robotics/catkin_ws/src

Robotics Course - ROS

Giovanni Pachera

Aprile 2025

Contents

1 pub_sub

1.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(pub_sub)

## Find catkin and any catkin packages
find_package(catkin REQUIRED COMPONENTS roscpp std_msgs)

## Declare a catkin package
catkin_package()

## Build talker and listener
include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(pub src/pub.cpp)
target_link_libraries(pub ${catkin_LIBRARIES})

add_executable(sub src/sub.cpp)
target_link_libraries(sub ${catkin_LIBRARIES})
```

1.2 launch

1.2.1 multi_turtle.launch

```
<launch>
  <group ns="turtlesim1">
    <node pkg="turtlesim" name="sim" type="turtlesim_node
       "/>
    <node pkg="turtlesim" name="teleop" type="
       turtle_teleop_key"/>
  </group>

  <group ns="turtlesim2">
    <node pkg="turtlesim" name="sim" type="turtlesim_node
       "/>
  </group>

  <node pkg="turtlesim" name="mimic" type="mimic">
    <remap from="input" to="turtlesim1/turtle1"/>
    <remap from="output" to="turtlesim2/turtle1"/>
  </node>
</launch>
```

1.2.2 simple_launch_namespace.launch

```
<launch>
    <!-- Publisher node with namespace -->
    <group ns="namespace">
    <node pkg="pub_sub" type="pub" name="talker" output="
       screen" />
    </group>
</launch>
```

1.2.3 simple_launch.launch

```
<launch>
    <!-- Publisher node -->
    <node pkg="pub_sub" type="pub" name="talker" output="
       screen" />
    <!-- Subscriber node -->
    <node pkg="pub_sub" type="sub" name="listener" output
       ="screen" />
</launch>
```

1.3 package.xml

```
<?xml version="1.0"?>
<package format="2">
  <name>pub_sub</name>
  <version>0.0.0</version>
  <description>The pub_sub package</description>
  <maintainer email="simone.mentasti@polimi.iy">simone</
     maintainer>
  <license>GPL</license>

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>

  <build_export_depend>roscpp</build_export_depend>
  <build_export_depend>std_msgs</build_export_depend>

  <exec_depend>roscpp</exec_depend>
  <exec_depend>std_msgs</exec_depend>

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

1.4 src

1.4.1 pub.cpp

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

int main(int argc, char **argv){
    ros::init(argc, argv, "talker"); //, ros::
       init_options::AnonymousName);
    ros::NodeHandle n;
    ros::Publisher chatter_pub = n.advertise<std_msgs::
       String>("chatter", 1);

    ros::Rate loop_rate(10);

    while (ros::ok()){
            std_msgs::String msg;
                msg.data = "hello world!";
            ROS_INFO("%s", msg.data.c_str());
            chatter_pub.publish(msg);

            ros::spinOnce();

            loop_rate.sleep();
    }

    return 0;
}
```

1.4.2 sub.cpp

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

void chatterCallback(const std_msgs::String::ConstPtr&
   msg){
  ROS_INFO("I heard: [%s]", msg->data.c_str());
}

int main(int argc, char **argv){
    ros::init(argc, argv, "listener");
    ros::NodeHandle n;
    ros::Subscriber sub = n.subscribe("chatter", 1,
       chatterCallback);

    ros::spin();

    return 0;
}
```

2 pub_sub_same

2.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(pub_sub_same)

find_package(catkin REQUIRED)
find_package(catkin REQUIRED COMPONENTS
  roscpp
)

catkin_package(
)

include_directories(
  include ${catkin_INCLUDE_DIRS}
)

add_executable(test_pub
  src/test_pub.cpp
)

add_dependencies(test_pub ${${PROJECT_NAME}
    _EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})

target_link_libraries(test_pub
  ${catkin_LIBRARIES}
)

add_executable(test_sub
  src/test_sub.cpp
)

add_dependencies(test_sub ${${PROJECT_NAME}
    _EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})

target_link_libraries(test_sub
  ${catkin_LIBRARIES}
)
```

2.2 package.xml

```
<?xml version="1.0"?>
<package format="2">
  <name>pub_sub_same</name>
  <version>0.0.0</version>
  <description>The test package</description>
  <!-- One maintainer tag required, multiple allowed, one
      person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe
     </maintainer> -->
  <maintainer email="simone@todo.todo">simone</maintainer
     >

  <!-- One license tag required, multiple allowed, one
     license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3,
     LGPLv2.1, LGPLv3 -->
  <license>TODO</license>
```

```
<!-- Url tags are optional, but multiple are allowed, one per tag -->
<!-- Optional attribute type can be: website, bugtracker, or repository -->
<!-- Example: -->
<!-- <url type="website">http://wiki.ros.org/test</url> -->
<!-- Author tags are optional, multiple are allowed, one per tag -->
<!-- Authors do not have to be maintainers, but could be -->
<!-- Example: -->
<!-- <author email="jane.doe@example.com">Jane Doe</author> -->
<!-- The *depend tags are used to specify dependencies -->
<!-- Dependencies can be catkin packages or system dependencies -->
<!-- Examples: -->
<!-- Use depend as a shortcut for packages that are both build and exec dependencies -->
<!--   <depend>roscpp</depend> -->
<!--   Note that this is equivalent to the following: -->
<!--   <build_depend>roscpp</build_depend> -->
<!--   <exec_depend>roscpp</exec_depend> -->
<!-- Use build_depend for packages you need at compile time: -->
<!--   <build_depend>message_generation</build_depend> -->
<!-- Use build_export_depend for packages you need in order to build against this package: -->
<!--   <build_export_depend>message_generation</build_export_depend> -->
<!-- Use buildtool_depend for build tool packages: -->
<!--   <buildtool_depend>catkin</buildtool_depend> -->
<!-- Use exec_depend for packages you need at runtime: -->
<!--   <exec_depend>message_runtime</exec_depend> -->
<!-- Use test_depend for packages you need only for testing: -->
<!--   <test_depend>gtest</test_depend> -->
```

```
  <!-- Use doc_depend for packages you need only for
     building documentation: -->
  <!--   <doc_depend>doxygen</doc_depend> -->
  <buildtool_depend>catkin</buildtool_depend>
  <!-- The export tag contains other, unspecified, tags
     -->

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

2.3 src

2.3.1 test_pub.cpp

```
#include "ros/ros.h"
#include "std_msgs/String.h"
#include <stdlib.h>
#include <sstream>

int main(int argc, char *argv[])
{
    ros::init(argc, argv, "publisher");
    ros::NodeHandle n;
    ros::Publisher chatter_pub = n.advertise<std_msgs::
       String>("chatter", 1000);
    ros::Publisher chatter_pub2 = n.advertise<std_msgs::
       String>("chatter2", 1000);

    ros::Rate loop_rate(10);
    int count = 0;

    while (ros::ok())
    {
        std_msgs::String msg;
        std::stringstream ss;
        ss << "hello world " << count;
        msg.data = ss.str();
        ROS_INFO("%s", msg.data.c_str());

        if (rand() % 10 <6){
            chatter_pub.publish(msg);
        }

        if (rand() % 10 <2){
            chatter_pub2.publish (msg);
        }

        ros::spinOnce();
        loop_rate.sleep();
        ++count;
    }

    return 0;
}
```

2.3.2 test_sub.cpp

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

class PubSubNode {
private:
    // ROS node handle
    ros::NodeHandle nh_;
    // Subscribers for the "/chatter" and "/chatter2"
       topics
    ros::Subscriber chatter_sub_;
    ros::Subscriber chatter2_sub_;
    // Publisher for the "/rechatter" topic
    ros::Publisher rechatter_pub_;
    // Timer for periodic publishing
    ros::Timer timer_
    // Latest received messages from the subscribed
       topics

    std_msgs::String chatterMsg_;
    std_msgs::String chatter2Msg_;
public:
    // Constructor: sets up subscribers, publisher, and
       timer
    PubSubNode() {
        // Subscribe to topics with a queue size of 1
        chatter_sub_ = nh_.subscribe("/chatter", 1, &PubSubNode::chatterCallback, this);
        chatter2_sub_ = nh_.subscribe("/chatter2", 1, &PubSubNode::chatter2Callback, this);
        // Advertise the publisher on "/rechatter" topic
        rechatter_pub_ = nh_.advertise<std_msgs::String>("/rechatter", 1);
        // Create a timer that triggers every 1 second to publish messages
        timer_ = nh_.createTimer(ros::Duration(1.0), &PubSubNode::timerCallback, this);
    }

    // Callback function for the "/chatter" topic
    void chatterCallback(const std_msgs::String::ConstPtr& msg) {
        chatterMsg_ = *msg;
    }

    // Callback function for the "/chatter2" topic
    void chatter2Callback(const std_msgs::String::ConstPtr& msg) {
        chatter2Msg_ = *msg;
    }

    // Timer callback function: publishes messages periodically
    void timerCallback(const ros::TimerEvent&) {
        // Publish the latest messages from both subscribers
        rechatter_pub_.publish(chatterMsg_);
        rechatter_pub_.publish(chatter2Msg_);
        ROS_INFO("Timer callback executed: messages published.");
    }
};
int main(int argc, char **argv) {
    ros::init(argc, argv, "subscribe_and_publish");
    PubSubNode node;

    ros::spin();

    return 0;
}
```

3 service

3.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(service)

## Find catkin and any catkin packages
find_package(catkin REQUIRED COMPONENTS roscpp std_msgs
   message_generation )

add_service_files(
  FILES
  AddTwoInts.srv
)

generate_messages(
  DEPENDENCIES
  std_msgs
)

## Declare a catkin package
catkin_package(CATKIN_DEPENDS message_runtime)

## Build talker and listener
include_directories(include ${catkin_INCLUDE_DIRS})
add_executable(add_two_int src/add_two_int.cpp)
target_link_libraries(add_two_int ${catkin_LIBRARIES})
add_dependencies(add_two_int ${catkin_EXPORTED_TARGETS})
add_executable(client src/client.cpp)
target_link_libraries(client ${catkin_LIBRARIES})
add_dependencies(client ${catkin_EXPORTED_TARGETS})
```

3.2 package.xml

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

```
  <name>service</name>
  <version>0.0.0</version>
  <description>The service package</description>
  <maintainer email="simone.mentasti@polimi.it">simone</maintainer>
  <license>GPL</license>

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_depend>message_generation</build_depend>

  <build_export_depend>roscpp</build_export_depend>
  <build_export_depend>std_msgs</build_export_depend>

  <exec_depend>roscpp</exec_depend>
  <exec_depend>std_msgs</exec_depend>
  <exec_depend>message_runtime</exec_depend>

  <!-- The export tag contains other, unspecified, tags -->
  <export>
    <!-- Other tools can request additional information be placed here -->
  </export>
</package>
```

3.3 src

3.3.1 add_two_int.cpp

```
#include "ros/ros.h"
#include "service/AddTwoInts.h"

bool add(service::AddTwoInts::Request  &req, service::AddTwoInts::Response &res)
{
  res.sum = req.a + req.b;
  ROS_INFO("request: x=%ld, y=%ld", (long int)req.a, (long int)req.b);
  ROS_INFO("sending back response: [%ld]", (long int)res.sum);
  return true;
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "add_two_ints_server");
  ros::NodeHandle n;
  ros::ServiceServer service = n.advertiseService("
     add_two_ints", add);
  ROS_INFO("Ready to add two ints.");

  ros::spin();

  return 0;
}
```

3.3.2 client.cpp

```
#include "ros/ros.h"
#include "service/AddTwoInts.h"

int main(int argc, char **argv)
{
  ros::init(argc, argv, "add_two_ints_client");

  if (argc != 3)
  {
    ROS_INFO("usage: add_two_ints_client X Y");
    return 1;
  }

  ros::NodeHandle n;
  ros::ServiceClient client = n.serviceClient<service::
     AddTwoInts>("add_two_ints");
  service::AddTwoInts srv;
  srv.request.a = atoll(argv[1]);
  srv.request.b = atoll(argv[2]);

  if (client.call(srv))
  {
    ROS_INFO("Sum: %ld", (long int)srv.response.sum);
  }
  else
  {
    ROS_ERROR("Failed to call service add_two_ints");

    return 1;
  }

  return 0;
}
```

3.4 srv

3.4.1 AddTwoInts.srv

```
int64 a
int64 b
---
int64 sum
```

4 custom_messages

4.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(custom_messages)

## Find catkin and any catkin packages
find_package(catkin REQUIRED COMPONENTS roscpp std_msgs
   message_generation)

add_message_files(
 FILES
 Num.msg
)

generate_messages(
   DEPENDENCIES
  std_msgs
)

```

```
## Declare a catkin package
catkin_package( CATKIN_DEPENDS message_runtime)

