



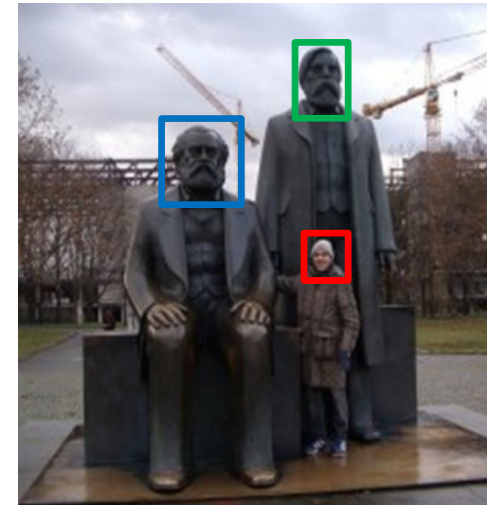
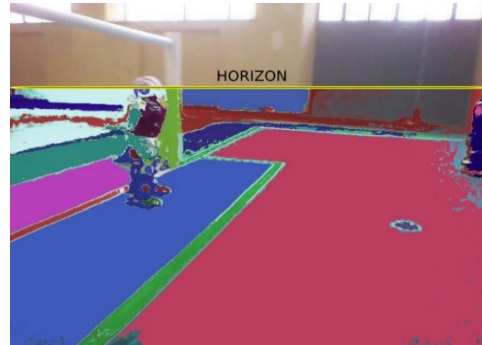
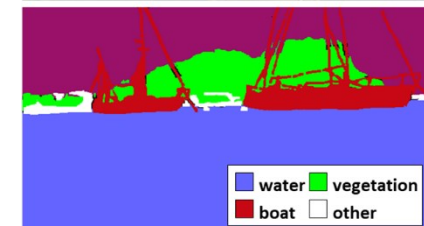
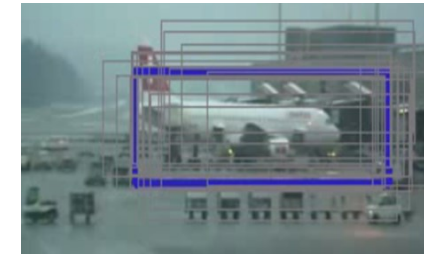
UNIVERSITÀ  
di **VERONA**

