

ECE 4703

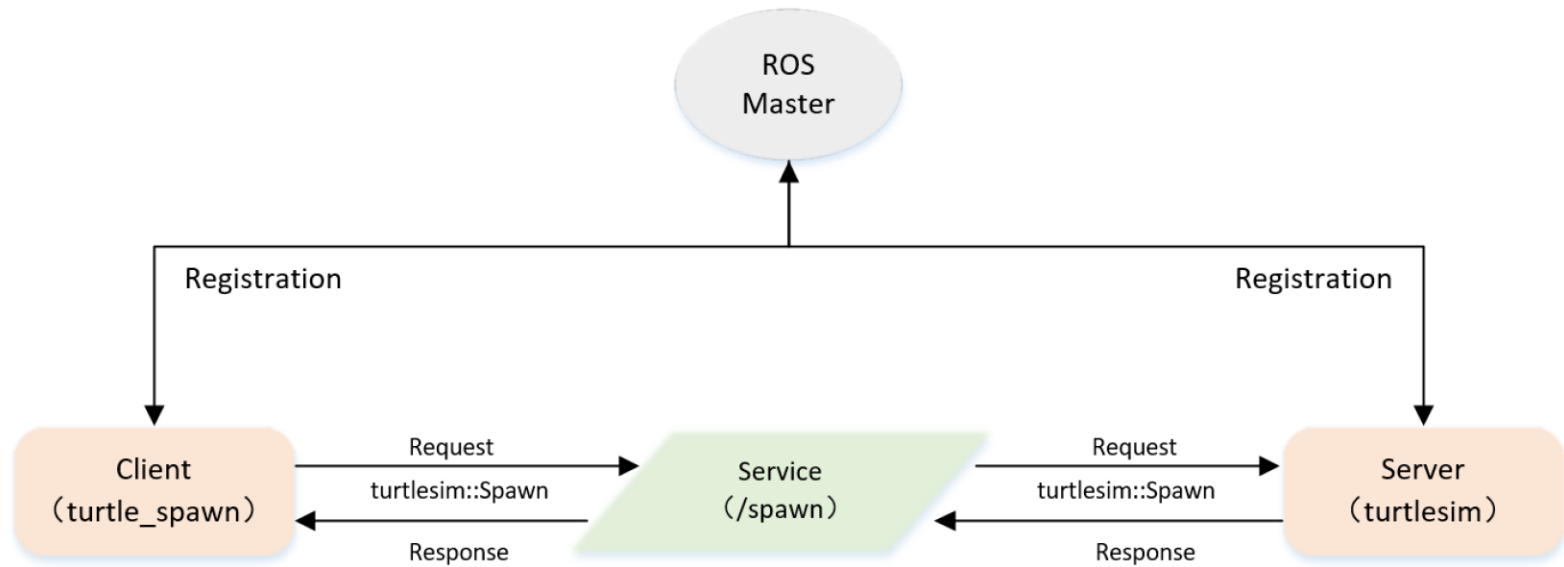
Mobile Autonomous Robots

Jenny Zhen Yu
zhenyu@cpp.edu

Department of Electrical and Computer Engineering
California State Polytechnic University, Pomona

Lecture 8: Client Program

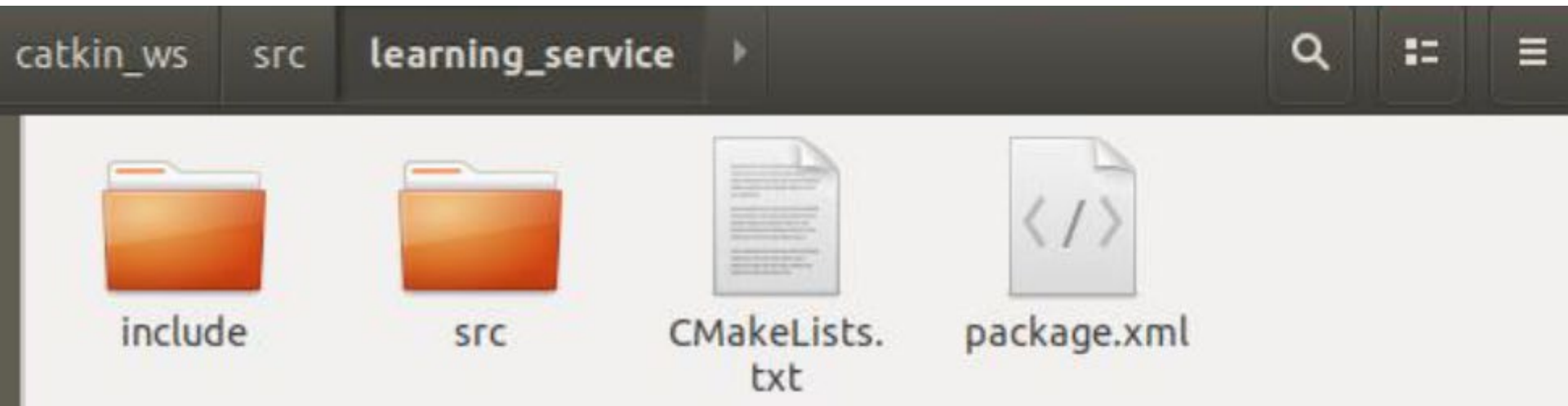
Publisher and Subscriber Compile



Build Package

```
$ cd ~/catkin_ws/src
```

```
$ catkin_create_pkg learning_service roscpp rospy std_msgs geometry_msgs turtlesim
```



C++ Code

```
Open ▼ [icon]
1 #include <ros/ros.h>
2 #include <turtlesim/Spawn.h>
3
4 int main(int argc, char** argv)
5 {
6
7     ros::init(argc, argv, "turtle_spawn");
8
9
10    ros::NodeHandle node;
11
12
13    ros::service::waitForService("/spawn");
14    ros::ServiceClient add_turtle = node.serviceClient<turtlesim::Spawn>("/spawn");
15
16
17    turtlesim::Spawn srv;
18    srv.request.x = 2.0;
19    srv.request.y = 2.0;
20    srv.request.name = "turtle2";
21
22
23    ROS_INFO("Call service to spawn turtle[x:%0.6f, y:%0.6f, name:%s]",
24            srv.request.x, srv.request.y, srv.request.name.c_str());
25
26    add_turtle.call(srv);
27
28
29    ROS_INFO("Spawn turtle successfully [name:%s]", srv.response.name.c_str());
30
31    return 0;
32 };
```

Compiling Code

```
## 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/learning_service_node.cpp)

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

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

Compiling Code

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

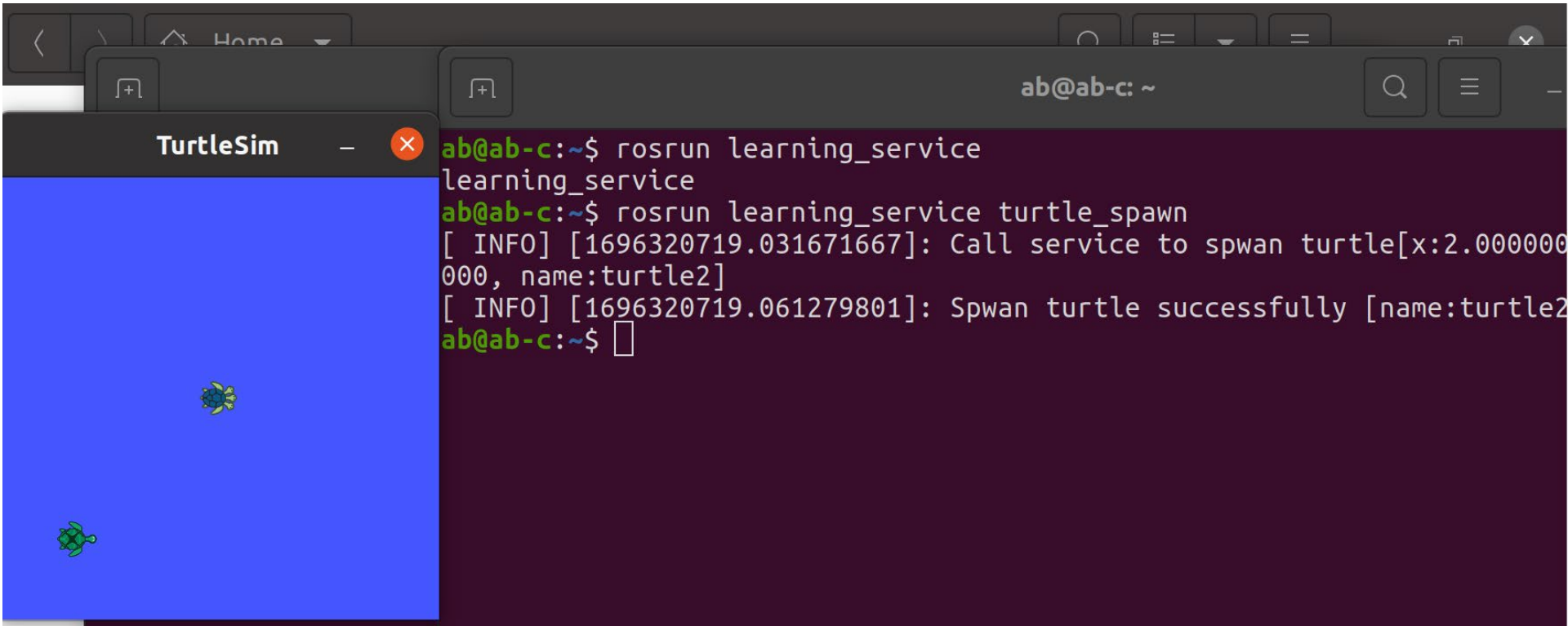
```
$ source devel/setup.bash
```

