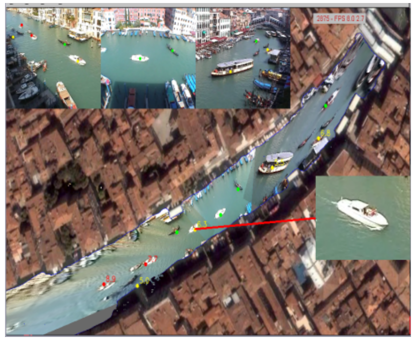
UNIVERSITÀ di VERONA | Dipartimento di INFORMATICA

Laurea magistrale in ingegneria e scienze informatiche

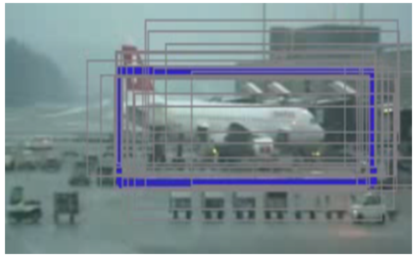
Corso di Robotica
Parte di Laboratorio

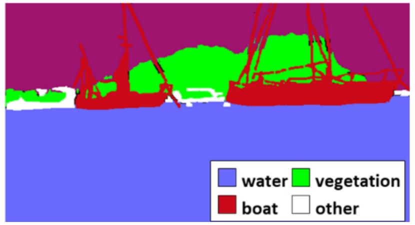


Esempio di applicazione

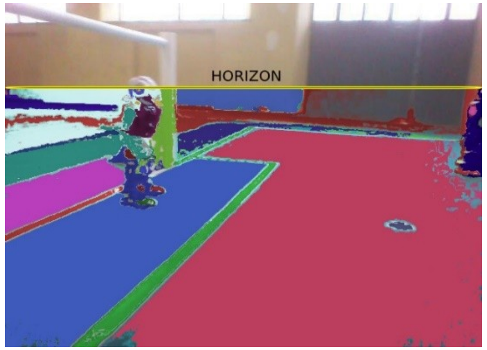
Docente:

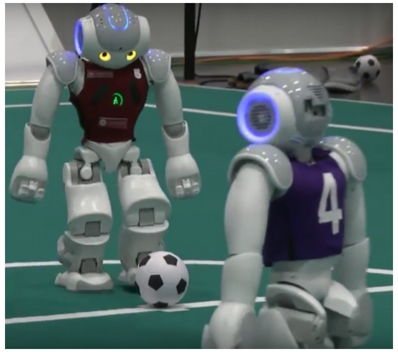
Domenico Daniele Bloisi





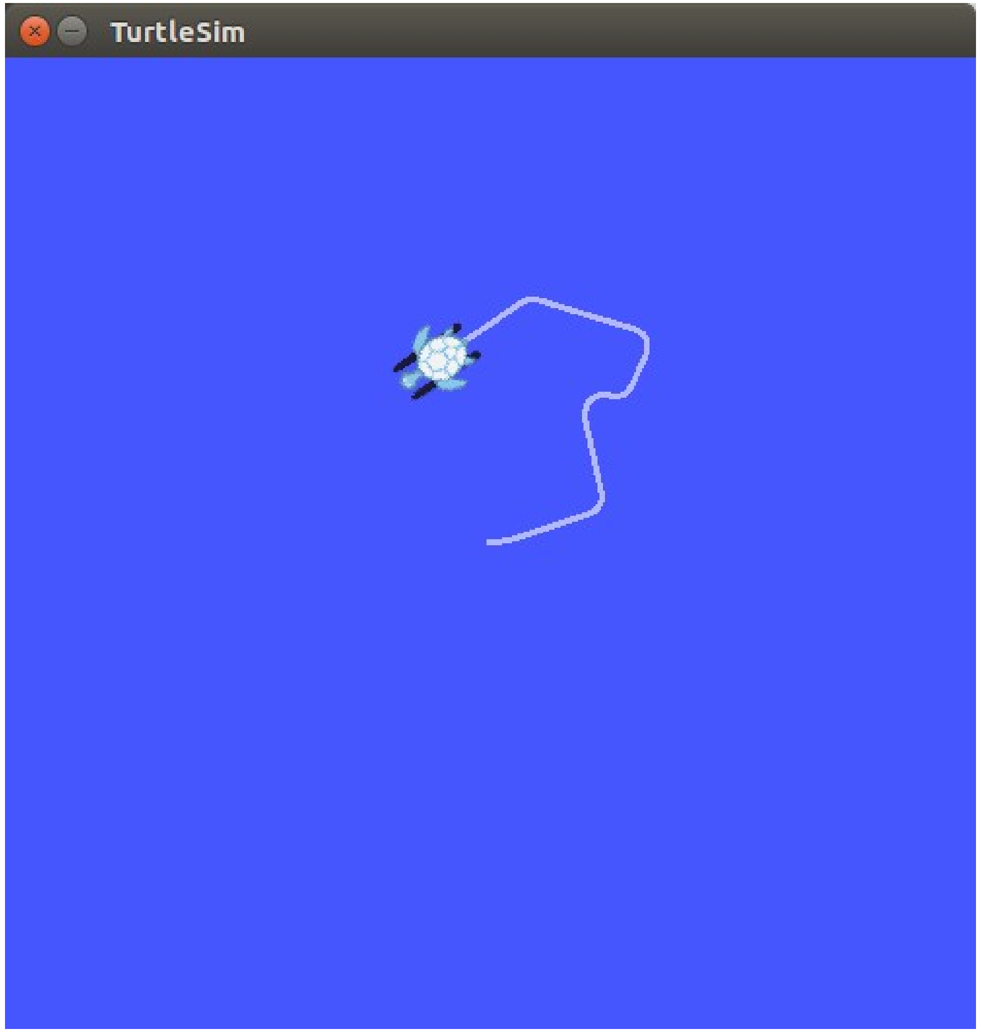
Dicembre 2017





Teleoperazione in ROS

Obiettivo: realizzare un nodo ROS per teleoperare da tastiera un robot simulato



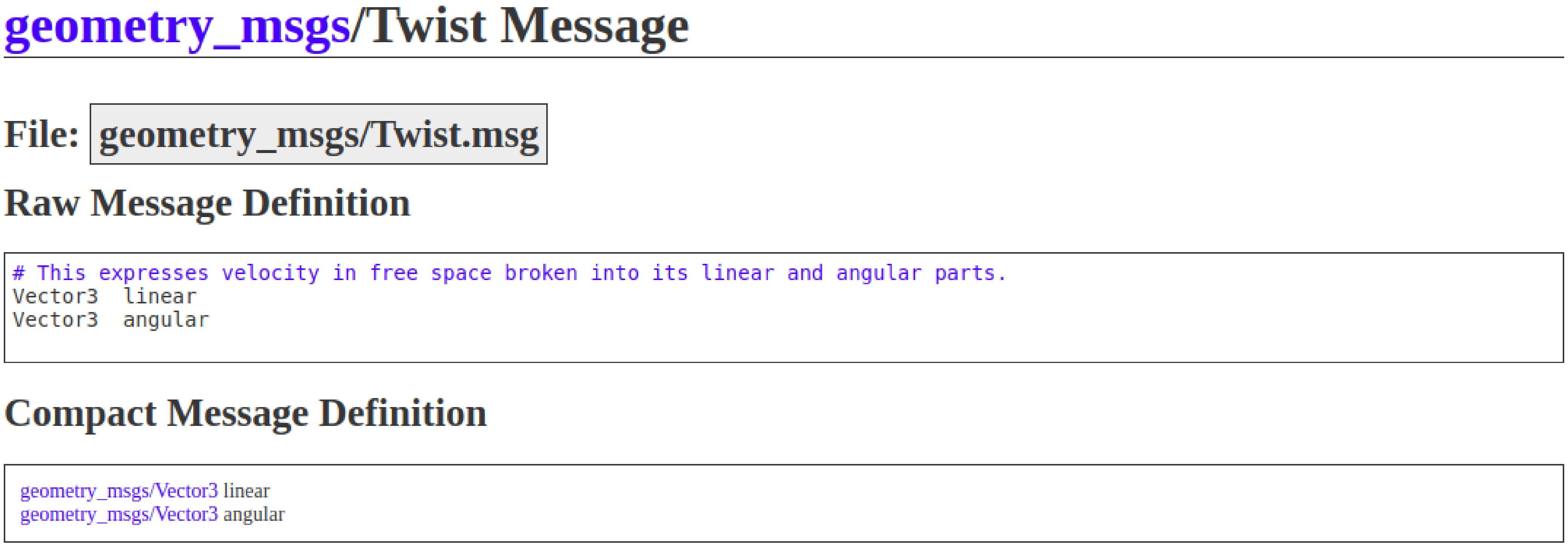
Idea

- Possiamo far muovere il robot controllandone la velocità

- Ci servirà controllare la velocità lineare e la velocità angolare

Comandi di velocità

- Per far muovere un robot in ROS è necessario pubblicare Twist messages sul topic cmd_vel



geometry_msgs/Twist Message

File: geometry_msgs/Twist.msg

Raw Message Definition

```
# This expresses velocity in free space broken into its linear and angular parts.
Vector3  linear
Vector3  angular
```

Compact Message Definition

```
geometry_msgs/Vector3 linear
geometry_msgs/Vector3 angular
```

