UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

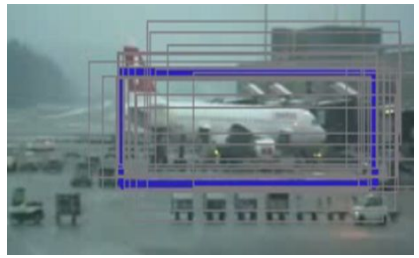


Corso di Visione e Percezione

A.A. 2019/2020

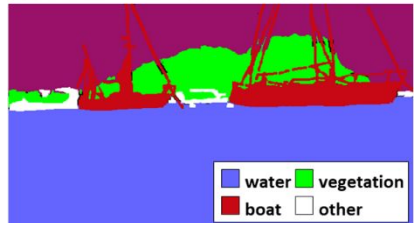
Docente

Domenico Daniele Bloisi

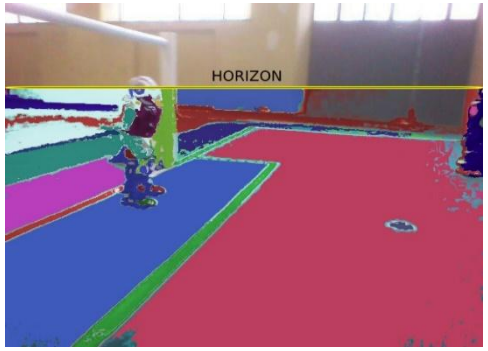



ROS launch file





Maggio 2020

Il corso

- Home page del corso
  http://web.unibas.it/bloisi/corsi/visione-e-percezione.html
- Docente: Domenico Daniele Bloisi
- Periodo: II semestre marzo 2020 – giugno 2020

roslaunch

roslaunch è un tool per semplificare

- il lancio di più nodi ROS
- il settaggio dei parametri

roslaunch utilizza i cosiddetti “launch file” che sono file XML contenenti la lista dei nodi da lanciare con i rispettivi parametri

roslaunch - sintassi

```
roslaunch <package> <launch file>
```

- i launch file hanno per convenzione un nome che termina con .launch

- `roscore` viene automaticamente lanciato quando si esegue `roslaunch`

Esempio launch file

```
<launch>
  <node name="talker" pkg="hello_ros" type="talker.py" output="screen"/>
  <node name="listener" pkg="hello_ros" type="listener.py" output="screen"/>
</launch>
```

Il tag <node> contiene gli attributi per l'esecuzione del nodo

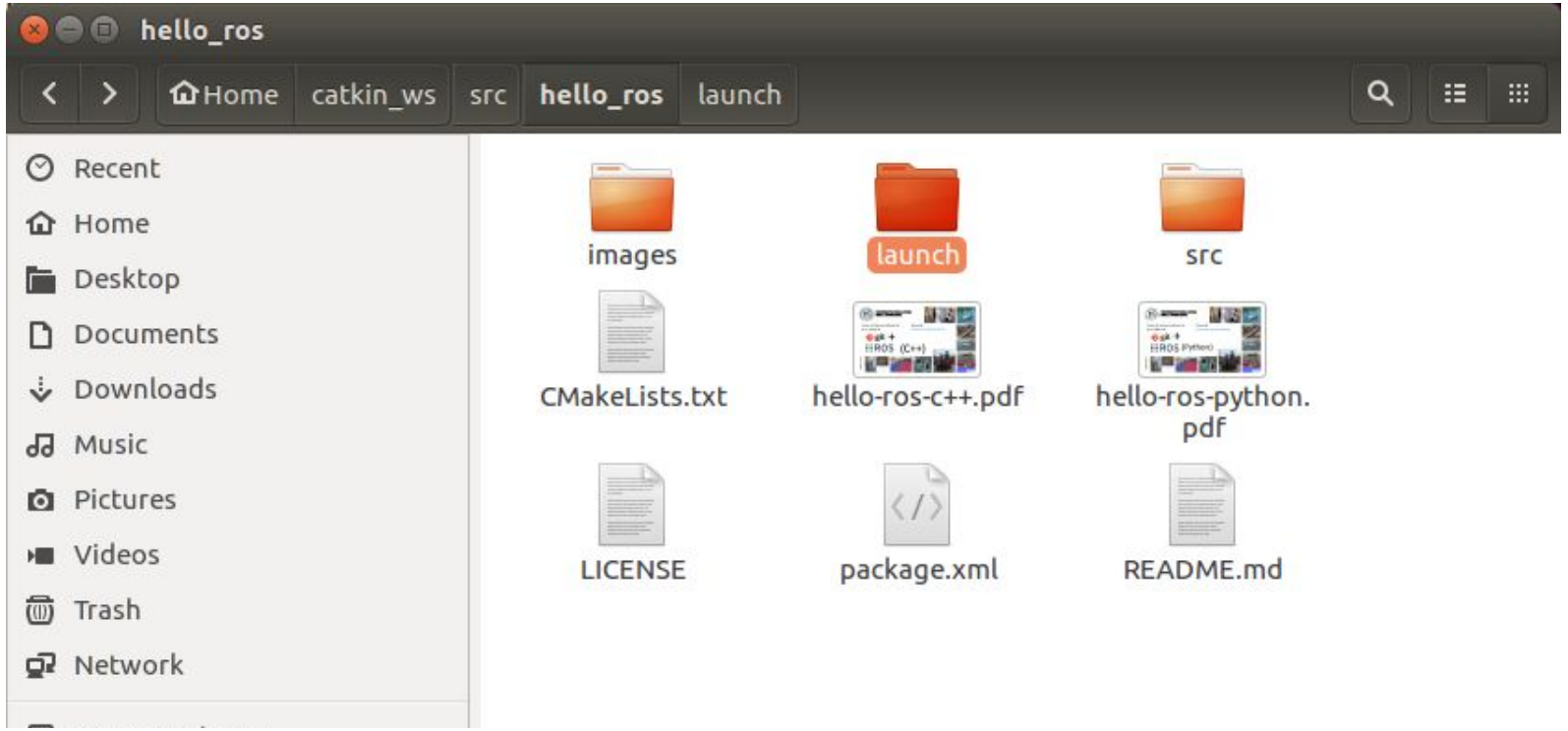
- name è il nome con cui il nodo verrà inserito nel grafo di ROS
- pkg indica il package nel quale può essere trovato il nodo
- type specifica il filename dell'eseguibile
- output posto a "screen" indica che i messaggi di log di ROS verranno mostrati sul terminale su cui verrà eseguito il comando `roslaunch`

hello_ros: git repo recap

```
bloisi@bloisi-U36SG:~/catkin_ws/src$ git clone https://github.com/dbloisi/hello_ros.git
Cloning into 'hello_ros'...
remote: Enumerating objects: 26, done.
remote: Counting objects: 100% (26/26), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 74 (delta 13), reused 0 (delta 0), pack-reused 48
Unpacking objects: 100% (74/74), done.
Checking connectivity... done.
bloisi@bloisi-U36SG:~/catkin_ws/src$ cd hello_ros
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$ ls
CMakeLists.txt     hello-ros-python.pdf  LICENSE      README.md
hello-ros-c++.pdf  images                package.xml  src
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$
```