## Build talker and listener
include_directories(include ${catkin_INCLUDE_DIRS})
add_executable(pub_custom src/pub.cpp)
add_dependencies(pub_custom
   custom_messages_generate_messages_cpp)

target_link_libraries(pub_custom ${catkin_LIBRARIES})
add_executable(sub_custom src/sub.cpp)

target_link_libraries(sub_custom ${catkin_LIBRARIES})
add_dependencies(sub_custom
   custom_messages_generate_messages_cpp)
```

4.2 package.xml

```
<?xml version="1.0"?>
<package format="2">
  <name>custom_messages</name>
  <version>0.0.0</version>
  <description>The custom mkessage package</description>
  <maintainer email="simone.mentasti@polimi.it">simone</
     maintainer>
  <license>GPL</license>

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_depend>message_generation</build_depend>

  <build_export_depend>roscpp</build_export_depend>
  <build_export_depend>std_msgs</build_export_depend>

  <exec_depend>roscpp</exec_depend>
  <exec_depend>std_msgs</exec_depend>
  <exec_depend>message_runtime</exec_depend>

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

4.3 msg

4.3.1 Num.msg

```
int64 num
```

4.4 src

4.4.1 pub.cpp

```
#include "ros/ros.h"
#include "std_msgs/String.h"
#include "custom_messages/Num.h"
#include <sstream>

int main(int argc, char **argv){
    ros::init(argc, argv, "talker");
    ros::NodeHandle n;
    ros::Publisher chatter_pub = n.advertise<
       custom_messages::Num>("chatter", 1000);

    ros::Rate loop_rate(10);
    int count = 0;

    while (ros::ok()){
        static int i=0;
        i=(i+1)%1000;

        custom_messages::Num msg;
        msg.num =i;

        chatter_pub.publish (msg);
    }

    return 0;
}
```

4.4.2 sub.cpp

```
#include "ros/ros.h"
#include "std_msgs/String.h"
#include "custom_messages/Num.h"

void chatterCallback(const custom_messages::Num::ConstPtr
   & msg){
  ROS_INFO("I heard: [%ld]", msg->num);
}

int main(int argc, char **argv){
    ros::init(argc, argv, "listener");
    ros::NodeHandle n;
    ros::Subscriber sub = n.subscribe("chatter", 1000,
       chatterCallback);

    ros::spin();

    return 0;
}
```

5 timer

5.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(timer)

## Find catkin and any catkin packages
find_package(catkin REQUIRED COMPONENTS roscpp std_msgs)

## Declare a catkin package
catkin_package()

## Build talker and listener
include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(timed_pub src/pub.cpp)
target_link_libraries(timed_pub ${catkin_LIBRARIES})
```

5.2 package.xml

```xml
<?xml version="1.0"?>
<package format="2">
  <name>timer</name>
  <version>0.0.0</version>
  <description>The timer package</description>
  <maintainer email="simone.mentasti@polimi.it">simone</
     maintainer>
  <license>GPL</license>

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>

  <build_export_depend>roscpp</build_export_depend>
  <build_export_depend>std_msgs</build_export_depend>

  <exec_depend>roscpp</exec_depend>
  <exec_depend>std_msgs</exec_depend>

  <!-- The export tag contains other, unspecified, tags
     -->
  <export>
    <!-- Other tools can request additional information
       be placed here -->
  </export>
</package>
```

5.3 src

5.3.1 pub.cpp

```cpp
#include "ros/ros.h"
#include "std_msgs/String.h"
#include <time.h>

void timerCallback(const ros::TimerEvent& ev){
    ROS_INFO_STREAM("Callback called at time: " <<  ros::
       Time::now());
}

```

```
int main(int argc, char **argv){
    ros::init(argc, argv, "timed_talker");
    ros::NodeHandle n;
    ros::Timer timer = n.createTimer(ros::Duration(0.1),
       timerCallback);
    ros::spin();

    return 0;
}
```

6 parameter_test

6.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(parameter_test)

## Find catkin and any catkin packages
find_package(catkin REQUIRED COMPONENTS roscpp std_msgs
   dynamic_reconfigure)

generate_dynamic_reconfigure_options(
  cfg/parameters.cfg
)

## Declare a catkin package
catkin_package()

## Build talker and listener
include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(param_first src/param_first.cpp)
target_link_libraries(param_first ${catkin_LIBRARIES})

add_executable(param_second src/param_second.cpp)

add_dependencies(param_second ${PROJECT_NAME}_gencfg)

target_link_libraries(param_second ${catkin_LIBRARIES})
```

6.2 package.xml

```xml
<?xml version="1.0"?>
<package format="2">
  <name>parameter_test</name>
  <version>0.0.0</version>
  <description>The pub_sub package</description>

  <!-- One maintainer tag required, multiple allowed, one
      person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe
     </maintainer> -->
  <maintainer email="simone@todo.todo">simone</maintainer
     >
  <!-- One license tag required, multiple allowed, one
     license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3,
     LGPLv2.1, LGPLv3 -->
  <license>TODO</license>
  <!-- Url tags are optional, but multiple are allowed,
     one per tag -->
  <!-- Optional attribute type can be: website,
     bugtracker, or repository -->
  <!-- Example: -->
  <!-- <url type="website">http://wiki.ros.org/pub_sub</
     url> -->
  <!-- Author tags are optional, multiple are allowed,
     one per tag -->
  <!-- Authors do not have to be maintainers, but could
     be -->
  <!-- Example: -->
  <!-- <author email="jane.doe@example.com">Jane Doe</
     author> -->
  <!-- The *depend tags are used to specify dependencies
     -->
  <!-- Dependencies can be catkin packages or system
     dependencies -->
  <!-- Examples: -->
  <!-- Use depend as a shortcut for packages that are
     both build and exec dependencies -->
  <!--   <depend>roscpp</depend> -->
  <!--   Note that this is equivalent to the following: -->
  <!--   <build_depend>roscpp</build_depend> -->
  <!--   <exec_depend>roscpp</exec_depend> -->
  <!-- Use build_depend for packages you need at compile time: -->
  <!--   <build_depend>message_generation</build_depend> -->
  <!-- Use build_export_depend for packages you need in order to build against this package: -->
  <!--   <build_export_depend>message_generation</build_export_depend> -->

  <!-- Use buildtool_depend for build tool packages: -->
  <!--   <buildtool_depend>catkin</buildtool_depend> -->
  <!-- Use exec_depend for packages you need at runtime: -->
  <!--   <exec_depend>message_runtime</exec_depend> -->
  <!-- Use test_depend for packages you need only for testing: -->
  <!--   <test_depend>gtest</test_depend> -->
  <!-- Use doc_depend for packages you need only for building documentation: -->

  <!--   <doc_depend>doxygen</doc_depend> -->
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_export_depend>roscpp</build_export_depend>
  <build_export_depend>std_msgs</build_export_depend>
  <exec_depend>roscpp</exec_depend>
  <exec_depend>std_msgs</exec_depend>
  <!-- The export tag contains other, unspecified, tags -->

  <export>
    <!-- Other tools can request additional information be placed here -->
  </export>

</package>
```

6.3 cfg

6.3.1 parameters.cfg

```
#!/usr/bin/env python
PACKAGE = "parameter_test"
from dynamic_reconfigure.parameter_generator_catkin import *

gen = ParameterGenerator()
gen.add("int_param",    int_t,    0, "An Integer parameter", 50,  0, 100)
gen.add("double_param", double_t, 1, "A double parameter", .5, 0,   1)
gen.add("str_param",    str_t,    2, "A string parameter",  "Hello World")
gen.add("bool_param",   bool_t,   3, "A Boolean parameter",  True)
size_enum = gen.enum([ gen.const("Small",      int_t, 0, "A small constant"),
                       gen.const("Medium",     int_t, 1, "A medium constant"),
                       gen.const("Large",      int_t, 2, "A large constant"),
                       gen.const("ExtraLarge", int_t, 3, "An extra large constant")],
                     "An enum to set size")
gen.add("size", int_t, 4, "A size parameter which is edited via an enum", 1, 0, 3, edit_method=size_enum)

exit(gen.generate(PACKAGE, "param_second", "parameters"))
```

6.4 launch

6.4.1 param_set.launch

```
<launch>
  <!-- Global parameter available to all nodes -->
  <param name="global_param" value="global_value" />
  <!-- Launch the node with a private parameter -->

  <node pkg="parameter_test" type="param_first" name="param_first" output="screen">
```

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

6.5 src

6.5.1 param_first.cpp

```
#include <ros/ros.h>
#include <string>

int main(int argc, char **argv)
{
  // Initialize the ROS node with the name "my_node"
  ros::init(argc, argv, "my_node");
  // Create a NodeHandle for global parameters
  ros::NodeHandle nh;
  // Create a NodeHandle in the private namespace for
     node-specific parameters
  ros::NodeHandle private_nh("~");

  // Variables to hold the parameter values
  std::string global_param_value;
  std::string private_param_value;
  // Retrieve a global parameter (set using an absolute
     name like "/global_param")

  if (nh.getParam("/global_param", global_param_value))
  {
    ROS_INFO("Global parameter: %s", global_param_value.
       c_str());
  }
  else
  {
    ROS_WARN("Global parameter not found, using default
       value");
    global_param_value = "default_global";
  }

```

```cpp
  // Retrieve a private (node-specific) parameter (set using the private namespace, e.g., "~private_param")
  if (private_nh.getParam("private_param", private_param_value))
  {
    ROS_INFO("Private parameter: %s", private_param_value.c_str());
  }
  else
  {
    ROS_WARN("Private parameter not found, using default value");
    private_param_value = "default_private";
  }

  // The node can now use these parameters as needed
  ros::spin();

  return 0;
}
```

6.5.2 param_second.cpp

```cpp
#include <ros/ros.h>
#include <dynamic_reconfigure/server.h>
#include <parameter_test/parametersConfig.h>

void callback(parameter_test::parametersConfig &config, uint32_t level) {
  ROS_INFO("Reconfigure Request: %d %f %s %s %d",
            config.int_param, config.double_param,
            config.str_param.c_str(),
            config.bool_param?"True":"False",
            config.size);
            ROS_INFO ("%d",level);
}

int main(int argc, char **argv) {
  ros::init(argc, argv, "param_second");
  dynamic_reconfigure::Server<parameter_test::parametersConfig> server;
dynamic_reconfigure::Server<parameter_test::parametersConfig>::CallbackType f;
  f = boost::bind(&callback, _1, _2);
  server.setCallback(f);
  ROS_INFO("Spinning node");

  ros::spin();

  return 0;
}
```

7 message_filters

7.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(message_filters_example)

## Find catkin and any catkin packages
find_package(catkin REQUIRED COMPONENTS roscpp std_msgs geometry_msgs message_filters)

## Declare a catkin package
catkin_package( CATKIN_DEPENDS geometry_msgs message_filters)

## Build talker and listener
include_directories(include ${catkin_INCLUDE_DIRS})
add_executable(multi_publisher src/pub.cpp)
target_link_libraries(multi_publisher ${catkin_LIBRARIES})
add_executable(filter_subscriber src/sub.cpp)
target_link_libraries(filter_subscriber ${catkin_LIBRARIES})
add_executable(filter_subscriber_policy src/sub_pol.cpp)
target_link_libraries(filter_subscriber_policy ${catkin_LIBRARIES})
```

7.2 package.xml

```xml
<?xml version="1.0"?>
<package format="2">
  <name>message_filters_example</name>
  <version>0.0.0</version>
  <description>The pub_sub package</description>

  <!-- One maintainer tag required, multiple allowed, one person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
  <maintainer email="simone@todo.todo">simone</maintainer>
  <!-- One license tag required, multiple allowed, one license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <license>TODO</license>
  <!-- Url tags are optional, but multiple are allowed, one per tag -->
  <!-- Optional attribute type can be: website, bugtracker, or repository -->
  <!-- Example: -->
  <!-- <url type="website">http://wiki.ros.org/pub_sub</url> -->
  <!-- Author tags are optional, multiple are allowed, one per tag -->
  <!-- Authors do not have to be maintainers, but could be -->
  <!-- Example: -->
  <!-- <author email="jane.doe@example.com">Jane Doe</author> -->
  <!-- The *depend tags are used to specify dependencies -->
  <!-- Dependencies can be catkin packages or system dependencies -->
  <!-- Examples: -->
  <!-- Use depend as a shortcut for packages that are both build and exec dependencies -->
  <!--   <depend>roscpp</depend> -->
  <!--   Note that this is equivalent to the following: -->
  <!--   <build_depend>roscpp</build_depend> -->
  <!--   <exec_depend>roscpp</exec_depend> -->
  <!-- Use build_depend for packages you need at compile time: -->
  <!--   <build_depend>message_generation</build_depend> -->
  <!-- Use build_export_depend for packages you need in order to build against this package: -->
  <!--   <build_export_depend>message_generation</build_export_depend> -->
  <!-- Use buildtool_depend for build tool packages: -->
  <!--   <buildtool_depend>catkin</buildtool_depend> -->
  <!-- Use exec_depend for packages you need at runtime: -->
  <!--   <exec_depend>message_runtime</exec_depend> -->
  <!-- Use test_depend for packages you need only for testing: -->
  <!--   <test_depend>gtest</test_depend> -->
  <!-- Use doc_depend for packages you need only for building documentation: -->
  <!--   <doc_depend>doxygen</doc_depend> -->