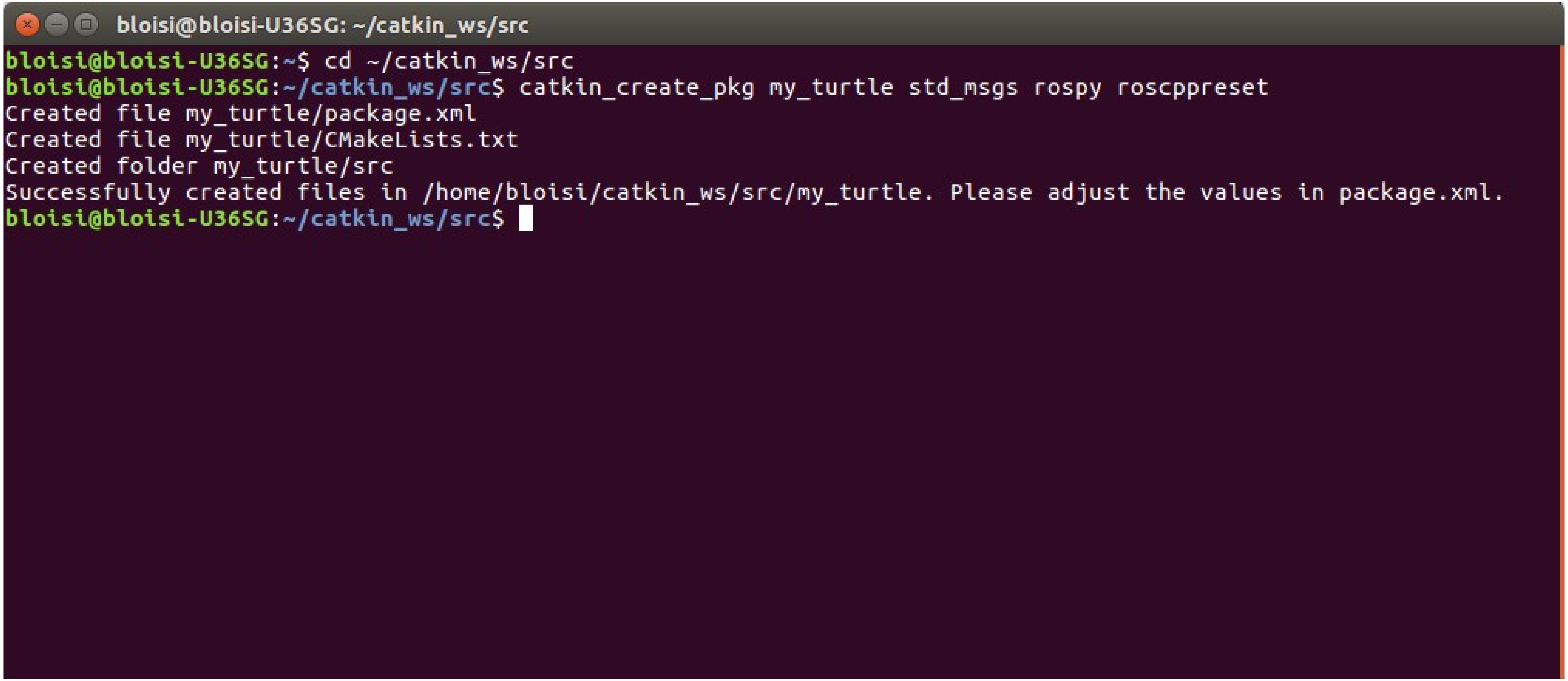
Package my_turtle

Iniziamo creando un package ROS my_turtle che conterrà codice del nodo e il launch file

Comandi:

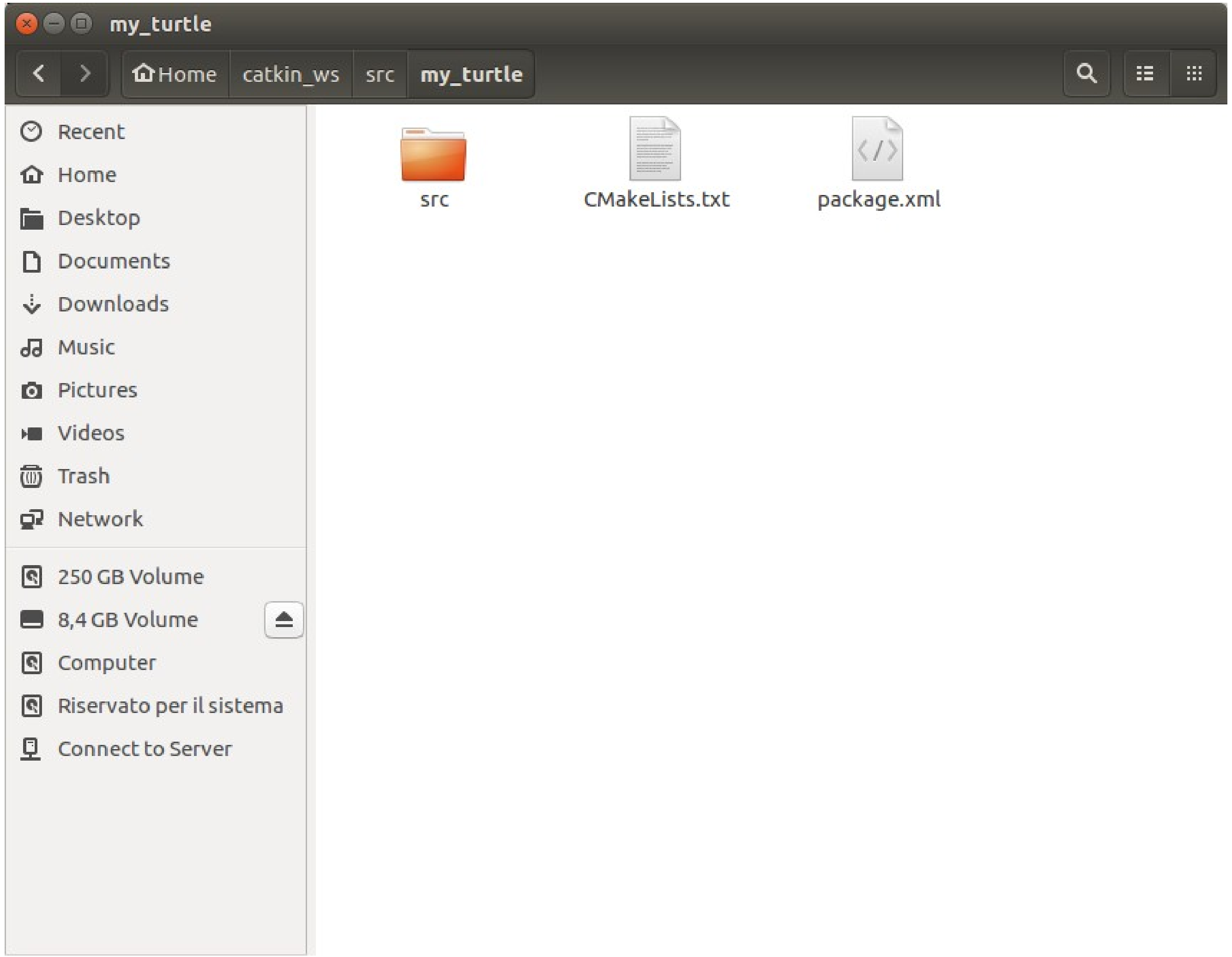
```
$ cd ~/catkin_ws/src
$ catkin_create_pkg my_turtle std_msgs rospy roscpp
```

Package my_turtle

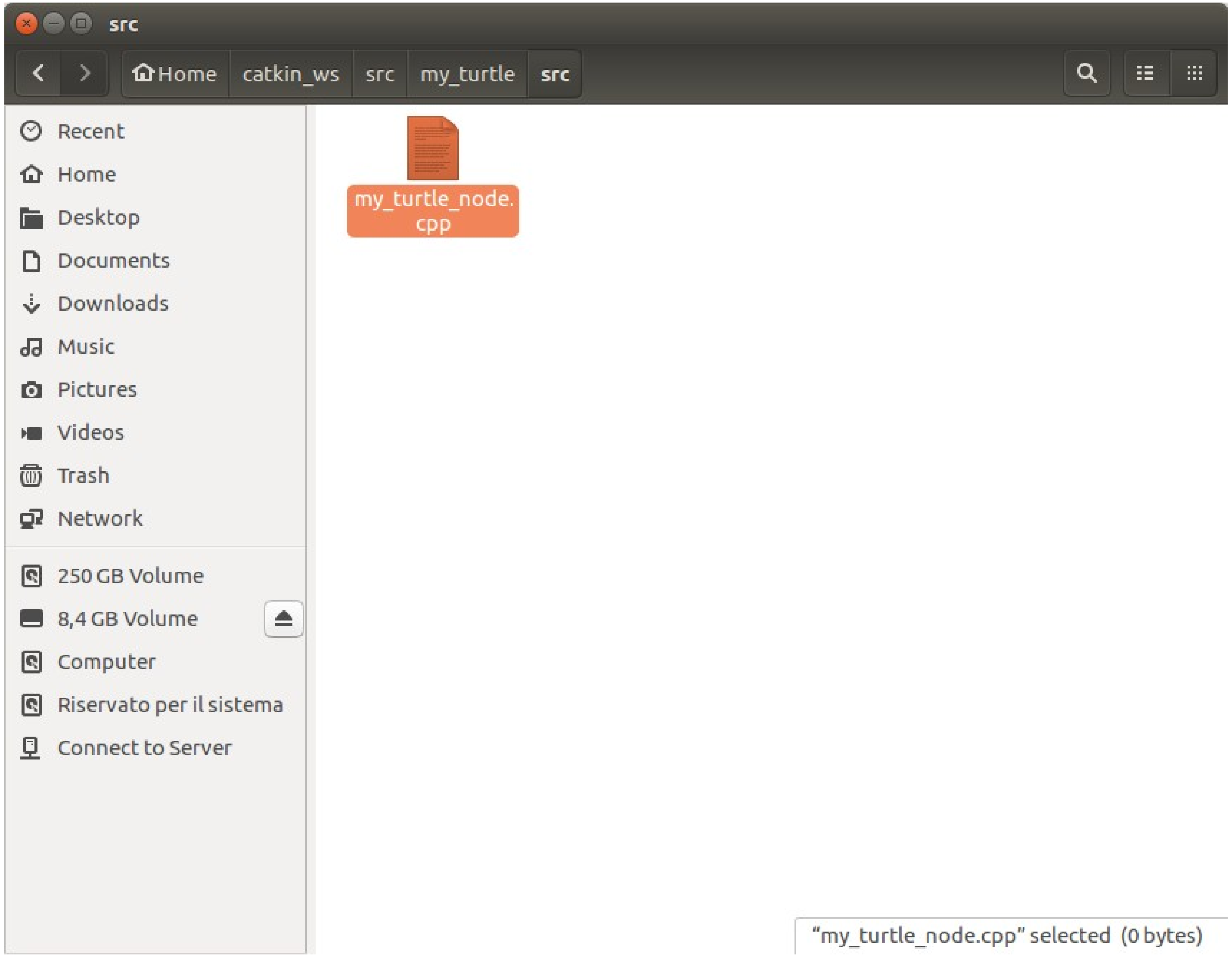


```
bloisi@bloisi-U36SG:~$ cd ~/catkin_ws/src
bloisi@bloisi-U36SG:~/catkin_ws/src$ catkin_create_pkg my_turtle std_msgs rospy roscppreset
Created file my_turtle/package.xml
Created file my_turtle/CMakeLists.txt
Created folder my_turtle/src
Successfully created files in /home/bloisi/catkin_ws/src/my_turtle. Please adjust the values in package.xml.
bloisi@bloisi-U36SG:~/catkin_ws/src$ 
```