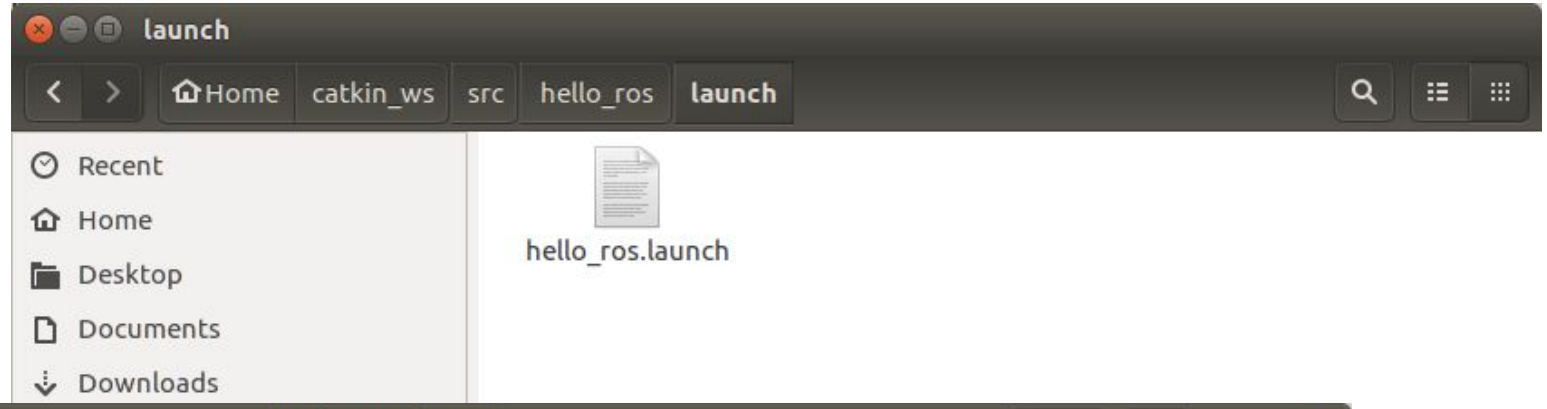
hello_ros launch file

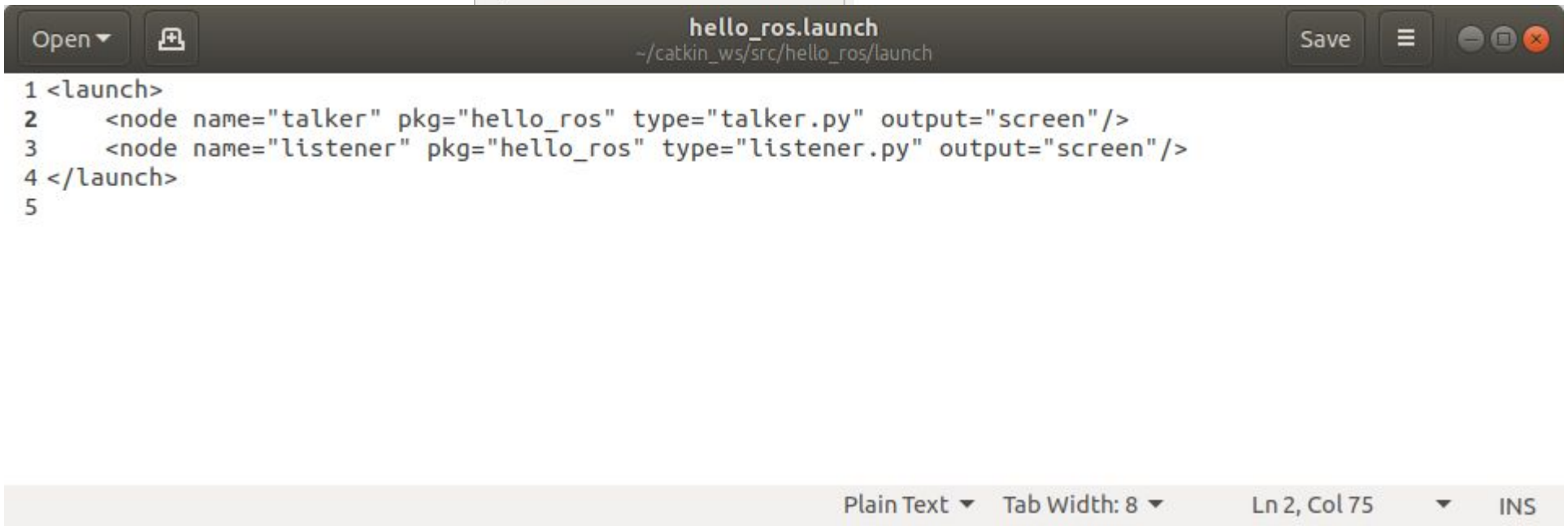
Creiamo una cartella launch



hello_ros launch file

Creiamo dentro la cartella `launch` il file `hello_ros.launch`





hello_ros.launch

esecuzione

```
bloisi@bloisi-U36SG: ~
File  Edit  View  Search  Terminal  Help
bloisi@bloisi-U36SG:~$ roslaunch hello_ros hello_ros.launch
... logging to /home/bloisi/.ros/log/ab261a1a-8de4-11ea-8461-50465dde6884/roslau
nch-bloisi-U36SG-4776.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:44199/

SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.5

NODES
  /
    listener (hello_ros/listener.py)
    talker (hello_ros/talker.py)

auto-starting new master
process[master]: started with pid [4786]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to ab261a1a-8de4-11ea-8461-50465dde6884
process[rosout-1]: started with pid [4797]
started core service [/rosout]
process[talker-2]: started with pid [4801]
process[listener-3]: started with pid [4805]
[INFO] [1588582386.444483]: hello world 1588582386.44
[INFO] [1588582386.545189]: hello world 1588582386.54
[INFO] [1588582386.549732]: /listenerI heard hello world 1588582386.54
[INFO] [1588582386.645215]: hello world 1588582386.64
[INFO] [1588582386.649691]: /listenerI heard hello world 1588582386.64
[INFO] [1588582386.745221]: hello world 1588582386.74
[INFO] [1588582386.750140]: /listenerI heard hello world 1588582386.74
```

hello_ros launch file: git

```
bloisi@bloisi-U36SG: ~/catkin_ws/src/hello_ros
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$ ls
CMakeLists.txt     hello-ros-python.pdf  launch   package.xml  src
hello-ros-c++.pdf  images                LICENSE  README.md
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$ git add launch
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$ git commit -m "adding launch fold
er and launch file"
[master eeefdd0] adding launch folder and launch file
 1 file changed, 5 insertions(+)
 create mode 100644 launch/hello_ros.launch
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$ git pull
Already up-to-date.
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$ git push origin master
Username for 'https://github.com': dbloisi
Password for 'https://dbloisi@github.com':
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 519 bytes | 0 bytes/s, done.
Total 5 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/dbloisi/hello_ros.git
   03675ef..eeefdd0  master -> master
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros$
```

Package unibas_turtle

Per creare il package digitiamo

```
cd ~/catkin_ws/src
catkin_create_pkg unibas_turtle geometry_msgs rospy
```

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

http://wiki.ros.org/turtlesim/Tutorials/Moving%20in%20a%20Straight%20Line

Package unibas_turtle: package.xml

Open ▾ *package.xml ~/catkin_ws/src/unibas_turtle Save

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

  <!-- One maintainer tag required, multiple allowed, one person per tag -->
  <!-- Example:  -->
  <!-- <maintainer email="jane.doe@example.com">Jane Doe</maintainer> -->
  <maintainer email="domenico.bloisi@gmail.com">domenico bloisi</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>GPLv3</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/unibas_turtle</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> -->

```

XML ▾ Tab Width: 8 ▾ Ln 16, Col 17 INS

Package unibas_turtle: catkin_make

Compiliamo con catkin_make

