enabled: false
60     Marker Alpha: 1
61     Marker Scale: 1
62     Name: TF
63     Show Arrows: true
64     Show Axes: true
65     Show Names: false
66     Tree:
67       {}
68     Update Interval: 0
69     Value: false
70   - Alpha: 1
71     Autocompute Intensity Bounds: true
72     Autocompute Value Bounds:
73       Max Value: 10
74       Min Value: -10
75       Value: true
76     Axis: Z
77     Channel Name: intensity
78     Class: rviz/LaserScan
79     Color: 255; 255; 255
80     Color Transformer: Intensity
81     Decay Time: 0
82     Enabled: true
83     Invert Rainbow: false
84     Max Color: 255; 255; 255
85     Min Color: 0; 0; 0
86     Name: LaserScan (kinect)
87     Position Transformer: XYZ
88     Queue Size: 10
89     Selectable: true
90     Size (Pixels): 3
91     Size (m): 0.009999999776482582
92     Style: Squares
93     Topic: /scan
94     Unreliable: false
```

```
95     Use Fixed Frame: true
96     Use rainbow: true
97     Value: true
98 - Alpha: 1
99     Autocompute Intensity Bounds: true
100    Autocompute Value Bounds:
101      Max Value: 10
102      Min Value: -10
103      Value: true
104    Axis: Z
105    Channel Name: intensity
106    Class: rviz/LaserScan
107    Color: 255; 255; 255
108    Color Transformer: Intensity
109    Decay Time: 0
110    Enabled: true
111    Invert Rainbow: false
112    Max Color: 255; 255; 255
113    Min Color: 0; 0; 0
114    Name: LaserScan (ir sensors)
115    Position Transformer: XYZ
116    Queue Size: 10
117    Selectable: true
118    Size (Pixels): 3
119    Size (m): 0.05000000074505806
120    Style: Flat Squares
121    Topic: /ir_scan
122    Unreliable: false
123    Use Fixed Frame: true
124    Use rainbow: true
125    Value: true
126 - Alpha: 1
127     Autocompute Intensity Bounds: true
128     Autocompute Value Bounds:
129       Max Value: 10
130       Min Value: -10
131       Value: true
132     Axis: Z
133     Channel Name: intensity
134     Class: rviz/LaserScan
135     Color: 255; 255; 255
136     Color Transformer: Intensity
137     Decay Time: 0
```



```
138     Enabled: true
139     Invert Rainbow: false
140     Max Color: 255; 255; 255
141     Min Color: 0; 0; 0
142     Name: LaserScan (virtual sensor)
143     Position Transformer: XYZ
144     Queue Size: 10
145     Selectable: true
146     Size (Pixels): 3
147     Size (m): 0.019999999552965164
148     Style: Flat Squares
149     Topic: /virtual_sensor_scan
150     Unreliable: false
151     Use Fixed Frame: true
152     Use rainbow: true
153     Value: true
154 - Alpha: 1
155     Autocompute Intensity Bounds: true
156     Autocompute Value Bounds:
157       Max Value: 10
158       Min Value: -10
159       Value: true
160     Axis: Z
161     Channel Name: intensity
162     Class: rviz/PointCloud2
163     Color: 255; 255; 255
164     Color Transformer: Intensity
165     Decay Time: 0
166     Enabled: true
167     Invert Rainbow: false
168     Max Color: 255; 255; 255
169     Min Color: 0; 0; 0
170     Name: PointCloud (bumpers)
171     Position Transformer: XYZ
172     Queue Size: 10
173     Selectable: true
174     Size (Pixels): 3
175     Size (m): 0.03999999910593033
176     Style: Flat Squares
177     Topic: /mobile_base/sensors/bumper_pointcloud
178     Unreliable: false
179     Use Fixed Frame: true
180     Use rainbow: true
```