Dipartimento  
di **INFORMATICA**

*Corso di Laboratorio Ciberfisico*  
*Modulo di Robot Programming with ROS*

# ROS Launch file

Docente:  
**Domenico Daniele  
Bloisi**



Marzo 2018

# 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="chat_pkg" type="talker" output="screen"/>
```

```
  <node name="listener" pkg="chat_pkg" type="listener" output="screen"/>
```

```
</launch>
```

- Il tag `<node>` contiene gli attributi per specificare il nome con cui il nome verrà inserito nel grafo di ROS, il package nel quale può essere trovato e il `type`, che è il filename dell'eseguibile
- L'attributo `output` posto a "screen" indica che i messaggi di log di ROS verranno mostrati sul terminale su cui verrà eseguito il comando `roslaunch`

# Package univr\_turtle

GitHub, Inc. [US] | [https://github.com/dbloisi/univr\\_turtle](https://github.com/dbloisi/univr_turtle)

This repository Search Pull requests Issues Marketplace Explore

dbloisi / univr\_turtle Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

ROS turtlesim based publisher/subscriber example Edit

Add topics

7 commits 1 branch 0 releases 1 contributor LGPL-3.0

Branch: master New pull request Create new file Upload files Find file Clone or download

dbloisi Update README.md Latest commit 9de51dd 4 days ago

images	course logo	4 days ago
launch	full code	4 days ago
src	full code	4 days ago
CMakeLists.txt	full code	4 days ago
LICENSE	Initial commit	5 days ago
README.md	Update README.md	4 days ago
package.xml	full code	4 days ago

README.md