```
cd ~/catkin_ws
catkin_make
```

```
bloisi@bloisi-U36SG: ~/catkin_ws
bloisi@bloisi-U36SG:~/catkin_ws/src/hello_ros/src$ cd ~/catkin_ws/
bloisi@bloisi-U36SG:~/catkin_ws$ catkin_make
Base path: /home/bloisi/catkin_ws
Source space: /home/bloisi/catkin_ws/src
Build space: /home/bloisi/catkin_ws/build
Devel space: /home/bloisi/catkin_ws/devel
Install space: /home/bloisi/catkin_ws/install
####
#### Running command: "make cmake_check_build_system" in "/home/bloisi/catkin_ws/build"
####
-- Using CATKIN_DEVEL_PREFIX: /home/bloisi/catkin_ws/devel
-- Using CMAKE_PREFIX_PATH: /home/bloisi/catkin_ws/devel;/opt/ros/kinetic
-- This workspace overlays: /home/bloisi/catkin_ws/devel;/opt/ros/kinetic
-- Using PYTHON_EXECUTABLE: /usr/bin/python
-- Using Debian Python package layout
-- Using empy: /usr/bin/empy
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/bloisi/catkin_ws/build/test_results
-- Found gmock sources under '/usr/src/gmock': gmock will be built
-- Found gtest sources under '/usr/src/gmock': gtests will be built
-- Using Python nosetests: /usr/bin/nosetests-2.7
-- catkin 0.7.14
```

http://wiki.ros.org/turtlesim/Tutorials/Moving%20in%20a%20Straight%20Line

Package unibas_turtle: src

Creiamo una cartella source che conterrà il codice sorgente

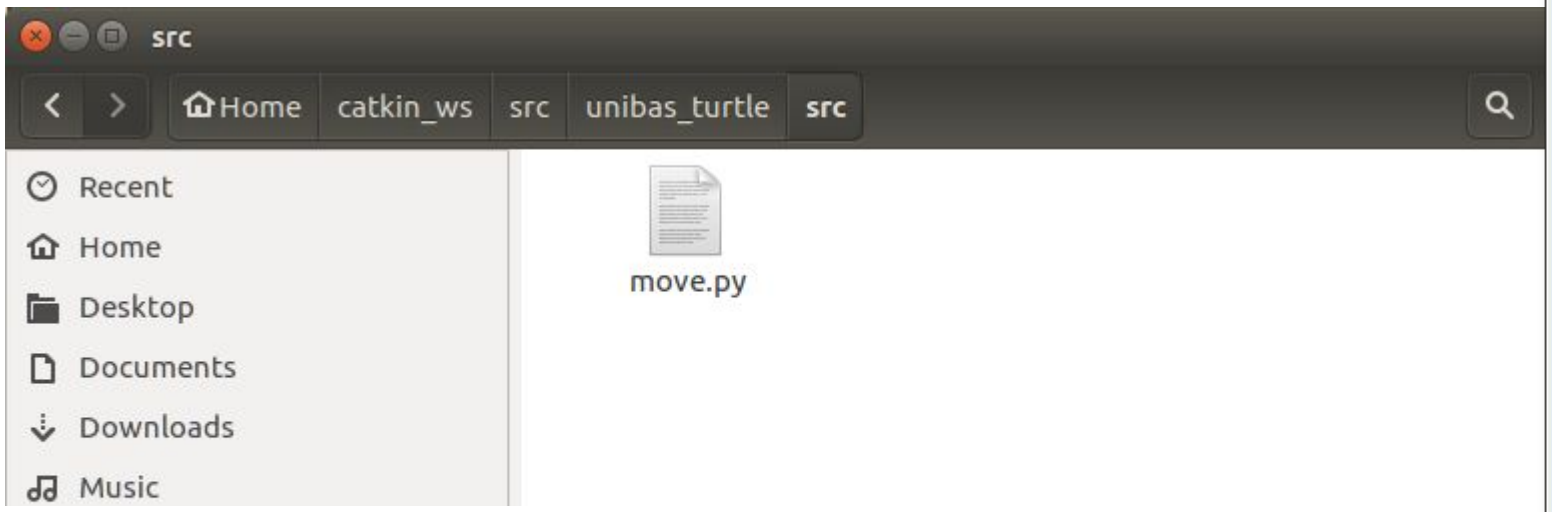
```
bloisi@bloisi-U36SG: ~/catkin_ws/src/unibas_turtle/src
bloisi@bloisi-U36SG:~/catkin_ws$ cd ~/catkin_ws/src/unibas_turtle
bloisi@bloisi-U36SG:~/catkin_ws/src/unibas_turtle$ ls
CMakeLists.txt  package.xml  src
bloisi@bloisi-U36SG:~/catkin_ws/src/unibas_turtle$ cd src
bloisi@bloisi-U36SG:~/catkin_ws/src/unibas_turtle/src$ ls
bloisi@bloisi-U36SG:~/catkin_ws/src/unibas_turtle/src$
```

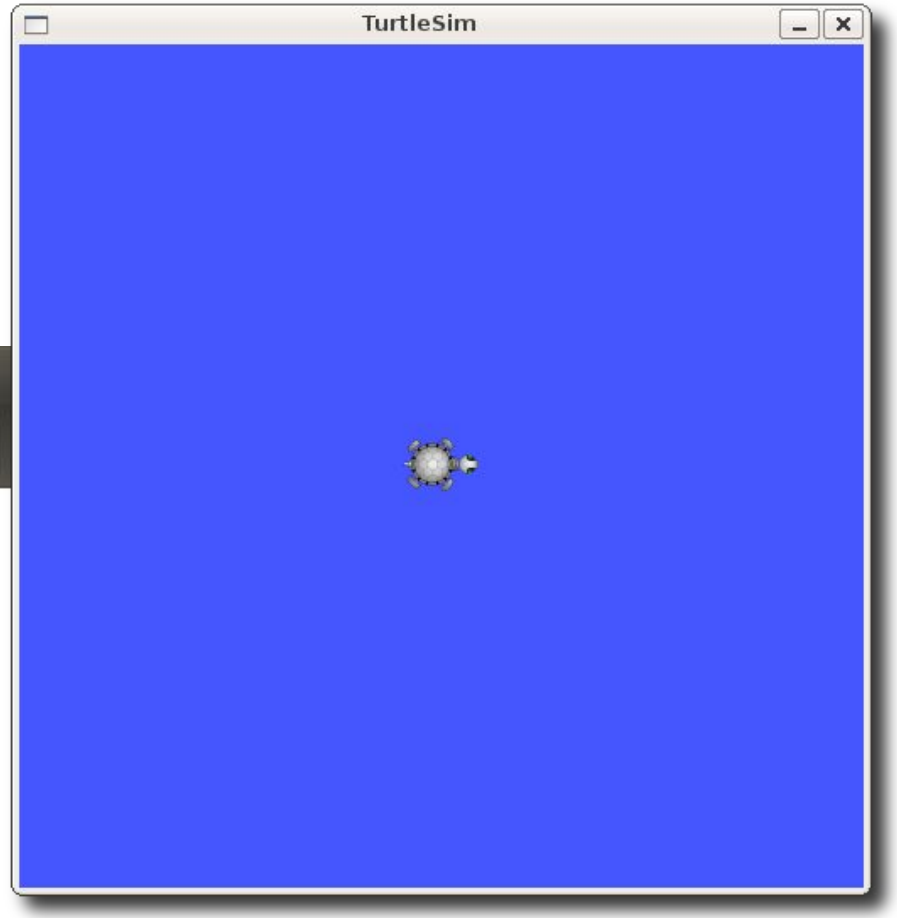
http://wiki.ros.org/turtlesim/Tutorials/Moving%20in%20a%20Straight%20Line

Package unibas_turtle: src

Creiamo un file `move.py` per far muovere

la tartaruga di `turtlesim`





http://wiki.ros.org/turtlesim/Tutorials/Moving%20in%20a%20Straight%20Line
http://wiki.ros.org/turtlesim

idea

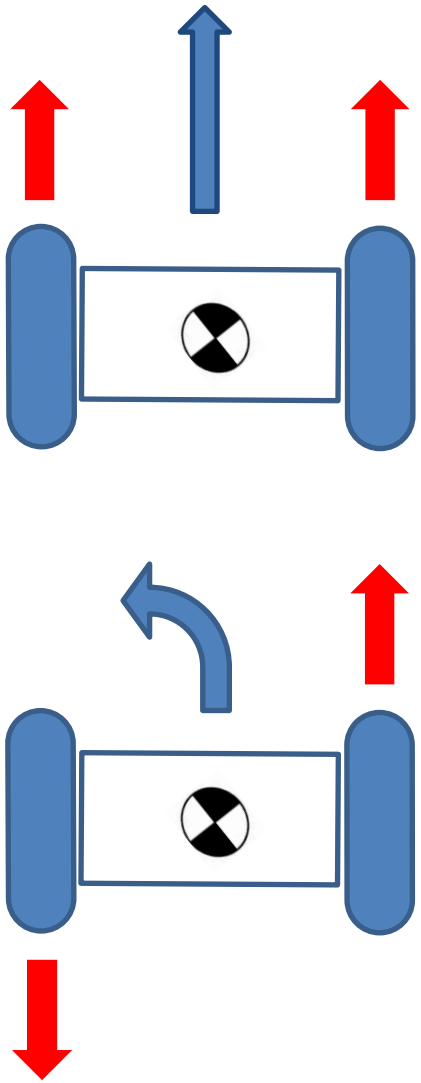
- Vogliamo far muovere la tartaruga controllandone la velocità

- Adottiamo per la tartaruga il modello di un robot differenziale

- Modifichiamo i valori di velocità lineare e angolare per controllare il moto

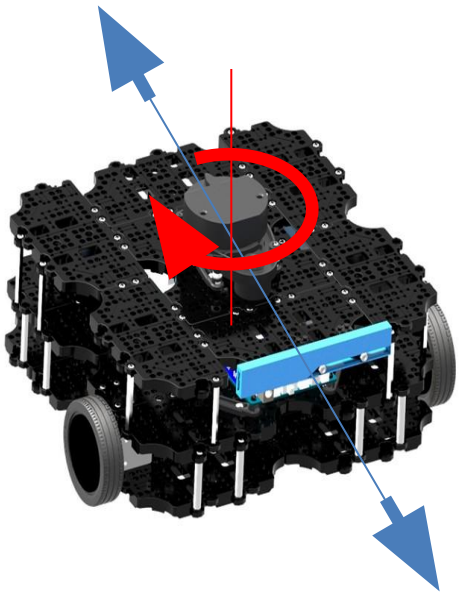
Differential drive robot

- Un robot differenziale su ruote è una base mobile avente due ruote motorizzate indipendenti
- Le ruote sono posizionate ai due lati opposti della scocca
- Il robot si muove in avanti quando entrambe le ruote gira in avanti, mentre gira sul posto quando una ruota gira in avanti e l'altra gira all'indietro



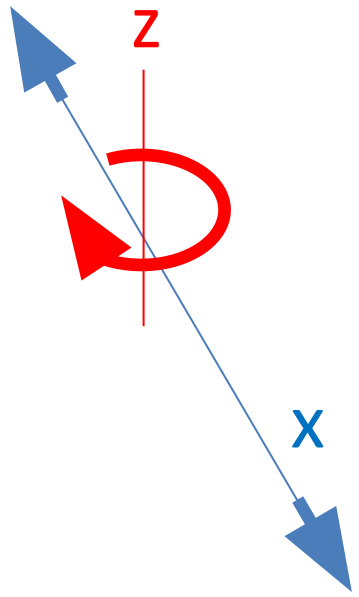
Movimento di un robot differenziale

Data la sua configurazione, un robot differenziale può muoversi solo in avanti o indietro lungo il suo asse longitudinale e può ruotare solo lungo il suo asse verticale



Movimento di un robot differenziale

- Il robot non potrà muoversi di lato o verticalmente
- Per tali motivi ci bastano la componente lineare x e la componente angolare z per controllare il movimento
- Nel caso di un robot omnidirezionale, avremo anche una componente y per lo spostamento laterale
- Quante componenti avremo per un robot underwater?



Comandi di velocità in ROS

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
```