```
$ roscore
```


```
$ rosrun turtlesim turtlesim_node
```

```
$ rosrun learning_service turtle_spawn
```

Compiling Code

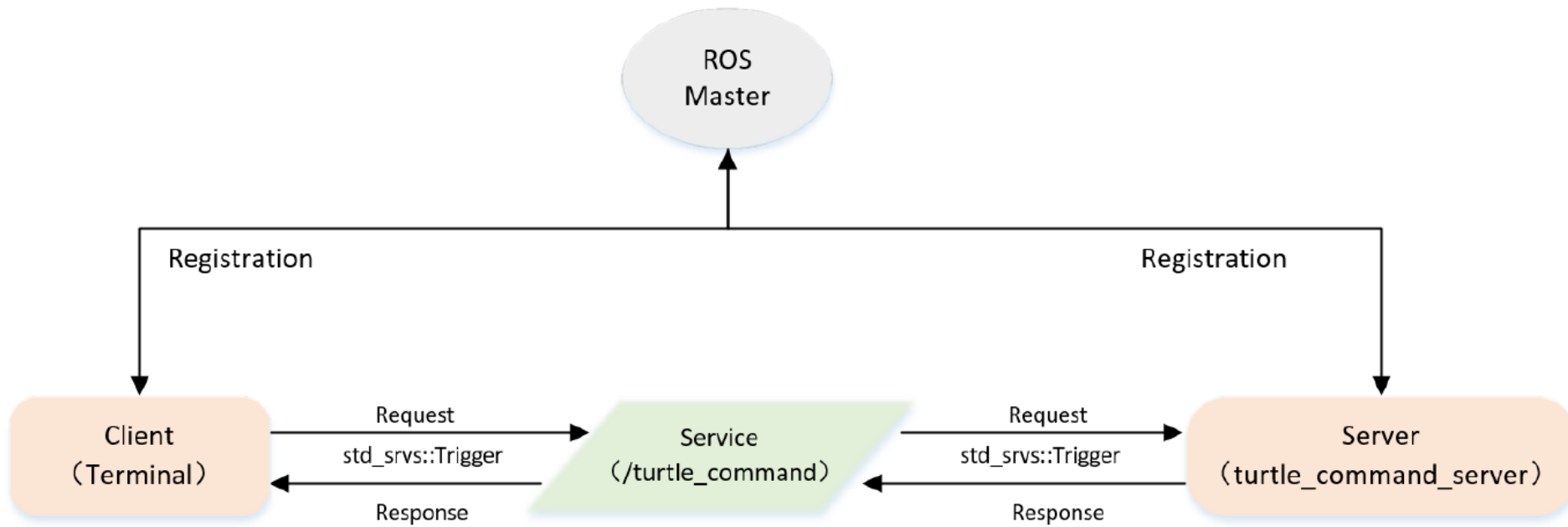


Python Code

```
Open ▼ 
1 import sys
2 import rospy
3 from turtlesim.srv import Spawn
4
5 def turtle_spawn():
6     rospy.init_node('turtle_spawn')
7
8
9
10    rospy.wait_for_service('/spawn')
11    try:
12        add_turtle = rospy.ServiceProxy('/spawn', Spawn)
13
14
15        response = add_turtle(2.0, 2.0, 0.0, "turtle2")
16        return response.name
17    except rospy.ServiceException, e:
18        print "Service call failed: %s"%e
19
20 if __name__ == "__main__":
21
22     print "Spwan turtle successfully [name:%s]" %(turtle_spawn())
23
24
```

Lecture 9: Server Program

Server Model



```

1 #include <ros/ros.h>
2 #include <geometry_msgs/Twist.h>
3 #include <std_srvs/Trigger.h>
4
5 ros::Publisher turtle_vel_pub;
6 bool pubCommand = false;
7
8 bool commandCallback(std_srvs::Trigger::Request &req,
9                     std_srvs::Trigger::Response &res)
10 {
11     pubCommand = !pubCommand;
12
13     ROS_INFO("Publish turtle velocity command [%s]", pubCommand==true?"Yes":"No");
14
15     res.success = true;
16     res.message = "Change turtle command state!";
17
18     return true;
19 }
20
21 int main(int argc, char **argv)
22 {
23     ros::init(argc, argv, "turtle_command_server");
24
25     ros::NodeHandle n;
26
27     ros::ServiceServer command_service = n.advertiseService("/turtle_command", commandCallback);
28
29     turtle_vel_pub = n.advertise<geometry_msgs::Twist>("/turtle1/cmd_vel", 10);
30
31     ROS_INFO("Ready to receive turtle command.");
32
33     ros::Rate loop_rate(10);
34
35     while(ros::ok())
36     {
37         ros::spinOnce();
38
39         if(pubCommand)
40         {
41             geometry_msgs::Twist vel_msg;
42             vel_msg.linear.x = 0.5;
43             vel_msg.angular.z = 0.2;
44             turtle_vel_pub.publish(vel_msg);
45         }
46
47         loop_rate.sleep();
48     }
49
50     return 0;
51 }

```

Compiling Code

```
## 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/learning_service_node.cpp)

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

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

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

Compiling Code

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

```
$ source devel/setup.bash
```

```
$ roscore
```

```
$ rosrun turtlesim turtlesim_node
```

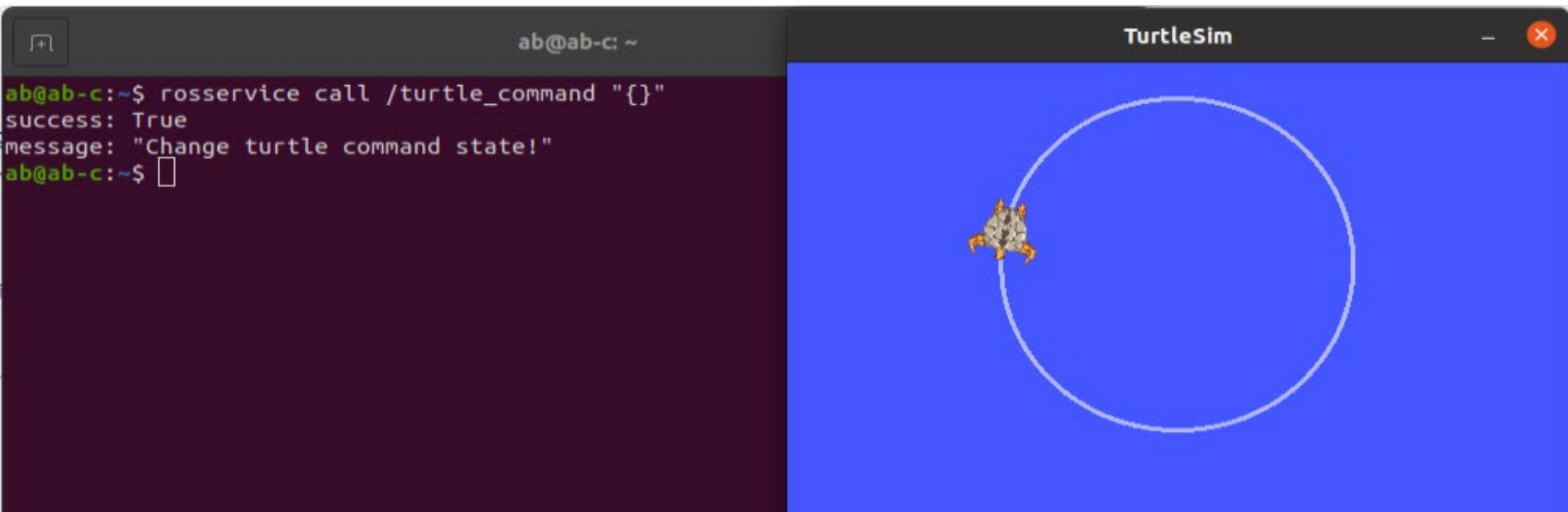
```
$ rosrun learning_service turtle_command_server
```

```
$ rosservice call /turtle_command "{}"
```