[https://github.com/dbloisi/univr\\_turtle](https://github.com/dbloisi/univr_turtle)

# idea

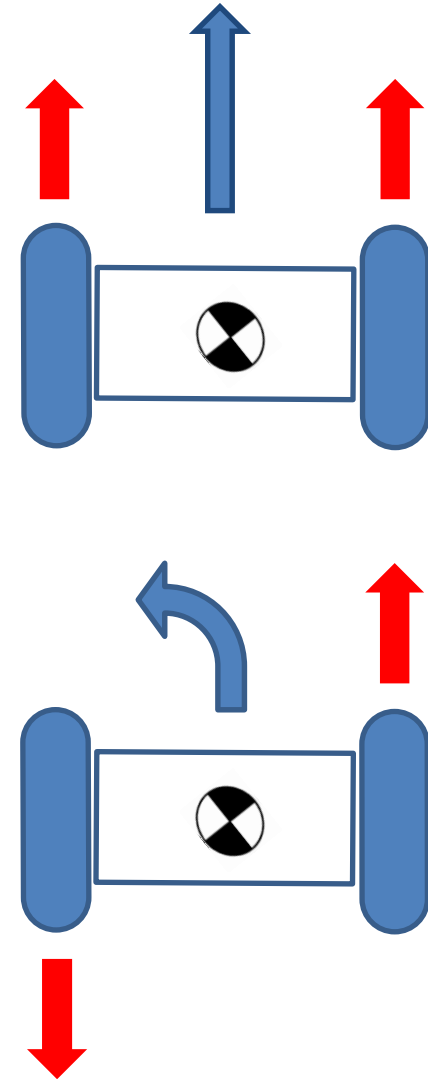
---

- Vogliamo far muovere la tartaruga controllandone la velocità
- Utilizziamo per la tartaruga il modello di un robot differenziale
- In questo modo possiamo utilizzare per il controllo la velocità lineare e la velocità angolare

# Differential drive robot

---

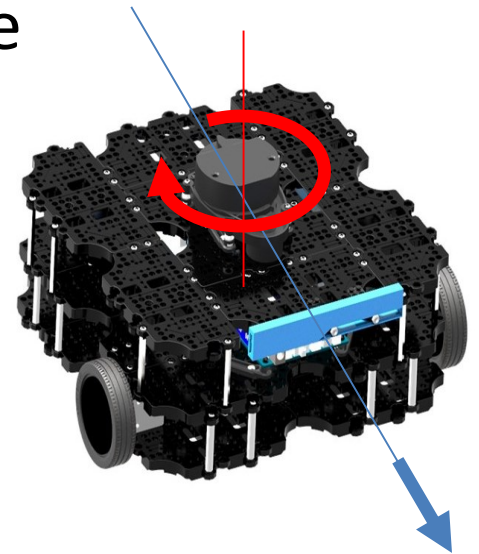
- Un robot differenziale su ruote è una base mobile avente due ruote motorizzate indipendenti
- Le ruote sono posizionate ai due lati opposti della scocca
- Il robot si muove in avanti quando entrambe le ruote girano in avanti, mentre gira sul posto quando una ruota gira in avanti e l'altra gira all'indietro



# Movimento del robot

---

- 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
- 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

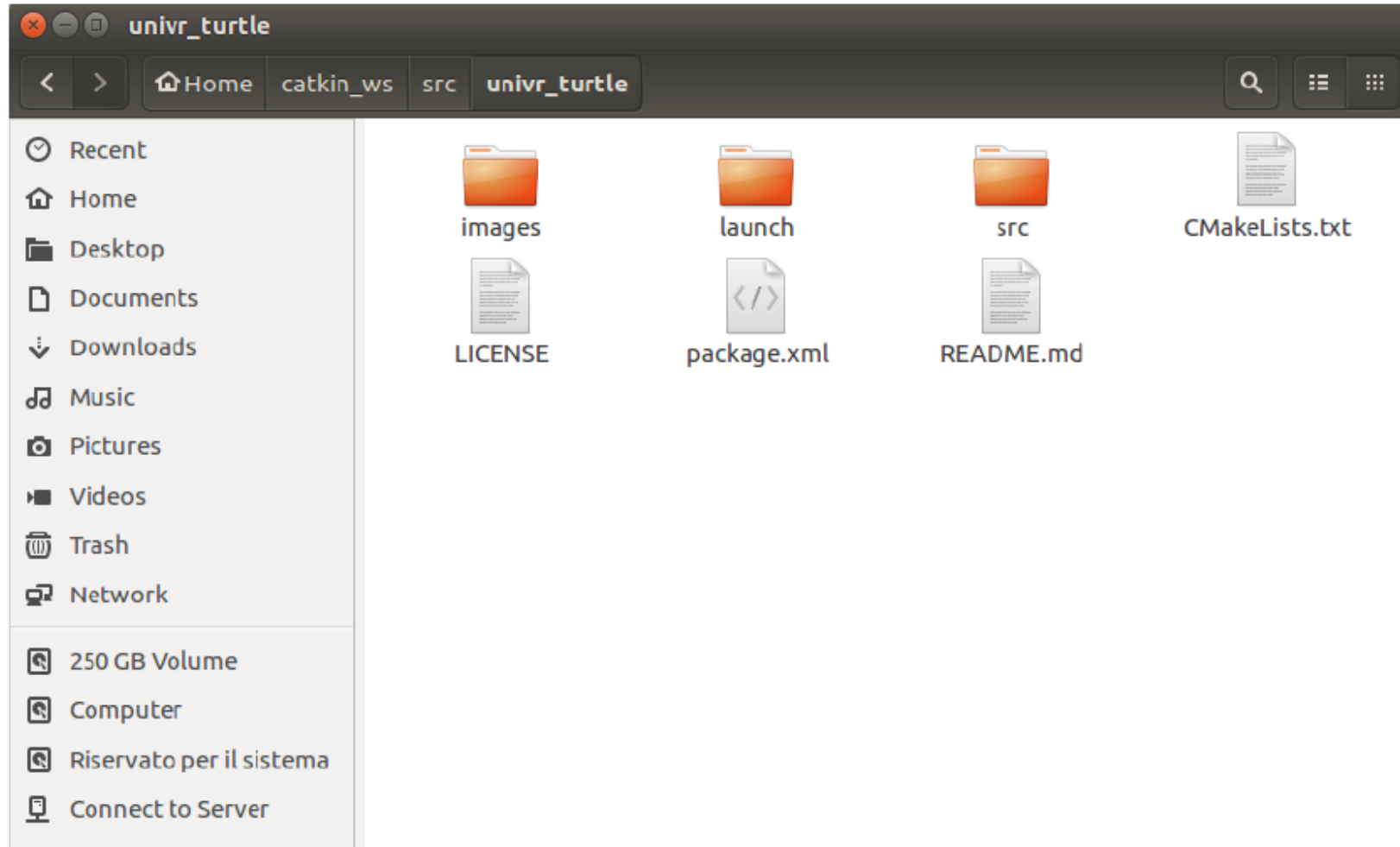
```
# This expresses velocity in free space broken into its linear and angular parts.  
Vector3 linear  
Vector3 angular
```

### Compact Message Definition

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

# File in univr\_turtle

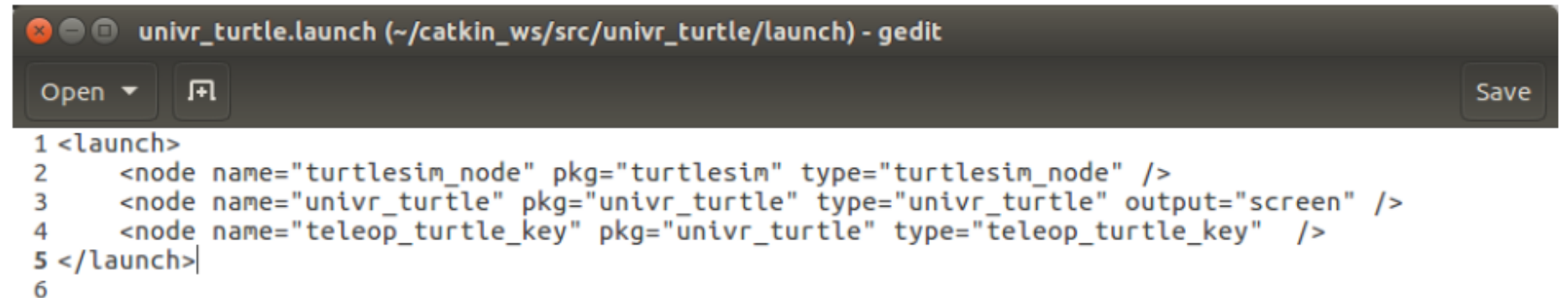
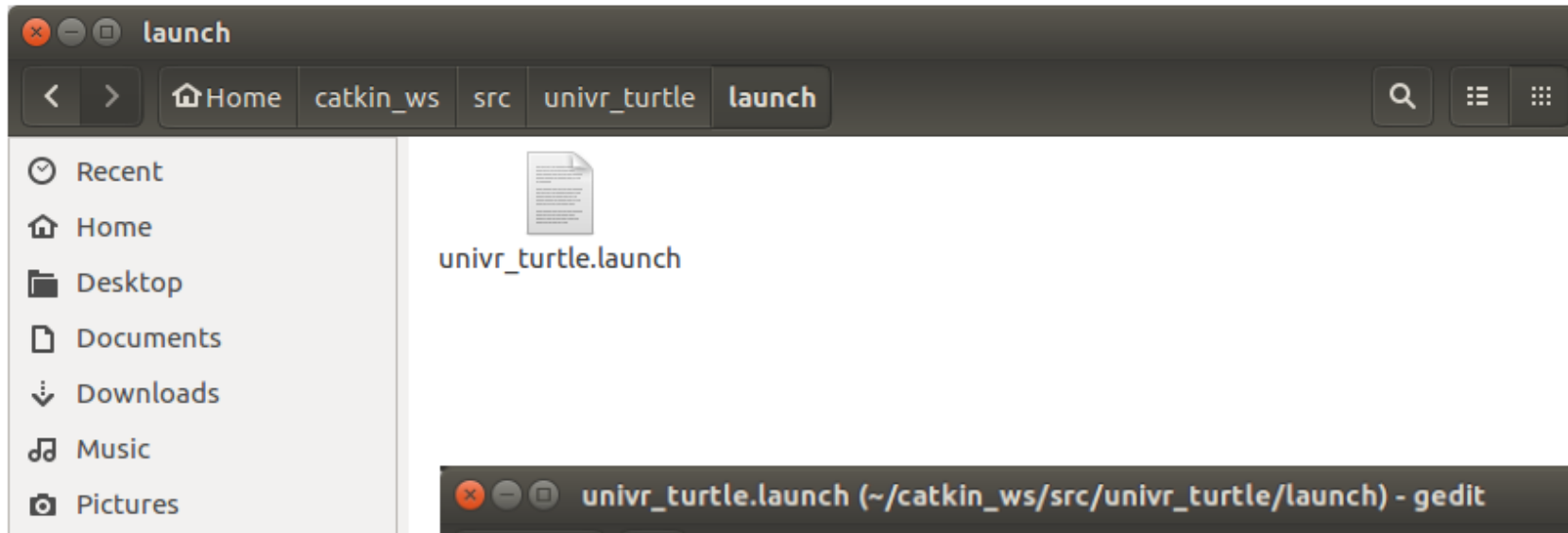
---



[https://github.com/dbloisi/univr\\_turtle](https://github.com/dbloisi/univr_turtle)

# univr\_turtle.launch

---

A screenshot of a gedit editor window titled 'univr\_turtle.launch (~/.catkin\_ws/src/univr\_turtle/launch) - gedit'. The editor shows the following XML code:

```
1 <launch>
2   <node name="turtlesim_node" pkg="turtlesim" type="turtlesim_node" />
3   <node name="univr_turtle" pkg="univr_turtle" type="univr_turtle" output="screen" />
4   <node name="teleop_turtle_key" pkg="univr_turtle" type="teleop_turtle_key" />
5 </launch>
6
```

The editor interface includes an 'Open' dropdown menu, a '+1' icon, and a 'Save' button.

[https://github.com/dbloisi/univr\\_turtle](https://github.com/dbloisi/univr_turtle)

# Esempio roslaunch

---

```
roslaunch univr_turtle univr_turtle.launch
```

ROS package name



launch file name



# Esecuzione roslaunch

```
bloisi@bloisi-U36SG:~$ roslaunch univr_turtle univr_turtle.launch
... logging to /home/bloisi/.ros/log/38399f00-314c-11e8-8f36-dc85de574b1d/roslau
nch-bloisi-U36SG-3795.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://localhost:41284/

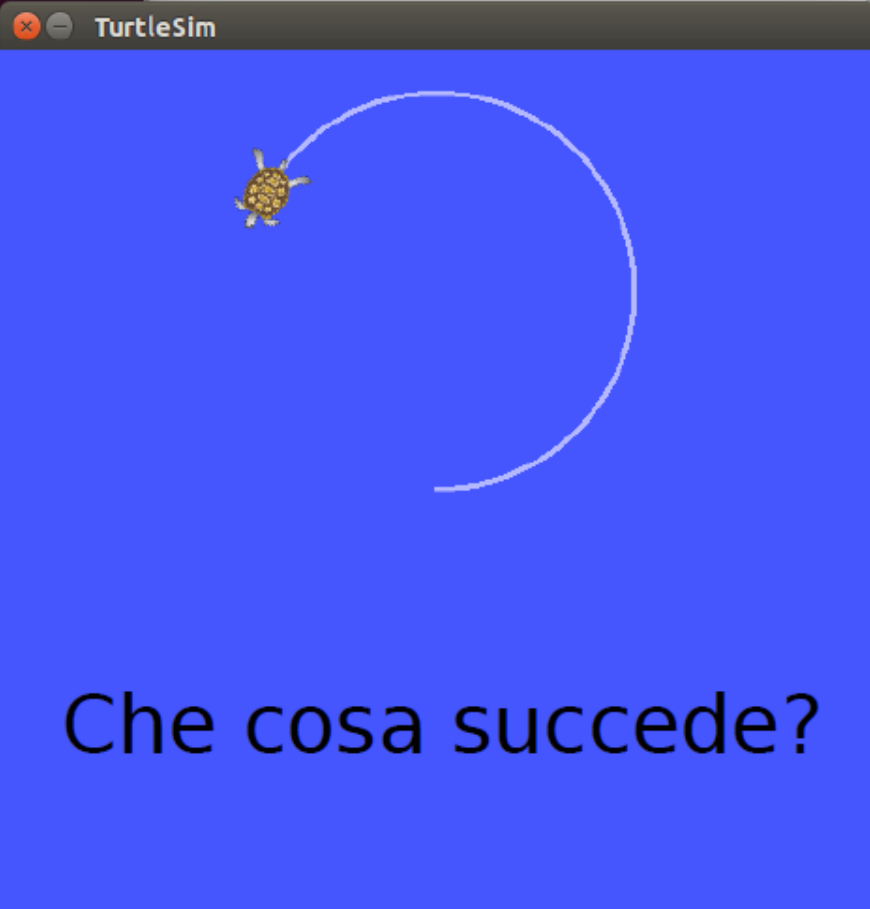
SUMMARY
=====

PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.13

NODES
/
  teleop_turtle_key (univr_turtle/teleop_turtle_key)
  turtlesim_node (turtlesim/turtlesim_node)
  univr_turtle (univr_turtle/univr_turtle)

auto-starting new master
process[master]: started with pid [3805]
ROS_MASTER_URI=http://localhost:11311