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_depend>geometry_msgs</build_depend>
  <build_depend>message_filters</build_depend>

  <build_export_depend>roscpp</build_export_depend>
  <build_export_depend>std_msgs</build_export_depend>
  <build_export_depend>message_filters</build_export_depend>

  <exec_depend>roscpp</exec_depend>
  <exec_depend>std_msgs</exec_depend>
  <exec_depend>geometry_msgs</exec_depend>
  <exec_depend>message_filters</exec_depend>

  <!-- The export tag contains other, unspecified, tags -->
  <export>
```

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

7.3 src

7.3.1 pub.cpp

```
#include "ros/ros.h"
#include "geometry_msgs/Vector3Stamped.h"
#include <sstream>

int main(int argc, char **argv){
    ros::init(argc, argv, "publisher");
    ros::NodeHandle n;
    ros::Publisher pub1 = n.advertise<geometry_msgs::
       Vector3Stamped>("topic1", 1000);
    ros::Publisher pub2 = n.advertise<geometry_msgs::
       Vector3Stamped>("topic2", 1000);

    ros::Rate loop_rate(1);
    int count = 0;

    while (ros::ok()){
        geometry_msgs::Vector3Stamped msg1;
        geometry_msgs::Vector3Stamped msg2;

        msg1.header.stamp = ros::Time::now();
        msg1.header.frame_id = "f1";

        msg1.vector.x = 1;
        msg1.vector.y = 1;
        msg1.vector.z = 1;

        msg2.header.stamp = ros::Time::now();
        msg2.header.frame_id = "f2";

        msg2.vector.x = 2;
        msg2.vector.y = 2;
        msg2.vector.z = 2;

        pub1.publish(msg1);
        pub2.publish(msg2);

        ROS_INFO ("Publishing message");

        ros::spinOnce();
        loop_rate.sleep();
        ++count;
    }

    return 0;
}
```

7.3.2 sub_pol.cpp

```
#include "ros/ros.h"
#include "geometry_msgs/Vector3Stamped.h"
#include <message_filters/subscriber.h>
#include <message_filters/time_synchronizer.h>
#include <message_filters/sync_policies/exact_time.h>
#include <message_filters/sync_policies/approximate_time.
   h>

void callback(const geometry_msgs::Vector3StampedConstPtr
   & msg1, const geometry_msgs::Vector3StampedConstPtr&
   msg2)
{
  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);
}

int main(int argc, char** argv)
{
  ros::init(argc, argv, "subscriber_sync");
  ros::NodeHandle n;

  message_filters::Subscriber<geometry_msgs::
     Vector3Stamped> sub1(n, "topic1", 1);
  message_filters::Subscriber<geometry_msgs::
     Vector3Stamped> sub2(n, "topic2", 1);

  //typedef message_filters::sync_policies::ExactTime<
     geometry_msgs::Vector3Stamped, geometry_msgs::
     Vector3Stamped> MySyncPolicy;
  typedef message_filters::sync_policies::ApproximateTime
     <geometry_msgs::Vector3Stamped, geometry_msgs::
     Vector3Stamped> MySyncPolicy;
  message_filters::Synchronizer<MySyncPolicy> sync(
     MySyncPolicy(10), sub1, sub2);
  sync.registerCallback(boost::bind(&callback, _1, _2));
  ros::spin();

  return 0;
}
```

7.3.3 sub.cpp

```
#include "ros/ros.h"
#include "geometry_msgs/Vector3Stamped.h"
#include <message_filters/subscriber.h>
#include <message_filters/time_synchronizer.h>

void callback(const geometry_msgs::Vector3StampedConstPtr
   & msg1, const geometry_msgs::Vector3StampedConstPtr&
   msg2)
{
  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);
}

int main(int argc, char** argv)
{
  ros::init(argc, argv, "subscriber");
  ros::NodeHandle n;
  message_filters::Subscriber<geometry_msgs::
     Vector3Stamped> sub1(n, "topic1", 1);
  message_filters::Subscriber<geometry_msgs::
     Vector3Stamped> sub2(n, "topic2", 1);
  message_filters::TimeSynchronizer<geometry_msgs::
     Vector3Stamped, geometry_msgs::Vector3Stamped> sync(
     sub1, sub2, 10);
  sync.registerCallback(boost::bind(&callback, _1, _2));

  ros::spin();

  return 0;
}
```

8 tf

8.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(tf_examples)

## Find catkin and any catkin packages
find_package(catkin REQUIRED COMPONENTS roscpp std_msgs
   tf)

## Declare a catkin package
catkin_package()

## Build talker and listener
include_directories(include ${catkin_INCLUDE_DIRS})
add_executable(tf_pub src/pub.cpp)
target_link_libraries(tf_pub ${catkin_LIBRARIES})
add_executable(get_tf src/get_tf.cpp)
target_link_libraries(get_tf ${catkin_LIBRARIES})
```

8.2 package.xml

```
<?xml version="1.0"?>
<package format="2">
  <name>tf_examples</name>
  <version>0.0.0</version>
  <description>The pub_sub package</description>

  <!-- One maintainer tag required, multiple allowed, one
       person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe
     </maintainer> -->
```

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

```
  <!--   <buildtool_depend>catkin</buildtool_depend> -->
  <!-- Use exec_depend for packages you need at runtime: -->
  <!--   <exec_depend>message_runtime</exec_depend> -->
  <!-- Use test_depend for packages you need only for testing: -->
  <!--   <test_depend>gtest</test_depend> -->
  <!-- Use doc_depend for packages you need only for building documentation: -->
  <!--   <doc_depend>doxygen</doc_depend> -->
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_export_depend>roscpp</build_export_depend>
  <build_export_depend>std_msgs</build_export_depend>
  <exec_depend>roscpp</exec_depend>
  <exec_depend>std_msgs</exec_depend>
  <!-- The export tag contains other, unspecified, tags -->

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

8.3 launch

8.3.1 turtle.launch

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

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

8.3.2 turtle.launch.old

```
<launch>
<node pkg="tf_examples" type = "tf_pub" name = "tf_pub"/>
<node pkg="turtlesim" type = "turtlesim_node" name = "
   turtlesim_node"/>
<node pkg="turtlesim" type = "turtle_teleop_key" name = "
   turtle_teleop_key"/>
<node pkg="tf" type="static_transform_publisher" name="
   back_right" args="0.3 -0.3 0 0 0 0 1 turtle FRleg 100"
    />
<node pkg="tf" type="static_transform_publisher" name="
   front_right" args="0.3 0.3 0 0 0 0 1 turtle FLleg 100"
    />
<node pkg="tf" type="static_transform_publisher" name="
   front_left" args="-0.3 0.3 0 0 0 0 1 turtle BLleg 100"
    />
<node pkg="tf" type="static_transform_publisher" name="
   back_left" args="-0.3 -0.3 0 0 0 0 1 turtle BRleg 100"
    />
</launch>
```

8.4 src

8.4.1 get_tf.cpp

```
#include <ros/ros.h>
#include <tf/transform_listener.h>
#include <geometry_msgs/TransformStamped.h>

int main(int argc, char** argv){
  // Initialize the ROS node
  ros::init(argc, argv, "world_to_frleg_listener");
  // Create a ROS node handle
  ros::NodeHandle node;
  // Create a TransformListener object that will listen
     to tf data
  tf::TransformListener listener;

  // Set the rate at which we want to check the
     transformation
  ros::Rate rate(10.0);

  while (node.ok()){
    tf::StampedTransform transform;

    try{
      // Wait for up to 1 second for the transform to
         become available
      listener.waitForTransform("/world", "/FRleg", ros::
         Time(0), ros::Duration(1.0));
      // Look up the transformation from "world" to "
         FRleg"
      listener.lookupTransform("/world", "/FRleg", ros::
         Time(0), transform);
    }
    catch (tf::TransformException &ex) {
      // If there is an exception print the error message
      ROS_ERROR("%s",ex.what());
      ros::Duration(1.0).sleep();
      continue;
    }

    // Print transformation (only pose, but you can get
       the orientation)
    ROS_INFO("Translation: x=%f, y=%f, z=%f",
             transform.getOrigin().x(),
             transform.getOrigin().y(),
             transform.getOrigin().z());

    // Sleep
    rate.sleep();
  }

  return 0;
}
```

8.4.2 pub.cpp

```cpp
#include "ros/ros.h"
#include "turtlesim/Pose.h"
#include <tf/transform_broadcaster.h>

class TfSubPub {
public:
    TfSubPub() {
        // Subscribe to the topic and bind the callback
           method.
        sub = n.subscribe("/turtle1/pose", 1000, &
           TfSubPub::callback, this);
    }

    void callback(const turtlesim::Pose::ConstPtr& msg) {
        // Update the transform's origin with the new
           pose
        transform.setOrigin(tf::Vector3(msg->x, msg->y,
           0));

        // Update the quaternion based on the new theta
           value
        q.setRPY(0, 0, msg->theta);
        transform.setRotation(q);

        // Publish the updated transform
        br.sendTransform(tf::StampedTransform(transform,
           ros::Time::now(), "world", "turtle"));
    }

private:
    ros::NodeHandle n;
    tf::TransformBroadcaster br;
    ros::Subscriber sub;
    // Declare transform and quaternion as class members
       to reuse them in each callback
    tf::Transform transform;
    tf::Quaternion q;
};

int main(int argc, char **argv) {
    ros::init(argc, argv, "subscribe_and_publish");
    TfSubPub myTfSubPub;
    ros::spin();

    return 0;
}
```

9 fibonacci

9.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(actionlib_tutorials)

## Compile as C++11, supported in ROS Kinetic and newer
add_compile_options(-std=c++11)

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED
   COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
  actionlib
  actionlib_msgs
  roscpp
)


## Generate actions in the 'action' folder
add_action_files(
  DIRECTORY action
  FILES Fibonacci.action
)

## Generate added messages and services with any
   dependencies listed here
generate_messages(
  DEPENDENCIES actionlib_msgs std_msgs  # Or other
     packages containing msgs
)

#
```

```
catkin_package(
  CATKIN_DEPENDS actionlib_msgs
)

###########
## Build ##
###########

## Specify additional locations of header files
## Your package locations should be listed before other
   locations
include_directories(
# include
  ${catkin_INCLUDE_DIRS}
)



add_executable(fibonacci_server src/fibonacci_server.cpp)

target_link_libraries(
  fibonacci_server
  ${catkin_LIBRARIES}
)

add_dependencies(
  fibonacci_server
  ${actionlib_tutorials_EXPORTED_TARGETS}
)



add_executable(fibonacci_client src/fibonacci_client.cpp)

target_link_libraries(
  fibonacci_client
  ${catkin_LIBRARIES}
)

add_dependencies(
  fibonacci_client
  ${actionlib_tutorials_EXPORTED_TARGETS}
)
```

```

add_executable(fibonacci_client2 src/fibonacci_client2.
    cpp)

target_link_libraries(
  fibonacci_client2
  ${catkin_LIBRARIES}
)

add_dependencies(
  fibonacci_client2
  ${actionlib_tutorials_EXPORTED_TARGETS}
)
```