Double click Tab

Rosrun

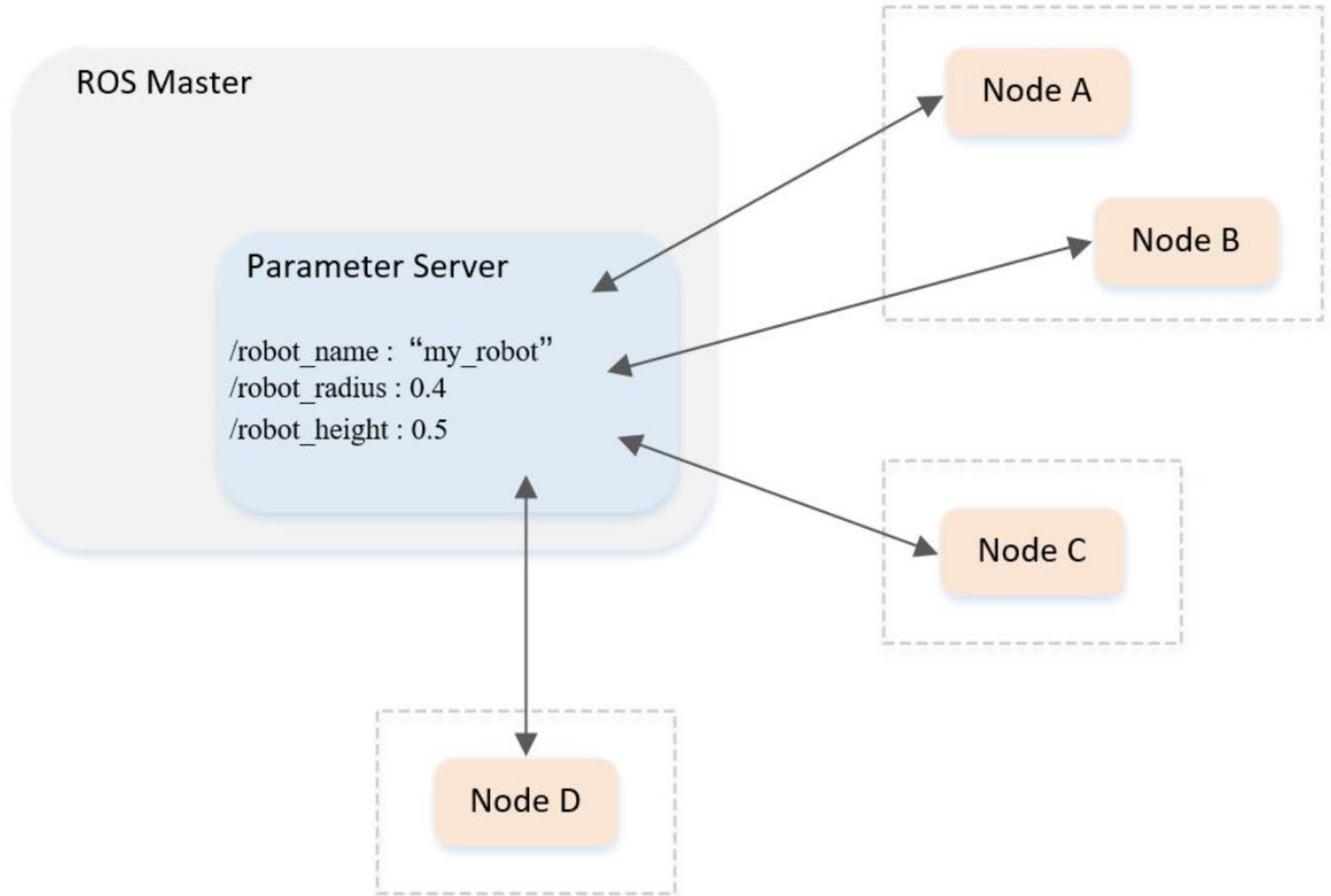


Python Code

```
1 import rospy
2 import thread,time
3 from geometry_msgs.msg import Twist
4 from std_srvs.srv import Trigger, TriggerResponse
5
6 pubCommand = False;
7 turtle_vel_pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)
8
9 def command_thread():
10     while True:
11         if pubCommand:
12             vel_msg = Twist()
13             vel_msg.linear.x = 0.5
14             vel_msg.angular.z = 0.2
15             turtle_vel_pub.publish(vel_msg)
16
17             time.sleep(0.1)
18
19 def commandCallback(req):
20     global pubCommand
21     pubCommand = bool(1-pubCommand)
22
23
24     rospy.loginfo("Publish turtle velocity command![%d]", pubCommand)
25
26
27     return TriggerResponse(1, "Change turtle command state!")
28
29 def turtle_command_server():
30
31     rospy.init_node('turtle_command_server')
32
33
34     s = rospy.Service('/turtle_command', Trigger, commandCallback)
35
36
37     print "Ready to receive turtle command."
38
39     thread.start_new_thread(command_thread, ())
40     rospy.spin()
41
42 if __name__ == "__main__":
43     turtle_command_server()
```