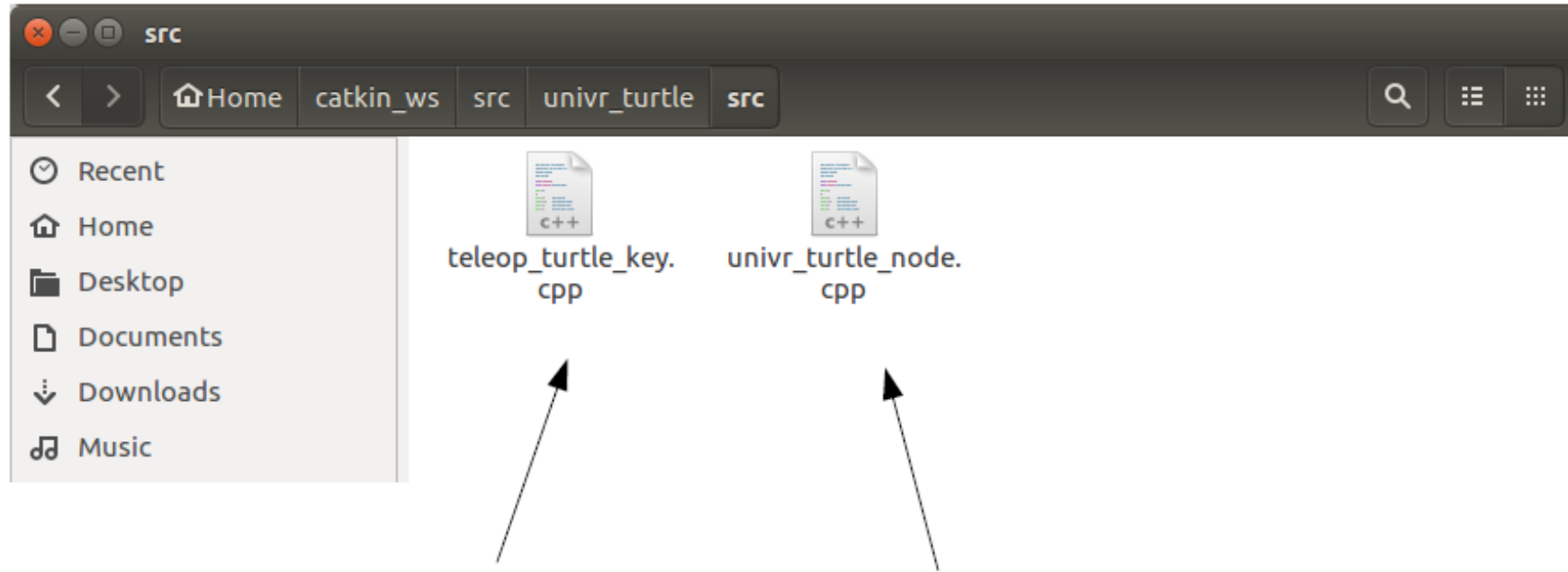
setting /run_id to 38399f00-314c-11e8-8f36-dc85de574b1d
process[rosout-1]: started with pid [3818]
started core service [/rosout]
process[turtlesim_node-2]: started with pid [3822]
process[univr_turtle-3]: started with pid [3827]
process[teleop_turtle_key-4]: started with pid [3835]
[ INFO] [1522106407.109770070]: Starting to move in circle
```



Che cosa succede?

# I nodi di univr\_turtle

---



Nodo per la gestione della teleoperazione

nodo per il controllo in velocità della tartaruga

# univr\_turtle\_node

```
univr_turtle_node.cpp (~catkin_ws/src/univr_turtle/src) - gedit
Open Save
1 /*
2  * univr_turtle_node.cpp
3  *
4  * This file is part of univr_turtle and it is distributed under the terms of the
5  * GNU Lesser General Public License (Lesser GPL)
6  *
7  * univr_turtle is included in the material of the course (in italian)
8  * Laboratorio Ciberfisico
9  * Robot Programming with ROS
10 * A.V. 2017/2018
11 * University of Verona (Italy)
12 * http://profs.scienze.univr.it/~bloisi/corsi/ciberfisico.html
13 *
14 * univr_turtle is distributed in the hope that it will be useful, but WITHOUT
15 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS
16 * FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.
17 *
18 * You should have received a copy of the GNU Lesser General Public License along with
19 * hello_ros.
20 * If not, see http://www.gnu.org/licenses/.
21 *
22 * The files in hello_ros contain the ROS based implementation for a simple publisher/subscriber
23 * mechanism.
24 *
25 * see this file for additional information (in italian)
26 *
27 * Please, report suggestions/comments/bugs to
28 * domenico.bloisi@gmail.com
29 */
30
```

C++ Tab Width: 8 Ln 88, Col 12 INS

# univr\_turtle\_node