Package my_turtle



Nodo my_turtle_node

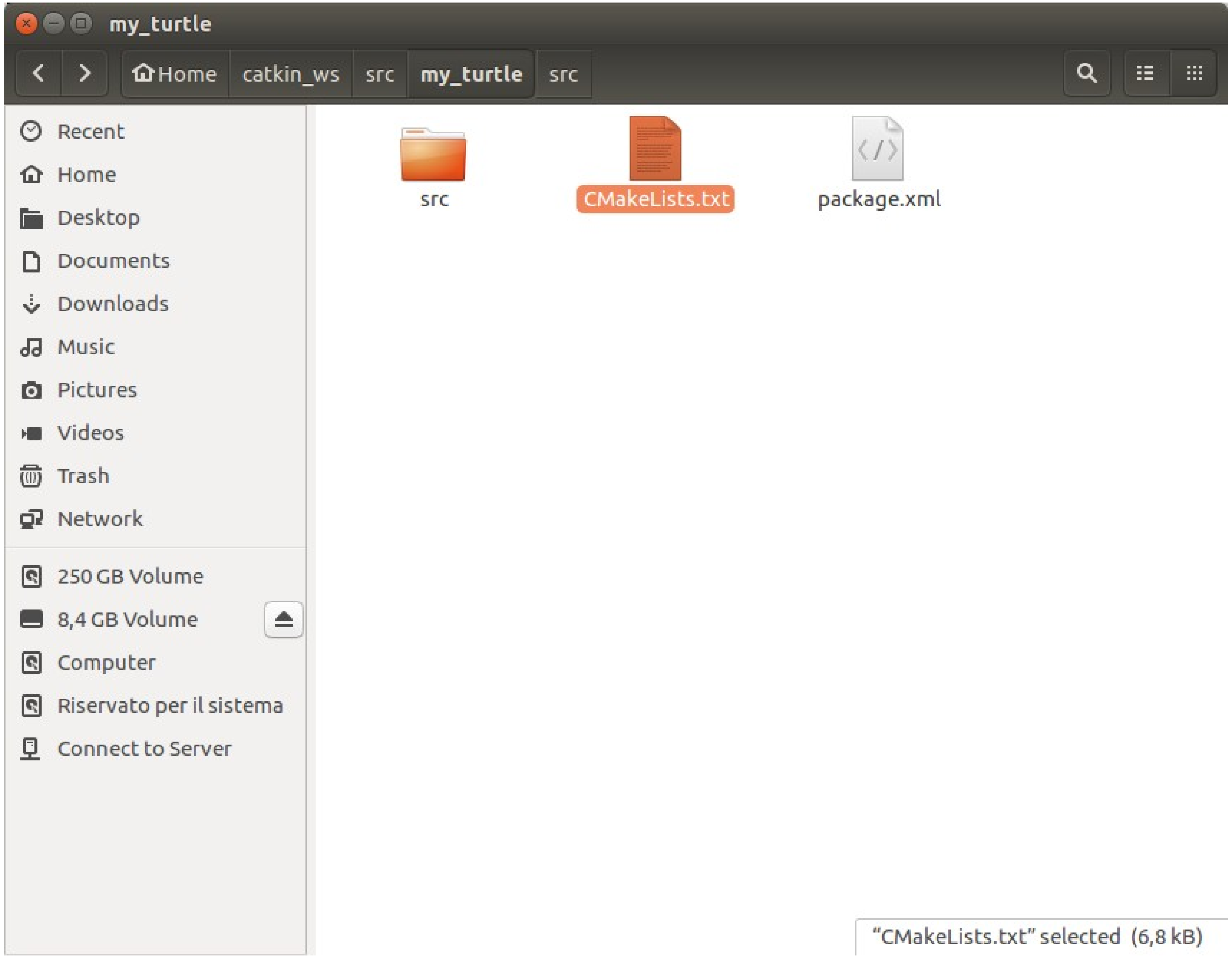


“my_turtle_node.cpp” selected (0 bytes)

Nodo my_turtle_node

```
#include "ros/ros.h"
#include "geometry_msgs/Twist.h"

int main(int argc, char **argv)
{
    const double FORWARD_SPEED_MPS = 0.5;

    // Initialize the node
    ros::init(argc, argv, "move_turtle");
    ros::NodeHandle node;

    // A publisher for the movement data
    ros::Publisher pub = node.advertise<geometry_msgs::Twist>("turtle1/cmd_vel", 10);

    // Drive forward at a given speed. The robot points up the x-axis.
    // The default constructor will set all commands to 0
    geometry_msgs::Twist msg;
    msg.linear.x = FORWARD_SPEED_MPS;

    // Loop at 10Hz, publishing movement commands until we shut down
    ros::Rate rate(10);
    ROS_INFO("Starting to move forward");
    while (ros::ok()) {
        pub.publish(msg);
        rate.sleep();
    }
}
```

CmakeLists.txt

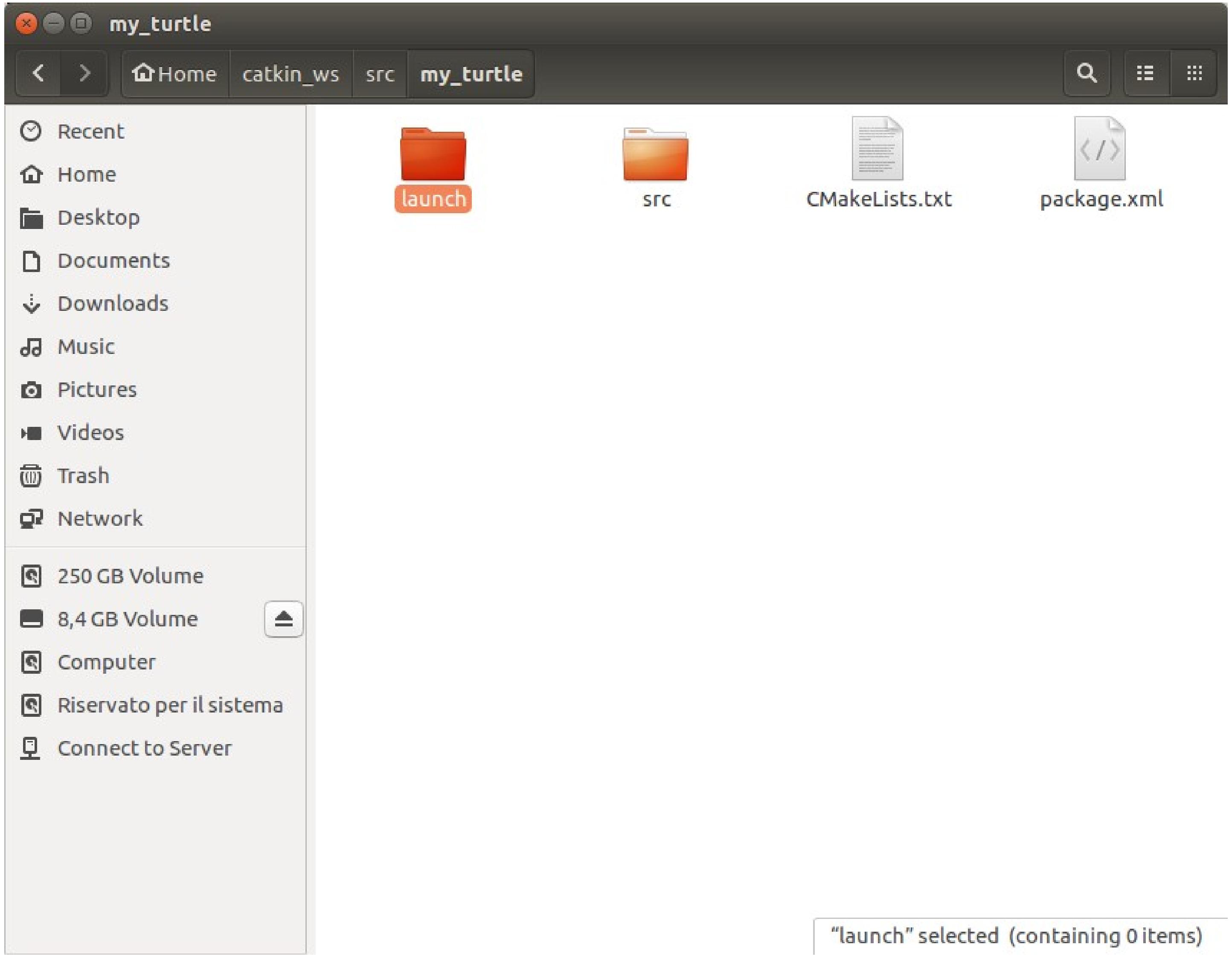


"CMakeLists.txt" selected (6,8 kB)