```
181     Value: true
182 - Alpha: 0.699999988079071
183   Class: rviz/Map
184   Color Scheme: map
185   Draw Behind: false
186   Enabled: true
187   Name: Map
188   Topic: /map
189   Unreliable: false
190   Use Timestamp: false
191   Value: true
192 - Class: rviz/Group
193   Displays:
194     - Alpha: 0.699999988079071
195       Class: rviz/Map
196       Color Scheme: costmap
197       Draw Behind: false
198       Enabled: false
199       Name: Costmap
200       Topic: /move_base/local_costmap/costmap
201       Unreliable: false
202       Use Timestamp: false
203       Value: false
204     - Alpha: 1
205       Buffer Length: 1
206       Class: rviz/Path
207       Color: 0; 12; 255
208       Enabled: true
209       Head Diameter: 0.30000001192092896
210       Head Length: 0.20000000298023224
211       Length: 0.30000001192092896
212       Line Style: Lines
213       Line Width: 0.029999999329447746
214       Name: Planner
215       Offset:
216         X: 0
217         Y: 0
218         Z: 0
219       Pose Color: 255; 85; 255
220       Pose Style: None
221       Queue Size: 10
222       Radius: 0.029999999329447746
223       Shaft Diameter: 0.10000000149011612
```

```
224         Shaft Length: 0.10000000149011612
225         Topic: /move_base/DWAPlanerROS/local_plan
226         Unreliable: false
227         Value: true
228     Enabled: true
229     Name: Local Planning
230 - Class: rviz/Group
231     Displays:
232     - Alpha: 0.4000000059604645
233       Class: rviz/Map
234       Color Scheme: costmap
235       Draw Behind: true
236       Enabled: false
237       Name: Costmap
238       Topic: /move_base/global_costmap/costmap
239       Unreliable: false
240       Use Timestamp: false
241       Value: false
242     - Alpha: 1
243       Buffer Length: 1
244       Class: rviz/Path
245       Color: 255; 0; 0
246       Enabled: true
247       Head Diameter: 0.30000001192092896
248       Head Length: 0.20000000298023224
249       Length: 0.30000001192092896
250       Line Style: Lines
251       Line Width: 0.029999999329447746
252       Name: Planner
253       Offset:
254         X: 0
255         Y: 0
256         Z: 0
257       Pose Color: 255; 85; 255
258       Pose Style: None
259       Queue Size: 10
260       Radius: 0.029999999329447746
261       Shaft Diameter: 0.10000000149011612
262       Shaft Length: 0.10000000149011612
263       Topic: /move_base/NavfnROS/plan
264       Unreliable: false
265       Value: true
266     Enabled: true
```

```
267     Name: Global Planning
268   - Alpha: 1
269     Axes Length: 1
270     Axes Radius: 0.10000000149011612
271     Class: rviz/Pose
272     Color: 255; 25; 0
273     Enabled: true
274     Head Length: 0.20000000298023224
275     Head Radius: 0.10000000149011612
276     Name: Pose (move_base)
277     Queue Size: 10
278     Shaft Length: 1
279     Shaft Radius: 0.05000000074505806
280     Shape: Arrow
281     Topic: /move_base/current_goal
282     Unreliable: false
283     Value: true
284   - Alpha: 1
285     Arrow Length: 0.20000000298023224
286     Axes Length: 0.30000001192092896
287     Axes Radius: 0.009999999776482582
288     Class: rviz/PoseArray
289     Color: 0; 192; 0
290     Enabled: true
291     Head Length: 0.07000000029802322
292     Head Radius: 0.029999999329447746
293     Name: ParticleCloud
294     Queue Size: 10
295     Shaft Length: 0.23000000417232513
296     Shaft Radius: 0.009999999776482582
297     Shape: Arrow (Flat)
298     Topic: /particlecloud
299     Unreliable: false
300     Value: true
301   - Alpha: 1
302     Autocompute Intensity Bounds: true
303     Autocompute Value Bounds:
304       Max Value: 10
305       Min Value: -10
306       Value: true
307     Axis: Z
308     Channel Name: total_cost
309     Class: rviz/PointCloud2
```