```
univr_turtle_node.cpp (~/.catkin_ws/src/univr_turtle/src) - gedit
Open Save
30
31 #include "ros/ros.h"
32 #include "geometry_msgs/Twist.h"
33 #include "turtlesim/Pose.h"
34
35 const float BASE_LIN_VEL = 0.5, BASE_ANG_VEL = 0.2;
36
37 float lin_vel_ = BASE_LIN_VEL;
38 float ang_vel_ = BASE_ANG_VEL;
39
40 ros::WallTime last_command_time_;
41
42 // Topic messages callback
43 void poseCallback(const turtlesim::PoseConstPtr& msg)
44 {
45     //ROS_INFO("x: %.2f, y: %.2f", msg->x, msg->y);
46 }
47
48 void velocityCallback(const geometry_msgs::Twist::ConstPtr& vel)
49 {
50     last_command_time_ = ros::WallTime::now();
51     lin_vel_ = vel->linear.x;
52     ang_vel_ = vel->angular.z;
53
54     ROS_INFO("keyboard: lin_vel: %.2f, ang_vel: %.2f", lin_vel_, ang_vel_);
55 }
56
C++ Tab Width: 8 Ln 88, Col 12 INS
```



# univr\_turtle\_node

---

```
univr_turtle_node.cpp (~/.catkin_ws/src/univr_turtle/src) - gedit
Open ▾ [+] Save
56
57 int main(int argc, char **argv)
58 {
59     // initialization
60     ros::init(argc, argv, "univr_turtle");
61     ros::NodeHandle node;
62
63     // A publisher for the movement data
64     ros::Publisher pub = node.advertise<geometry_msgs::Twist>("turtle1/cmd_vel", 10);
65
66     // A listener for pose
67     ros::Subscriber sub = node.subscribe("turtle1/pose", 10, poseCallback);
68
69     // A listener for left, right, up, and down keyboard arrow commands
70     ros::Subscriber velocity_sub = node.subscribe("keyboard/cmd_vel", 1, velocityCallback);
71
72     // set the publishing rate at 10Hz
73     ros::Rate rate(10);
74
75     ROS_INFO("Starting to move in a circle");
76
C++ ▾ Tab Width: 8 ▾ Ln 88, Col 12 ▾ INS
```

# univr\_turtle\_node