CmakeLists.txt

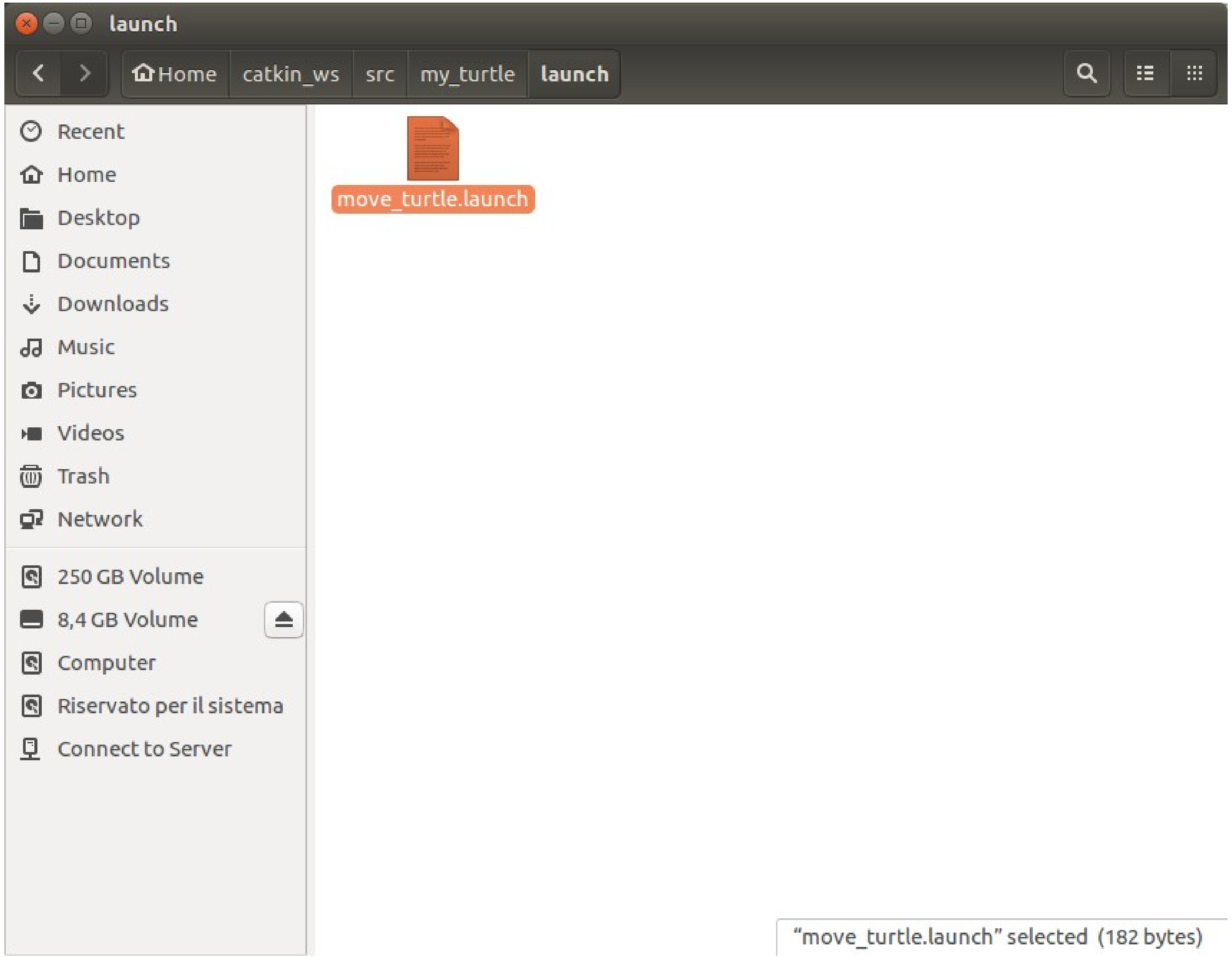
```
cmake_minimum_required(VERSION 2.8.3)
project(my_turtle)

add_compile_options(-std=c++11)

find_package(catkin REQUIRED COMPONENTS
  roscpp
  rospy
  std_msgs
)

catkin_package()

include_directories(
  src/
  ${catkin_INCLUDE_DIRS}
)

add_executable(${PROJECT_NAME}_node src/my_turtle_node.cpp)

target_link_libraries(${PROJECT_NAME}_node
   ${catkin_LIBRARIES}
)
```

Launch File



“launch” selected (containing 0 items)

Launch File



“move_turtle.launch” selected (182 bytes)

Launch File

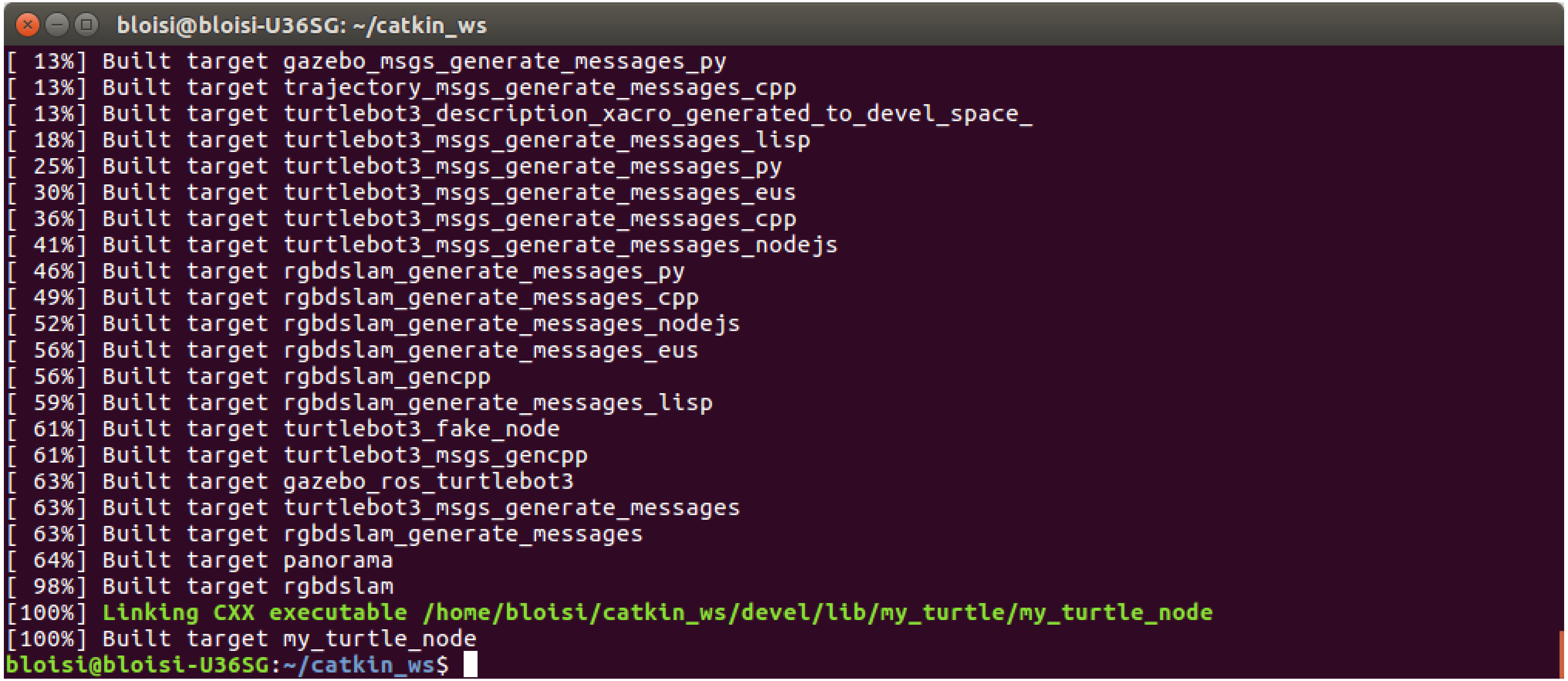
```
<launch>
    <node name="turtlesim_node" pkg="turtlesim" type="turtlesim_node" />
    <node name="my_turtle_node" pkg="my_turtle" type="my_turtle_node" output="screen" />
</launch>
```

catkin_make

Comandi:

```
$ cd ~/catkin_ws
$ catkin_make
```

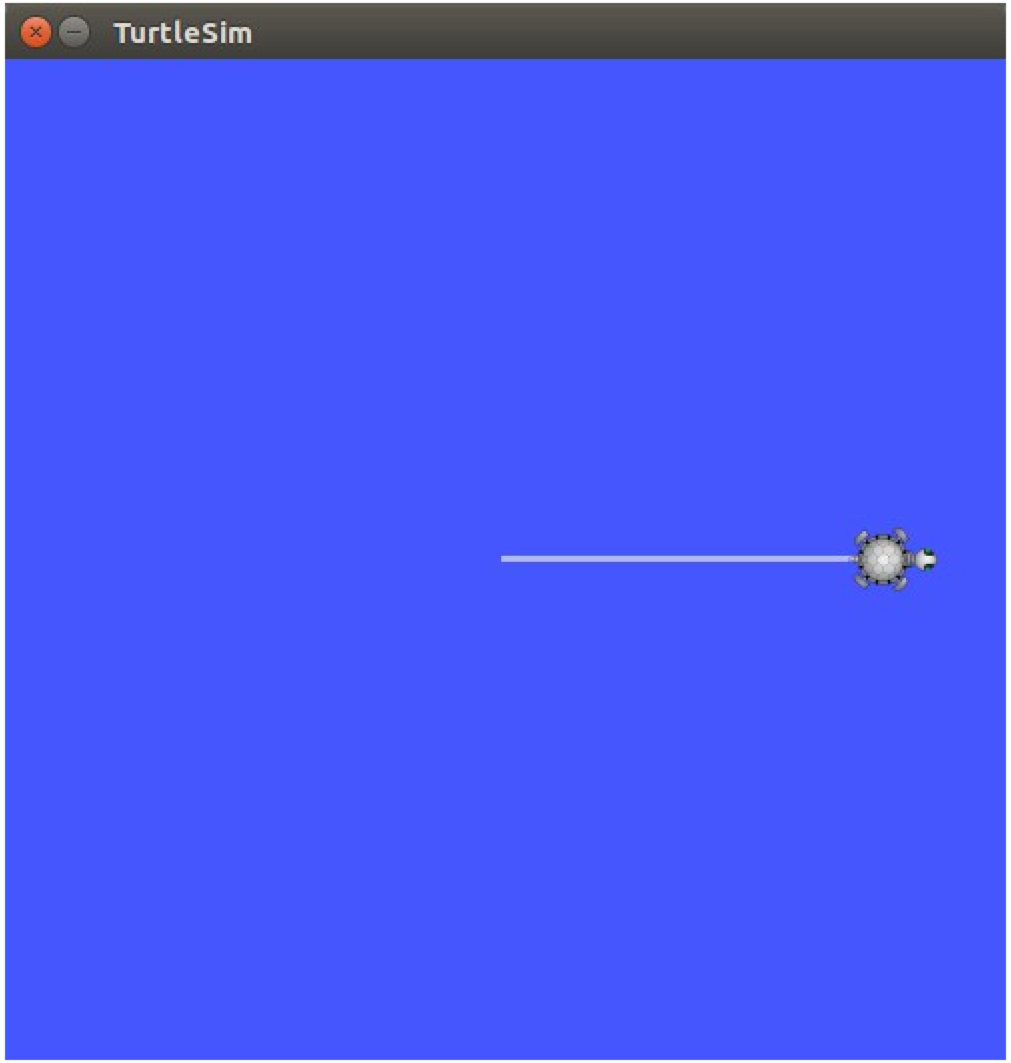
catkin_make



```
[ 13%] Built target gazebo_msgs_generate_messages_py
[ 13%] Built target trajectory_msgs_generate_messages_cpp
[ 13%] Built target turtlebot3_description_xacro_generated_to_devel_space_
[ 18%] Built target turtlebot3_msgs_generate_messages_lisp
[ 25%] Built target turtlebot3_msgs_generate_messages_py
[ 30%] Built target turtlebot3_msgs_generate_messages_eus
[ 36%] Built target turtlebot3_msgs_generate_messages_cpp
[ 41%] Built target turtlebot3_msgs_generate_messages_nodejs
[ 46%] Built target rgbdslam_generate_messages_py
[ 49%] Built target rgbdslam_generate_messages_cpp
[ 52%] Built target rgbdslam_generate_messages_nodejs
[ 56%] Built target rgbdslam_generate_messages_eus
[ 56%] Built target rgbdslam_gencpp
[ 59%] Built target rgbdslam_generate_messages_lisp
[ 61%] Built target turtlebot3_fake_node
[ 61%] Built target turtlebot3_msgs_gencpp
[ 63%] Built target gazebo_ros_turtlebot3
[ 63%] Built target turtlebot3_msgs_generate_messages
[ 63%] Built target rgbdslam_generate_messages
[ 64%] Built target panorama
[ 98%] Built target rgbdslam
[100%] Linking CXX executable /home/bloisi/catkin_ws/devel/lib/my_turtle/my_turtle_node
[100%] Built target my_turtle_node
bloisi@bloisi-U36SG:~/catkin_ws$
```