Lecture 11: Parameter Server

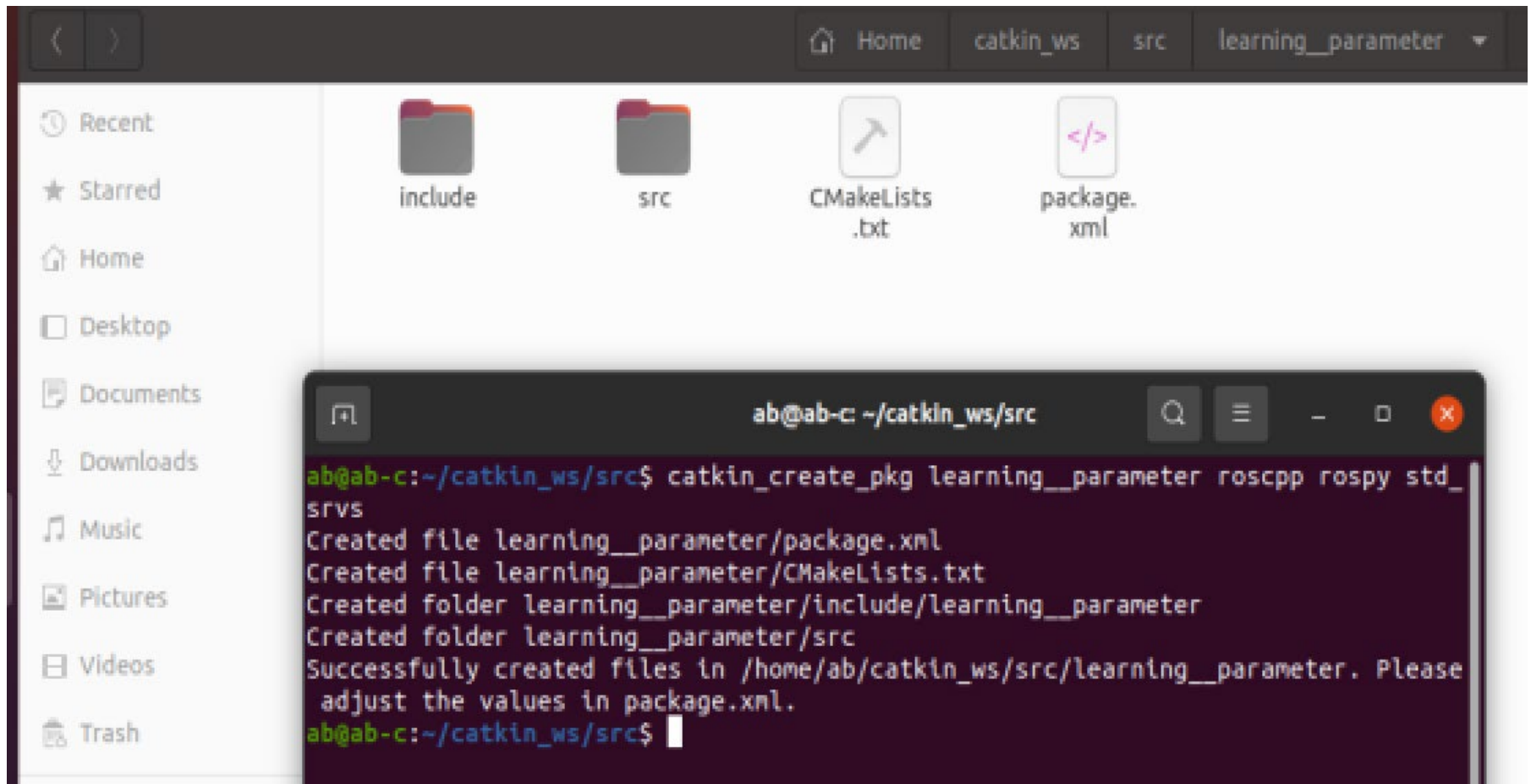
Parameter Server Model



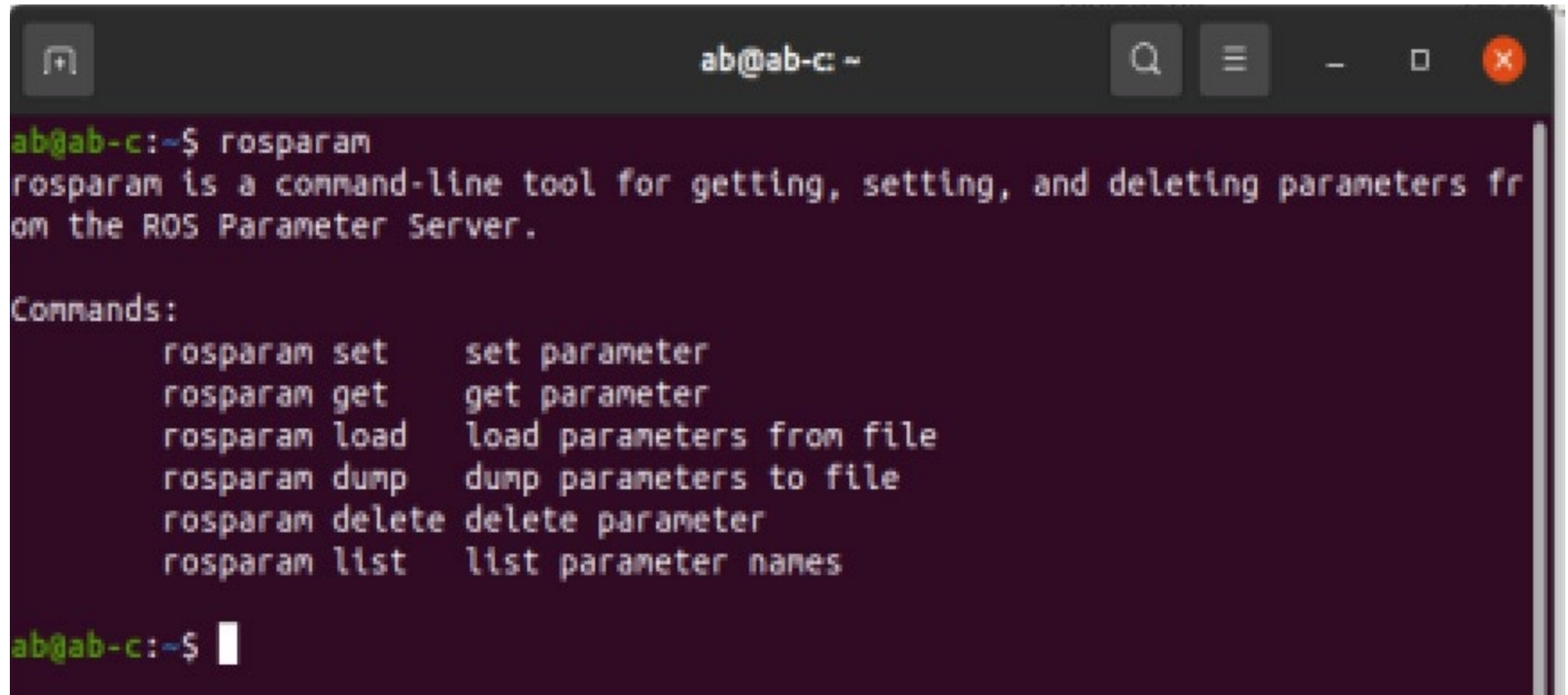
Create Package

```
$ cd ~/catkin_ws/src
```

```
$ catkin_create_pkg learning_parameter roscpp rospy std_srvs
```



rosparam

A terminal window titled 'ab@ab-c: ~' with standard window controls. The terminal shows the command 'rosparam' being executed, followed by a help message and a list of commands.

```
ab@ab-c:~$ rosparam
rosparam is a command-line tool for getting, setting, and deleting parameters fr
om the ROS Parameter Server.

Commands:
    rosparam set      set parameter
    rosparam get      get parameter
    rosparam load     load parameters from file
    rosparam dump     dump parameters to file
    rosparam delete   delete parameter
    rosparam list     list parameter nanes

ab@ab-c:~$
```

Rosparam: list parameter

```
ab@ab-c: ~  
ab@ab-c:~$ rosparam  
rosparam is a command-line tool for getting, setting, and deleting parameters from the ROS Parameter Server.  
  
Commands:  
    rosparam set      set parameter  
    rosparam get      get parameter  
    rosparam load     load parameters from file  
    rosparam dump     dump parameters to file  
    rosparam delete   delete parameter  
    rosparam list     list parameter names  
  
ab@ab-c:~$ rosparam list  
/rostdistro  
/roslaunch/urls/host_ab_c__39333  
/rosversion  
/run_id  
/turtlesim/background_b  
/turtlesim/background_g  
/turtlesim/background_r  
ab@ab-c:~$
```

Rosparam: display parameter

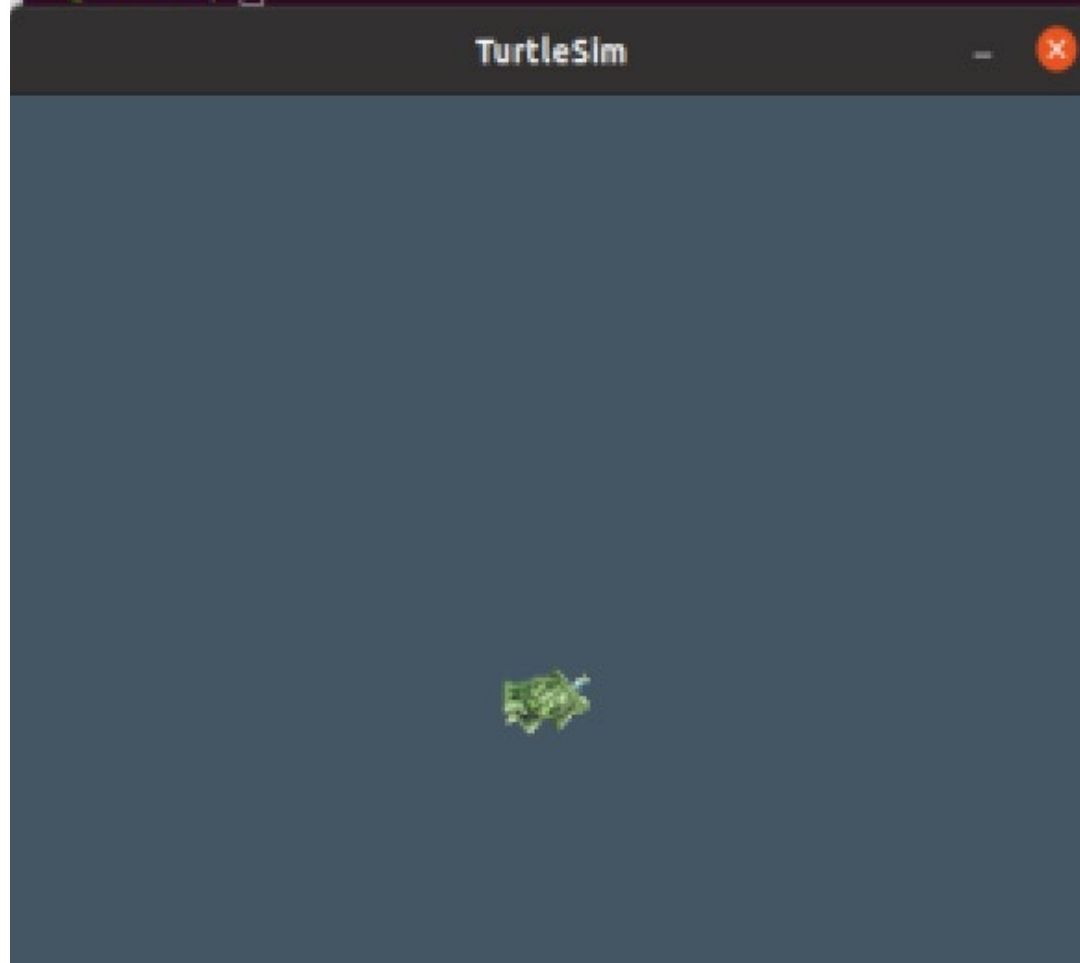
```
ab@ab-c:~$ rosparam get /turtlesim/background_b  
255
```

```
ab@ab-c:~$ rosparam get /turtlesim/background_r  
69
```

```
ab@ab-c:~$ rosparam get /rosversion  
'1.16.0'
```

Rosparam: set parameter

```
abgab-c:~$ rosparam set /turtlesim/background_b 100
abgab-c:~$ rosparam get /turtlesim/background_b
100
abgab-c:~$
abgab-c:~$ rosservice call /clear "{}"
abgab-c:~$
```