```
310     Color: 255; 255; 255
311     Color Transformer: Intensity
312     Decay Time: 0
313     Enabled: false
314     Invert Rainbow: false
315     Max Color: 255; 255; 255
316     Min Color: 0; 0; 0
317     Name: PointCloud2
318     Position Transformer: XYZ
319     Queue Size: 10
320     Selectable: true
321     Size (Pixels): 3
322     Size (m): 0.009999999776482582
323     Style: Points
324     Topic: /move_base/DWAPlannerROS/cost_cloud
325     Unreliable: false
326     Use Fixed Frame: true
327     Use rainbow: true
328     Value: false
329 - Alpha: 0.699999988079071
330   Class: rviz/Map
331   Color Scheme: map
332   Draw Behind: false
333   Enabled: false
334   Name: Map
335   Topic: /move_base/local_costmap/costmap
336   Unreliable: false
337   Use Timestamp: false
338   Value: false
339 - Alpha: 0.699999988079071
340   Class: rviz/Map
341   Color Scheme: map
342   Draw Behind: false
343   Enabled: false
344   Name: Map
345   Topic: /move_base/global_costmap/costmap
346   Unreliable: false
347   Use Timestamp: false
348   Value: false
349 - Alpha: 1
350   Class: rviz/Polygon
351   Color: 25; 255; 0
352   Enabled: true
```

```
353     Name: Polygon
354     Queue Size: 10
355     Topic: /move_base/local_costmap/footprint
356     Unreliable: false
357     Value: true
358 Enabled: true
359 Global Options:
360     Background Color: 48; 48; 48
361     Default Light: true
362     Fixed Frame: map
363     Frame Rate: 30
364 Name: root
365 Tools:
366     - Class: rviz/MoveCamera
367     - Class: rviz/Interact
368       Hide Inactive Objects: true
369     - Class: rviz/Select
370     - Class: rviz/SetInitialPose
371       Theta std deviation: 0.2617993950843811
372       Topic: /initialpose
373       X std deviation: 0.5
374       Y std deviation: 0.5
375     - Class: rviz/SetGoal
376       Topic: /move_base_simple/goal
377     - Class: rviz/Measure
378 Value: true
379 Views:
380     Current:
381       Class: rviz/Orbit
382       Distance: 24.896236419677734
383       Enable Stereo Rendering:
384         Stereo Eye Separation: 0.059999999865889549
385         Stereo Focal Distance: 1
386         Swap Stereo Eyes: false
387         Value: false
388       Field of View: 0.7853981852531433
389       Focal Point:
390         X: 3.1884381771087646
391         Y: 3.1455135345458984
392         Z: -0.1958555430173874
393       Focal Shape Fixed Size: true
394       Focal Shape Size: 0.05000000074505806
395       Invert Z Axis: false
```

```

396     Name: Current View
397     Near Clip Distance: 0.009999999776482582
398     Pitch: 1.5697963237762451
399     Target Frame: base_link
400     Yaw: 6.27159309387207
401     Saved: ~
402 Window Geometry:
403     Displays:
404         collapsed: false
405     Height: 1016
406     Hide Left Dock: false
407     Hide Right Dock: false
408     QMainWindow State: 000000
        ff00000000fd00000004000000000000001e10000039dfc02000000005fb0000001200
409     Selection:
410         collapsed: false
411     Time:
412         collapsed: false
413     Tool Properties:
414         collapsed: false
415     Views:
416         collapsed: false
417     Width: 1920
418     X: -2
419     Y: 0

```

14.6 launch

14.6.1 amcl.launch.xml

```

1 <launch>
2   <arg name="use_map_topic"    default="true"/>
3   <arg name="scan_topic"       default="scan"/>
4   <arg name="initial_pose_x"   default="0.0"/>
5   <arg name="initial_pose_y"   default="0.0"/>
6   <arg name="initial_pose_a"   default="0.0"/>
7   <arg name="odom_frame_id"     default="odom"/>
8   <arg name="base_frame_id"     default="base_footprint"/>
9   <arg name="global_frame_id"  default="map"/>
10
11   <node pkg="amcl" type="amcl" name="amcl">

```