```
univr_turtle_node.cpp (~/.catkin_ws/src/univr_turtle/src) - gedit
Open [v] [F1] Save
76
77 while (ros::ok()) {
78     geometry_msgs::Twist msg;
79     msg.linear.x = lin_vel_;
80     msg.angular.z = ang_vel_;
81     pub.publish(msg);
82     ros::spinOnce(); // processing of incoming messages
83     // When a message arrives, ROS pushes your subscriber callback onto a queue.
84     // It does not call it immediately. ROS only processes your callbacks when
85     // you tell it to with ros::spinOnce()
86     rate.sleep();
87     // if no messages are received from the keyboard for 0.5 seconds, then
88     // the turtle starts moving in a circle
89     if (ros::WallTime::now() - last_command_time_ > ros::WallDuration(0.5))
90     {
91         lin_vel_ = BASE_LIN_VEL;
92         ang_vel_ = BASE_ANG_VEL;
93     }
94 }
95 }
96
C++ Tab Width: 8 Ln 88, Col 12 INS
```

# teleop\_turtle\_key

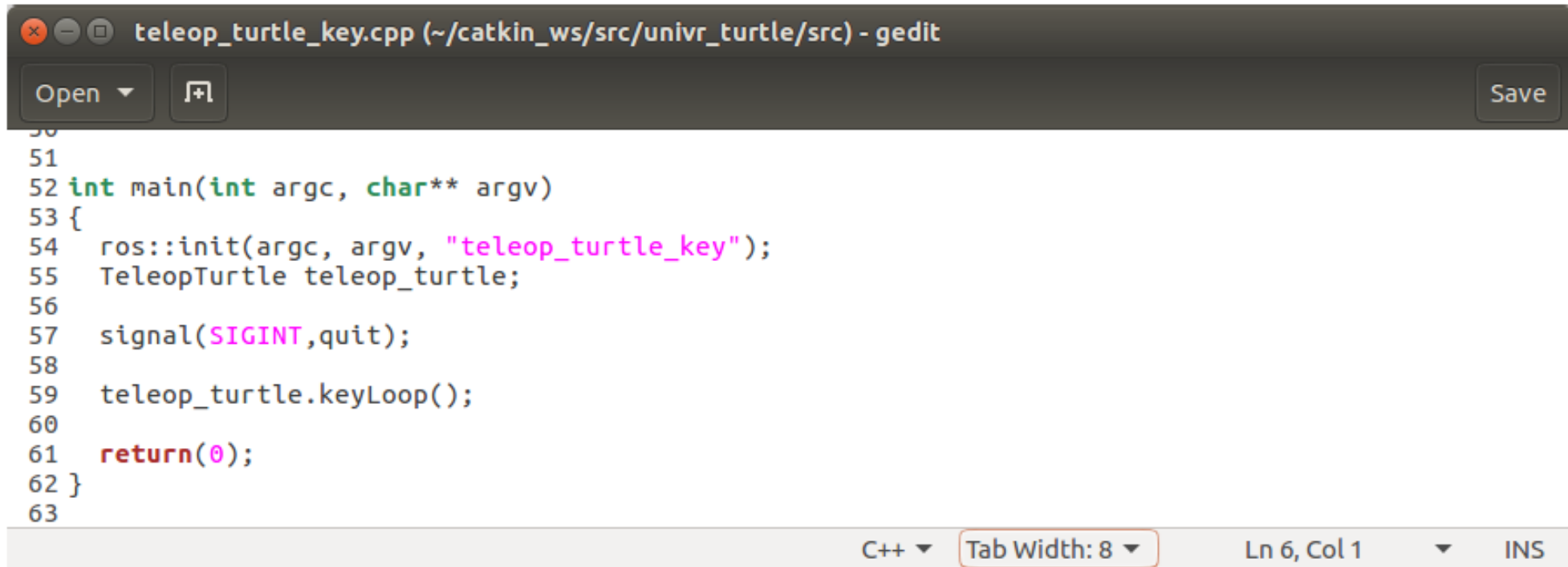
```
teleop_turtle_key.cpp (~catkin_ws/src/univr_turtle/src) - gedit
Open [icon] Save
1 #include <ros/ros.h>
2 #include <geometry_msgs/Twist.h>
3 #include <signal.h>
4 #include <termios.h>
5 #include <stdio.h>
6
7 #define KEYCODE_R 0x43
8 #define KEYCODE_L 0x44
9 #define KEYCODE_U 0x41
10 #define KEYCODE_D 0x42
11 #define KEYCODE_Q 0x71
12
13 class TeleopTurtle
14 {
15 public:
16   TeleopTurtle();
17   void keyLoop();
18
19 private:
20
21
22   ros::NodeHandle nh_;
23   double linear_, angular_, l_scale_, a_scale_;
24   ros::Publisher twist_pub_;
25
26 };
C++ Tab Width: 8 Ln 6, Col 1 INS
```

# teleop\_turtle\_key

```
teleop_turtle_key.cpp (~/.catkin_ws/src/univr_turtle/src) - gedit
Open [+] Save
27
28 TeleopTurtle::TeleopTurtle():
29     linear_(0),
30     angular_(0),
31     l_scale_(0.5),
32     a_scale_(0.5)
33 {
34     nh_.param("scale_angular", a_scale_, a_scale_);
35     nh_.param("scale_linear", l_scale_, l_scale_);
36
37     twist_pub_ = nh_.advertise<geometry_msgs::Twist>("keyboard/cmd_vel", 1);
38 }
39
40 int kfd = 0;
41 struct termios cooked, raw;
42
43 void quit(int sig)
44 {
45     (void)sig;
46     tcsetattr(kfd, TCSANOW, &cooked);
47     ros::shutdown();
48     exit(0);
49 }
50
C++ Tab Width: 8 Ln 6, Col 1 INS
```

# teleop\_turtle\_key

---



```
teleop_turtle_key.cpp (~/.catkin_ws/src/univr_turtle/src) - gedit
50
51
52 int main(int argc, char** argv)
53 {
54     ros::init(argc, argv, "teleop_turtle_key");
55     TeleopTurtle teleop_turtle;
56
57     signal(SIGINT, quit);
58
59     teleop_turtle.keyLoop();
60
61     return(0);
62 }
63
```

C++ ▾ Tab Width: 8 ▾ Ln 6, Col 1 ▾ INS

# teleop\_turtle\_key

---

```
teleop_turtle_key.cpp (~/.catkin_ws/src/univr_turtle/src) - gedit
Open [ ] Save
64
65 void TeleopTurtle::keyLoop()
66 {
67     char c;
68     bool dirty=false;
69
70
71     // get the console in raw mode
72     tcgetattr(kfd, &cooked);
73     memcpy(&raw, &cooked, sizeof(struct termios));
74     raw.c_lflag &=~ (ICANON | ECHO);
75     // Setting a new line, then end of file
76     raw.c_cc[VEOL] = 1;
77     raw.c_cc[VEOF] = 2;
78     tcsetattr(kfd, TCSANOW, &raw);
79
80     puts("Reading from keyboard");
81     puts("-----");
82     puts("Use arrow keys to move the turtle.");
83
C++ Tab Width: 8 Ln 6, Col 1 INS
```

# teleop\_turtle\_key