9.2 package.xml

```
<?xml version="1.0"?>
<package format="2">
  <name>actionlib_tutorials</name>
  <version>0.0.0</version>
  <description>The fibonacci package</description>

  <!-- One maintainer tag required, multiple allowed, one
      person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe
      </maintainer> -->
  <maintainer email="simone@todo.todo">simone</maintainer
      >


  <!-- One license tag required, multiple allowed, one
      license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3,
      LGPLv2.1, LGPLv3 -->
  <license>TODO</license>


  <!-- Url tags are optional, but multiple are allowed,
      one per tag -->
  <!-- Optional attribute type can be: website,
     bugtracker, or repository -->
  <!-- Example: -->
  <!-- <url type="website">http://wiki.ros.org/
     fibonacci_server</url> -->


  <!-- Author tags are optional, multiple are allowed,
     one per tag -->
  <!-- Authors do not have to be maintainers, but could
     be -->
  <!-- Example: -->
  <!-- <author email="jane.doe@example.com">Jane Doe</
     author> -->


  <!-- The *depend tags are used to specify dependencies
     -->
  <!-- Dependencies can be catkin packages or system
     dependencies -->
  <!-- Examples: -->
  <!-- Use depend as a shortcut for packages that are
     both build and exec dependencies -->
  <!--   <depend>roscpp</depend> -->
  <!--   Note that this is equivalent to the following:
     -->
  <!--   <build_depend>roscpp</build_depend> -->
  <!--   <exec_depend>roscpp</exec_depend> -->
  <!-- Use build_depend for packages you need at compile
     time: -->
  <!--   <build_depend>message_generation</build_depend>
     -->
  <!-- Use build_export_depend for packages you need in
     order to build against this package: -->
  <!--   <build_export_depend>message_generation</
     build_export_depend> -->
  <!-- Use buildtool_depend for build tool packages: -->
  <!--   <buildtool_depend>catkin</buildtool_depend> -->
  <!-- Use exec_depend for packages you need at runtime:
     -->
  <!--   <exec_depend>message_runtime</exec_depend> -->
  <!-- Use test_depend for packages you need only for
     testing: -->
  <!--   <test_depend>gtest</test_depend> -->
  <!-- Use doc_depend for packages you need only for
     building documentation: -->
  <!--   <doc_depend>doxygen</doc_depend> -->
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>actionlib</build_depend>
  <build_depend>actionlib_msgs</build_depend>
  <build_depend>roscpp</build_depend>
  <build_export_depend>actionlib</build_export_depend>
  <build_export_depend>actionlib_msgs</
     build_export_depend>
  <build_export_depend>roscpp</build_export_depend>
  <exec_depend>actionlib</exec_depend>
  <exec_depend>actionlib_msgs</exec_depend>
  <exec_depend>roscpp</exec_depend>
  <exec_depend>message_generation</exec_depend>


  <!-- The export tag contains other, unspecified, tags
     -->
  <export>
    <!-- Other tools can request additional information
       be placed here -->

  </export>
</package>
```

9.3 action

9.3.1 Fibonacci.action

```
#goal definition
int32 order
---
#result definition
int32[] sequence
---
#feedback
int32[] sequence
```

9.4 launch

9.4.1 launcher.launch

```
<launch>
<node name="fibonacci_client" pkg="actionlib_tutorials"
   type="fibonacci_client" output ="screen"/>

<param name="order" value="10" type="int"/>
<param name="duration" value="15" type="double"/>
</launch>
```

9.4.2 launcher2.launch

```
<launch>
<node name="fibonacci_client" pkg="actionlib_tutorials"
   type="fibonacci_client2" output ="screen"/>

<param name="order" value="10" type="int"/>
<param name="duration" value="10" type="double"/>
</launch>
```

9.5 src

9.5.1 fibonacci_client.cpp

```
#include <ros/ros.h>
#include <actionlib/client/simple_action_client.h>
#include <actionlib/client/terminal_state.h>
#include <actionlib_tutorials/FibonacciAction.h>

int main (int argc, char **argv)
{
  ros::init(argc, argv, "test_fibonacci");

  // create the action client
  actionlib::SimpleActionClient<actionlib_tutorials::
     FibonacciAction> ac("fibonacci", true);

  ROS_INFO("Waiting for action server to start.");
  // wait for the action server to start
  ac.waitForServer(); //will wait for infinite time

  ROS_INFO("Action server started, sending goal.");
  // send a goal to the action
  actionlib_tutorials::FibonacciGoal goal;
  int order =10;
  double duration =1.0;
  ros::param::get("order",order);
  ros::param::get("duration",duration);
  goal.order = order;
  ac.sendGoal(goal);

  //wait for the action to return
  bool finished_before_timeout = ac.waitForResult(ros::
     Duration(duration));


  if (finished_before_timeout)
  {
    actionlib::SimpleClientGoalState state = ac.getState
       ();
    ROS_INFO("Action finished: %s",state.toString().c_str
       ());

    actionlib_tutorials::FibonacciResultConstPtr  result
       = ac.getResult();
    for (int i=0; i<result->sequence.size();i++){
       ROS_INFO ("%d ", result->sequence[i]);
    }

  }
  else{
    ROS_INFO("Action did not finish before the time out."
       );
      //ac.cancelGoal ();
  }

  //exit
  return 0;
}
```

9.5.2 fibonacci_client2.cpp

```cpp
#include <ros/ros.h>
#include <actionlib/client/simple_action_client.h>
#include <actionlib/client/terminal_state.h>
#include <actionlib_tutorials/FibonacciAction.h>

typedef actionlib::SimpleActionClient<actionlib_tutorials::FibonacciAction> Client;

void doneCb(const actionlib::SimpleClientGoalState& state,
            const actionlib_tutorials::FibonacciResultConstPtr& result) {
    ROS_INFO("Finished in state [%s]", state.toString().c_str());
    std::stringstream ss;
    for (auto value : result->sequence) {
        ss << value << " ";
    }
    ROS_INFO("Result: %s", ss.str().c_str());
}

void activeCb() {
    ROS_INFO("Goal just went active");
}

void feedbackCb(const actionlib_tutorials::FibonacciFeedbackConstPtr& feedback) {
    ROS_INFO("Got Feedback of length %lu", feedback->sequence.size());
}

void preemptTimerCallback(const ros::TimerEvent&, Client* client) {
    if (client->getState() == actionlib::SimpleClientGoalState::ACTIVE ||
        client->getState() == actionlib::SimpleClientGoalState::PENDING) {
        ROS_INFO("Preempting the current goal due to timeout.");
        client->cancelGoal();
    }
}

int main (int argc, char **argv) {
    ros::init(argc, argv, "test_fibonacci");
    ros::NodeHandle nh;

    Client client("fibonacci", true);
    ROS_INFO("Waiting for action server to start.");
    client.waitForServer();

    ROS_INFO("Action server started, sending goal.");
    actionlib_tutorials::FibonacciGoal goal;
    int order = 10; // Default order
    double duration = 5.0; // Default duration in seconds
    nh.param("order", order, 10); // Retrieve order if specified in parameters
    nh.param("duration", duration, 5.0); // Retrieve duration if specified in parameters

    goal.order = order;
    client.sendGoal(goal, &doneCb, &activeCb, &feedbackCb);

    // Setup a timer to preempt the goal after the specified duration
    ros::Timer timer = nh.createTimer(ros::Duration(duration), boost::bind(preemptTimerCallback, _1, &client), true);

    ros::Rate loop_rate(1);

    while (ros::ok()){
          ROS_INFO("doing other processing");
        ros::spinOnce();

        loop_rate.sleep();
    }

    return 0;
}
```

9.5.3 fibonacci_server.cpp

```cpp
#include <ros/ros.h>
#include <actionlib/server/simple_action_server.h>
#include <actionlib_tutorials/FibonacciAction.h>

class FibonacciAction
{
private:
  ros::NodeHandle nh_;
  actionlib::SimpleActionServer<actionlib_tutorials::FibonacciAction> as_;
  std::string action_name_;
  // create messages that are used to published feedback/result
  actionlib_tutorials::FibonacciFeedback feedback_;
  actionlib_tutorials::FibonacciResult result_;

public:

  FibonacciAction(std::string name) :
    as_(nh_, name, boost::bind(&FibonacciAction::executeCB, this, _1), false),
    action_name_(name)
  {
    as_.start();
  }

  ~FibonacciAction(void)
  {
  }

  void executeCB(const actionlib_tutorials::FibonacciGoalConstPtr &goal)
  {
    // helper variables
    ros::Rate r(1); //simulate compute time
    bool success = true;

    // clear and set first two values
    feedback_.sequence.clear();
    feedback_.sequence.push_back(0);
    feedback_.sequence.push_back(1);

    // publish info to the console for the user
    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]);

    // start executing the action
    for(int i=1; i<=goal->order; i++)
    {
      // check that preempt has not been requested by the client
      if (as_.isPreemptRequested() || !ros::ok())
      {
        ROS_INFO("%s: Preempted", action_name_.c_str());
        // set the action state to preempted
        as_.setPreempted();
        success = false;
        break;
      }
      feedback_.sequence.push_back(feedback_.sequence[i] + feedback_.sequence[i-1]);
      // publish the feedback
      as_.publishFeedback(feedback_);
      // this sleep is not necessary, we simulate compute time
      r.sleep();
    }

    if(success)
    {
      result_.sequence = feedback_.sequence;
      ROS_INFO("%s: Succeeded", action_name_.c_str());
      // set the action state to succeeded
      as_.setSucceeded(result_);
    }
  }


};


int main(int argc, char** argv)
```

```
{
  ros::init(argc, argv, "fibonacci");

  FibonacciAction fibonacci("fibonacci");
  ros::spin();

  return 0;
}
```

10 pub_latched

10.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(pub_latched)

## Find catkin and any catkin packages
find_package(catkin REQUIRED COMPONENTS roscpp std_msgs)


## Declare a catkin package
catkin_package()

## Build talker and listener
include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(pub_latched src/pub.cpp)
target_link_libraries(pub_latched ${catkin_LIBRARIES})
```

10.2 package.xml

```
<?xml version="1.0"?>
<package format="2">
  <name>pub_latched</name>
  <version>0.0.0</version>
  <description>The latched package</description>

  <!-- One maintainer tag required, multiple allowed, one
      person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe
     </maintainer> -->
  <maintainer email="simone@todo.todo">simone</maintainer
     >

  <!-- One license tag required, multiple allowed, one
     license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3,
     LGPLv2.1, LGPLv3 -->
  <license>TODO</license>

  <!-- Url tags are optional, but multiple are allowed, one per tag -->
  <!-- Optional attribute type can be: website, bugtracker, or repository -->
  <!-- Example: -->
  <!-- <url type="website">http://wiki.ros.org/pub_sub</url> -->

  <!-- Author tags are optional, multiple are allowed, one per tag -->
  <!-- Authors do not have to be maintainers, but could be -->
  <!-- Example: -->
  <!-- <author email="jane.doe@example.com">Jane Doe</author> -->

  <!-- The *depend tags are used to specify dependencies -->
  <!-- Dependencies can be catkin packages or system dependencies -->
  <!-- Examples: -->
  <!-- Use depend as a shortcut for packages that are both build and exec dependencies -->
  <!--   <depend>roscpp</depend> -->
  <!--   Note that this is equivalent to the following: -->
  <!--   <build_depend>roscpp</build_depend> -->
  <!--   <exec_depend>roscpp</exec_depend> -->
  <!-- Use build_depend for packages you need at compile time: -->
  <!--   <build_depend>message_generation</build_depend> -->
  <!-- Use build_export_depend for packages you need in order to build against this package: -->
  <!--   <build_export_depend>message_generation</build_export_depend> -->
  <!-- Use buildtool_depend for build tool packages: -->
  <!--   <buildtool_depend>catkin</buildtool_depend> -->
  <!-- Use exec_depend for packages you need at runtime: -->
  <!--   <exec_depend>message_runtime</exec_depend> -->
  <!-- Use test_depend for packages you need only for
     testing: -->
  <!--   <test_depend>gtest</test_depend> -->
  <!-- Use doc_depend for packages you need only for
     building documentation: -->
  <!--   <doc_depend>doxygen</doc_depend> -->
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_export_depend>roscpp</build_export_depend>
  <build_export_depend>std_msgs</build_export_depend>
  <exec_depend>roscpp</exec_depend>
  <exec_depend>std_msgs</exec_depend>


  <!-- The export tag contains other, unspecified, tags
     -->
  <export>
    <!-- Other tools can request additional information
       be placed here -->

  </export>
</package>
```

10.3 src

10.3.1 pub.cpp

```
#include "ros/ros.h"
#include "std_msgs/String.h"
#include <sstream>

int main(int argc, char **argv){

    ros::init(argc, argv, "talker");
    ros::NodeHandle n;

    ros::Publisher chatter_pub = n.advertise<std_msgs::
       String>("chatter", 1, true);

    ros::Rate loop_rate(0.05);

    int count = 0;

    while (ros::ok()){

            std_msgs::String msg;

            std::stringstream ss;
            ss << "hello world " << count;
            msg.data = ss.str();

            ROS_INFO("%s", msg.data.c_str());

            chatter_pub.publish(msg);

            ros::spinOnce();

            loop_rate.sleep();
            ++count;
    }

    return 0;
}
```

10.4 launch

10.4.1 launcher.launch

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

11 asynch

11.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(sub_asynch)

```

```
## Find catkin and any catkin packages
find_package(catkin REQUIRED COMPONENTS roscpp std_msgs)

## Declare a catkin package
catkin_package()

## Build talker and listener
include_directories(include ${catkin_INCLUDE_DIRS})

add_executable(standard_pub src/pub.cpp)
target_link_libraries(standard_pub ${catkin_LIBRARIES})

add_executable(standard_sub src/standard_sub.cpp)
target_link_libraries(standard_sub ${catkin_LIBRARIES})

add_executable(asynch_sub src/asynch_sub.cpp)
target_link_libraries(asynch_sub ${catkin_LIBRARIES})
```

