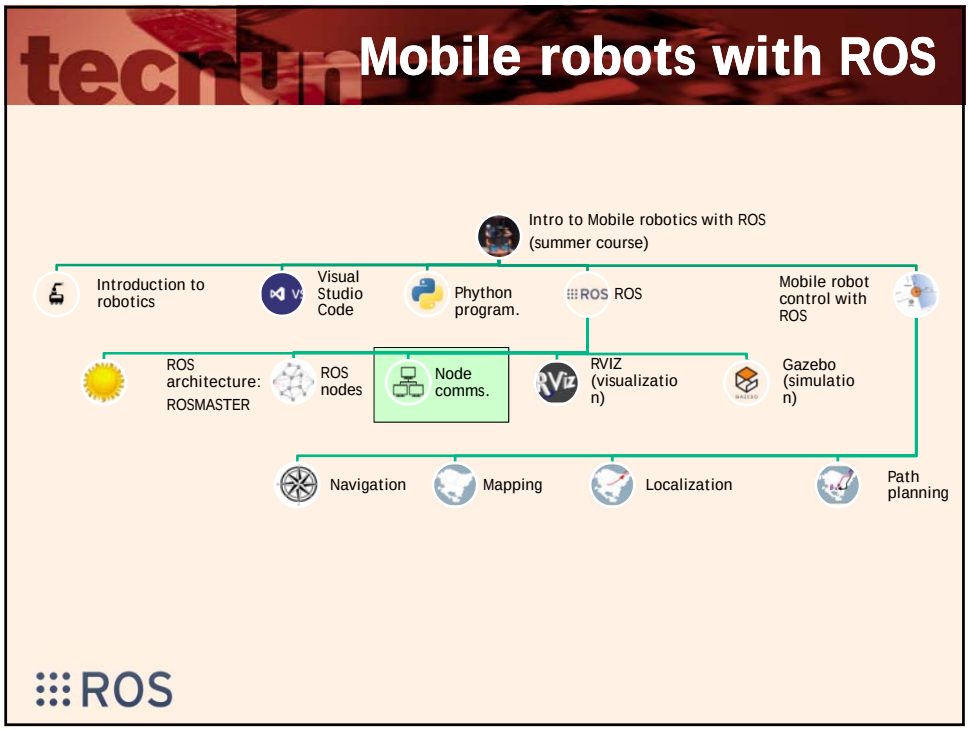
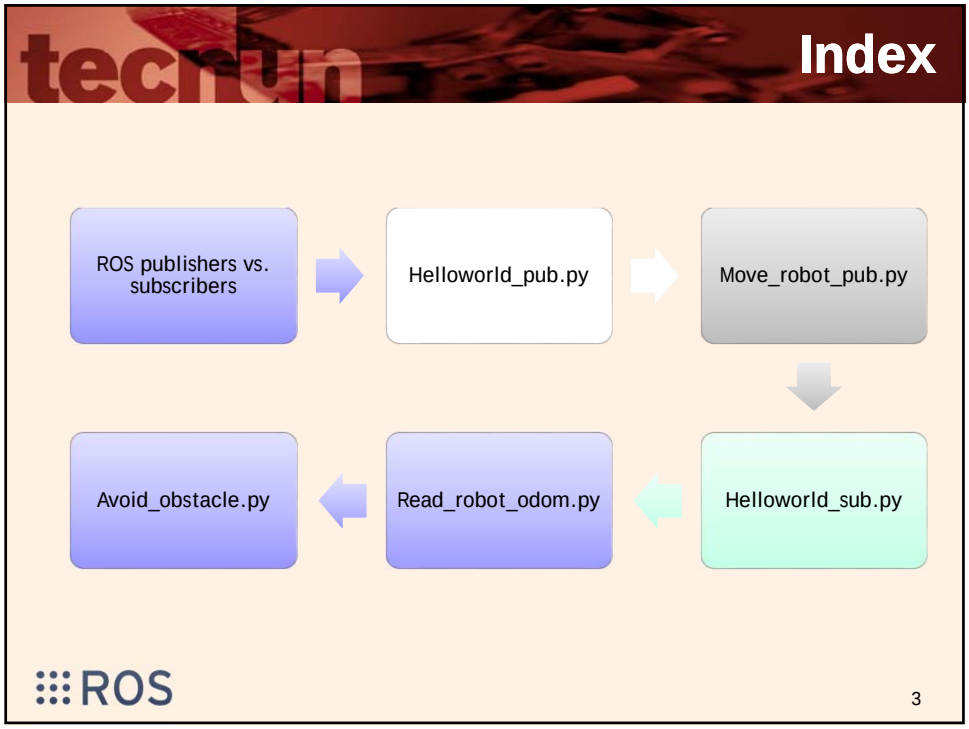