```

12     <param name="use_map_topic"           value="$(arg
        use_map_topic)"/>
13     <!-- Publish scans from best pose at a max of 10 Hz
        -->
14     <param name="odom_model_type"         value="diff-
        corrected"/>
15     <param name="odom_alpha5"             value="0.1"/>
16     <param name="gui_publish_rate"        value="10.0"
        />
17     <param name="laser_max_beams"         value="60"
        />
18     <param name="laser_max_range"         value="12.0"
        />
19     <param name="min_particles"           value="500"/>
20     <param name="max_particles"          value="2000"
        />
21     <param name="kld_err"                 value="0.05"
        />
22     <param name="kld_z"                   value="0.99"
        />
23     <param name="odom_alpha1"             value="0.2"/>
24     <param name="odom_alpha2"             value="0.2"/>
25     <param name="odom_alpha3"             value="0.2"/>
26     <param name="odom_alpha4"             value="0.2"/>
27     <param name="laser_z_hit"             value="0.5"/>
28     <param name="laser_z_short"          value="0.05"
        />
29     <param name="laser_z_max"             value="0.05"
        />
30     <param name="laser_z_rand"            value="0.5"/>
31     <param name="laser_sigma_hit"         value="0.2"/>
32     <param name="laser_lambda_short"      value="0.1"/>
33     <param name="laser_model_type"        value="
        likelihood_field"/>
34     <!-- <param name="laser_model_type" value="beam"/>
        -->
35     <param name="laser_likelihood_max_dist" value="2.0"/>
36     <param name="update_min_d"            value="0.25"
        />
37     <param name="update_min_a"            value="0.2"/>
38     <param name="odom_frame_id"           value="$(arg
        odom_frame_id)"/>

```



```

39     <param name="base_frame_id"           value="$(arg
        base_frame_id)"/>
40     <param name="global_frame_id"         value="$(arg
        global_frame_id)"/>
41     <param name="resample_interval"        value="1"/>
42     <!-- Increase tolerance because the computer can get
        quite busy -->
43     <param name="transform_tolerance"      value="1.0"/>
44     <param name="recovery_alpha_slow"     value="0.0"/>
45     <param name="recovery_alpha_fast"     value="0.0"/>
46     <param name="initial_pose_x"          value="$(arg
        initial_pose_x)"/>
47     <param name="initial_pose_y"          value="$(arg
        initial_pose_y)"/>
48     <param name="initial_pose_a"          value="$(arg
        initial_pose_a)"/>
49     <remap from="scan"                    to="$(arg
        scan_topic)"/>
50 </node>
51 </launch>

```

14.6.2 gmapping.launch.xml

```

1 <launch>
2   <arg name="scan_topic"   default="scan" />
3   <arg name="base_frame"   default="base_footprint"/>
4   <arg name="odom_frame"   default="odom"/>
5
6   <node pkg="gmapping" type="slam_gmapping" name="
       slam_gmapping" output="screen">
7     <param name="base_frame" value="$(arg base_frame)"/>
8     <param name="odom_frame" value="$(arg odom_frame)"/>
9     <param name="map_update_interval" value="5.0"/>
10    <param name="maxUrange" value="6.0"/>
11    <param name="maxRange" value="8.0"/>
12    <param name="sigma" value="0.05"/>
13    <param name="kernelSize" value="1"/>
14    <param name="lstep" value="0.05"/>
15    <param name="astep" value="0.05"/>
16    <param name="iterations" value="5"/>
17    <param name="lsigma" value="0.075"/>
18    <param name="ogain" value="3.0"/>

```