11.2 package.xml

```
<?xml version="1.0"?>
<package format="2">
  <name>sub_asynch</name>
  <version>0.0.0</version>
  <description>The asynch package</description>

  <!-- One maintainer tag required, multiple allowed, one
      person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe
     </maintainer> -->
  <maintainer email="simone@todo.todo">simone</maintainer
     >

  <!-- One license tag required, multiple allowed, one
     license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3,
     LGPLv2.1, LGPLv3 -->
  <license>TODO</license>

  <!-- Url tags are optional, but multiple are allowed, one per tag -->
  <!-- Optional attribute type can be: website, bugtracker, or repository -->
  <!-- Example: -->
  <!-- <url type="website">http://wiki.ros.org/pub_sub</url> -->


  <!-- Author tags are optional, multiple are allowed, one per tag -->
  <!-- Authors do not have to be maintainers, but could be -->
  <!-- Example: -->
  <!-- <author email="jane.doe@example.com">Jane Doe</author> -->

  <!-- The *depend tags are used to specify dependencies -->
  <!-- Dependencies can be catkin packages or system dependencies -->
  <!-- Examples: -->
  <!-- Use depend as a shortcut for packages that are both build and exec dependencies -->
  <!--   <depend>roscpp</depend> -->
  <!--   Note that this is equivalent to the following: -->
  <!--   <build_depend>roscpp</build_depend> -->
  <!--   <exec_depend>roscpp</exec_depend> -->
  <!-- Use build_depend for packages you need at compile time: -->
  <!--   <build_depend>message_generation</build_depend> -->
  <!-- Use build_export_depend for packages you need in order to build against this package: -->
  <!--   <build_export_depend>message_generation</build_export_depend> -->
  <!-- Use buildtool_depend for build tool packages: -->
  <!--   <buildtool_depend>catkin</buildtool_depend> -->
  <!-- Use exec_depend for packages you need at runtime: -->
  <!--   <exec_depend>message_runtime</exec_depend> -->
  <!-- Use test_depend for packages you need only for
     testing: -->
  <!--   <test_depend>gtest</test_depend> -->
  <!-- Use doc_depend for packages you need only for
     building documentation: -->
  <!--   <doc_depend>doxygen</doc_depend> -->
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_export_depend>roscpp</build_export_depend>
  <build_export_depend>std_msgs</build_export_depend>
  <exec_depend>roscpp</exec_depend>
  <exec_depend>std_msgs</exec_depend>


  <!-- The export tag contains other, unspecified, tags
     -->
  <export>
    <!-- Other tools can request additional information
       be placed here -->

  </export>
</package>
```

11.3 src

11.3.1 pub.cpp

```
#include "ros/ros.h"
#include "std_msgs/String.h"
#include <sstream>

int main(int argc, char **argv){
    ros::init(argc, argv, "talker");
    ros::NodeHandle n;
    ros::Publisher chatter_pub1 = n.advertise<std_msgs::
       String>("talker1", 1);
    ros::Publisher chatter_pub2 = n.advertise<std_msgs::
       String>("talker2", 1);
    ros::Rate loop_rate(3.0);

    std_msgs::String msg1;
    std_msgs::String msg2;

     int count = 0;

    while (ros::ok()){

      std::stringstream ss1;

      std::stringstream ss2;

      ss1 << "Hey 1:" << count;
        msg1.data = ss1.str();

        ss2 << "Hey 2:" << count;
        msg2.data = ss2.str();

      count++;

        chatter_pub1.publish(msg1);
        chatter_pub2.publish(msg2);

        ros::spinOnce();

        loop_rate.sleep();
    }

    return 0;
}
```

11.3.2 standard_sub.cpp

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

```

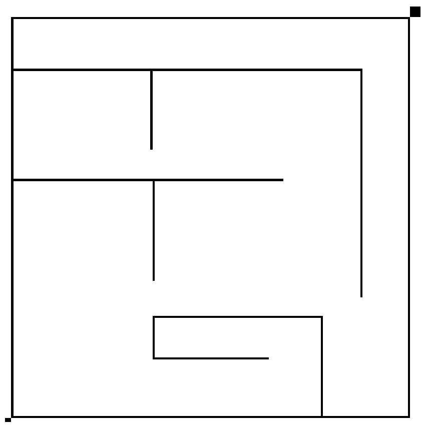
```cpp
void callbackTalker2(const std_msgs::String::ConstPtr &
    msg)
{
    ROS_INFO_STREAM("Message from callback 2:");
    ROS_INFO("%s", msg->data.c_str());
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "talker_subscribers");
    ros::NodeHandle nh;
    ros::Subscriber counter1_sub = nh.subscribe("talker1"
        , 1, callbackTalker1);
    ros::Subscriber counter2_sub = nh.subscribe("talker2"
        , 1, callbackTalker2);
    ros::spin();
}
```

11.3.3 asynch_sub.cpp

```cpp
#include <ros/ros.h>
#include <std_msgs/String.h>

void callbackTalker1(const std_msgs::String::ConstPtr &
    msg)
{
    ROS_INFO_STREAM("Message from callback 1:" );
    ros::Duration(2.0).sleep();
    ROS_INFO("%s", msg->data.c_str());
}

void callbackTalker2(const std_msgs::String::ConstPtr &
    msg)
{
    ROS_INFO_STREAM("Message from callback 2:");
    ROS_INFO("%s", msg->data.c_str());
}

int main(int argc, char **argv)
{
    ros::init(argc, argv, "talker_subscribers_asynch");
    ros::NodeHandle nh;

    ros::AsyncSpinner spinner(0); // num of thread
    spinner.start();

    ros::Subscriber counter1_sub = nh.subscribe("talker1"
        , 1, callbackTalker1);
    ros::Subscriber counter2_sub = nh.subscribe("talker2"
        , 1, callbackTalker2);
    ros::waitForShutdown();
}
```

12 stage

12.1 maze.png



12.2 maze.world

```
include "turtlebot.inc"

define floorplan model
(
  # sombre, sensible, artistic
  color "gray30"

  # most maps will need a bounding box
  boundary 1

  gui_nose 0
  gui_grid 0
  gui_outline 0
  gripper_return 0
  fiducial_return 0
  ranger_return 1
)

resolution 0.02
interval_sim 100  # simulation timestep in milliseconds

window
(
  size [ 600.0 700.0 ]
  center [ 0.0 0.0 ]
  rotate [ 0.0 0.0 ]
  scale 60
)

floorplan
(
  name "maze"
  bitmap "maze.png"
  size [ 10.0 10.0 2.0 ]
  pose [  5.0  5.0 0.0 0.0 ]
)

# throw in a robot
turtlebot
(
  pose [ 2.0 2.0 0.0 0.0 ]
  name "turtlebot"
  color "black"
)
```

12.3 turtlebot.inc

```
define kinect ranger
(
  sensor
  (
    pose [ -0.1 0.0 -0.11 0.0 ]
    size [0.1 0.1 0.1 ]
    range  [0 6.5]
    fov 120.0
    samples 640
  )
  # generic model properties
  color "black"
  size [ 0.06 0.15 0.03 ]
)

define turtlebot position
(
  pose [ 0.0 0.0 0.0 0.0 ]

  localization "odom"


  odom_error [0.2 0.2 0.0 0.3 ]

  size [ 0.2552 0.2552 0.40 ]
  origin [ 0.0 0.0 0.0 0.0 ]
  gui_nose 1
  drive "diff"
  color "grey"

  kinect()
)
```

13 demo_mapping

13.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(demo_mapping)

## Find catkin and any catkin packages
find_package(catkin REQUIRED COMPONENTS roscpp std_msgs)

## Declare a catkin package
catkin_package()
```

13.2 package.xml

```xml
<?xml version="1.0"?>
<package format="2">
  <name>demo_mapping</name>
  <version>0.0.0</version>
  <description>The demo_mapping package</description>

  <!-- One maintainer tag required, multiple allowed, one person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
  <maintainer email="simone@todo.todo">simone</maintainer>


  <!-- One license tag required, multiple allowed, one license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3, LGPLv2.1, LGPLv3 -->
  <license>TODO</license>


  <!-- Url tags are optional, but multiple are allowed, one per tag -->
  <!-- Optional attribute type can be: website, bugtracker, or repository -->
  <!-- Example: -->
  <!-- <url type="website">http://wiki.ros.org/pub_sub</url> -->


  <!-- Author tags are optional, multiple are allowed, one per tag -->
  <!-- Authors do not have to be maintainers, but could be -->
  <!-- Example: -->
  <!-- <author email="jane.doe@example.com">Jane Doe</author> -->


  <!-- The *depend tags are used to specify dependencies -->
  <!-- Dependencies can be catkin packages or system dependencies -->
  <!-- Examples: -->
  <!-- Use depend as a shortcut for packages that are both build and exec dependencies -->
  <!--   <depend>roscpp</depend> -->
  <!--   Note that this is equivalent to the following: -->
  <!--   <build_depend>roscpp</build_depend> -->
  <!--   <exec_depend>roscpp</exec_depend> -->
  <!-- Use build_depend for packages you need at compile time: -->
  <!--   <build_depend>message_generation</build_depend> -->
  <!-- Use build_export_depend for packages you need in order to build against this package: -->
  <!--   <build_export_depend>message_generation</build_export_depend> -->
  <!-- Use buildtool_depend for build tool packages: -->
  <!--   <buildtool_depend>catkin</buildtool_depend> -->
  <!-- Use exec_depend for packages you need at runtime: -->
  <!--   <exec_depend>message_runtime</exec_depend> -->
  <!-- Use test_depend for packages you need only for testing: -->
  <!--   <test_depend>gtest</test_depend> -->
  <!-- Use doc_depend for packages you need only for building documentation: -->
  <!--   <doc_depend>doxygen</doc_depend> -->
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_export_depend>roscpp</build_export_depend>
  <build_export_depend>std_msgs</build_export_depend>
  <exec_depend>roscpp</exec_depend>
  <exec_depend>std_msgs</exec_depend>


  <!-- The export tag contains other, unspecified, tags -->
  <export>
```

```
    <!-- Other tools can request additional information
       be placed here -->

  </export>
</package>
```

13.3 launch

13.3.1 gmapping.launch

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

13.3.2 slam_toolbox.launch

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

14 nav2d_conf

14.1 CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
project(nav2d_conf)

## Compile as C++11, supported in ROS Kinetic and newer
# add_compile_options(-std=c++11)

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
  move_base
  std_msgs
)

## System dependencies are found with CMake's conventions
# find_package(Boost REQUIRED COMPONENTS system)


## Uncomment this if the package has a setup.py. This macro ensures
## modules and global scripts declared therein get installed
## See http://ros.org/doc/api/catkin/html/user_guide/setup_dot_py.html
# catkin_python_setup()

################################################
## Declare ROS messages, services and actions ##
################################################

## To declare and build messages, services or actions from within this
## package, follow these steps:
## * Let MSG_DEP_SET be the set of packages whose message types you use in
##   your messages/services/actions (e.g. std_msgs, actionlib_msgs, ...).
## * In the file package.xml:
##   * add a build_depend tag for "message_generation"
##   * add a build_depend and a run_depend tag for each package in MSG_DEP_SET
##   * If MSG_DEP_SET isn't empty the following dependency has been pulled in
##     but can be declared for certainty nonetheless:
##     * add a run_depend tag for "message_runtime"
## * In this file (CMakeLists.txt):
##   * add "message_generation" and every package in MSG_DEP_SET to
##     find_package(catkin REQUIRED COMPONENTS ...)
##   * add "message_runtime" and every package in MSG_DEP_SET to
##     catkin_package(CATKIN_DEPENDS ...)
##   * uncomment the add_*_files sections below as needed
##     and list every .msg/.srv/.action file to be processed
##   * uncomment the generate_messages entry below
##   * add every package in MSG_DEP_SET to generate_messages(DEPENDENCIES ...)