move.py

```
#!/usr/bin/env python
import rospy
from geometry_msgs.msg import Twist

def move():
    # Starts a new node
    rospy.init_node('move', anonymous=True)
    velocity_publisher = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)
    vel_msg = Twist()
```

move.py

```
#Receiveing the user's input
print("Let's move your robot")
speed = input("Input your speed:")
distance = input("Type your distance:")
isForward = input("Foward?: ")#True or False

#Checking if the movement is forward or backwards
if(isForward):
    vel_msg.linear.x = abs(speed)
else:
    vel_msg.linear.x = -abs(speed)

#Since we are moving just in x-axis
vel_msg.linear.y = 0
vel_msg.linear.z = 0
vel_msg.angular.x = 0
vel_msg.angular.y = 0
vel_msg.angular.z = 0
```

move.py

```
while not rospy.is_shutdown():
    #Setting the current time for distance calculus
    t0 = rospy.Time.now().to_sec()
    current_distance = 0

    #Loop to move the turtle in an specified distance
    while(current_distance < distance):
        #Publish the velocity
        velocity_publisher.publish(vel_msg)
        #Takes actual time to velocity calculus
        t1=rospy.Time.now().to_sec()
        #Calculates distancePoseStamped
        current_distance= speed*(t1-t0)
    #After the loop, stops the robot
    vel_msg.linear.x = 0
    #Force the robot to stop
    velocity_publisher.publish(vel_msg)
```

move.py

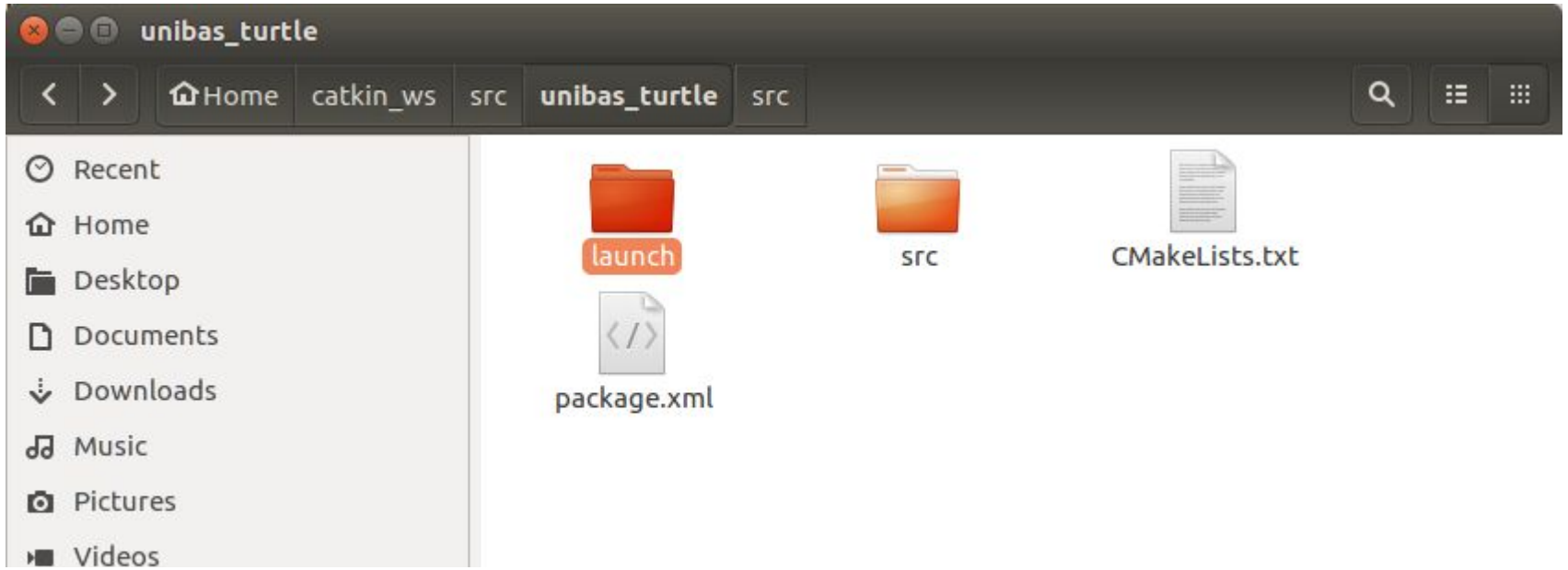
```
if __name__ == '__main__':
    try:
        #Testing our function
        move()
     except rospy.ROSInterruptException: pass
```

permessi per move.py

```
bloisi@bloisi-U36SG: ~/catkin_ws/src/unibas_turtle/src
bloisi@bloisi-U36SG:~/catkin_ws/src/unibas_turtle/src$ ls
move.py
bloisi@bloisi-U36SG:~/catkin_ws/src/unibas_turtle/src$ chmod u+x ~/catkin_ws/src/unibas_turtle/src/move.py
bloisi@bloisi-U36SG:~/catkin_ws/src/unibas_turtle/src$
```

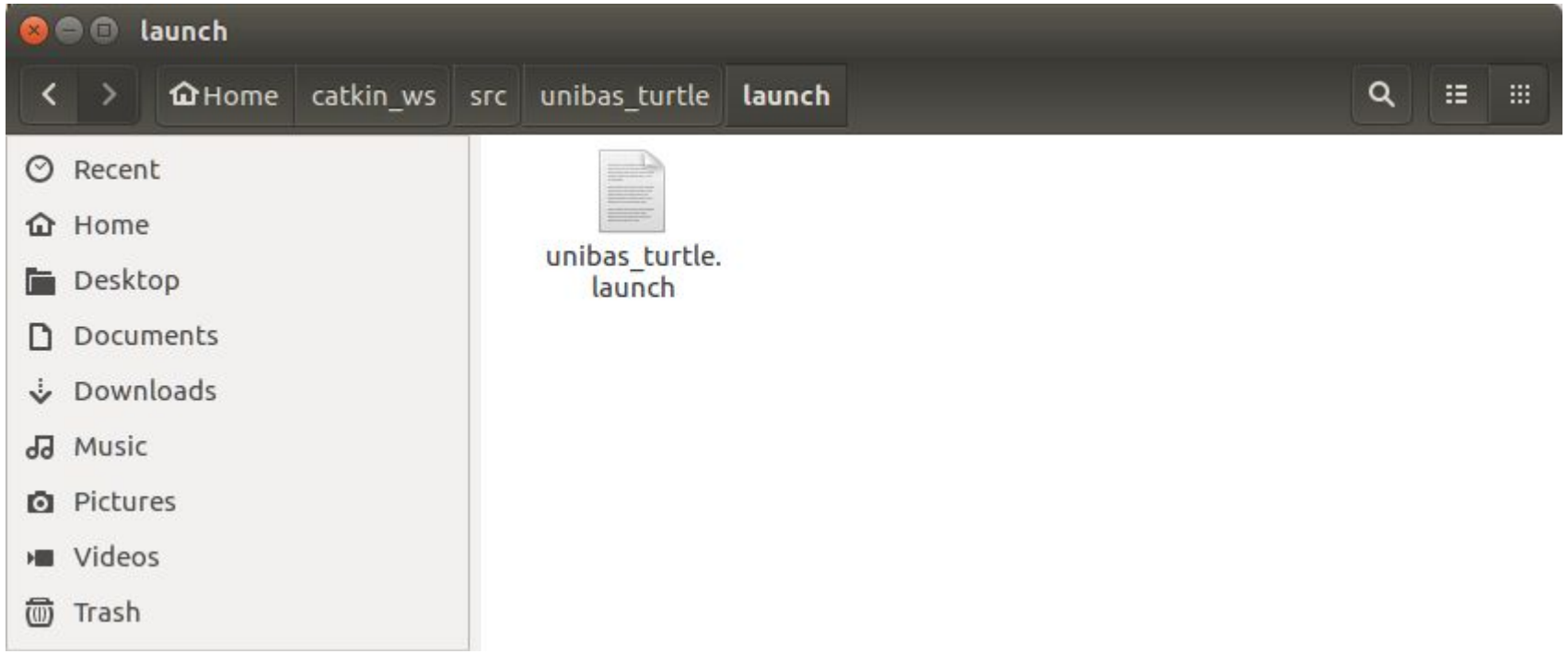
Launch file per unibas_turtle

Creiamo una cartella launch



Launch file per unibas_turtle

Creiamo un file unibas_turtle.launch dentro la cartella launch



unibas_turtle.launch

unibas_turtle.launch (~/catkin_ws/src/unibas_turtle/launch) - gedit

Open

```
<launch>
    <node name="turtlesim_node" pkg="turtlesim" type="turtlesim_node" output="screen"/>
    <node name="move" pkg="unibas_turtle" type="move.py" output="screen"/>
</launch>
```