Rosparam: Change Turtle Color

```
ab@ab-c:~$ rosparam set /turtlesim/background_b 255
ab@ab-c:~$ rosparam set /turtlesim/background_r 255
ab@ab-c:~$ rosparam set /turtlesim/background_g 255
ab@ab-c:~$ rosservice call /clear "{}"

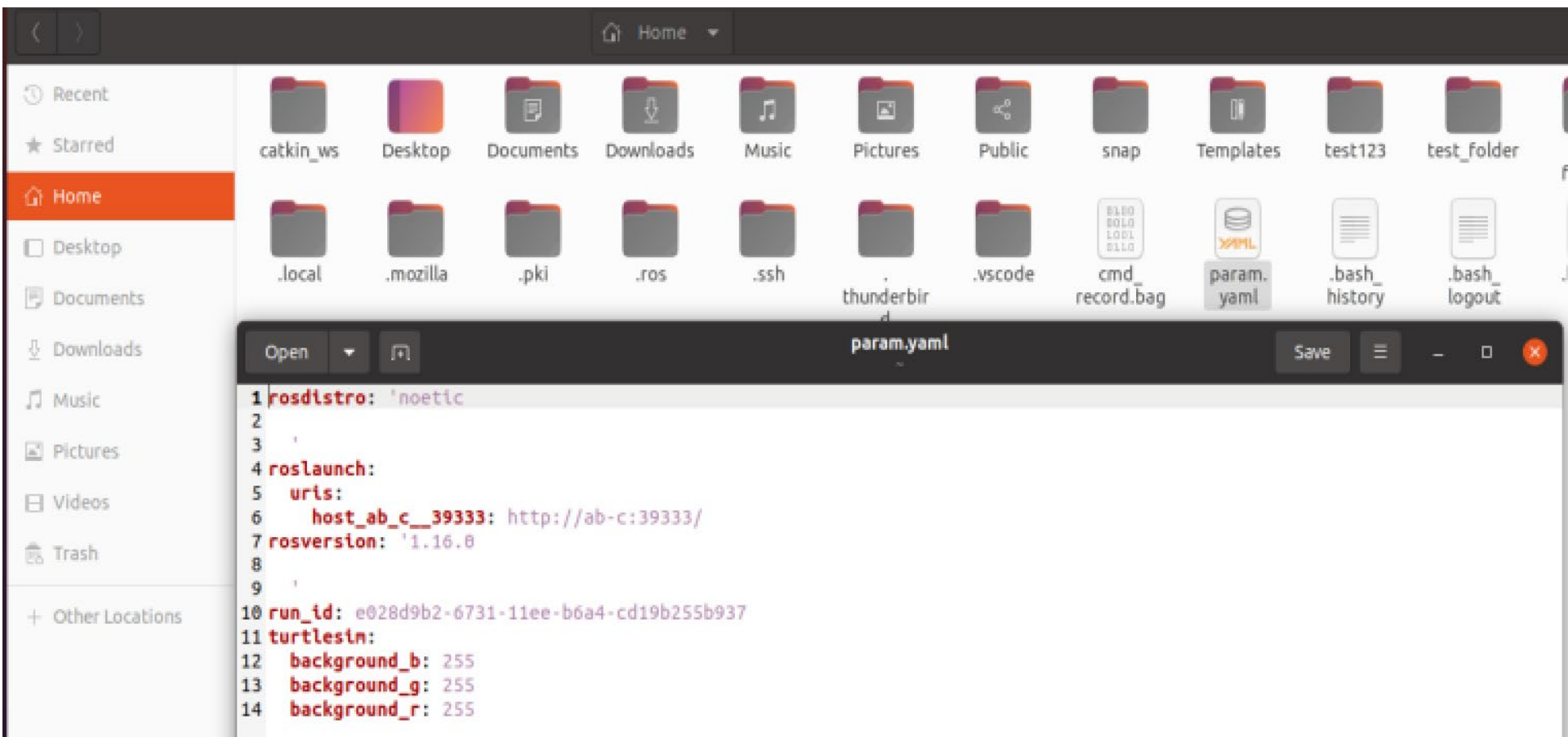
ab@ab-c:~$
```

TurtleSim



Rosparam: save parameter

```
ab@ab-c:~$ rosparam dump param.yaml
ab@ab-c:~$
```

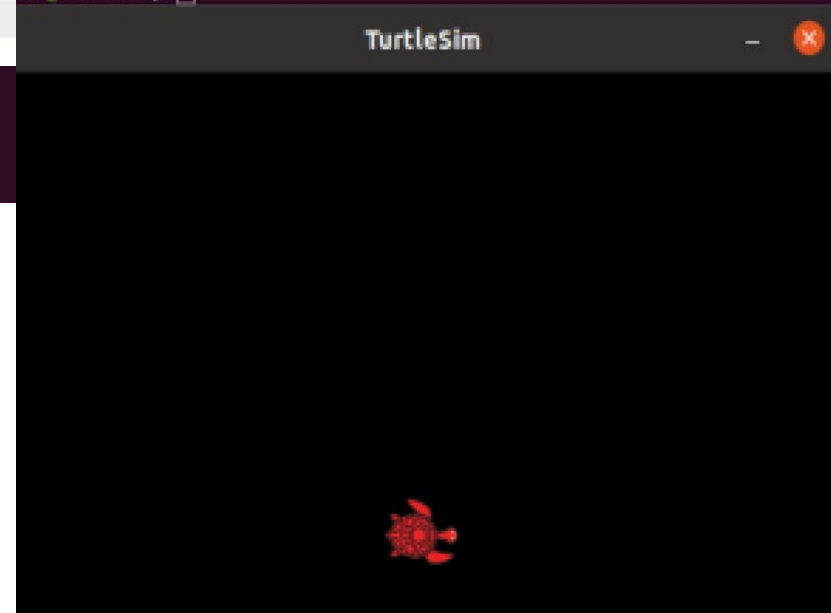


Rosparam: load parameter

```
Open  param.yaml
1 rosdistro: 'noetic'
2
3
4 roslaunch:
5   uris:
6     host_ab_c_39333: http://ab-c:39333/
7 rosversion: '1.16.0'
8
9
10 run_id: e028d9b2-6731-11ee-b6a4-cd19b255b937
11 turtlesim:
12   background_b: 0
13   background_g: 0
14   background_r: 0

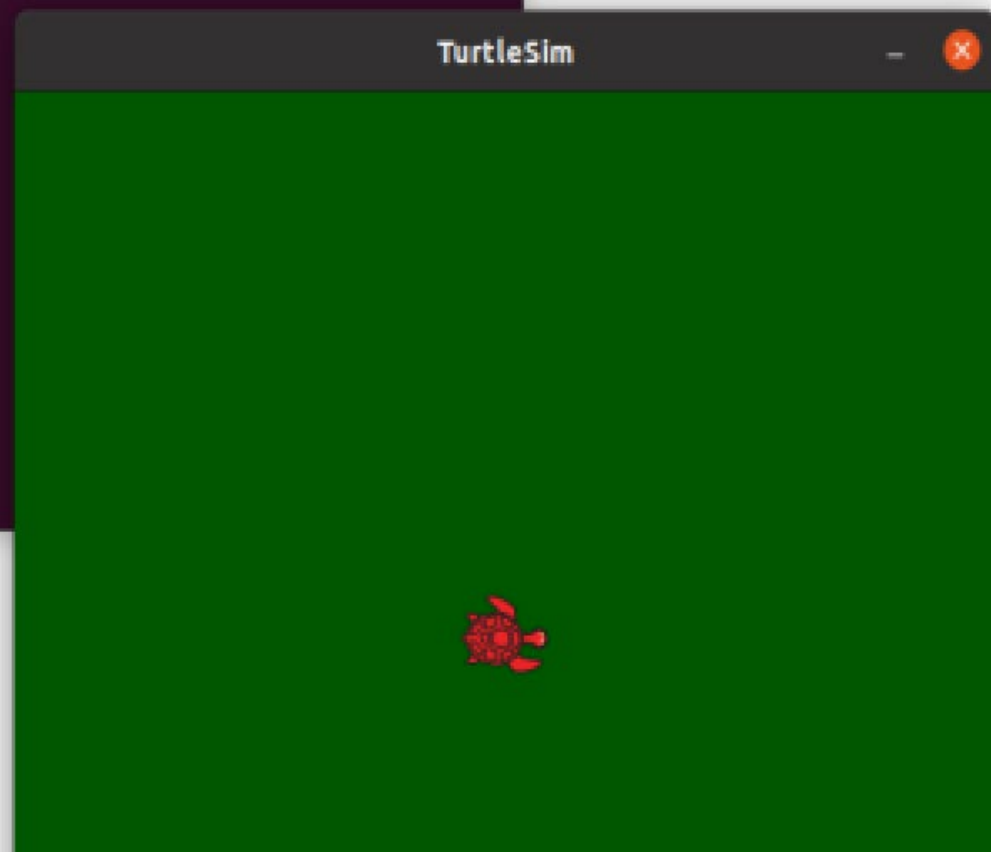
ab@ab-c:~$ rosparam load param.yaml
ab@ab-c:~$ rosparam get /turtlesim/background_b
0
```

```
ab@ab-c:~$ rosservice call /clear "{}"
ab@ab-c:~$
```



Rosparam: delete parameter

```
ab@ab-c:~$ rosparam delete /turtlesim/background_g
ab@ab-c:~$
ab@ab-c:~$ rosparam list
/rosdistro
/roslaunch/uris/host_ab_c__39333
/rosversion
/run_id
/turtlesim/background_b
/turtlesim/background_r
ab@ab-c:~$
ab@ab-c:~$ rosservice call /clear "{}"
ab@ab-c:~$
```



Create Parameter Code C++

```
Open  [v] [f] [l] ~/catkin

1 #include <string>
2 #include <ros/ros.h>
3 #include <std_srvs/Empty.h>
4
5 int main(int argc, char **argv)
6 {
7     int red, green, blue;
8
9
10    ros::init(argc, argv, "parameter_config");
11
12
13    ros::NodeHandle node;
14
15
16    ros::param::get("/background_r", red);
17    ros::param::get("/background_g", green);
18    ros::param::get("/background_b", blue);
19
20    ROS_INFO("Get Background Color[%d, %d, %d]", red, green, blue);
21
22
23    ros::param::set("/background_r", 255);
24    ros::param::set("/background_g", 255);
25    ros::param::set("/background_b", 255);
26
27    ROS_INFO("Set Background Color[255, 255, 255]");
28
29
30    ros::param::get("/background_r", red);
31    ros::param::get("/background_g", green);
32    ros::param::get("/background_b", blue);
33
34    ROS_INFO("Re-get Background Color[%d, %d, %d]", red, green, blue);
35
36
37    ros::service::waitForService("/clear");
38    ros::ServiceClient clear_background = node.serviceClient<std_srvs::Empty>("/clear");
39    std_srvs::Empty srv;
40    clear_background.call(srv);
41
42    sleep(1);
43
44    return 0;
45 }
```