## Generate messages in the 'msg' folder
# add_message_files(
#   FILES
#   Message1.msg
#   Message2.msg
# )

## Generate services in the 'srv' folder
# add_service_files(
#   FILES
#   Service1.srv
#   Service2.srv
# )

## Generate actions in the 'action' folder
# add_action_files(
#   FILES
#   Action1.action
#   Action2.action
# )

## Generate added messages and services with any dependencies listed here
# generate_messages(
#   DEPENDENCIES
#   std_msgs
# )

################################################
## Declare ROS dynamic reconfigure parameters ##
################################################

```

```
## To declare and build dynamic reconfigure parameters
   within this
## package, follow these steps:
## * In the file package.xml:
##   * add a build_depend and a run_depend tag for "
   dynamic_reconfigure"
## * In this file (CMakeLists.txt):
##   * add "dynamic_reconfigure" to
##     find_package(catkin REQUIRED COMPONENTS ...)
##   * uncomment the "
   generate_dynamic_reconfigure_options" section below
##     and list every .cfg file to be processed

## Generate dynamic reconfigure parameters in the 'cfg'
   folder
# generate_dynamic_reconfigure_options(
#   cfg/DynReconf1.cfg
#   cfg/DynReconf2.cfg
# )

###################################
## catkin specific configuration ##
###################################
## The catkin_package macro generates cmake config files
   for your package
## Declare things to be passed to dependent projects
## INCLUDE_DIRS: uncomment this if your package contains
   header files
## LIBRARIES: libraries you create in this project that
   dependent projects also need
## CATKIN_DEPENDS: catkin_packages dependent projects
   also need
## DEPENDS: system dependencies of this project that
   dependent projects also need
catkin_package(
#  INCLUDE_DIRS include
#  LIBRARIES 2dnav_conf
#  CATKIN_DEPENDS move_base std_msgs
#  DEPENDS system_lib
)

###########
## Build ##
```

```
###########

## Specify additional locations of header files
## Your package locations should be listed before other locations
include_directories(
# include
  ${catkin_INCLUDE_DIRS}
)

## Declare a C++ library
# add_library(${PROJECT_NAME}
#   src/${PROJECT_NAME}/2dnav_conf.cpp
# )

## Add cmake target dependencies of the library
## as an example, code may need to be generated before libraries
## either from message generation or dynamic reconfigure
# add_dependencies(${PROJECT_NAME} ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})

## Declare a C++ executable
## With catkin_make all packages are built within a single CMake context
## The recommended prefix ensures that target names across packages don't collide
# add_executable(${PROJECT_NAME}_node src/2dnav_conf_node.cpp)

## Rename C++ executable without prefix
## The above recommended prefix causes long target names, the following renames the
## target back to the shorter version for ease of user use
## e.g. "rosrun someones_pkg node" instead of "rosrun someones_pkg someones_pkg_node"
# set_target_properties(${PROJECT_NAME}_node PROPERTIES OUTPUT_NAME node PREFIX "")

## Add cmake target dependencies of the executable
## same as for the library above
```

```
# add_dependencies(${PROJECT_NAME}_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})

## Specify libraries to link a library or executable target against
# target_link_libraries(${PROJECT_NAME}_node
#   ${catkin_LIBRARIES}
# )

#############
## Install ##
#############

# all install targets should use catkin DESTINATION variables
# See http://ros.org/doc/api/catkin/html/adv_user_guide/variables.html

## Mark executable scripts (Python etc.) for installation
## in contrast to setup.py, you can choose the destination
# install(PROGRAMS
#   scripts/my_python_script
#   DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
# )

## Mark executables and/or libraries for installation
# install(TARGETS ${PROJECT_NAME} ${PROJECT_NAME}_node
#   ARCHIVE DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
#   LIBRARY DESTINATION ${CATKIN_PACKAGE_LIB_DESTINATION}
#   RUNTIME DESTINATION ${CATKIN_PACKAGE_BIN_DESTINATION}
# )

## Mark cpp header files for installation
# install(DIRECTORY include/${PROJECT_NAME}/
#   DESTINATION ${CATKIN_PACKAGE_INCLUDE_DESTINATION}
#   FILES_MATCHING PATTERN "*.h"
#   PATTERN ".svn" EXCLUDE
# )

## Mark other files for installation (e.g. launch and bag files, etc.)
# install(FILES
#   # myfile1
#   # myfile2
#   DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}
# )

#############
## Testing ##
#############

## Add gtest based cpp test target and link libraries
# catkin_add_gtest(${PROJECT_NAME}-test test/
   test_2dnav_conf.cpp)
# if(TARGET ${PROJECT_NAME}-test)
#   target_link_libraries(${PROJECT_NAME}-test ${
   PROJECT_NAME})
# endif()

## Add folders to be run by python nosetests
# catkin_add_nosetests(test)
```

14.2 package.xml

```
<?xml version="1.0"?>
<package format="2">
  <name>nav2d_conf</name>
  <version>0.0.0</version>
  <description>The 2dnav_conf package</description>

  <!-- One maintainer tag required, multiple allowed, one
      person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe
     </maintainer> -->
  <maintainer email="alessandro@todo.todo">alessandro</
     maintainer>

  <!-- One license tag required, multiple allowed, one
     license per tag -->
  <!-- Commonly used license strings: -->
  <!--   BSD, MIT, Boost Software License, GPLv2, GPLv3,
     LGPLv2.1, LGPLv3 -->
  <license>TODO</license>

  <!-- Url tags are optional, but multiple are allowed,
     one per tag -->
  <!-- Optional attribute type can be: website,
     bugtracker, or repository -->
  <!-- Example: -->
  <!-- <url type="website">http://wiki.ros.org/2dnav_conf
     </url> -->


  <!-- Author tags are optional, multiple are allowed,
     one per tag -->
  <!-- Authors do not have to be maintainers, but could
     be -->
  <!-- Example: -->
  <!-- <author email="jane.doe@example.com">Jane Doe</
     author> -->

  <!-- The *depend tags are used to specify dependencies
     -->
  <!-- Dependencies can be catkin packages or system
     dependencies -->
  <!-- Examples: -->
  <!-- Use depend as a shortcut for packages that are
     both build and exec dependencies -->
  <!--   <depend>roscpp</depend> -->
  <!--   Note that this is equivalent to the following:
     -->
  <!--   <build_depend>roscpp</build_depend> -->
  <!--   <exec_depend>roscpp</exec_depend> -->
  <!-- Use build_depend for packages you need at compile
     time: -->
  <!--   <build_depend>message_generation</build_depend>
     -->
  <!-- Use build_export_depend for packages you need in
     order to build against this package: -->
  <!--   <build_export_depend>message_generation</
     build_export_depend> -->
  <!-- Use buildtool_depend for build tool packages: -->
  <!--   <buildtool_depend>catkin</buildtool_depend> -->
  <!-- Use exec_depend for packages you need at runtime:
     -->
  <!--   <exec_depend>message_runtime</exec_depend> -->
  <!-- Use test_depend for packages you need only for
     testing: -->
  <!--   <test_depend>gtest</test_depend> -->
  <!-- Use doc_depend for packages you need only for
     building documentation: -->
  <!--   <doc_depend>doxygen</doc_depend> -->
  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>move_base</build_depend>
  <build_depend>std_msgs</build_depend>
  <build_export_depend>move_base</build_export_depend>
  <build_export_depend>std_msgs</build_export_depend>
  <exec_depend>move_base</exec_depend>
  <exec_depend>std_msgs</exec_depend>


  <!-- The export tag contains other, unspecified, tags
     -->
  <export>
    <!-- Other tools can request additional information
       be placed here -->

  </export>
</package>
```

14.3 cfg

14.3.1 costmap_common_params.yaml

```
# Obstacle Cost Shaping (http://wiki.ros.org/costmap_2d/
   hydro/inflation)
robot_radius: 0.20  # distance a circular robot should be
    clear of the obstacle (kobuki: 0.18)
# footprint: [[x0, y0], [x1, y1], ... [xn, yn]]  # if the
    robot is not circular

map_type: costmap_2d

obstacle_layer:
  enabled:              true
  unknown_threshold:    15
  mark_threshold:       0
  combination_method:   1
  track_unknown_space:  true    #true needed for disabling global path planning through unknown space
  obstacle_range: 2.5  # maximum range in meters at which to insert obstacles into the costmap using sensor data
  raytrace_range: 3.0 # maximum range in meters at which to raytrace out obstacles from the map using sensor data
  observation_sources:  scan
  scan:
    data_type: LaserScan
    topic: scan
    marking: true
    clearing: true
    min_obstacle_height: 0.25
    max_obstacle_height: 0.35


#cost_scaling_factor and inflation_radius were now moved to the inflation_layer ns
inflation_layer:
  enabled:              true
  cost_scaling_factor:  5.0  # exponential rate at which the obstacle cost drops off (default: 10)
  inflation_radius:     0.6  # max. distance from an obstacle at which costs are incurred for planning paths.

static_layer:
  enabled:              true
```

14.3.2 dwa_local_planner_params.yaml

```
DWAPlannerROS:

# Robot Configuration Parameters - Kobuki
  max_vel_x: 0.5  # 0.55
  min_vel_x: 0.0

  max_vel_y: 0.0  # diff drive robot
  min_vel_y: 0.0  # diff drive robot

  max_trans_vel: 0.4 # choose slightly less than the base's capability
  min_trans_vel: 0.1  # this is the min trans velocity when there is negligible rotational velocity
  trans_stopped_vel: 0.1

  # Warning!
  #   do not set min_trans_vel to 0.0 otherwise dwa will always think translational velocities
  #   are non-negligible and small in place rotational velocities will be created.

  max_rot_vel: 3.0  # choose slightly less than the base's capability
  min_rot_vel: 0.4  # this is the min angular velocity when there is negligible translational velocity
  rot_stopped_vel: 0.4

  acc_lim_x: 0.5 # maximum is theoretically 2.0, but we don't want to crash/do strange stuff/overshoot
  acc_lim_theta: 1.0 #rad
  acc_lim_y: 0.0      # diff drive robot

# Goal Tolerance Parameters
  yaw_goal_tolerance: 0.3  # 0.05
  xy_goal_tolerance: 0.10  # 0.10


# Forward Simulation Parameters
  sim_time: 2.0       # 1.7 The amount of time to forward-simulate trajectories in seconds
  vx_samples: 3       # 3 The number of samples to use when exploring the x velocity space
  vy_samples: 10       # diff drive robot
  vtheta_samples: 20  # 20

# Trajectory Scoring Parameters
#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))
```

```
  path_distance_bias: 10.0      # 32.0   - weighting for how much it should stick to the global path plan
  goal_distance_bias: 24.0      # 24.0   - wighting for how much it should attempt to reach its goal
  occdist_scale: 0.2            # 0.01   - weighting for how much the controller should avoid obstacles
  forward_point_distance: 0.25 # 0.325  - how far along to place an additional scoring point
  stop_time_buffer: 0.2         # 0.2    - amount of time a robot must stop in before colliding for a valid traj.
  scaling_speed: 0.25           # 0.25   - absolute velocity at which to start scaling the robot's footprint
  max_scaling_factor: 0.2       # 0.2    - how much to scale the robot's footprint when at speed.

# Oscillation Prevention Parameters
  oscillation_reset_dist: 0.15  # 0.05   - how far to travel before resetting oscillation flags

# Debugging
  publish_traj_pc : true
  publish_cost_grid_pc: true
  publish_cost_grid: true
  global_frame_id: odom
```

14.3.3 global_costmap_params.yaml

```
global_costmap:

   global_frame: map
   robot_base_frame: base_footprint
   update_frequency: 1.0
   publish_frequency: 0.5
   static_map: true
   transform_tolerance: 0.5
   plugins:
     - {name: static_layer,            type: "costmap_2d::StaticLayer"}
     - {name: inflation_layer,         type: "costmap_2d::InflationLayer"}
```

14.3.4 local_costmap_params.yaml

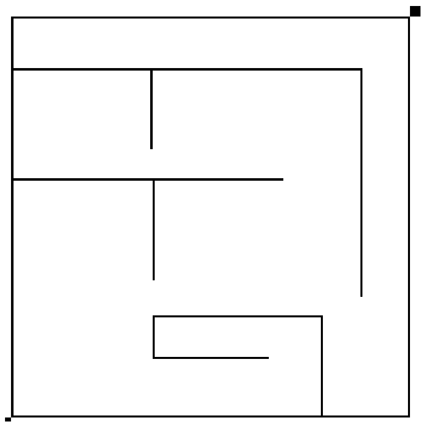
```
local_costmap:
   global_frame: odom
   robot_base_frame: base_footprint
   update_frequency: 5.0
   publish_frequency: 2.0
   static_map: false
   rolling_window: true
   width: 6.0
   height: 6.0
   resolution: 0.05
   transform_tolerance: 0.5 #maximum amount of latency allowed between tf
   plugins:
    - {name: obstacle_layer,      type: "costmap_2d::VoxelLayer"}
    - {name: inflation_layer,     type: "costmap_2d::InflationLayer"}
```

14.3.5 move_base_params.yaml

```
shutdown_costmaps: false

controller_frequency: 5.0 #5
controller_patience: 3.0  #3

planner_frequency: 1.0
planner_patience: 5.0 #5

oscillation_timeout: 10.0
oscillation_distance: 0.2

# local planner - default is trajectory rollout
base_local_planner: "dwa_local_planner/DWAPlannerROS"

base_global_planner: "navfn/NavfnROS" #alternatives: global_planner/GlobalPlanner, carrot_planner/CarrotPlanner navfn/NavfnROS
```

14.4 maps

14.4.1 maze.png



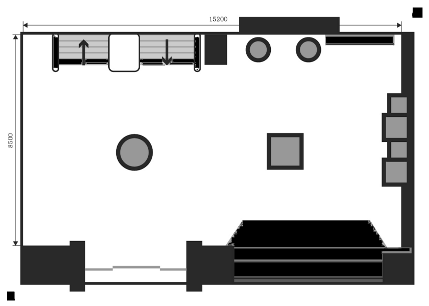
14.4.2 maze.yaml