```
teleop_turtle_key.cpp (~/.catkin_ws/src/univr_turtle/src) - gedit
Open [ ] Save
84
85 for(;;)
86 {
87     // get the next event from the keyboard
88     if(read(kfd, &c, 1) < 0)
89     {
90         perror("read()");
91         exit(-1);
92     }
93
94     llinear_=angular_=0;
95     ROS_DEBUG("value: 0x%02X\n", c);
96
97     switch(c)
98     {
99         case KEYCODE_L:
100             ROS_DEBUG("LEFT");
101             angular_ = 1.0;
102             dirty = true;
103             break;
104         case KEYCODE_R:
105             ROS_DEBUG("RIGHT");
106             angular_ = -1.0;
107             dirty = true;
108             break;
109         case KEYCODE_U:
110             ROS_DEBUG("UP");
111             linear_ = 1.0;
112             dirty = true;
113             break;
114         case KEYCODE_D:
115             ROS_DEBUG("DOWN");
116             linear_ = -1.0;
117             dirty = true;
118             break;
119     }
120
```

C++ Tab width: 8 Ln 6, Col 1 INS

# teleop\_turtle\_key

---

```
teleop_turtle_key.cpp (~/.catkin_ws/src/univr_turtle/src) - gedit
Open [+] Save
121
122     geometry_msgs::Twist twist;
123     twist.angular.z = a_scale_*angular_;
124     twist.linear.x = l_scale_*linear_;
125     if(dirty ==true)
126     {
127         twist_pub_.publish(twist);
128         dirty=false;
129     }
130 }
131
132
133     return;
134 }
135
136
```

C++ ▾ Tab Width: 8 ▾ Ln 6, Col 1 ▾ INS



# rqt\_graph

The screenshot displays a Linux desktop environment with the following components:

- Terminal Window:** Shows the execution of the `roslaunch univr_turtle univr_turtle.launch` command, followed by the `rqt_graph` command.
- rqt\_graph Window:** Displays a ROS Node Graph with the following structure:

```
graph LR; A([/teleop_turtle_key]) -- /keyboard/cmd_vel --> B([/univr_turtle]); B -- /turtle1/cmd_vel --> C([/turtlesim_node]); C -- /turtle1/pose --> B;
```
- TurtleSim Window:** Shows a 2D simulation of a turtle on a blue plane.

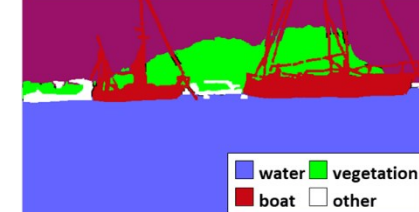
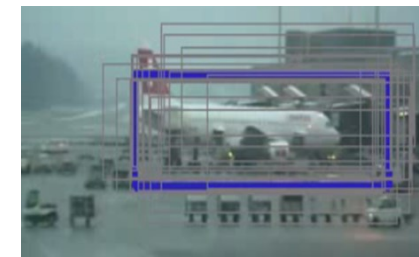


UNIVERSITÀ  
di **VERONA**

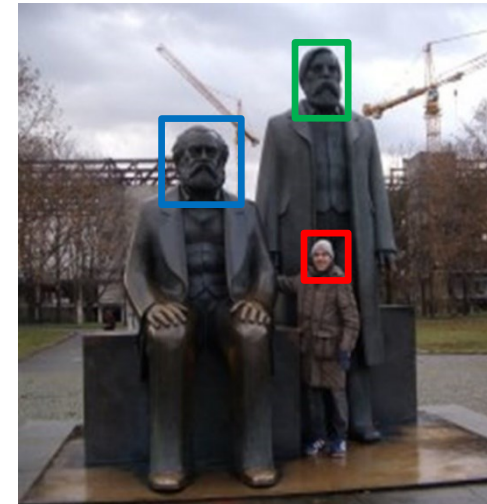
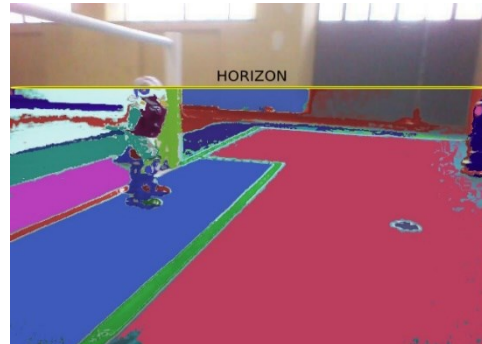
Dipartimento  
di **INFORMATICA**

*Corso di Laboratorio Ciberfisico*  
*Modulo di Robot Programming with ROS*

# ROS Launch file



Docente:  
**Domenico Daniele  
Bloisi**



Marzo 2018