ROS publishers and
ROS subscribers
Dr. Emilio Sánchez Tapia
tecnun

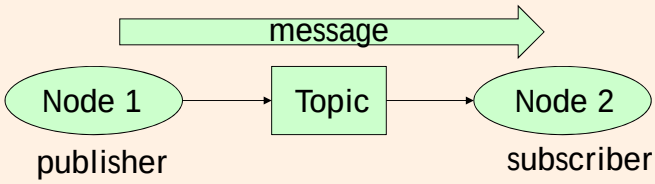

tecnun
Mobile robots with ROS
Intro to Mobile robotics with ROS
(summer course)
Introduction to
robotics
Visual
Studio
Code
Phython
program.
ROS
Mobile robot
control with
ROS
ROS
architecture:
ROSMASTER
ROS
nodes
Node
comms.
RVIZ
(visualizatio
n)
Gazebo
(simulatio
n)
Navigation
Mapping
Localization
Path
planning
ROS


Index
ROS publishers vs. subscribers
Helloworld_pub.py
Move_robot_pub.py
Helloworld_sub.py
Read_robot_odom.py
Avoid_obstacle.py
ROS
3


ROS publishers vs. subscribers
ROS

ROS publisher vs ROS subscriber

- A publisher is a node that keeps sending a message into a topic at a predefined rate
- Remember→ a topic is a pipe that interconnects nodes
- A subscriber reads the message in the topic sent by the publisher



ROS

5

ROS publisher: helloworld_pub.py

```
#! /usr/bin/env python
import rospy
from std_msgs.msg import Int32

rospy.init_node('helloworld_publisher')
pub = rospy.Publisher('/counter', Int32, queue_size=1)
rate = rospy.Rate(0.2)
count = Int32()
count.data = 0

while not rospy.is_shutdown():
  pub.publish(count)
  count.data += 1
  rate.sleep()
```

ROS

7

ROS publisher: helloworld_pub.py

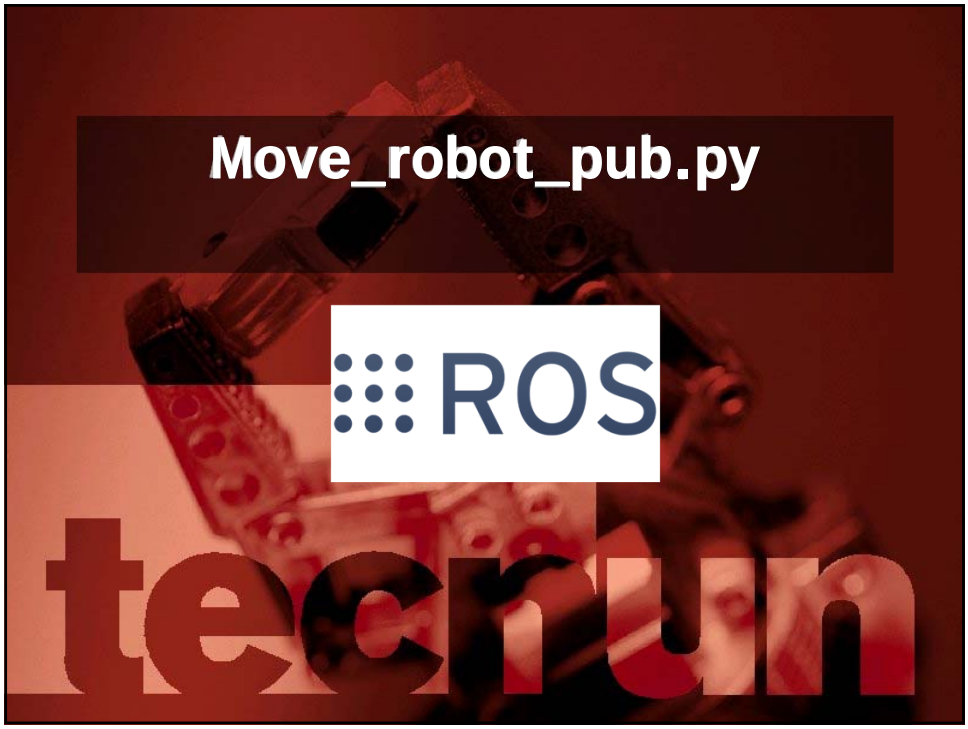
```
#! /usr/bin/env python
import rospy                                # Import the Python library
for ROS
from std_msgs.msg import Int32              # Import the Int32 message
from the std_msgs package

rospy.init_node(' helloworld_publisher')         # Initiate a Node
named 'topic_publisher'
# Create a Publisher object, that will publish on the /counter topic
pub = rospy.Publisher('counter', Int32) #  messages of type Int32
rate = rospy.Rate(0.2)                    # Set a publish rate of 2 Hz
count = Int32()                          # Create a var of type Int32
count.data = 0                           # Initialize 'count' variable

# Create a loop that will go until someone stops the program execution
while not rospy.is_shutdown():
     # Publish the message within the 'count' variable
     pub.publish(count)
     count.data += 1                     # Increment 'count' variable
     rate.sleep()        # Make sure the publish rate maintains at 2 Hz
```