Eseguire il launch file

```
$ roslaunch my_turtle move_turtle.launch
```



Stampare la Robot Pose

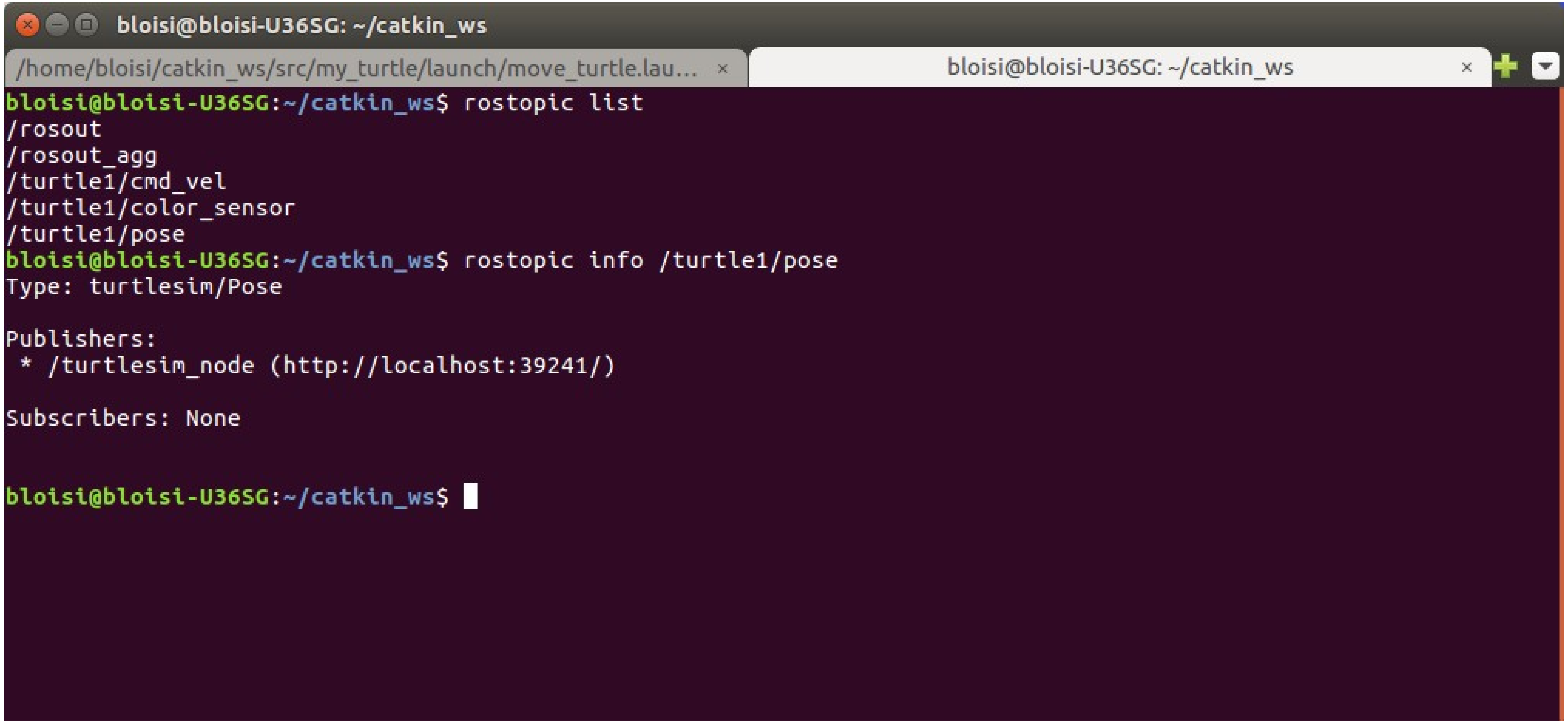
Per poter stampare la robot pose abbiamo bisogno di creare un subscriber al topic turtle1/pose

```
ros::Subscriber sub = node.subscribe("turtle1/pose", 10, poseCallback);
```

Va creata anche una opportuna callback che stampi il messaggio

```
void poseCallback(const turtlesim::PoseConstPtr& msg)
{
    ROS_INFO("x: %.2f, y: %.2f", msg->x, msg->y);
}
```

Tipo del messaggio per Robot Pose



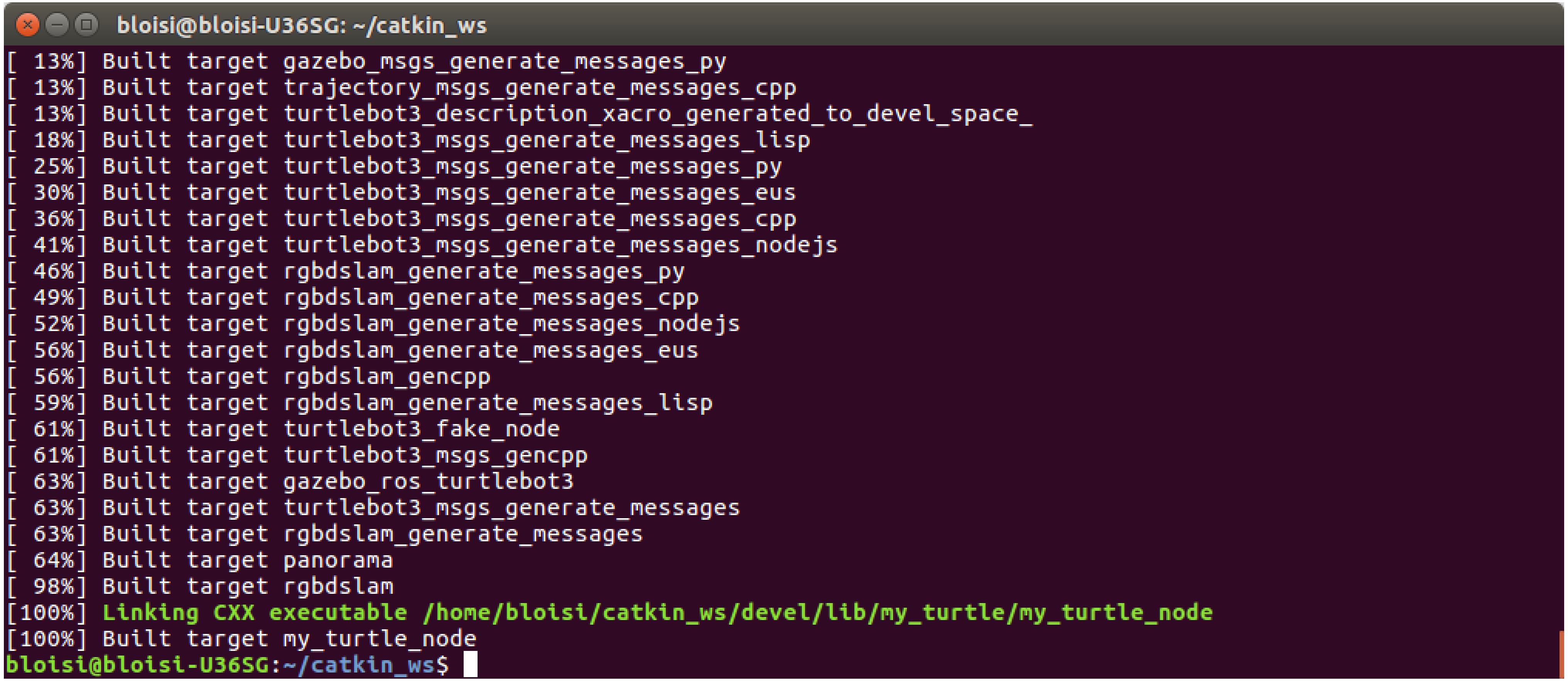
bloisi@bloisi-U36SG: ~/catkin_ws
/home/bloisi/catkin_ws/src/my_turtle/launch/move_turtle.lau...
bloisi@bloisi-U36SG: ~/catkin_ws

```
bloisi@bloisi-U36SG:~/catkin_ws$ rostopic list
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
bloisi@bloisi-U36SG:~/catkin_ws$ rostopic info /turtle1/pose
Type: turtlesim/Pose

Publishers:
 * /turtlesim_node (http://localhost:39241/)

Subscribers: None


bloisi@bloisi-U36SG:~/catkin_ws$ 
```