Compiling Code

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

```
143
144 ## Add cmake target dependencies of the executable
145 ## same as for the library above
146 # add_dependencies(${PROJECT_NAME}_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
147
148 ## Specify libraries to link a library or executable target against
149 # target_link_libraries(${PROJECT_NAME}_node
150 #   ${catkin_LIBRARIES}
151 # )
152
153 add_executable(parameter_config src/parameter_config.cpp)
154 target_link_libraries(parameter_config ${catkin_LIBRARIES})
155 |
156 #####
157 ## Install ##
158 #####
```

Compiling Code

```
$ cd ~/catkin_ws
```

```
$ catkin_make
```

```
$ source devel/setup.bash
```

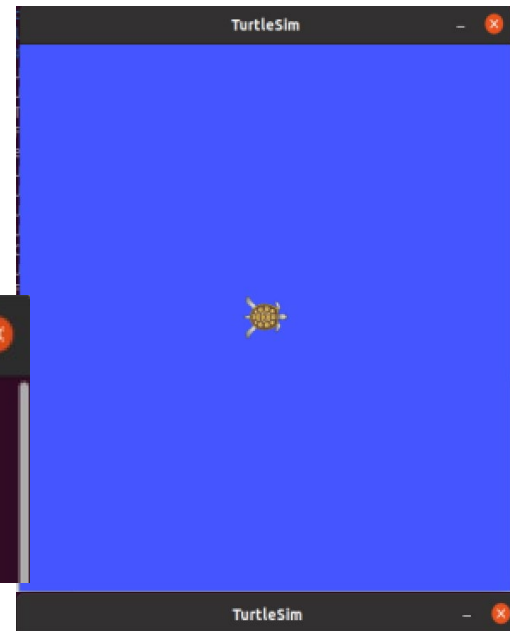
```
$ roscore
```

```
$ rosrun turtlesim turtlesim_node
```

```
$ rosrun learning_parameter parameter_config
```

Rosrun

```
ab@ab-c: ~  
ab@ab-c:~$ rosrn learning_parameter parameter_config  
[ INFO] [1696923237.602951927]: Get Backgroud Color[69, 86, 255]  
[ INFO] [1696923237.605239044]: Set Backgroud Color[255, 255, 255]  
[ INFO] [1696923237.606016400]: Re-get Backgroud Color[255, 255, 255]  
ab@ab-c:~$
```



Create Parameter Code Python

```
Text Editor ▼
Open ▼ [icon] parameter_config.py
~/catkin_ws/scripts

1 import sys
2 import rospy
3 from std_srvs.srv import Empty
4
5 def parameter_config():
6
7     rospy.init_node('parameter_config', anonymous=True)
8
9
10    red = rospy.get_param('/background_r')
11    green = rospy.get_param('/background_g')
12    blue = rospy.get_param('/background_b')
13
14    rospy.loginfo("Get Background Color[%d, %d, %d]", red, green, blue)
15
16
17    rospy.set_param("/background_r", 255);
18    rospy.set_param("/background_g", 255);
19    rospy.set_param("/background_b", 255);
20
21    rospy.loginfo("Set Background Color[255, 255, 255]");
22
23
24    red = rospy.get_param('/background_r')
25    green = rospy.get_param('/background_g')
26    blue = rospy.get_param('/background_b')
27
28    rospy.loginfo("Get Background Color[%d, %d, %d]", red, green, blue)
29
30
31    rospy.wait_for_service('/clear')
32    try:
33        clear_background = rospy.ServiceProxy('/clear', Empty)
34
35        |
36        response = clear_background()
37        return response
38    except rospy.ServiceException, e:
39        print "Service call failed: %s"%e
40
41 if __name__ == "__main__":
42     parameter_config()
```

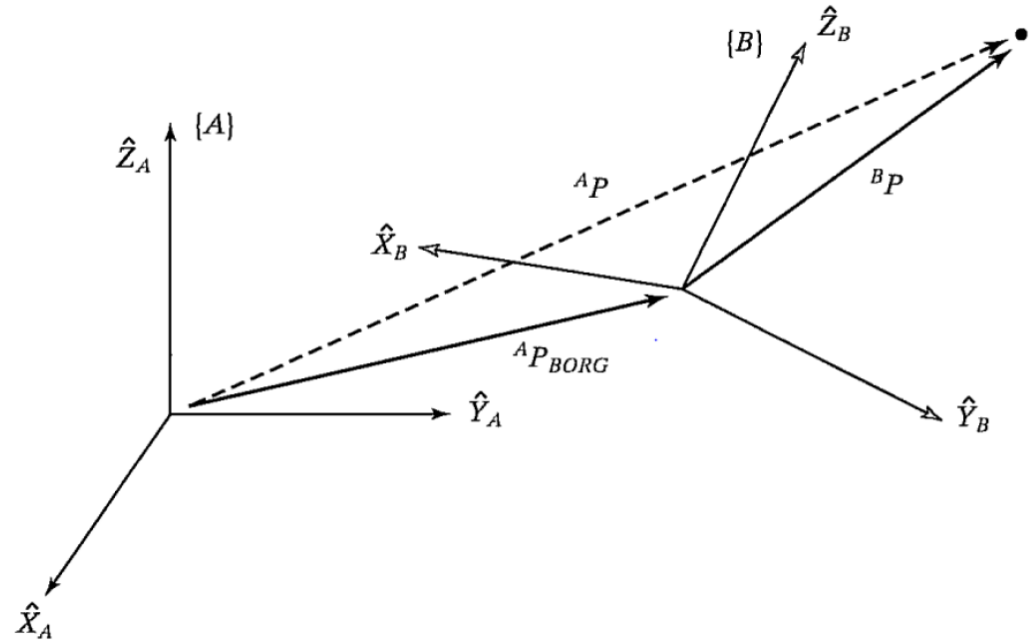

Lecture 12: ROS Coordinate System

Coordinate Transformation

$${}^A P = {}^A R {}^B P + {}^A P_{BORG}.$$

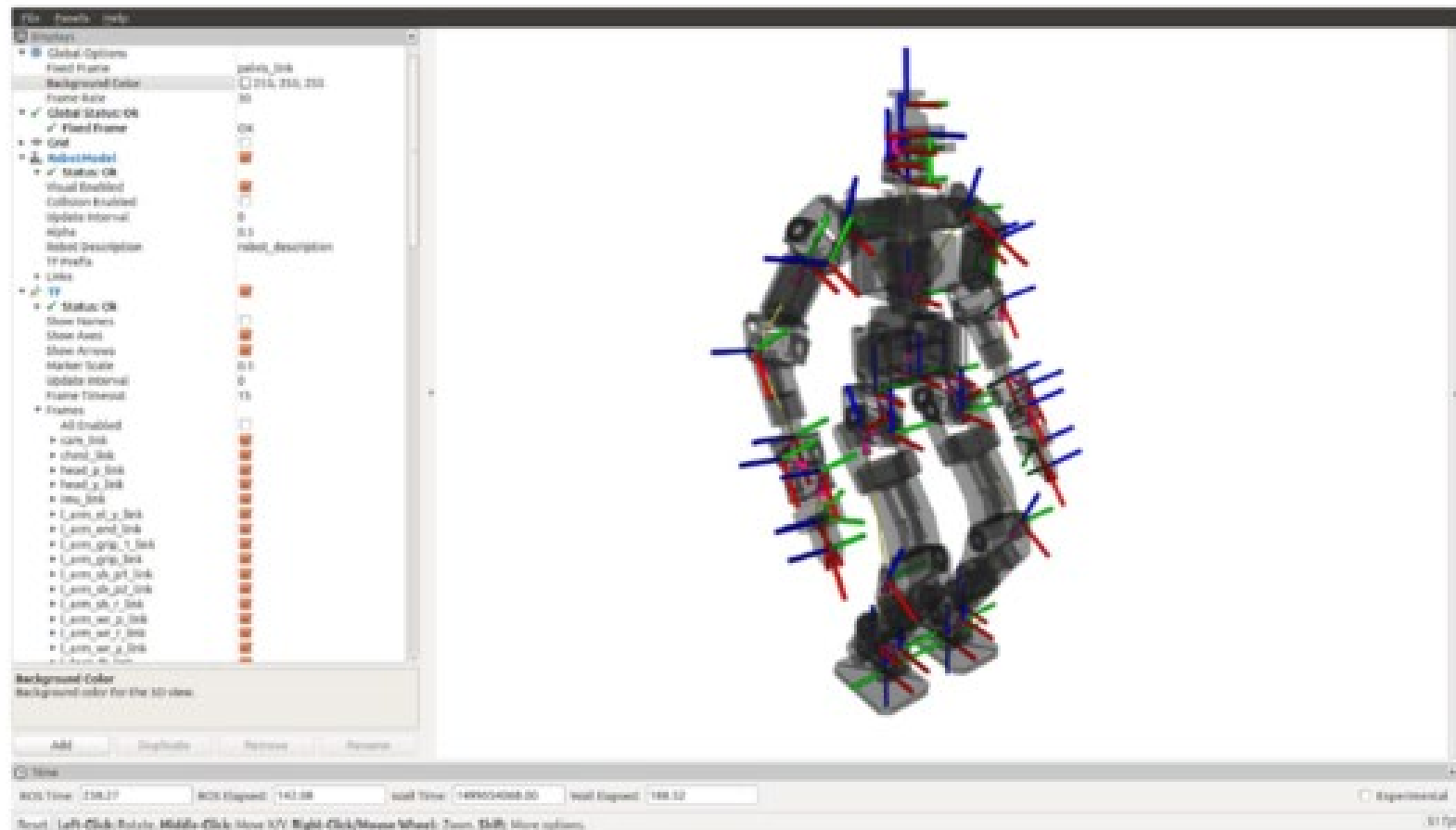
$${}^A P = {}^A T {}^B P.$$

$$\begin{bmatrix} {}^A P \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} {}^A R & & & {}^A P_{BORG} \\ 0 & 0 & 0 & 1 \end{array} \right] \begin{bmatrix} {}^B P \\ 1 \end{bmatrix}.$$