ROS

8



ROS publisher: move_robot_pub.launch

```
<launch>
   <node pkg="helloworld_pkg" type="move_robot.py"
name="move_robot_node" output="screen" />
</launch>
```

ROS

10

ROS subscriber: helloworld_pub.py test

- Try the subscriber without running the publisher.
- We can simulate the publisher by:

```
> rostopic pub <topic_name> <message_type> <value>
```

- Run the subscriber and try this command:

```
> rostopic pub /counter std_msgs/Int32 28
```

ROS

11

ROS publisher: move_robot.py

```
import rospy
from geometry_msgs.msg import Twist
rospy.init_node('move_robot_node')
pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
rate = rospy.Rate(0.2)
move_cmd = Twist()
#differential motion: Vx,OMEGAz
#Move the robot with a linear velocity in the x axis
move_cmd.linear.x = 0.5
#Move the with an angular velocity in the z axis
move_cmd.angular.z = 0.5

while not rospy.is_shutdown():
    pub.publish(move_cmd)
    rate.sleep()
```

ROS

12

ROS subscriber: helloworld_sub.py

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

import rospy
from std_msgs.msg import Int32

def callback(msg):
  print msg.data

rospy.init_node('helloworld_subscriber')
sub = rospy.Subscriber('counter', Int32, callback)
rospy.spin()
```

ROS

14

ROS subscriber: helloworld_sub.py

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

import rospy
from std_msgs.msg import Int32

# a callback function that manages the incoming msg
def callback(msg):
    # Print the value 'data' inside 'msg'
    print msg.data

# Initiate a Node called ‘helloworld_subscriber'
rospy.init_node(‘helloworld_subscriber')

# Create a Subscriber object that will listen to the /counter topic
# the callback function will be called automatically each time it reads
# a new message
sub = rospy.Subscriber('/counter', Int32, callback)

rospy.spin()    # Create a loop that will keep the program in execution
```

ROS

15

ROS subscriber: read_robot_odom.launch

```
<launch>
    <node pkg="helloworld_pkg" type="read_robot_odom.py"
name="read_robot_odom_node" output="screen" />
</launch>
```

ROS

17

ROS subscriber: read_robot_odom.py

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

import rospy
from nav_msgs.msg import Odometry

def callback(msg):
  print msg # print the whole Odometry message
  # print msg.header # print the header section
  # print msg.pose # pose section

rospy.init_node('read_robot_odom_node')
sub = rospy.Subscriber('/odom', Odometry, callback)
rospy.spin()
```

ROS

18

Avoid_obstacle: laser msg

- A code that makes the robot to avoid a obstacle in front of it→ reading the laser scan topic.

```
$ rosmsg show sensor_msgs/LaserScan
            std_msgs/Header header
              uint32 seq
              time stamp
              string frame_id
            float32 angle_min
            float32 angle_max
            float32 angle_increment
            float32 time_increment
            float32 scan_time
            float32 range_min
            float32 range_max
            float32[] ranges <-- Use only this one
            float32[] intensities
```

ROS

20

Avoid_obstacle: launch

- Avoid_obstacle.launch

```
<launch>
   <node pkg="helloworld_pkg" type="avoid_obstacle.py"
name="avoid_obstacle_node" output="screen" />

</launch>
```

ROS

21

Avoid obstacle (1 of 5)

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

def callback(msg):
  left=90
  frontleft=30
  frontright=330
  right=270
  front=0
  back=180

  #print len(msg.ranges)
```

ROS

22

Avoid obstacle (2 of 5)

```
  #If the distance to an obstacle in front of the robot is bigger than
1 meter, the robot will move forward
  if msg.ranges[front] > 1 and msg.ranges[frontleft]>1 and
msg.ranges[frontleft]>1:
      move.linear.x = 0.2
      move.angular.z = 0.0


else:
 #If the distance to an obstacle in front of the robot is smaller than
1 meter, the robot will turn left
      if msg.ranges[front] <= 1:
         print "obstacle ahead, turning left"
         move.linear.x = -0.5
         move.angular.z = 5.0
```

ROS

23

Avoid obstacle (3 of 5)

```
#If the distance to an obstacle at the front-left side of the robot is
smaller than 1 meters, the robot will turn right
      elif msg.ranges[frontleft] < 1:
         print "obstacle on the front-left, turning right"
         move.linear.x = -0.2
         move.angular.z = -2


     #If the distance to an obstacle at the front-right side of the
robot is smaller than 1 meters, the robot will turn left
      elif msg.ranges[frontright] < 1:
         print "obstacle on the front-right, turning left"
         move.linear.x = -0.2
         move.angular.z = 2
```