Modifica a my_turtle_node.cpp

```
#include "ros/ros.h"
#include "geometry_msgs/Twist.h"
#include "turtlesim/Pose.h"

// Topic messages callback
void poseCallback(const turtlesim::PoseConstPtr& msg)
{
    ROS_INFO("x: %.2f, y: %.2f", msg->x, msg->y);
}

int main(int argc, char **argv)
{
    const double FORWARD_SPEED_MPS = 0.5;

    // Initialize the node
    ros::init(argc, argv, "move_turtle");
    ros::NodeHandle node;

    // A publisher for the movement data
    ros::Publisher pub = node.advertise<geometry_msgs::Twist>("turtle1/cmd_vel", 10);

    // A listener for pose
    ros::Subscriber sub = node.subscribe("turtle1/pose", 10, poseCallback);
    // Drive forward at a given speed. The robot points up the x-axis.
    // The default constructor will set all commands to 0
    geometry_msgs::Twist msg;
    msg.linear.x = FORWARD_SPEED_MPS;

    // Loop at 10Hz, publishing movement commands until we shut down
    ros::Rate rate(10);
    ROS_INFO("Starting to move forward");
    while (ros::ok()) {
        pub.publish(msg);
        ros::spinOnce(); // Allow processing of incoming messages
        rate.sleep();
    }
}
```

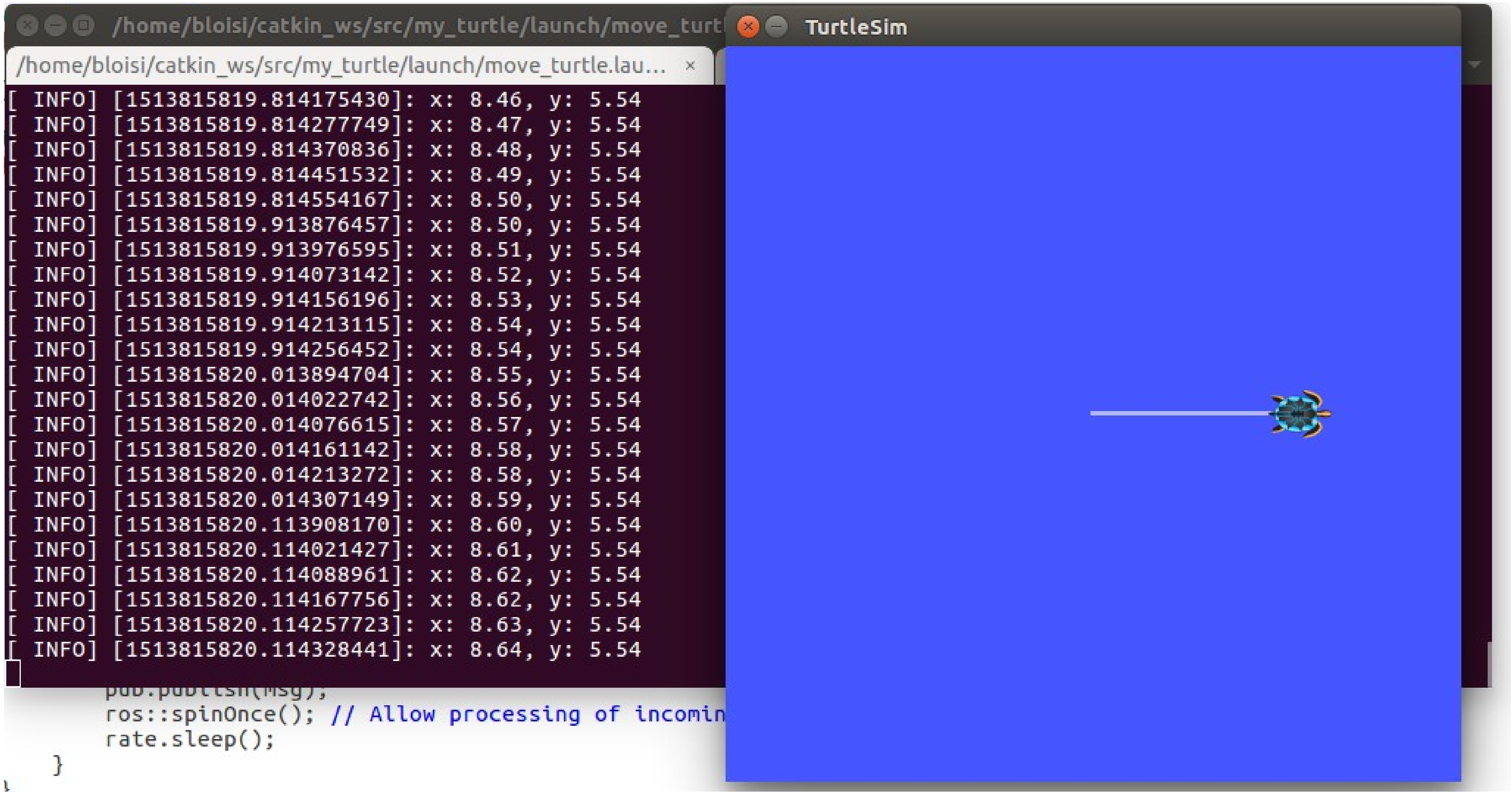
catkin_make



```
[ 13%] Built target gazebo_msgs_generate_messages_py
[ 13%] Built target trajectory_msgs_generate_messages_cpp
[ 13%] Built target turtlebot3_description_xacro_generated_to_devel_space_
[ 18%] Built target turtlebot3_msgs_generate_messages_lisp
[ 25%] Built target turtlebot3_msgs_generate_messages_py
[ 30%] Built target turtlebot3_msgs_generate_messages_eus
[ 36%] Built target turtlebot3_msgs_generate_messages_cpp
[ 41%] Built target turtlebot3_msgs_generate_messages_nodejs
[ 46%] Built target rgbdslam_generate_messages_py
[ 49%] Built target rgbdslam_generate_messages_cpp
[ 52%] Built target rgbdslam_generate_messages_nodejs
[ 56%] Built target rgbdslam_generate_messages_eus
[ 56%] Built target rgbdslam_gencpp
[ 59%] Built target rgbdslam_generate_messages_lisp
[ 61%] Built target turtlebot3_fake_node
[ 61%] Built target turtlebot3_msgs_gencpp
[ 63%] Built target gazebo_ros_turtlebot3
[ 63%] Built target turtlebot3_msgs_generate_messages
[ 63%] Built target rgbdslam_generate_messages
[ 64%] Built target panorama
[ 98%] Built target rgbdslam
[100%] Linking CXX executable /home/bloisi/catkin_ws/devel/lib/my_turtle/my_turtle_node
[100%] Built target my_turtle_node
bloisi@bloisi-U36SG:~/catkin_ws$ 
```

Esecuzione

```
$ roslaunch my_turtle move_turtle.launch
```



/home/bloisi/catkin_ws/src/my_turtle/launch/move_turtl
/home/bloisi/catkin_ws/src/my_turtle/launch/move_turtle.lau...
TurtleSim

```
[ INFO] [1513815819.814175430]: x: 8.46, y: 5.54
[ INFO] [1513815819.814277749]: x: 8.47, y: 5.54
[ INFO] [1513815819.814370836]: x: 8.48, y: 5.54
[ INFO] [1513815819.814451532]: x: 8.49, y: 5.54
[ INFO] [1513815819.814554167]: x: 8.50, y: 5.54
[ INFO] [1513815819.913876457]: x: 8.50, y: 5.54
[ INFO] [1513815819.913976595]: x: 8.51, y: 5.54
[ INFO] [1513815819.914073142]: x: 8.52, y: 5.54
[ INFO] [1513815819.914156196]: x: 8.53, y: 5.54
[ INFO] [1513815819.914213115]: x: 8.54, y: 5.54
[ INFO] [1513815819.914256452]: x: 8.54, y: 5.54
[ INFO] [1513815820.013894704]: x: 8.55, y: 5.54
[ INFO] [1513815820.014022742]: x: 8.56, y: 5.54
[ INFO] [1513815820.014076615]: x: 8.57, y: 5.54
[ INFO] [1513815820.014161142]: x: 8.58, y: 5.54
[ INFO] [1513815820.014213272]: x: 8.58, y: 5.54
[ INFO] [1513815820.014307149]: x: 8.59, y: 5.54
[ INFO] [1513815820.113908170]: x: 8.60, y: 5.54
[ INFO] [1513815820.114021427]: x: 8.61, y: 5.54
[ INFO] [1513815820.114088961]: x: 8.62, y: 5.54
[ INFO] [1513815820.114167756]: x: 8.62, y: 5.54
[ INFO] [1513815820.114257723]: x: 8.63, y: 5.54
[ INFO] [1513815820.114328441]: x: 8.64, y: 5.54
```

```
        pub.publish(msg);
        ros::spinOnce(); // Allow processing of incomin
        rate.sleep();
    }
```