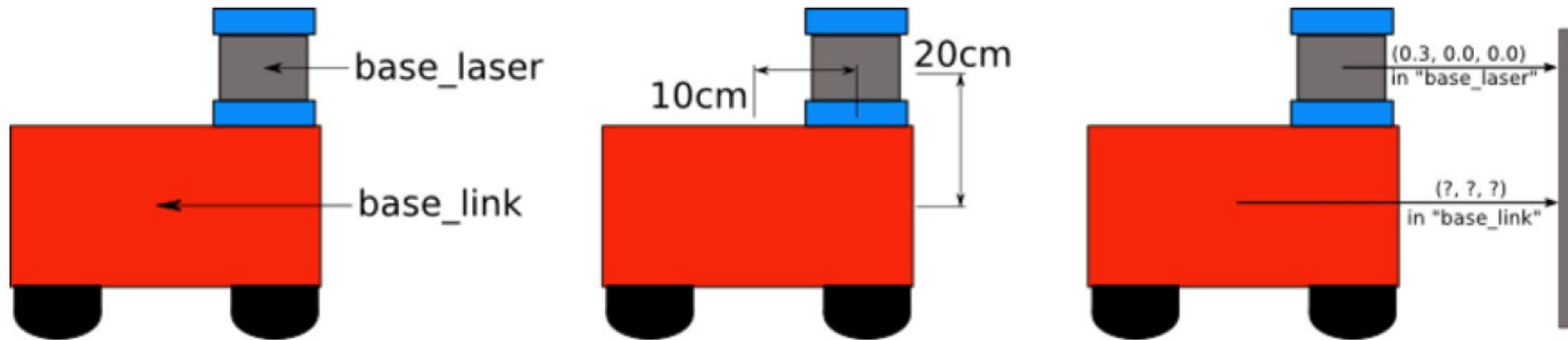


Coordinate transformation(TF, transform)

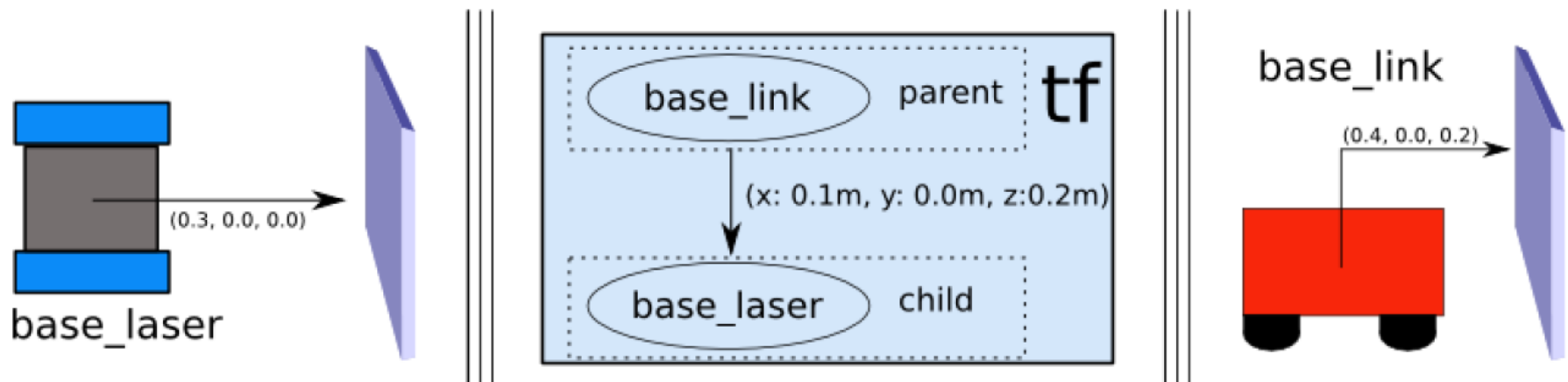
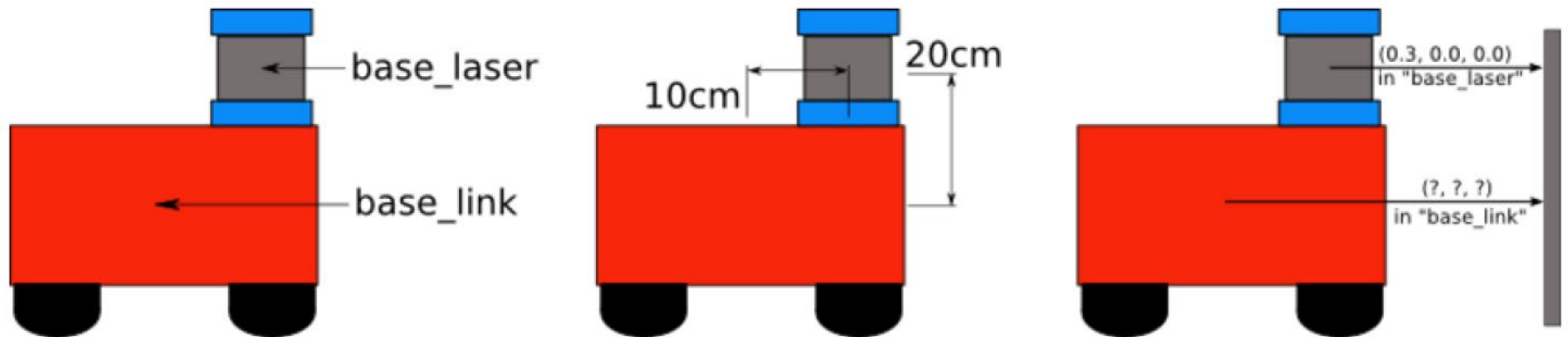
- Relative coordinate transformation of each joint
 - Indicates the relationship between joints in the form of tree structure



Coordinate transformation(TF, transform)



Coordinate transformation(TF, transform)



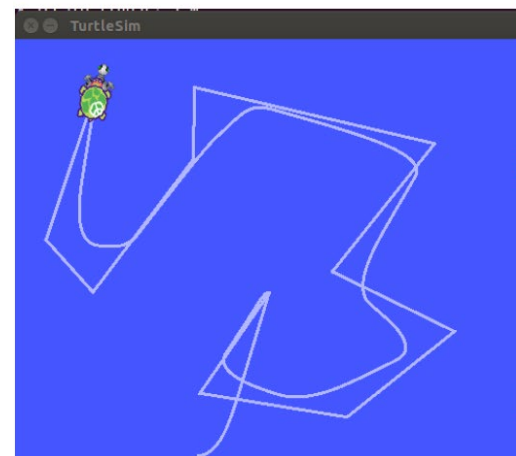
Coordinate transformation(TF, transform)

```
ab@ab-c:~$ sudo apt-get install ros-noetic-turtle-tf
```

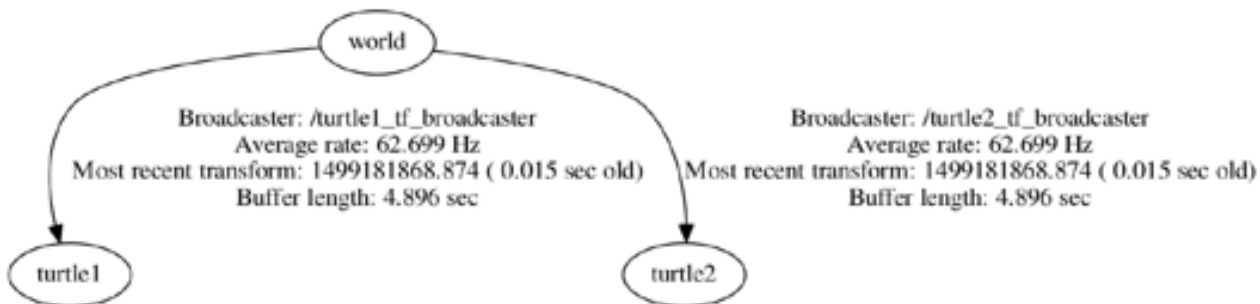
```
ab@ab-c:~$ sudo apt install python3-is-python3
```

```
ab@ab-c:~$ roslaunch turtle_tf turtle_tf_demo.launch
```

```
ab@ab-c:~$ rosrn turtlesim turtle_teleop_key
```