Esempio roslaunch

```
roslaunch unibas_turtle unibas_turtle.launch
```

ROS package name

launch file name

Esecuzione unibas_turtle.roslaunch

/home/bloisi/catkin_ws/src/unibas_turtle/launch/unibas_turtle.launch http://localhost:11311

File Edit View Search Terminal Help

```
bloisi@bloisi-U36SG:~/catkin_ws$ roslaunch unibas_turtle unibas_turtle.launch
... logging to /home/bloisi/.ros/log/9cdf77f4-8de8-11ea-8461-50465dde6884/ros
42.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:33819/

SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.5

NODES
  /
    move (unibas_turtle/move.py)
    turtlesim_node (turtlesim/turtlesim_node)

auto-starting new master
process[master]: started with pid [5952]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 9cdf77f4-8de8-11ea-8461-50465dde6884
process[rosout-1]: started with pid [5963]
started core service [/rosout]
process[turtlesim_node-2]: started with pid [5967]
process[move-3]: started with pid [5971]
Let's move your robot
Input your speed:
```

TurtleSim

Let's move your robot

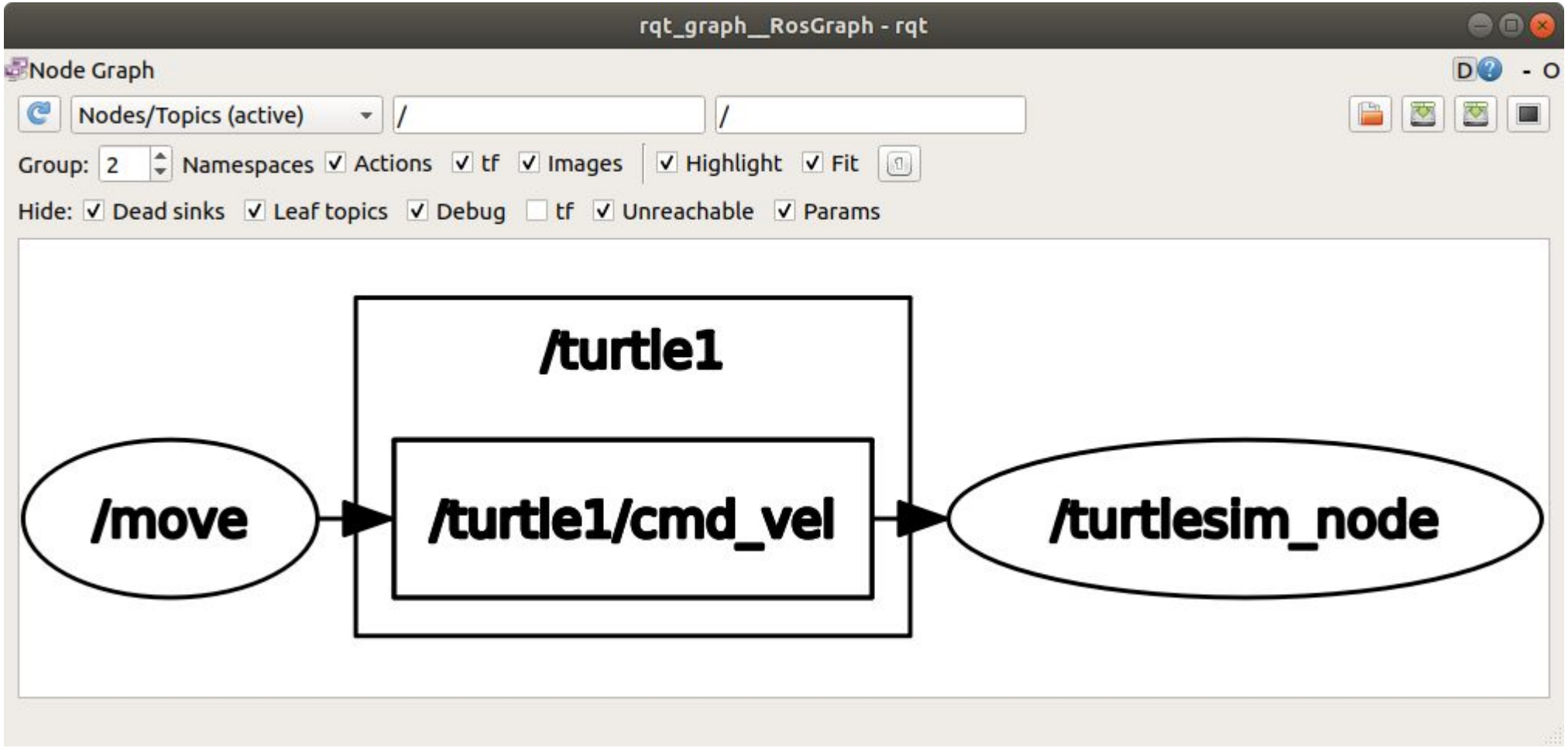
/home/bloisi/catkin_ws/src/unibas_turtle/launch/unibas_turtle.launch http://localhost:11311

File Edit View Search Terminal Help

```
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:33709/

SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.5

NODES
  /
    move (unibas_turtle/move.py)
    turtlesim_node (turtlesim/turtlesim_node)

auto-starting new master
process[master]: started with pid [6065]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to f97acaea-8de8-11ea-8461-50465dde6884
process[rosout-1]: started with pid [6076]
started core service [/rosout]
process[turtlesim_node-2]: started with pid [6082]
process[move-3]: started with pid [6084]
Let's move your robot
Input your speed:3
Type your distance:2
Foward?: 1
```

TurtleSim

rqt_graph


rqt_graph__RosGraph - rqt
Node Graph
Nodes/Topics (active)
/
/
Group: 2
Namespaces
Actions
tf
Images
Highlight
Fit
Hide:
Dead sinks
Leaf topics
Debug
tf
Unreachable
Params
/turtle1
/move
/turtle1/cmd_vel
/turtlesim_node