```
image: maze.png
resolution: 0.05
origin: [0.0, 0.0, 0.0]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196
```

14.4.3 robopark_plan.yaml

```
image: robopark2.bmp
resolution: 0.014
origin: [-0.6, -2.24, 0.0] #The 2-D pose of the lower-
   left pixel in the map, as (x, y, yaw)
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196

# 2150x700 pix
# 491 pix -> 13.900 m
# res = 0.0283 pix/m
# 1012 >>> 28.6396
# 340  >>> 9.622
```

14.4.4 robopark2.bmp



14.4.5 stage

maze.world

```
include "turtlebot.inc"

define floorplan model
(
  # sombre, sensible, artistic
  color "gray30"

  # most maps will need a bounding box
  boundary 1

  gui_nose 0
  gui_grid 0
  gui_outline 0
  gripper_return 0
  fiducial_return 0
  laser_return 1
)

resolution 0.02
interval_sim 100  # simulation timestep in milliseconds

window
(
  size [ 600.0 700.0 ]
  center [ 0.0 0.0 ]
  rotate [ 0.0 0.0 ]
  scale 60
)
```

```

floorplan
(
  name "maze"
  bitmap "../maze.png"
  size [ 10.0 10.0 2.0 ]
  pose [  5.0  5.0 0.0 0.0 ]
)

# throw in a robot
turtlebot
(
  pose [ 2.0 2.0 0.0 0.0 ]
  name "turtlebot"
  color "black"
)
```

robopark_plan.world

```
include "turtlebot.inc"

# Definition for an obstacle placed on the map.
define block model
(
  size [0.500 0.500 1.500]
  gui_nose 0
)

# throw in a robot
turtlebot
(
  pose [ 2.000 2.000 0.000 0.000 ]
  name "turtlebot1"
  color "gray"
  gui_nose 1
)

# throw in an obstacle
block( pose [ 4.000 4.000 0.000 0.000 ] color "red")


define floorplan model
```

```
(
  # sombre, sensible, artistic
  color "gray30"

  # most maps will need a bounding box
  boundary 1

  gui_nose 0
  gui_grid 0

  gui_outline 0
  gripper_return 0
  fiducial_return 0
  laser_return 1
)

# set the resolution of the underlying raytrace model in
   meters
resolution 0.01

interval_sim 100  # simulation timestep in milliseconds

window
(
  size [ 600 424 ]
  rotate [ 0.000 0.000 ]
)

# load an environment bitmap
floorplan
(
  name "Robo Park floor"
  bitmap "../robopark2.bmp"
  size [16.800 11.870 1.000]

  pose [ 0.000 0.000 0.000 0.000 ]
  #for test
  origin [ 7.800 3.700 0.000 0.000] # specify the
     position of the object's center, relative to its
     pose
)
```

turtlebot.inc

```
define kinect ranger
(
  sensor
  (
    range_max 8.0
    fov 360.0
    samples 640
  )
  # generic model properties
  color "black"
  size [ 0.06 0.15 0.03 ]
)

define turtlebot position
(
  pose [ 0.0 0.0 0.0 0.0 ]

  odom_error [0.01 0.01 999999 999999 999999 0.01]

  size [ 0.2552 0.2552 0.40 ]
  origin [ 0.0 0.0 0.0 0.0 ]
  gui_nose 1
  drive "diff"
  color "gray"

  kinect(pose [ -0.1 0.0 -0.11 0.0 ])
)
```

14.5 rviz

14.5.1 robot_navigation.rviz

```
Panels:
  - Class: rviz/Displays
    Help Height: 78
    Name: Displays
    Property Tree Widget:
      Expanded:
        - /TF1/Frames1
        - /TF1/Tree1
      Splitter Ratio: 0.5
    Tree Height: 786
  - Class: rviz/Selection
    Name: Selection
  - Class: rviz/Tool Properties
    Expanded:
      - /2D Pose Estimate1
      - /2D Nav Goal1
    Name: Tool Properties
    Splitter Ratio: 0.5886790156364441
  - Class: rviz/Views
    Expanded:
      - /Current View1
    Name: Views
    Splitter Ratio: 0.5
  - Class: rviz/Time
    Name: Time
    SyncMode: 0
    SyncSource: LaserScan (kinect)
Preferences:
  PromptSaveOnExit: true
Toolbars:
  toolButtonStyle: 2
Visualization Manager:
  Class: ""
  Displays:
    - Alpha: 0.5
      Cell Size: 1
      Class: rviz/Grid
      Color: 160; 160; 164
      Enabled: true
      Line Style:
        Line Width: 0.029999999329447746
        Value: Lines
      Name: Grid
      Normal Cell Count: 0
      Offset:
        X: 0
        Y: 0
        Z: 0
      Plane: XY
      Plane Cell Count: 10
      Reference Frame: <Fixed Frame>
```

```
        Value: true
    - Class: rviz/TF
      Enabled: false
      Filter (blacklist): ""
      Filter (whitelist): ""
      Frame Timeout: 15
      Frames:
        All Enabled: false
      Marker Alpha: 1
      Marker Scale: 1
      Name: TF
      Show Arrows: true
      Show Axes: true
      Show Names: false
      Tree:
        {}
      Update Interval: 0
      Value: false
    - Alpha: 1
      Autocompute Intensity Bounds: true
      Autocompute Value Bounds:
        Max Value: 10
        Min Value: -10
        Value: true
      Axis: Z
      Channel Name: intensity
      Class: rviz/LaserScan
      Color: 255; 255; 255
      Color Transformer: Intensity
      Decay Time: 0
      Enabled: true
      Invert Rainbow: false
      Max Color: 255; 255; 255
      Min Color: 0; 0; 0
      Name: LaserScan (kinect)
      Position Transformer: XYZ
      Queue Size: 10
      Selectable: true
      Size (Pixels): 3
      Size (m): 0.009999999776482582
      Style: Squares
      Topic: /scan
      Unreliable: false
      Use Fixed Frame: true
      Use rainbow: true
      Value: true
    - Alpha: 1
      Autocompute Intensity Bounds: true
      Autocompute Value Bounds:
        Max Value: 10
        Min Value: -10
        Value: true
      Axis: Z
      Channel Name: intensity
      Class: rviz/LaserScan
      Color: 255; 255; 255
      Color Transformer: Intensity
      Decay Time: 0
      Enabled: true
      Invert Rainbow: false
      Max Color: 255; 255; 255
      Min Color: 0; 0; 0
      Name: LaserScan (ir sensors)
      Position Transformer: XYZ
      Queue Size: 10
      Selectable: true
      Size (Pixels): 3
      Size (m): 0.05000000074505806
      Style: Flat Squares
      Topic: /ir_scan
      Unreliable: false
      Use Fixed Frame: true
      Use rainbow: true
      Value: true
    - Alpha: 1
      Autocompute Intensity Bounds: true
      Autocompute Value Bounds:
        Max Value: 10
        Min Value: -10
        Value: true
      Axis: Z
      Channel Name: intensity
      Class: rviz/LaserScan
      Color: 255; 255; 255
      Color Transformer: Intensity
      Decay Time: 0
      Enabled: true
      Invert Rainbow: false
      Max Color: 255; 255; 255
      Min Color: 0; 0; 0
      Name: LaserScan (virtual sensor)
      Position Transformer: XYZ
      Queue Size: 10
      Selectable: true
      Size (Pixels): 3
      Size (m): 0.019999999552965164
      Style: Flat Squares
      Topic: /virtual_sensor_scan
      Unreliable: false
      Use Fixed Frame: true
      Use rainbow: true
      Value: true
    - Alpha: 1
      Autocompute Intensity Bounds: true
      Autocompute Value Bounds:
        Max Value: 10
        Min Value: -10
        Value: true
      Axis: Z
      Channel Name: intensity
      Class: rviz/PointCloud2
      Color: 255; 255; 255
      Color Transformer: Intensity
      Decay Time: 0
      Enabled: true
      Invert Rainbow: false
      Max Color: 255; 255; 255
      Min Color: 0; 0; 0
      Name: PointCloud (bumpers)
      Position Transformer: XYZ
      Queue Size: 10
      Selectable: true
      Size (Pixels): 3
      Size (m): 0.03999999910593033
      Style: Flat Squares
      Topic: /mobile_base/sensors/bumper_pointcloud
      Unreliable: false
      Use Fixed Frame: true
      Use rainbow: true
```

```
        Value: true
      - Alpha: 0.699999988079071
        Class: rviz/Map
        Color Scheme: map
        Draw Behind: false
        Enabled: true
        Name: Map
        Topic: /map
        Unreliable: false
        Use Timestamp: false
        Value: true
      - Class: rviz/Group
        Displays:
          - Alpha: 0.699999988079071
            Class: rviz/Map
            Color Scheme: costmap
            Draw Behind: false
            Enabled: false
            Name: Costmap
            Topic: /move_base/local_costmap/costmap
            Unreliable: false
            Use Timestamp: false
            Value: false
          - Alpha: 1
            Buffer Length: 1
            Class: rviz/Path
            Color: 0; 12; 255
            Enabled: true
            Head Diameter: 0.30000001192092896
            Head Length: 0.20000000298023224
            Length: 0.30000001192092896
            Line Style: Lines
            Line Width: 0.029999999329447746
            Name: Planner
            Offset:
              X: 0
              Y: 0
              Z: 0
            Pose Color: 255; 85; 255
            Pose Style: None
            Queue Size: 10
            Radius: 0.029999999329447746
            Shaft Diameter: 0.10000000149011612
            Shaft Length: 0.10000000149011612
            Topic: /move_base/DWAPlannerROS/local_plan
            Unreliable: false
            Value: true
        Enabled: true
        Name: Local Planning
      - Class: rviz/Group
        Displays:
          - Alpha: 0.4000000059604645
            Class: rviz/Map
            Color Scheme: costmap
            Draw Behind: true
            Enabled: false
            Name: Costmap
            Topic: /move_base/global_costmap/costmap
            Unreliable: false
            Use Timestamp: false
            Value: false
          - Alpha: 1
            Buffer Length: 1
            Class: rviz/Path
            Color: 255; 0; 0
            Enabled: true
            Head Diameter: 0.30000001192092896
            Head Length: 0.20000000298023224
            Length: 0.30000001192092896
            Line Style: Lines
            Line Width: 0.029999999329447746
            Name: Planner
            Offset:
              X: 0
              Y: 0
              Z: 0
            Pose Color: 255; 85; 255
            Pose Style: None
            Queue Size: 10
            Radius: 0.029999999329447746
            Shaft Diameter: 0.10000000149011612
            Shaft Length: 0.10000000149011612
            Topic: /move_base/NavfnROS/plan
            Unreliable: false
            Value: true
        Enabled: true
        Name: Global Planning
      - Alpha: 1
        Axes Length: 1
        Axes Radius: 0.10000000149011612
        Class: rviz/Pose
        Color: 255; 25; 0
        Enabled: true
        Head Length: 0.20000000298023224
        Head Radius: 0.10000000149011612
        Name: Pose (move_base)
        Queue Size: 10
        Shaft Length: 1
        Shaft Radius: 0.05000000074505806
        Shape: Arrow
        Topic: /move_base/current_goal
        Unreliable: false
        Value: true
      - Alpha: 1
        Arrow Length: 0.20000000298023224
        Axes Length: 0.30000001192092896
        Axes Radius: 0.009999999776482582
        Class: rviz/PoseArray
        Color: 0; 192; 0
        Enabled: true
        Head Length: 0.07000000029802322
        Head Radius: 0.029999999329447746
        Name: ParticleCloud
        Queue Size: 10
        Shaft Length: 0.23000000417232513
        Shaft Radius: 0.009999999776482582
        Shape: Arrow (Flat)
        Topic: /particlecloud
        Unreliable: false
        Value: true
      - Alpha: 1
        Autocompute Intensity Bounds: true
        Autocompute Value Bounds:
          Max Value: 10
          Min Value: -10
          Value: true
        Axis: Z
        Channel Name: total_cost
        Class: rviz/PointCloud2
        Color: 255; 255; 255
        Color Transformer: Intensity
        Decay Time: 0
        Enabled: false
        Invert Rainbow: false
        Max Color: 255; 255; 255
        Min Color: 0; 0; 0
        Name: PointCloud2
        Position Transformer: XYZ
        Queue Size: 10
        Selectable: true
        Size (Pixels): 3
        Size (m): 0.009999999776482582
        Style: Points
        Topic: /move_base/DWAPlannerROS/cost_cloud
        Unreliable: false
        Use Fixed Frame: true
        Use rainbow: true
        Value: false
      - Alpha: 0.699999988079071
        Class: rviz/Map
        Color Scheme: map
        Draw Behind: false
        Enabled: false
        Name: Map
        Topic: /move_base/local_costmap/costmap
        Unreliable: false
        Use Timestamp: false
        Value: false
      - Alpha: 0.699999988079071
        Class: rviz/Map
        Color Scheme: map
        Draw Behind: false
        Enabled: false
        Name: Map
        Topic: /move_base/global_costmap/costmap
        Unreliable: false
        Use Timestamp: false
        Value: false
      - Alpha: 1
        Class: rviz/Polygon
        Color: 25; 255; 0
        Enabled: true
        Name: Polygon
        Queue Size: 10
        Topic: /move_base/local_costmap/footprint
        Unreliable: false
        Value: true
    Enabled: true
    Global Options:
      Background Color: 48; 48; 48
      Default Light: true
      Fixed Frame: map
      Frame Rate: 30
    Name: root
    Tools:
      - Class: rviz/MoveCamera
      - Class: rviz/Interact
        Hide Inactive Objects: true
      - Class: rviz/Select
      - Class: rviz/SetInitialPose
        Theta std deviation: 0.2617993950843811
        Topic: /initialpose
        X std deviation: 0.5
        Y std deviation: 0.5
      - Class: rviz/SetGoal
        Topic: /move_base_simple/goal
      - Class: rviz/Measure
    Value: true
    Views:
      Current:
        Class: rviz/Orbit
        Distance: 24.896236419677734
        Enable Stereo Rendering:
          Stereo Eye Separation: 0.05999999865889549
          Stereo Focal Distance: 1
          Swap Stereo Eyes: false
          Value: false
        Field of View: 0.7853981852531433
        Focal Point:
          X: 3.1884381771087646
          Y: 3.1455135345458984
          Z: -0.1958555430173874
        Focal Shape Fixed Size: true
        Focal Shape Size: 0.05000000074505806
        Invert Z Axis: false
        Name: Current View
        Near Clip Distance: 0.009999999776482582
        Pitch: 1.5697963237762451
        Target Frame: base_link
        Yaw: 6.27159309387207
      Saved: ~
  Window Geometry:
    Displays:
      collapsed: false
    Height: 1016
    Hide Left Dock: false
    Hide Right Dock: false
    QMainWindow State: 000000
       ff00000000fd00000004000000000000001e10000039dfc0200000005fb0000001200