Ricevere comandi da tastiera

Per poter guidare il robot da tastiera abbiamo bisogno di creare

1. un subscriber per i comandi per la teleoperazione

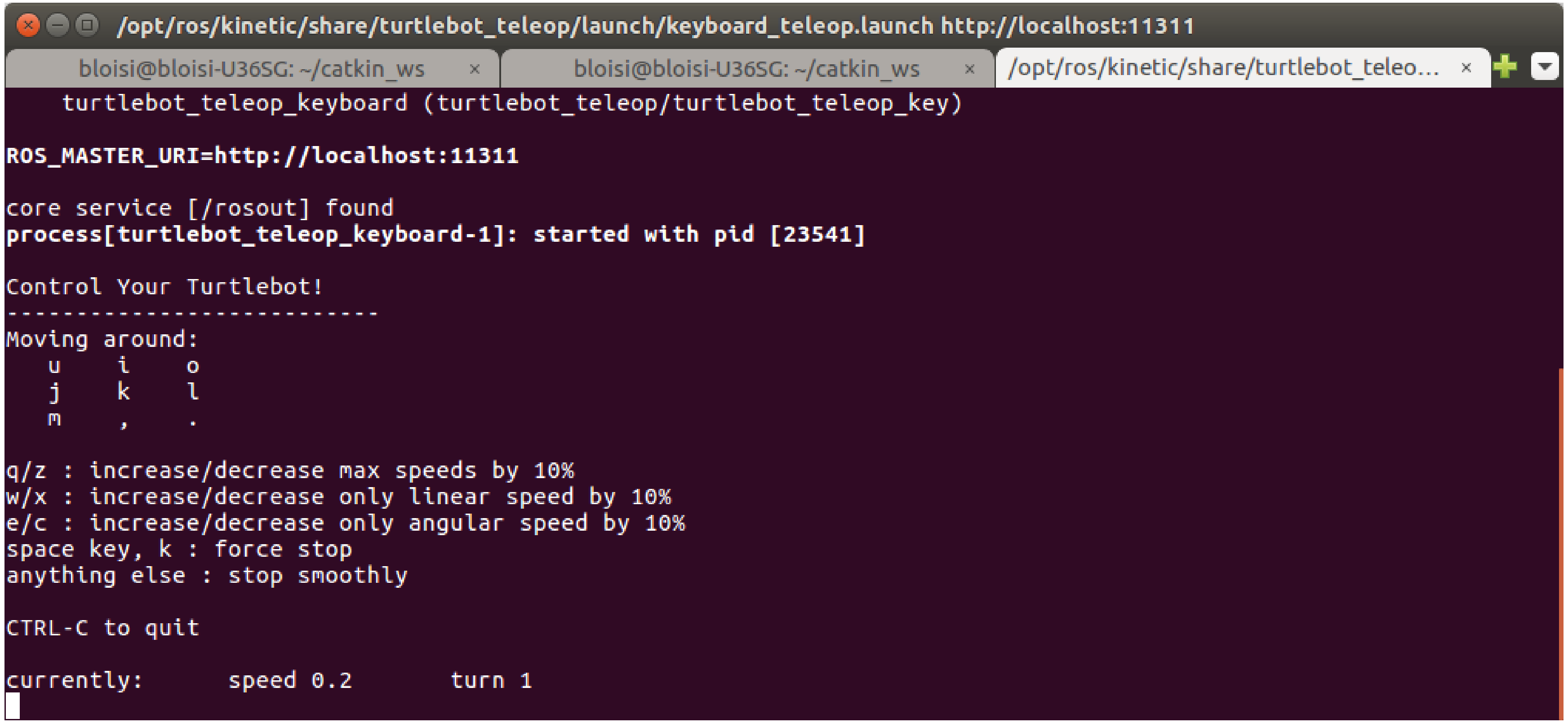
2. un publisher per comunicare al robot come intendiamo trasformare i comandi provenienti dalla tastiera in comandi di velocità

comandi da tastiera

Lanciamo in un terminal il comando

$ roslaunch turtlebot_teleop keyboard_teleop.launch

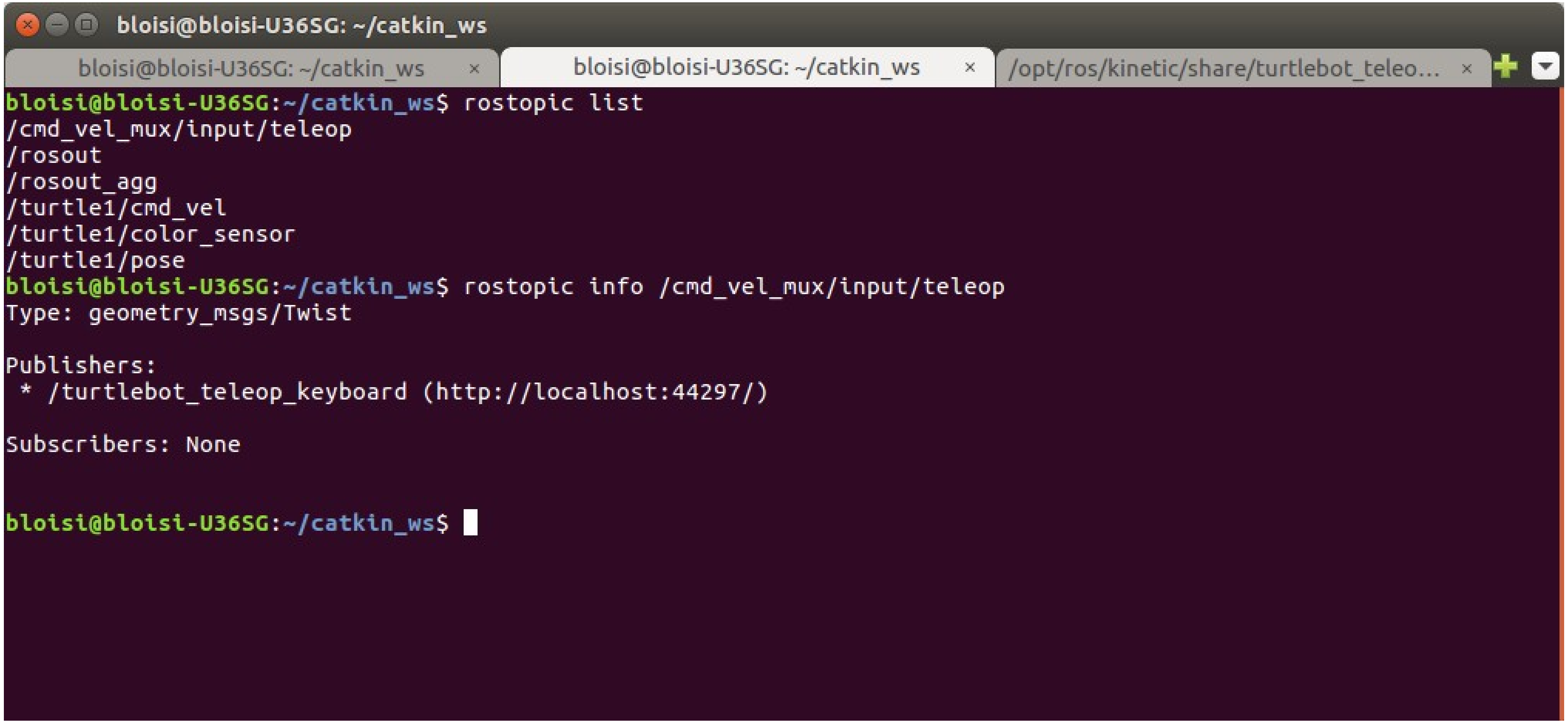
per poter acquisire i comandi per la teleoperazione (da tastiera)



/opt/ros/kinetic/share/turtlebot_teleop/launch/keyboard_teleop.launch http://localhost:11311
bloisi@bloisi-U36SG: ~/catkin_ws
bloisi@bloisi-U36SG: ~/catkin_ws
/opt/ros/kinetic/share/turtlebot_teleo...

```
    turtlebot_teleop_keyboard (turtlebot_teleop/turtlebot_teleop_key)

ROS_MASTER_URI=http://localhost:11311

core service [/rosout] found
process[turtlebot_teleop_keyboard-1]: started with pid [23541]

Control Your Turtlebot!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
space key, k : force stop
anything else : stop smoothly

CTRL-C to quit

currently:      speed 0.2       turn 1
```

/cmd_vel_mux/input/teleop



bloisi@bloisi-U36SG: ~/catkin_ws
bloisi@bloisi-U36SG: ~/catkin_ws
bloisi@bloisi-U36SG: ~/catkin_ws
/opt/ros/kinetic/share/turtlebot_teleo...

```
bloisi@bloisi-U36SG:~/catkin_ws$ rostopic list
/cmd_vel_mux/input/teleop
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
bloisi@bloisi-U36SG:~/catkin_ws$ rostopic info /cmd_vel_mux/input/teleop
Type: geometry_msgs/Twist

Publishers:
 * /turtlebot_teleop_keyboard (http://localhost:44297/)

Subscribers: None


bloisi@bloisi-U36SG:~/catkin_ws$ 
```

Gestire la teleoperazione

Per poter gestire la teleoperazione abbiamo bisogno di creare un subscriber al topic cmd_vel_mux/input/teleop

```
ros::Subscriber velocity_sub =
    node.subscribe("cmd_vel_mux/input/teleop", 1, velocityCallback);
```

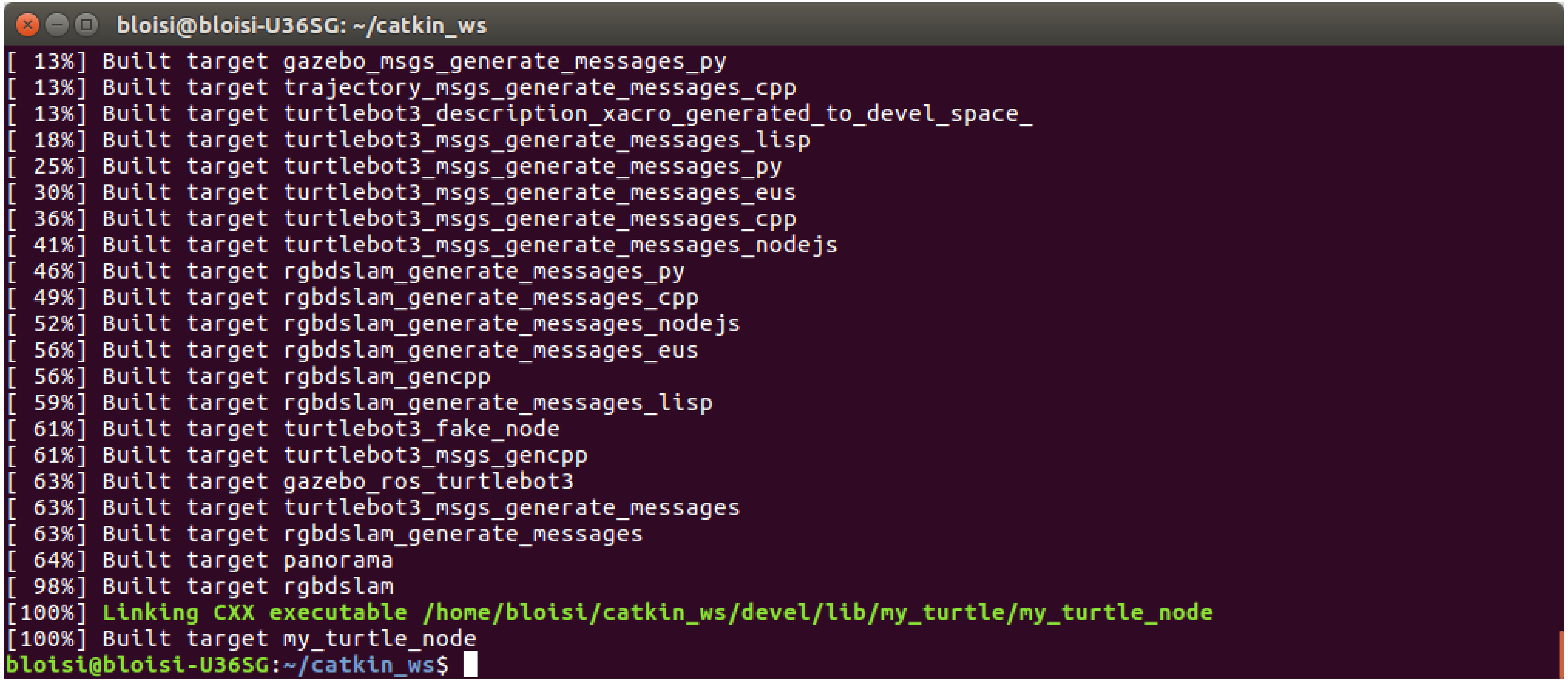
Va creata anche una opportuna callback che gestisca il messaggio

```
void velocityCallback(const geometry_msgs::Twist::ConstPtr& vel)
{
  lin_vel_ = vel->linear.x;
  ang_vel_ = vel->angular.z;
}
```