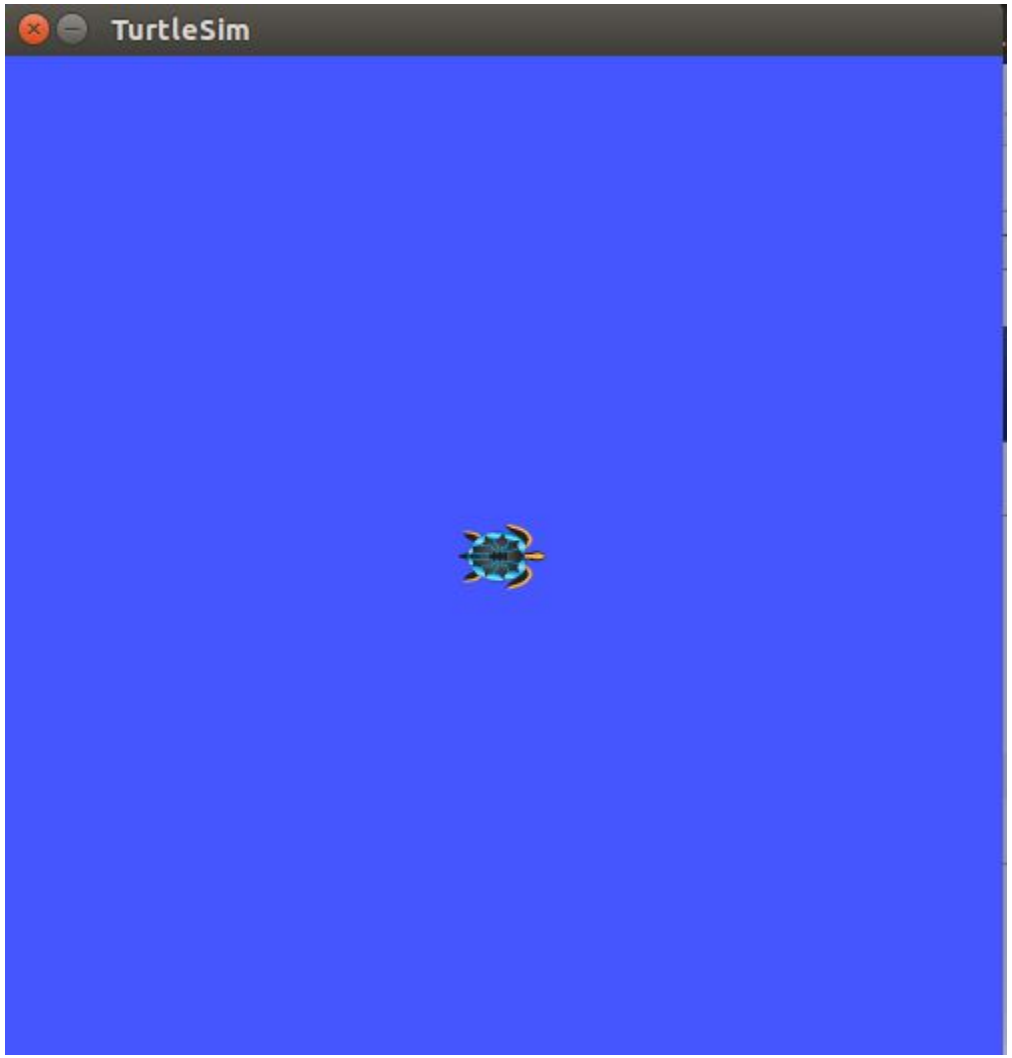
topic e message

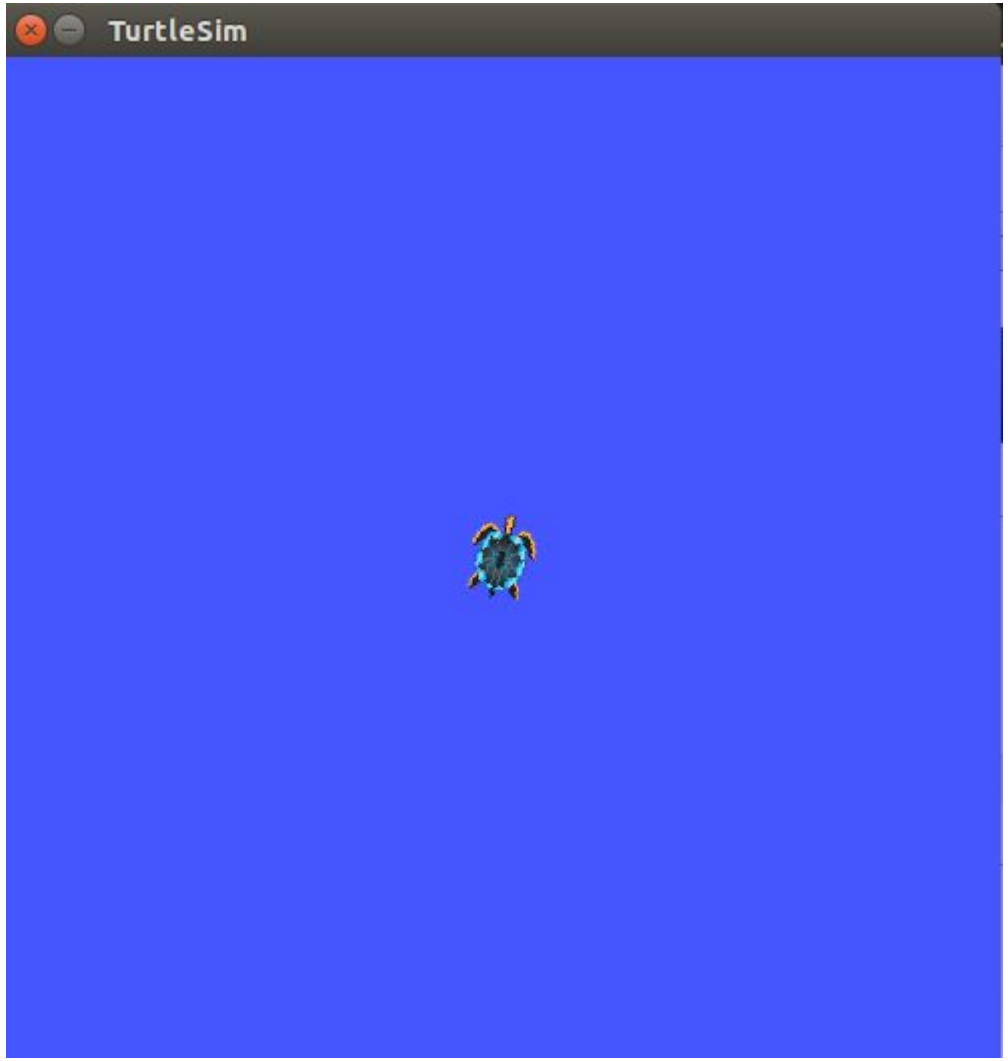
bloisi@bloisi-U36SG: ~

File Edit View Search Terminal Help

```
bloisi@bloisi-U36SG:~$ rostopic info /turtle1/cmd_vel
Type: geometry_msgs/Twist

Publishers:
 * /move (http://localhost:38655/)

Subscribers:
 * /turtlesim_node (http://localhost:42133/)


bloisi@bloisi-U36SG:~$ rosmsg show geometry_msgs/Twist
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z

bloisi@bloisi-U36SG:~$
```

Rotating left and right

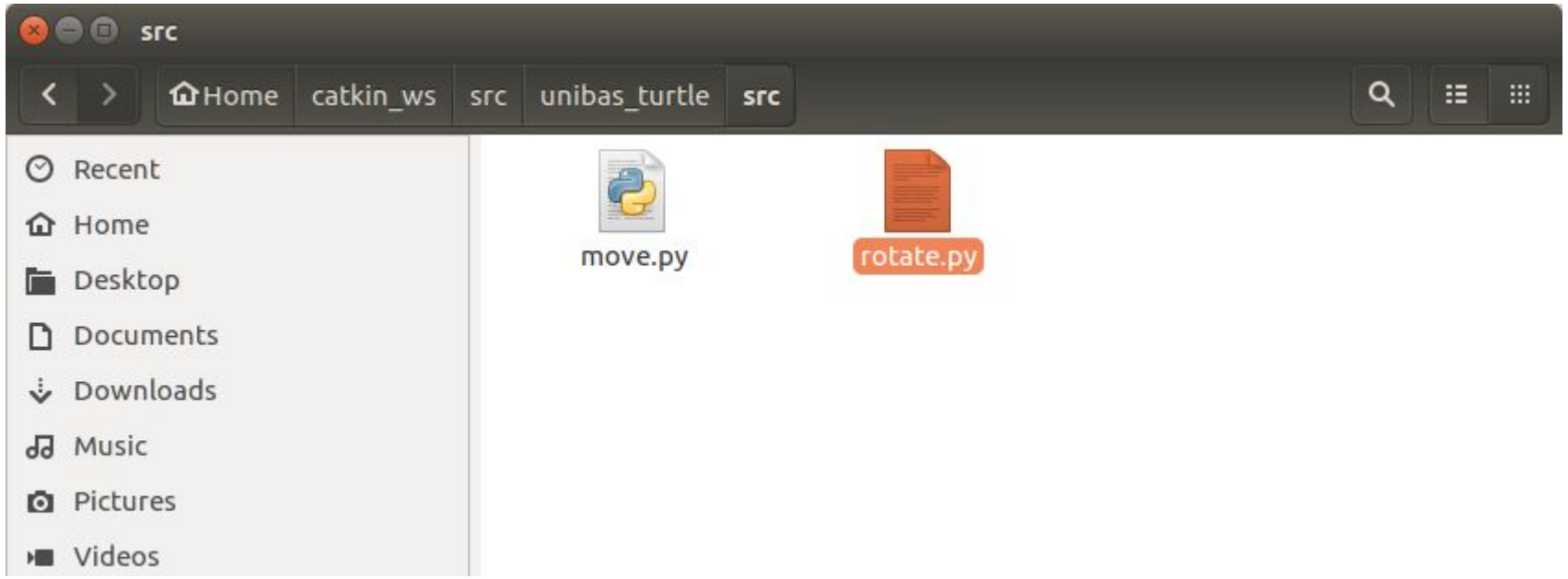




http://wiki.ros.org/turtlesim/Tutorials/Rotating%20Left%20and%20Right

creazione di rotate.py

Creiamo un file rotate.py dentro la cartella src



http://wiki.ros.org/turtlesim/Tutorials/Rotating%20Left%20and%20Right

rotate.py

```
#!/usr/bin/env python
import rospy
from geometry_msgs.msg import Twist
PI = 3.1415926535897

def rotate():
    #Starts a new node
    rospy.init_node('rotate', anonymous=True)
    velocity_publisher = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)
    vel_msg = Twist()
```

rotate.py

```
 # Receiveing the user's input
print("Let's rotate your robot")
speed = input("Input your speed (degrees/sec):")
angle = input("Type your distance (degrees):")
clockwise = input("Clockwise?: ") #True or false

#Converting from angles to radians
angular_speed = speed*2*PI/360
relative_angle = angle*2*PI/360

#We wont use linear components
vel_msg.linear.x=0
vel_msg.linear.y=0
vel_msg.linear.z=0
vel_msg.angular.x = 0
vel_msg.angular.y = 0
```

rotate.py

```
# Checking if our movement is CW or CCW
if clockwise:
    vel_msg.angular.z = -abs(angular_speed)
else:
    vel_msg.angular.z = abs(angular_speed)
# Setting the current time for distance calculus
t0 = rospy.Time.now().to_sec()
current_angle = 0

while(current_angle < relative_angle):
    velocity_publisher.publish(vel_msg)
    t1 = rospy.Time.now().to_sec()
    current_angle = angular_speed*(t1-t0)
```