    Selection:
      collapsed: false
    Time:
      collapsed: false
    Tool Properties:
      collapsed: false
    Views:
      collapsed: false
    Width: 1920
    X: -2
    Y: 0
```

14.6 launch

14.6.1 amcl.launch.xml

```
<launch>
  <arg name="use_map_topic"   default="true"/>
  <arg name="scan_topic"      default="scan"/>
  <arg name="initial_pose_x"  default="0.0"/>
  <arg name="initial_pose_y"  default="0.0"/>
  <arg name="initial_pose_a"  default="0.0"/>
  <arg name="odom_frame_id"   default="odom"/>
  <arg name="base_frame_id"   default="base_footprint"/>
  <arg name="global_frame_id" default="map"/>

  <node pkg="amcl" type="amcl" name="amcl">
```

```
    <param name="use_map_topic"             value="$(arg use_map_topic)"/>
    <!-- Publish scans from best pose at a max of 10 Hz -->
    <param name="odom_model_type"           value="diff-corrected"/>
    <param name="odom_alpha5"               value="0.1"/>
    <param name="gui_publish_rate"          value="10.0"/>
    <param name="laser_max_beams"             value="60"/>
    <param name="laser_max_range"           value="12.0"/>
    <param name="min_particles"             value="500"/>
    <param name="max_particles"             value="2000"/>
    <param name="kld_err"                   value="0.05"/>
    <param name="kld_z"                     value="0.99"/>
    <param name="odom_alpha1"               value="0.2"/>
    <param name="odom_alpha2"               value="0.2"/>
    <param name="odom_alpha3"               value="0.2"/>
    <param name="odom_alpha4"               value="0.2"/>
    <param name="laser_z_hit"               value="0.5"/>
    <param name="laser_z_short"             value="0.05"/>
    <param name="laser_z_max"               value="0.05"/>
    <param name="laser_z_rand"              value="0.5"/>
    <param name="laser_sigma_hit"           value="0.2"/>
    <param name="laser_lambda_short"        value="0.1"/>
    <param name="laser_model_type"          value="likelihood_field"/>
    <!-- <param name="laser_model_type" value="beam"/> -->
    <param name="laser_likelihood_max_dist" value="2.0"/>
    <param name="update_min_d"              value="0.25"/>
    <param name="update_min_a"              value="0.2"/>
    <param name="odom_frame_id"             value="$(arg odom_frame_id)"/>
    <param name="base_frame_id"              value="$(arg base_frame_id)"/>
    <param name="global_frame_id"            value="$(arg global_frame_id)"/>
    <param name="resample_interval"          value="1"/>
    <!-- Increase tolerance because the computer can get quite busy -->
    <param name="transform_tolerance"        value="1.0"/>
    <param name="recovery_alpha_slow"        value="0.0"/>
    <param name="recovery_alpha_fast"        value="0.0"/>
    <param name="initial_pose_x"             value="$(arg initial_pose_x)"/>
    <param name="initial_pose_y"             value="$(arg initial_pose_y)"/>
    <param name="initial_pose_a"             value="$(arg initial_pose_a)"/>
    <remap from="scan"                       to="$(arg scan_topic)"/>
  </node>
</launch>
```

14.6.2 gmapping.launch.xml

```
<launch>
  <arg name="scan_topic"  default="scan" />
  <arg name="base_frame"  default="base_footprint"/>
  <arg name="odom_frame"  default="odom"/>

  <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping" output="screen">
    <param name="base_frame" value="$(arg base_frame)"/>
    <param name="odom_frame" value="$(arg odom_frame)"/>
    <param name="map_update_interval" value="5.0"/>
    <param name="maxUrange" value="6.0"/>
    <param name="maxRange" value="8.0"/>
    <param name="sigma" value="0.05"/>
    <param name="kernelSize" value="1"/>
    <param name="lstep" value="0.05"/>
    <param name="astep" value="0.05"/>
    <param name="iterations" value="5"/>
    <param name="lsigma" value="0.075"/>
    <param name="ogain" value="3.0"/>
    <param name="lskip" value="0"/>
    <param name="minimumScore" value="200"/>
    <param name="srr" value="0.1"/>
    <param name="srt" value="0.2"/>
    <param name="str" value="0.1"/>
    <param name="stt" value="0.2"/>
    <param name="linearUpdate" value="0.5"/>
    <param name="angularUpdate" value="0.1"/>
    <param name="temporalUpdate" value="-1.0"/>
    <param name="resampleThreshold" value="0.5"/>
    <param name="particles" value="30"/>

    <param name="xmin" value="-10.0"/>
    <param name="ymin" value="-10.0"/>
    <param name="xmax" value="10.0"/>
    <param name="ymax" value="10.0"/>

    <param name="delta" value="0.05"/>
    <param name="llsamplerange" value="0.01"/>
    <param name="llsamplestep" value="0.01"/>
    <param name="lasamplerange" value="0.005"/>
    <param name="lasamplestep" value="0.005"/>
    <remap from="scan" to="$(arg scan_topic)"/>
  </node>
</launch>
```

14.6.3 turtlebot_amcl.launch

```

<launch>
  <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
kobuki_hexagons_kinect kobuki_hexagons_r200
   roomba_circles_asus_xtion_pro roomba_circles_kinect
   -->

  <!-- Name of the map to use and initial position -->
  <arg name="map_file"       default="$(find nav2d_conf)/
     maps/maze.yaml"/>
  <arg name="world_file"     default="$(find nav2d_conf)/maps/stage/maze.world"/>
  <arg name="initial_pose_x" default="2.0"/>
  <arg name="initial_pose_y" default="2.0"/>
  <arg name="initial_pose_a" default="0.0"/>

  <arg name="odom_frame_id"   default="odom"/>
  <arg name="global_frame_id" default="map"/>

  <param name="/use_sim_time" value="true"/>

 <!--  ***************** Robot simulation ****************  -->
  <node pkg="stage_ros" type="stageros" name="stageros" args="$(arg world_file)">
    <param name="base_watchdog_timeout" value="0.5"/>
    <remap from="base_scan" to="scan"/>
  </node>

  <!--  ***************** Robot Model ****************  -->
  <include file="$(find turtlebot3_bringup)/launch/includes/description.launch.xml">
    <arg name="model" value="$(arg base)" />
  </include>
  <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher">
    <param name="use_gui" value="false"/>
  </node>

  <!--  ************** Navigation  ***************  -->
   <node pkg="move_base" type="move_base" respawn="false" name="move_base" output="screen">
    <rosparam file="$(find nav2d_conf)/cfg/costmap_common_params.yaml" command="load" ns="global_costmap" />
    <rosparam file="$(find nav2d_conf)/cfg/costmap_common_params.yaml" command="load" ns="local_costmap" />
    <rosparam file="$(find nav2d_conf)/cfg/local_costmap_params.yaml" command="load" />
    <rosparam file="$(find nav2d_conf)/cfg/global_costmap_params.yaml" command="load" />
    <rosparam file="$(find nav2d_conf)/cfg/dwa_local_planner_params.yaml" command="load" />
    <rosparam file="$(find nav2d_conf)/cfg/move_base_params.yaml" command="load" />

    <param name="base_global_planner" value="navfn/NavfnROS" />  <!--  planner  navfn/NavfnROS  alternative  carrot_planner/CarrotPlanner -->
    <param name="planner_frequency" value="1.0" />
    <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.  -->

    <param name="base_local_planner" value="dwa_local_planner/DWAPlannerROS" />
    <param name="controller_frequency" value="5.0" />
    <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.  -->

    <param name="clearing_rotation_allowed" value="true" />
  </node>

  <!--  ****** Maps *****  -->
  <node name="map_server" pkg="map_server" type="map_server" args="$(arg map_file)">
    <param name="frame_id" value="map"/>
  </node>

  <!--  ****** amcl *****  -->
  <include file="$(find nav2d_conf)/launch/amcl.launch.xml">
    <arg name="scan_topic" value="scan"/>
    <arg name="use_map_topic" value="true"/>
    <arg name="initial_pose_x" value="$(arg initial_pose_x)"/>
    <arg name="initial_pose_y" value="$(arg initial_pose_y)"/>
    <arg name="initial_pose_a" value="$(arg initial_pose_a)"/>
  </include>


  <!--  *************** Visualisation ***************
     -->
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(
     find nav2d_conf)/rviz/robot_navigation.rviz"/>
</launch>
```

14.6.4 turtlebot_gmapping.launch

```
<launch>
  <arg name="base"       default="burger"/>  <!-- create,
      rhoomba -->

  <!-- Name of the map to use and initial position -->
  <arg name="world_file"     default="$(find nav2d_conf)/
     maps/stage/maze.world"/>

  <arg name="odom_frame_id"   default="odom"/>
  <arg name="global_frame_id" default="map"/>

  <param name="/use_sim_time" value="true"/>

 <!--  **************** Robot simulation
    ****************  -->
  <node pkg="stage_ros" type="stageros" name="stageros"
     args="$(arg world_file)">
    <param name="base_watchdog_timeout" value="0.5"/>
    <remap from="base_scan" to="scan"/>
  </node>

  <!--  **************** Robot Model ****************
     -->
  <include file="$(find turtlebot3_bringup)/launch/
     includes/description.launch.xml">
    <arg name="model" value="$(arg base)" />
  </include>

  <!--  ************* Navigation  **************  -->
   <node pkg="move_base" type="move_base" respawn="false"
       name="move_base" output="screen">
    <rosparam file="$(find nav2d_conf)/cfg/costmap_common_params.yaml" command="load" ns="global_costmap" />
    <rosparam file="$(find nav2d_conf)/cfg/costmap_common_params.yaml" command="load" ns="local_costmap" />
    <rosparam file="$(find nav2d_conf)/cfg/local_costmap_params.yaml" command="load" />
    <rosparam file="$(find nav2d_conf)/cfg/global_costmap_params.yaml" command="load" />
    <rosparam file="$(find nav2d_conf)/cfg/dwa_local_planner_params.yaml" command="load" />
    <rosparam file="$(find nav2d_conf)/cfg/move_base_params.yaml" command="load" />

    <param name="base_global_planner" value="navfn/NavfnROS" />
    <param name="planner_frequency" value="1.0" />
    <param name="planner_patience" value="10.0" />

    <param name="base_local_planner" value="dwa_local_planner/DWAPlannerROS" />
    <param name="controller_frequency" value="5.0" />
    <param name="controller_patience" value="5.0" />

    <param name="clearing_rotation_allowed" value="true" />
  </node>

 <!-- Gmapping -->
  <include file="$(find nav2d_conf)/launch/gmapping.launch.xml"/>

  <!--  **************** Visualisation *************** -->
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find nav2d_conf)/rviz/robot_navigation.rviz"/>
</launch>
```