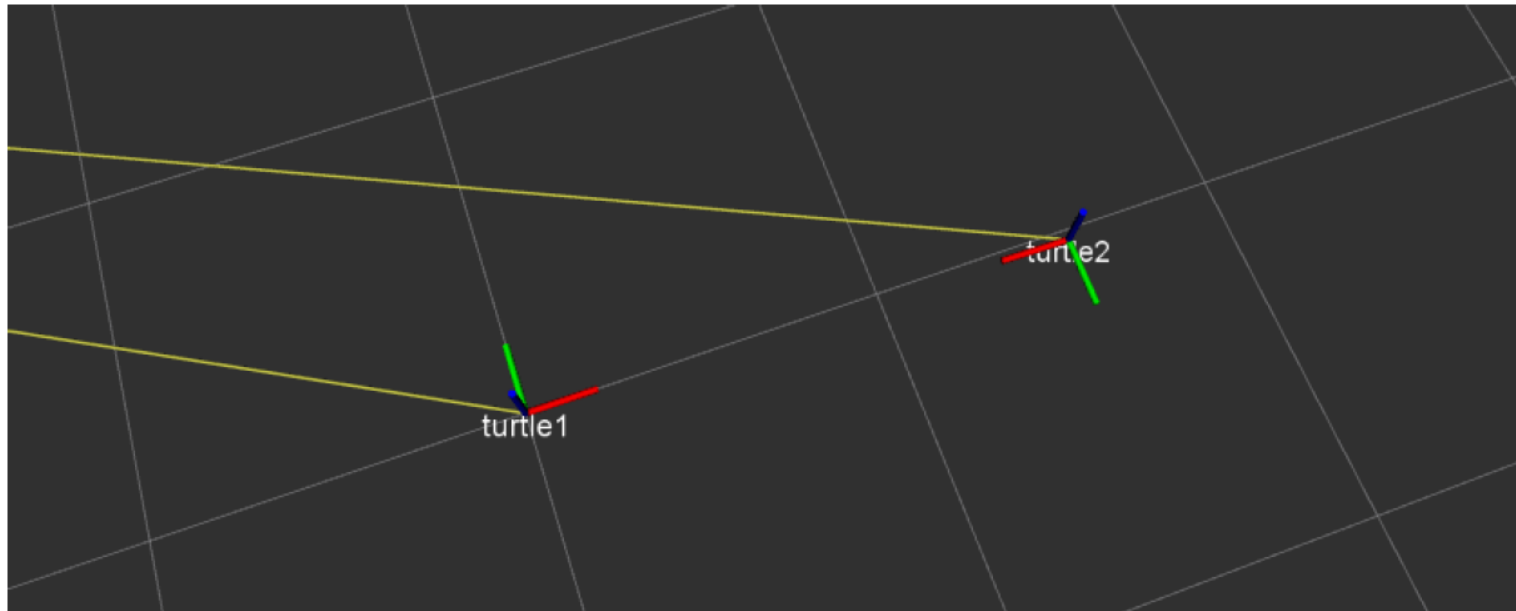


view_frames Result
Recorded at time: 1499181868.889



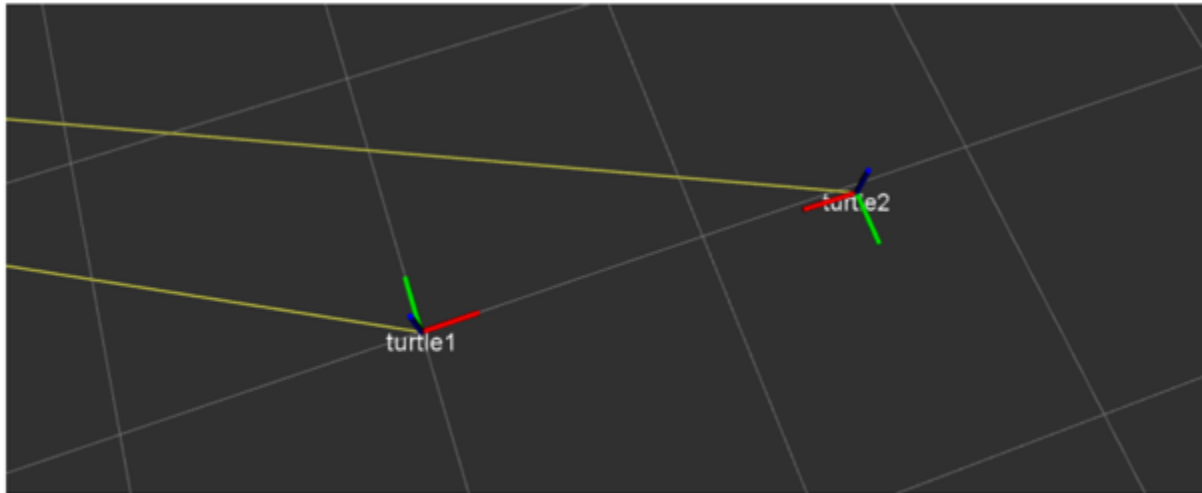
Coordinate transformation(TF, transform)

```
→ ~ rosrun tf tf_echo turtle1 turtle2
At time 1504942486.329
- Translation: [0.000, 0.000, 0.000]
- Rotation: in Quaternion [0.000, 0.000, 0.311, 0.950]
             in RPY (radian) [0.000, -0.000, 0.633]
             in RPY (degree) [0.000, -0.000, 36.290]
At time 1504942487.018
- Translation: [0.000, 0.000, 0.000]
- Rotation: in Quaternion [0.000, 0.000, 0.311, 0.950]
             in RPY (radian) [0.000, -0.000, 0.633]
             in RPY (degree) [0.000, -0.000, 36.290]
```

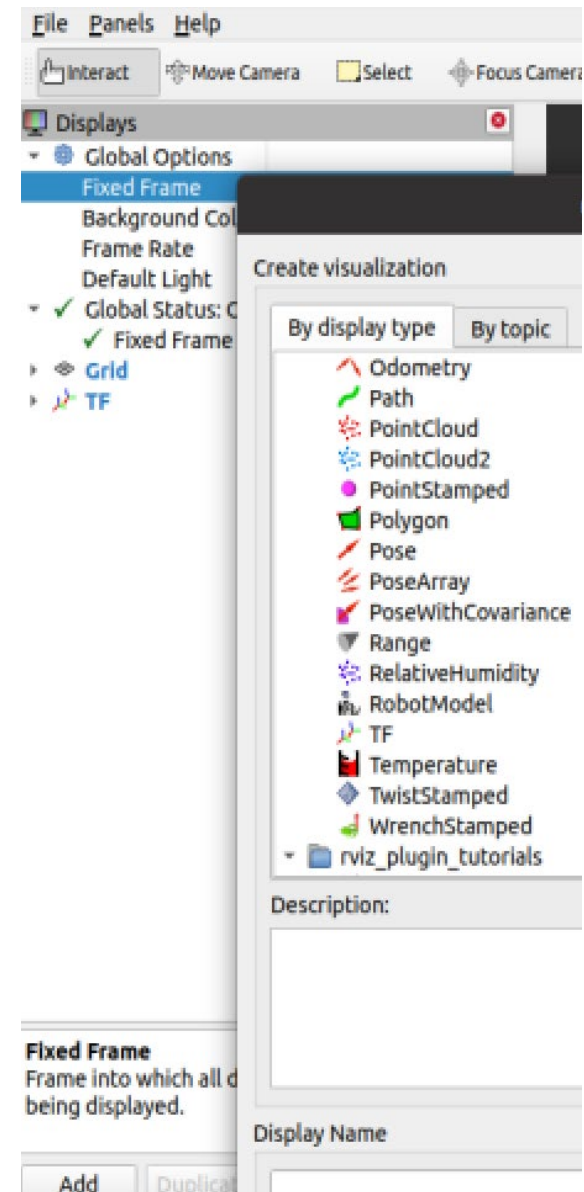
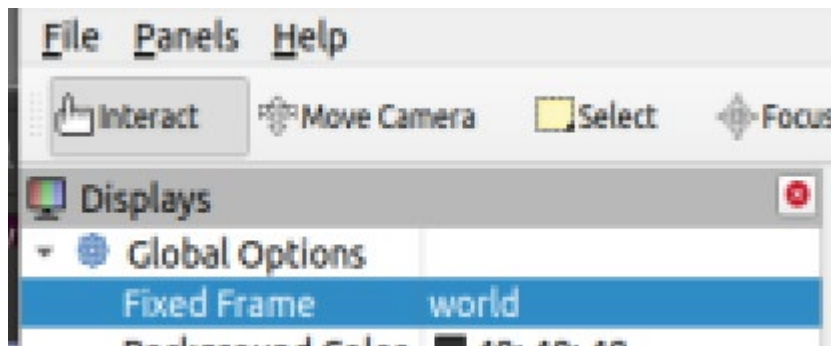


```
$ rosrun rviz rviz -d `rospack find turtle_tf` /rviz/turtle_rviz.rviz
```

Coordinate transformation(TF, transform)



```
$ rosrn rviz rviz -d `rospack find turtle_tf` /rviz/turtle_rviz.rviz
```



Reference



Free

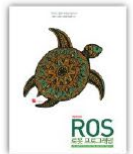


Download link



Language:

English, chinese, Japanese, Korean



“ROS Robot Programming”

A Handbook is written by TurtleBot3 Developers

Reference

- ❑ **R. Siegwart, I. R. Nourbakhsh, D. Scaramuzza. Introduction to Autonomous Mobile Robots. MIT Press, 2nd Edition, 2011, ISBN-10: 0262015358.**
- ❑ **Y. Pyo, H. Cho, R. Jung, and T. Lim, ROS Robot Programming, ROBOTIS Co., Ltd., 2017, ISBN 979-11-962307-1-5**
- ❑ **J. O’Kane, A Gentle Introduction to ROS, CreateSpace Independent Publishing Platform, 2013, ISBN-13: 978-1492143239**