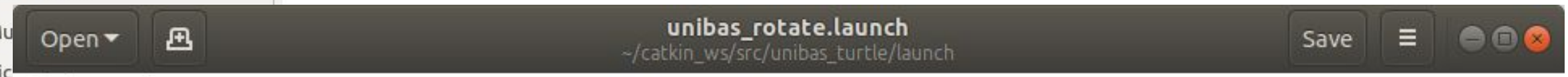
rotate.py

```
    #Forcing our robot to stop
    vel_msg.angular.z = 0
    velocity_publisher.publish(vel_msg)
    rospy.spin()

if __name__ == '__main__':
    try:
        # Testing our function
        rotate()
    except rospy.ROSInterruptException:
         pass
```

permessi per rotate.py

```
bloisi@bloisi-U36SG: ~
bloisi@bloisi-U36SG:~$ chmod u+x ~/catkin_ws/src/unibas_turtle/src/rotate.py
bloisi@bloisi-U36SG:~$
```

launch file per il nodo rotate



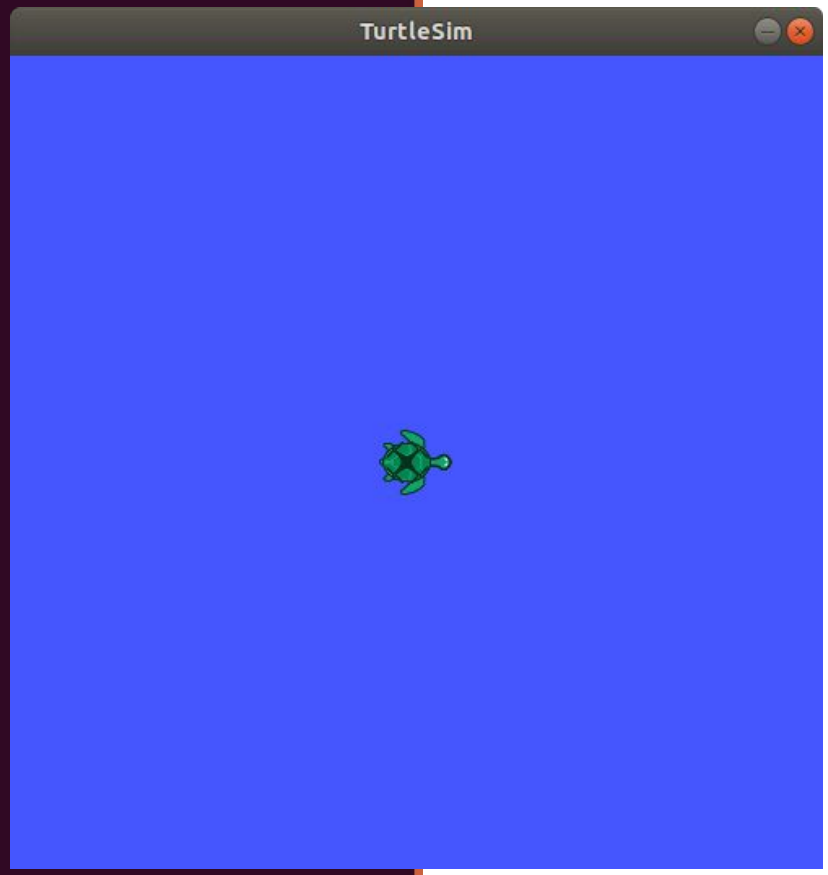

```
<launch>
    <node name="turtlesim_node" pkg="turtlesim" type="turtlesim_node"/>
    <node name="rotate" pkg="unibas_turtle" type="rotate.py" output="screen"/>
</launch>
```

esecuzione per il nodo rotate

/home/bloisi/catkin_ws/src/unibas_turtle/launch/unibas_rotate.launch http://localhost:11311

File Edit View Search Terminal Help

```
bloisi@bloisi-U36SG:~/catkin_ws$ roslaunch unibas_turtle unibas_rotate.launch
... logging to /home/bloisi/.ros/log/7639f460-8dea-11ea-8461-50465dde6884/roslaunch-bloisi-U36SG-70
10.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:37645/

SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.5

NODES
  /
    rotate (unibas_turtle/rotate.py)
    turtlesim_node (turtlesim/turtlesim_node)

auto-starting new master
process[master]: started with pid [7020]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 7639f460-8dea-11ea-8461-50465dde6884
process[rosout-1]: started with pid [7031]
started core service [/rosout]
process[turtlesim_node-2]: started with pid [7037]
process[rotate-3]: started with pid [7039]
Let's rotate your robot
Input your speed (degrees/sec):
```