```

19   <param name="lskip" value="0"/>
20   <param name="minimumScore" value="200"/>
21   <param name="srr" value="0.1"/>
22   <param name="srt" value="0.2"/>
23   <param name="str" value="0.1"/>
24   <param name="stt" value="0.2"/>
25   <param name="linearUpdate" value="0.5"/>
26   <param name="angularUpdate" value="0.1"/>
27   <param name="temporalUpdate" value="-1.0"/>
28   <param name="resampleThreshold" value="0.5"/>
29   <param name="particles" value="30"/>
30
31   <param name="xmin" value="-10.0"/>
32   <param name="ymin" value="-10.0"/>
33   <param name="xmax" value="10.0"/>
34   <param name="ymax" value="10.0"/>
35
36   <param name="delta" value="0.05"/>
37   <param name="llsamplerange" value="0.01"/>
38   <param name="llsamplestep" value="0.01"/>
39   <param name="lasamplerange" value="0.005"/>
40   <param name="lasamplestep" value="0.005"/>
41   <remap from="scan" to="$(arg scan_topic)"/>
42 </node>
43 </launch>

```

14.6.3 turtlebot_amcl.launch

```

1
2 <launch>
3   <arg name="base"          default="burger"/> <!--
      create_circles_asus_xtion_pro create_circles_kinect
      kobuki_hexagons_astra kobuki_hexagons_asus_xtion_pro
      kobuki_hexagons_asus_xtion_pro
4 kobuki_hexagons_kinect kobuki_hexagons_r200
      roomba_circles_asus_xtion_pro roomba_circles_kinect
      -->
5
6   <!-- Name of the map to use and initial position -->
7   <arg name="map_file"      default="$(find nav2d_conf)/
      maps/maze.yaml"/>

```

```

8   <arg name="world_file"      default="$(find nav2d_conf)/
    maps/stage/maze.world"/>
9   <arg name="initial_pose_x" default="2.0"/>
10  <arg name="initial_pose_y" default="2.0"/>
11  <arg name="initial_pose_a" default="0.0"/>
12
13  <arg name="odom_frame_id"    default="odom"/>
14  <arg name="global_frame_id" default="map"/>
15
16  <param name="/use_sim_time" value="true"/>
17
18  <!-- ***** Robot simulation
    ***** -->
19  <node pkg="stage_ros" type="stageros" name="stageros"
    args="$(arg world_file)">
20    <param name="base_watchdog_timeout" value="0.5"/>
21    <remap from="base_scan" to="scan"/>
22  </node>
23
24  <!-- ***** Robot Model *****
    -->
25  <include file="$(find turtlebot3_bringup)/launch/
    includes/description.launch.xml">
26    <arg name="model" value="$(arg base)" />
27  </include>
28  <node name="joint_state_publisher" pkg="
    joint_state_publisher" type="joint_state_publisher">
29    <param name="use_gui" value="false"/>
30  </node>
31
32  <!-- ***** Navigation ***** -->
33  <node pkg="move_base" type="move_base" respawn="false"
    name="move_base" output="screen">
34    <rosparam file="$(find nav2d_conf)/cfg/
    costmap_common_params.yaml" command="load" ns="
    global_costmap" />
35    <rosparam file="$(find nav2d_conf)/cfg/
    costmap_common_params.yaml" command="load" ns="
    local_costmap" />
36    <rosparam file="$(find nav2d_conf)/cfg/
    local_costmap_params.yaml" command="load" />
37    <rosparam file="$(find nav2d_conf)/cfg/
    global_costmap_params.yaml" command="load" />

```