Guidare il robot

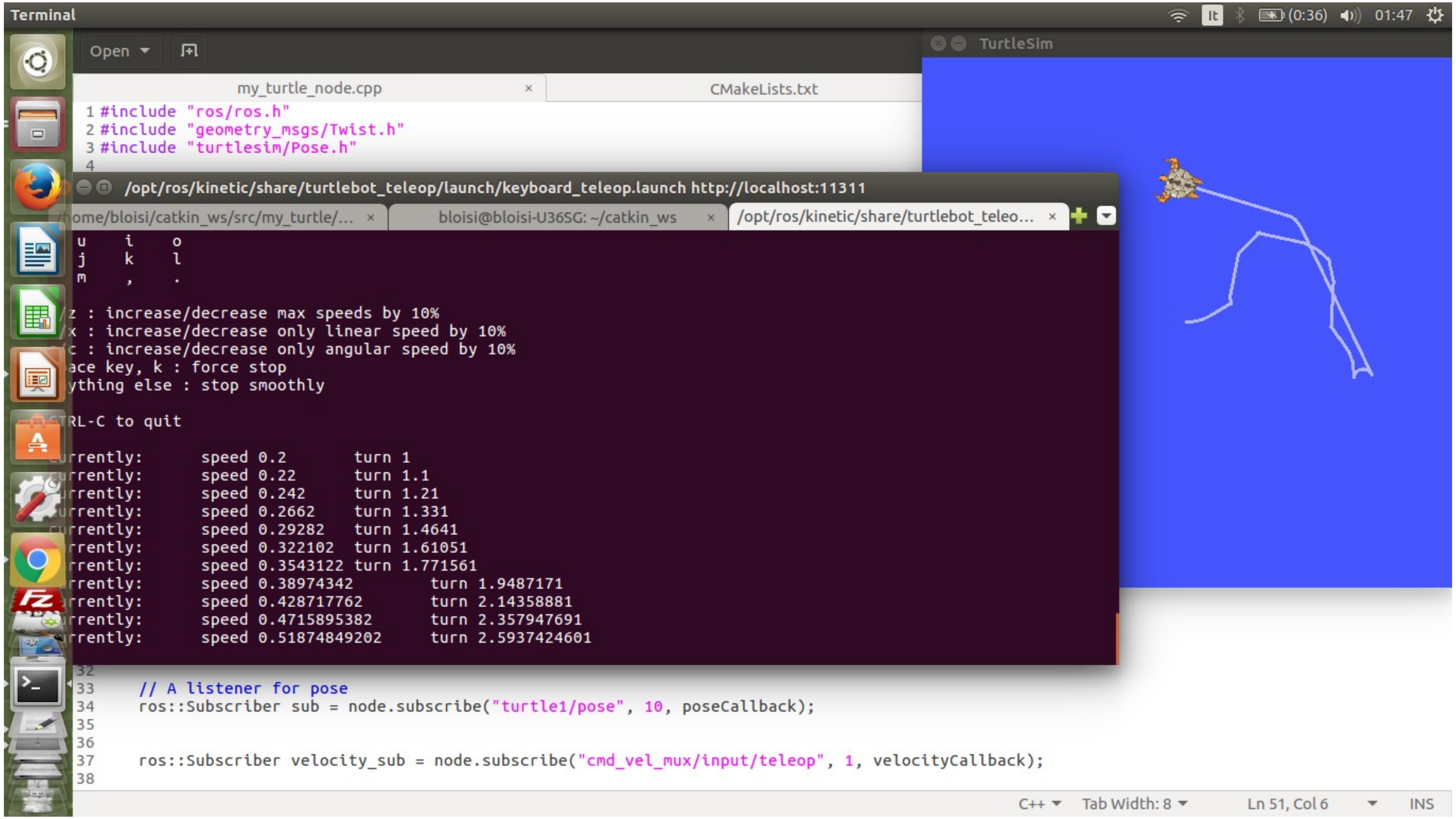
```
ros::NodeHandle node;

ros::Publisher pub = node.advertise<geometry_msgs::Twist>("turtle1/cmd_vel", 10);

ros::Subscriber velocity_sub = node.subscribe("cmd_vel_mux/input/teleop", 1, velocityCallback);

// Loop at 10Hz, publishing movement commands until we shut down
ros::Rate rate(10);
ROS_INFO("Starting to move forward");
while (ros::ok()) {
    geometry_msgs::Twist msg;
    msg.linear.x = lin_vel_;
    msg.angular.z = ang_vel_;
    pub.publish(msg);
    ros::spinOnce(); // Allow processing of incoming messages
    rate.sleep();
}
```

catkin_make



```
[ 13%] Built target gazebo_msgs_generate_messages_py
[ 13%] Built target trajectory_msgs_generate_messages_cpp
[ 13%] Built target turtlebot3_description_xacro_generated_to_devel_space_
[ 18%] Built target turtlebot3_msgs_generate_messages_lisp
[ 25%] Built target turtlebot3_msgs_generate_messages_py
[ 30%] Built target turtlebot3_msgs_generate_messages_eus
[ 36%] Built target turtlebot3_msgs_generate_messages_cpp
[ 41%] Built target turtlebot3_msgs_generate_messages_nodejs
[ 46%] Built target rgbdslam_generate_messages_py
[ 49%] Built target rgbdslam_generate_messages_cpp
[ 52%] Built target rgbdslam_generate_messages_nodejs
[ 56%] Built target rgbdslam_generate_messages_eus
[ 56%] Built target rgbdslam_gencpp
[ 59%] Built target rgbdslam_generate_messages_lisp
[ 61%] Built target turtlebot3_fake_node
[ 61%] Built target turtlebot3_msgs_gencpp
[ 63%] Built target gazebo_ros_turtlebot3
[ 63%] Built target turtlebot3_msgs_generate_messages
[ 63%] Built target rgbdslam_generate_messages
[ 64%] Built target panorama
[ 98%] Built target rgbdslam
[100%] Linking CXX executable /home/bloisi/catkin_ws/devel/lib/my_turtle/my_turtle_node
[100%] Built target my_turtle_node
bloisi@bloisi-U36SG:~/catkin_ws$ 
```

Esercizio


Terminal
It
(0:36)
01:47
Open
TurtleSim
my_turtle_node.cpp
CMakeLists.txt
/opt/ros/kinetic/share/turtlebot_teleop/launch/keyboard_teleop.launch http://localhost:11311
ome/bloisi/catkin_ws/src/my_turtle/...
bloisi@bloisi-U36SG: ~/catkin_ws
/opt/ros/kinetic/share/turtlebot_teleo...
C++
Tab Width: 8
Ln 51, Col 6
INS

```
#include "ros/ros.h"
#include "geometry_msgs/Twist.h"
#include "turtlesim/Pose.h"
```

```
u    i    o
j    k    l
m    ,    .

z : increase/decrease max speeds by 10%
x : increase/decrease only linear speed by 10%
c : increase/decrease only angular speed by 10%
ace key, k : force stop
ything else : stop smoothly

RL-C to quit

rrently:      speed 0.2       turn 1
rrently:      speed 0.22      turn 1.1
rrently:      speed 0.242     turn 1.21
rrently:      speed 0.2662    turn 1.331
rrently:      speed 0.29282   turn 1.4641
rrently:      speed 0.322102  turn 1.61051
rrently:      speed 0.3543122 turn 1.771561
rrently:      speed 0.38974342        turn 1.9487171
rrently:      speed 0.428717762       turn 2.14358881
rrently:      speed 0.4715895382      turn 2.357947691
rrently:      speed 0.51874849202     turn 2.5937424601
```

```

    // A listener for pose
    ros::Subscriber sub = node.subscribe("turtle1/pose", 10, poseCallback);


    ros::Subscriber velocity_sub = node.subscribe("cmd_vel_mux/input/teleop", 1, velocityCallback);

```

References and Credits

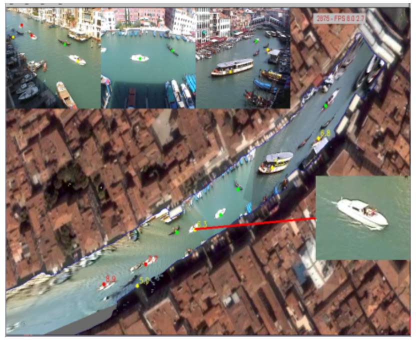
Alcune slide e parte del codice contenuto in questa presentazione sono stati adattati da

https://www.ldv.ei.tum.de/fileadmin/w00bfa/www/Vorlesungen/cpp/leistungskurs/ws1617/turtlesim.pdf

UNIVERSITÀ di VERONA | Dipartimento di INFORMATICA

Laurea magistrale in ingegneria e scienze informatiche

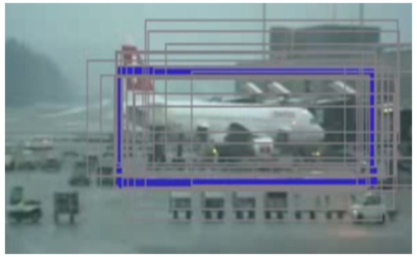
Corso di Robotica
Parte di Laboratorio

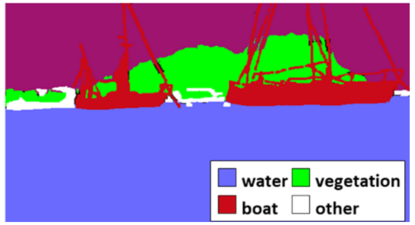


Esempio di applicazione

Docente:

Domenico Daniele Bloisi





Dicembre 2017