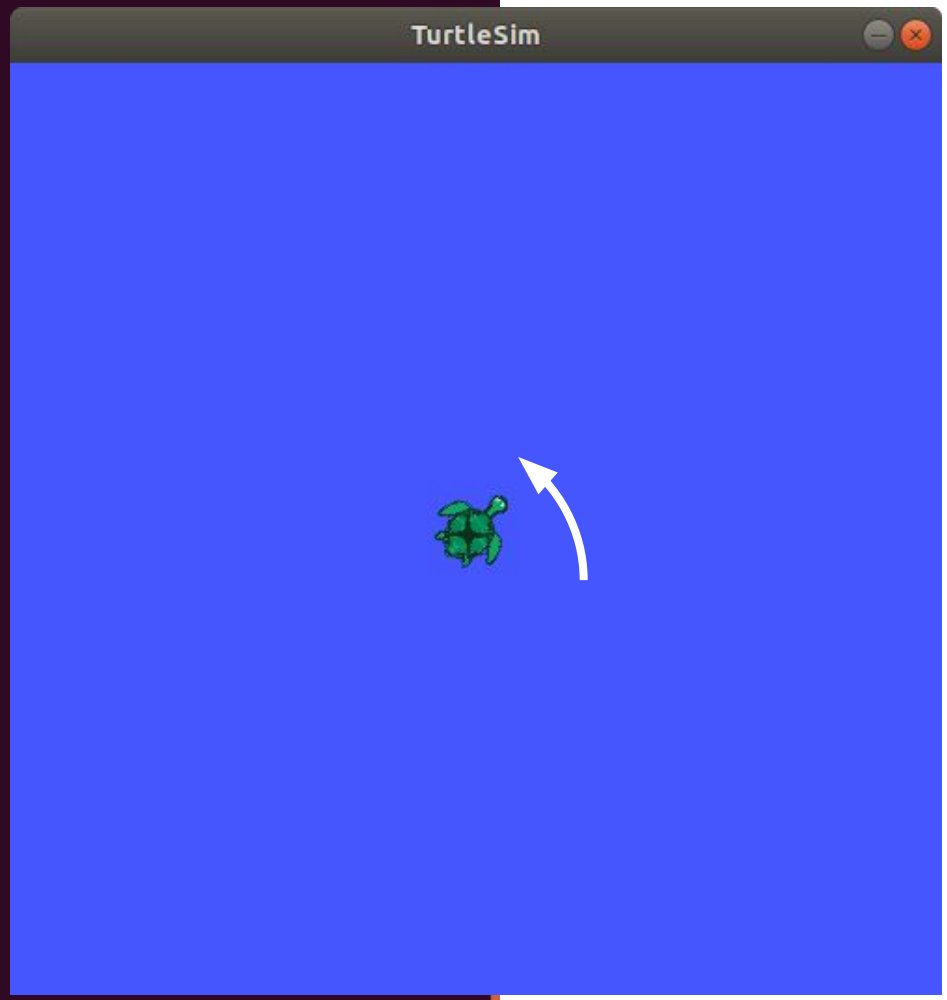


esecuzione per il nodo rotate

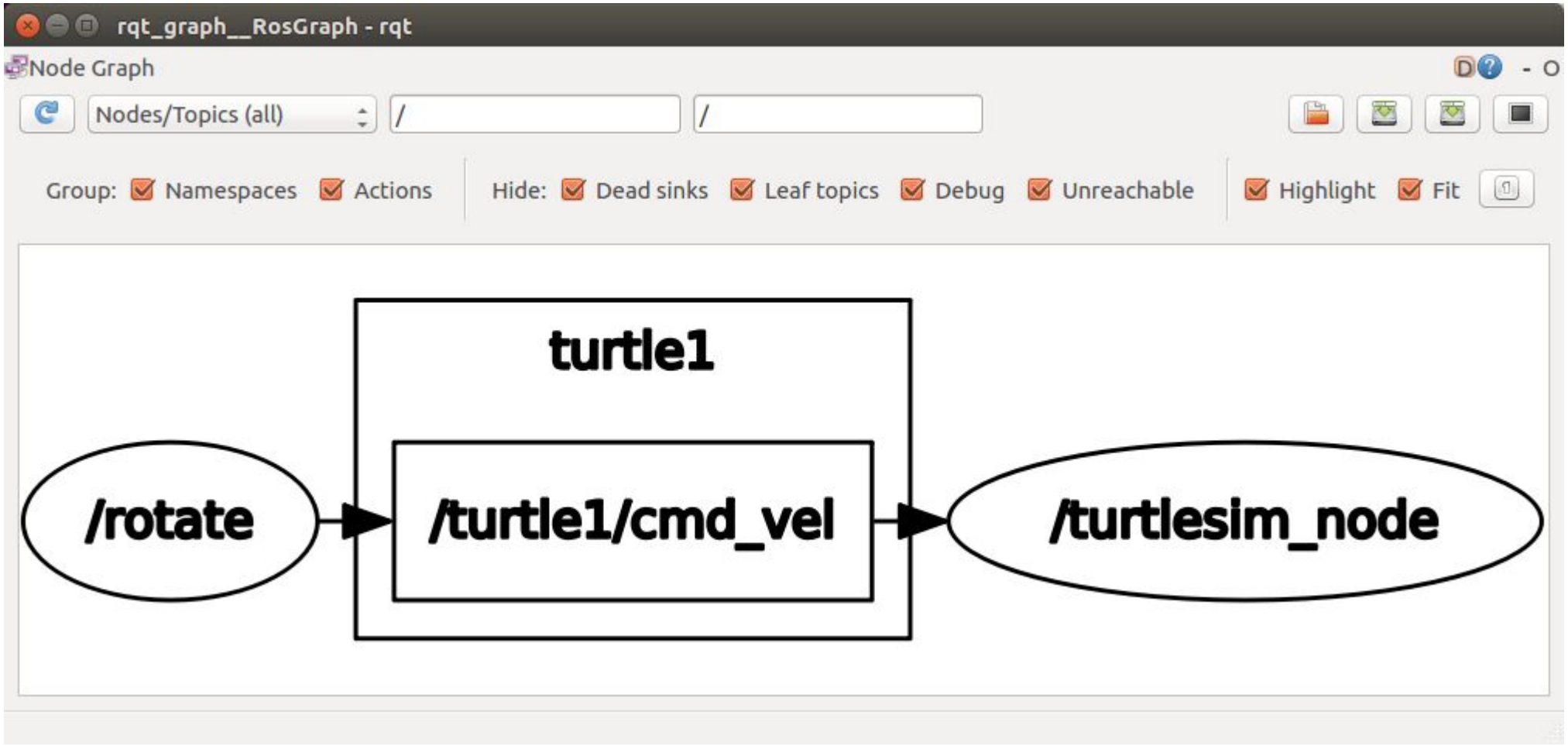
/home/bloisi/catkin_ws/src/unibas_turtle/launch/unibas_rotate.launch http://localhost:11311

File Edit View Search Terminal Help

```
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:37645/

SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.5

NODES
  /
    rotate (unibas_turtle/rotate.py)
    turtlesim_node (turtlesim/turtlesim_node)

auto-starting new master
process[master]: started with pid [7020]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 7639f460-8dea-11ea-8461-50465dde6884
process[rosout-1]: started with pid [7031]
started core service [/rosout]
process[turtlesim_node-2]: started with pid [7037]
process[rotate-3]: started with pid [7039]
Let's rotate your robot
Input your speed (degrees/sec):3
Type your distance (degrees):45
Clockwise?: 0
```



rqt_graph


rqt_graph__RosGraph - rqt
Node Graph
Nodes/Topics (all)
/
/
Group:
Namespaces
Actions
Hide:
Dead sinks
Leaf topics
Debug
Unreachable
Highlight
Fit
turtle1
/rotate
/turtle1/cmd_vel
/turtlesim_node

Il package unibas_teleop

Branch: master ▾ unibas_teleop / src / key_teleop.py / <> Jump to ▾

dbloisi first commit

0 contributors

Executable File | 91 lines (69 sloc) | 1.92 KB

```
#!/usr/bin/env python

from __future__ import print_function

import roslib; roslib.load_manifest('unibas_teleop')
import rospy

from geometry_msgs.msg import Twist

import sys, select, termios, tty

```

https://github.com/dbloisi/unibas_teleop

Il package unibas_teleop

```
msg = """
Reading from keyboard
---------------------------
Use the following keys to move the robot.
   w
a     d
   z

ESC key to quit

"""

```

https://github.com/dbloisi/unibas_teleop

Il package unibas_teleop

```
linear_ = 0.
angular_ = 0.
l_scale_ = 0.5
a_scale_ = 0.5
dirty = False

KEYCODE_R = 'd'
KEYCODE_L = 'a'
KEYCODE_U = 'w'
KEYCODE_D = 'z'

bindings = {
    KEYCODE_L:(0.0, 1.0, True),
    KEYCODE_R:(0.0, -1.0, True),
    KEYCODE_U:(1.0, 0.0, True),
    KEYCODE_D:(-1.0, 0.0, True)
}

def getKey():
    tty.setraw(sys.stdin.fileno())
    select.select([sys.stdin], [], [], 0)
    key = sys.stdin.read(1)
    termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
    return key

```

https://github.com/dbloisi/unibas_teleop

Il package unibas_teleop

```
50  if __name__=="__main__":
51      settings = termios.tcgetattr(sys.stdin)
52
53      pub = rospy.Publisher('turtle1/cmd_vel', Twist, queue_size = 1)
54      rospy.init_node('key_teleop')
55
56      try:
57          print(msg)
58          run = True
59          while(run):
60              key = getKey()
61              linear_ = 0.
62              angular = 0.
63              dirty = False
64
65              if key in bindings.keys():
66                  linear_ = bindings[key][0]
67                  angular_ = bindings[key][1]
68                  dirty = bindings[key][2]
69              elif ord(key) == 27: #ESC key
70                  print('quit')
71                  run = False
72                  continue
```

https://github.com/dbloisi/unibas_teleop

Il package unibas_teleop

```
            twist = Twist()
            twist.linear.x = l_scale_*linear_
            twist.linear.y = 0;
            twist.linear.z = 0;
            twist.angular.x = 0;
            twist.angular.y = 0;
            twist.angular.z = a_scale_*angular_
            if dirty is True:
                pub.publish(twist)
                termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
                dirty = False


    except Exception as e:
        print(e)

```

https://github.com/dbloisi/unibas_teleop

unibas_key_teleop.launch

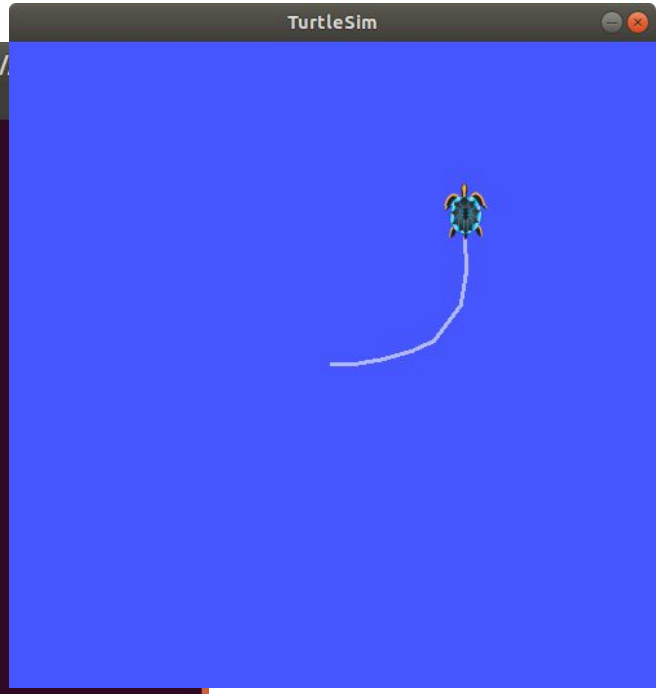
Open ▾ | unibas_key_teleop.launch
~/catkin_ws/src/unibas_teleop/launch | Save

```
<launch>
    <node name="turtlesim_node" pkg="turtlesim" type="turtlesim_node"/>
    <node name="key_teleop" pkg="unibas_teleop" type="key_teleop.py" output="screen"/>
</launch>
```

/home/bloisi/catkin_ws/src/unibas_teleop/launch/unibas_key_teleop.launch http:/

File Edit View Search Terminal Help

```
    key_teleop (unibas_teleop/key_teleop.py)
    turtlesim_node (turtlesim/turtlesim_node)

auto-starting new master
process[master]: started with pid [4062]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 76a4e73a-8dee-11ea-be24-50465dde6884
process[rosout-1]: started with pid [4073]
started core service [/rosout]
process[turtlesim_node-2]: started with pid [4079]
process[key_teleop-3]: started with pid [4081]

Reading from keyboard
---------------------------
Use the following keys to move the robot.
   w
a     d
   z

ESC key to quit
```



https://github.com/dbloisi/unibas_teleop

UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

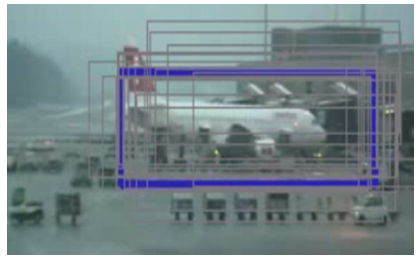


Corso di Visione e Percezione

A.A. 2019/2020

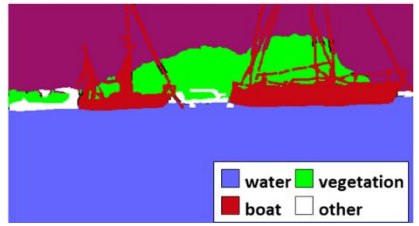
Docente

Domenico Daniele Bloisi

ROS launch file





Maggio 2020