```

38 <rosparam file="$(find nav2d_conf)/cfg/
    dwa_local_planner_params.yaml" command="load" />
39 <rosparam file="$(find nav2d_conf)/cfg/
    move_base_params.yaml" command="load" />
40
41 <param name="base_global_planner" value="navfn/
    NavfnROS" /> <!-- planner navfn/NavfnROS
    alternative carrot_planner/CarrotPlanner -->
42 <param name="planner_frequency" value="1.0" />
43 <param name="planner_patience" value="5.0" /> <!--
    How long the planner will wait in seconds in an
    attempt to find a valid plan before space-clearing
    operations are performed. -->
44
45 <param name="base_local_planner" value="
    dwa_local_planner/DWAPlannerROS" />
46 <param name="controller_frequency" value="5.0" />
47 <param name="controller_patience" value="5.0" /> <!--
    How long the controller will wait in seconds
    without receiving a valid control before space-
    clearing operations are performed. -->
48
49 <param name="clearing_rotation_allowed" value="true"
    />
50 </node>
51
52 <!-- ***** Maps ***** -->
53 <node name="map_server" pkg="map_server" type="
    map_server" args="$(arg map_file)">
54 <param name="frame_id" value="map"/>
55 </node>
56
57 <!-- ***** amcl ***** -->
58 <include file="$(find nav2d_conf)/launch/amcl.launch.
    xml">
59 <arg name="scan_topic" value="scan"/>
60 <arg name="use_map_topic" value="true"/>
61 <arg name="initial_pose_x" value="$(arg
    initial_pose_x)"/>
62 <arg name="initial_pose_y" value="$(arg
    initial_pose_y)"/>
63 <arg name="initial_pose_a" value="$(arg
    initial_pose_a)"/>

```

```

64 </include>
65
66
67 <!-- ***** Visualisation *****
-->
68 <node name="rviz" pkg="rviz" type="rviz" args="-d $(
    find nav2d_conf)/rviz/robot_navigation.rviz"/>
69 </launch>

```

14.6.4 turtlebot_gmapping.launch

```

1 <launch>
2   <arg name="base"          default="burger"/>  <!-- create,
    rhoomba -->
3
4   <!-- Name of the map to use and initial position -->
5   <arg name="world_file"    default="$(find nav2d_conf)/
    maps/stage/maze.world"/>
6
7   <arg name="odom_frame_id"  default="odom"/>
8   <arg name="global_frame_id" default="map"/>
9
10  <param name="/use_sim_time" value="true"/>
11
12  <!-- ***** Robot simulation
    ***** -->
13  <node pkg="stage_ros" type="stageros" name="stageros"
    args="$(arg world_file)">
14    <param name="base_watchdog_timeout" value="0.5"/>
15    <remap from="base_scan" to="scan"/>
16  </node>
17
18  <!-- ***** Robot Model *****
    -->
19  <include file="$(find turtlebot3_bringup)/launch/
    includes/description.launch.xml">
20    <arg name="model" value="$(arg base)" />
21  </include>
22
23  <!-- ***** Navigation ***** -->
24  <node pkg="move_base" type="move_base" respawn="false"
    name="move_base" output="screen">

```

```

25     <rosparam file="$(find nav2d_conf)/cfg/
        costmap_common_params.yaml" command="load" ns="
        global_costmap" />
26     <rosparam file="$(find nav2d_conf)/cfg/
        costmap_common_params.yaml" command="load" ns="
        local_costmap" />
27     <rosparam file="$(find nav2d_conf)/cfg/
        local_costmap_params.yaml" command="load" />
28     <rosparam file="$(find nav2d_conf)/cfg/
        global_costmap_params.yaml" command="load" />
29     <rosparam file="$(find nav2d_conf)/cfg/
        dwa_local_planner_params.yaml" command="load" />
30     <rosparam file="$(find nav2d_conf)/cfg/
        move_base_params.yaml" command="load" />
31
32     <param name="base_global_planner" value="navfn/
        NavfnROS" />
33     <param name="planner_frequency" value="1.0" />
34     <param name="planner_patience" value="10.0" />
35
36     <param name="base_local_planner" value="
        dwa_local_planner/DWAPlannerROS" />
37     <param name="controller_frequency" value="5.0" />
38     <param name="controller_patience" value="5.0" />
39
40     <param name="clearing_rotation_allowed" value="true"
        />
41 </node>
42
43 <!-- Gmapping -->
44 <include file="$(find nav2d_conf)/launch/gmapping.
        launch.xml" />
45
46 <!-- ***** Visualisation *****
        -->
47 <node name="rviz" pkg="rviz" type="rviz" args="-d $(
        find nav2d_conf)/rviz/robot_navigation.rviz"/>
48 </launch>

```