ROS

24

Avoid obstacle (4 of 5)

```
 #If the distance to an obstacle at the left side of the robot is
smaller than 0.4 meters, the robot will turn right
  if msg.ranges[left] < 0.4:
         print "obstacle on the left, turning right"
         move.linear.x = 0.0
         move.angular.z = -1

     #If the distance to an obstacle at the right side of the robot is
smaller than 0.4 meters, the robot will turn left
  if msg.ranges[right] < 0.4:
         print "obstacle on the right, turning left"
         move.linear.x = 0.0
         move.angular.z = 1
```

ROS

25

Avoid obstacle (5 of 5)

```
#print some info for debugging
 print "FL=",msg.ranges[frontleft],"F=",msg.ranges[front],
"FR=",msg.ranges[frontright],
"L=",msg.ranges[left],"R=",msg.ranges[right],"B=",msg.ranges[back]
   #publish the msg
  pub.publish(move)

#################### node main script  ############################
#node creation
rospy.init_node('avoid_obstacle_node')
#subscriber creation
sub = rospy.Subscriber('/scan', LaserScan, callback) #We subscribe to
the laser's topic
#publisher creation
pub = rospy.Publisher('/cmd_vel', Twist,queue_size=10)
move = Twist()

rospy.spin()
```

ROS

26

Avoid obstacle launch gazebo simulation

- Create avoid_obstacle.launch
- Create avoid_obstacle.py
- Test with gazebo
  - Terminal 1

    ```
    roscore
    ```
  - Terminal 2

    ```
    export SVGA_VGPU10=0
    ```

  Try with veggie

  ```
  roslaunch prk_gazebo veggiebot_gazebo.launch
  ```

  Or try with turtlebot3 `${TB3_MODEL}=Burger|waffle|waffle_pi`

```
export TURTLEBOT3_MODEL=${TB3_MODEL}
roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
roslaunch turtlebot3_gazebo turtlebot3_world.launch
roslaunch turtlebot3_gazebo turtlebot3_world.launch
```

ROS

27

Avoid obstacle: robot cmds from a terminal

- Terminal 3 → activate robot
  - Console cmds
    - Vx→ linear speed along x-axis
    - Wz → angular speed along z-axis

```
rostopic pub /cmd_vel geometry_msgs/Twist "linear:
  x: Vx
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: Wz“
```

Try with different values, example:

Vx=0.0 or Vx=1.0 or Vx=-1.0 or

Wx=0.0 or Wx=1.0 or Wx=-1.0 or

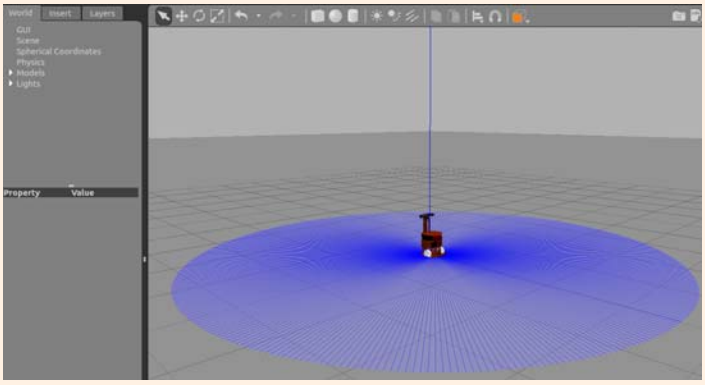
- When you have tested the robot in gazebo can move, execute the python script

```
roslaunch helloworld_pkg avoid_obstacle.py
```

ROS

28

avoid obstacle: veggie in Gazebo

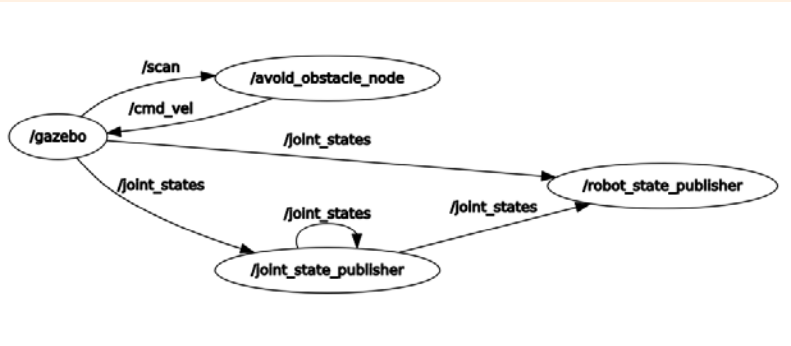


ROS

29

avoid obstacle: ROS graph

- Use rpt_graph for displaying the running nodes
- Or



ROS

30