



Lecture 4

2024

INFO 802

Master Advanced Mechatronics

Luc Marechal



ROS

**Robot control
Turtlesim**

Course 4 : Robot control

Outline

- Turtlesim topic, messages and commands
- Move Turtle
- Gazebo TurtleBot simulation

Course 4 : Robot control

Objectives

- Know which topics are at stake in a node
- Know what type of message is at stake and what is the source package
- Know how to use Twist, Pose, Odometry messages
- Write a publisher function
- Write a subscriber function and understand its callback

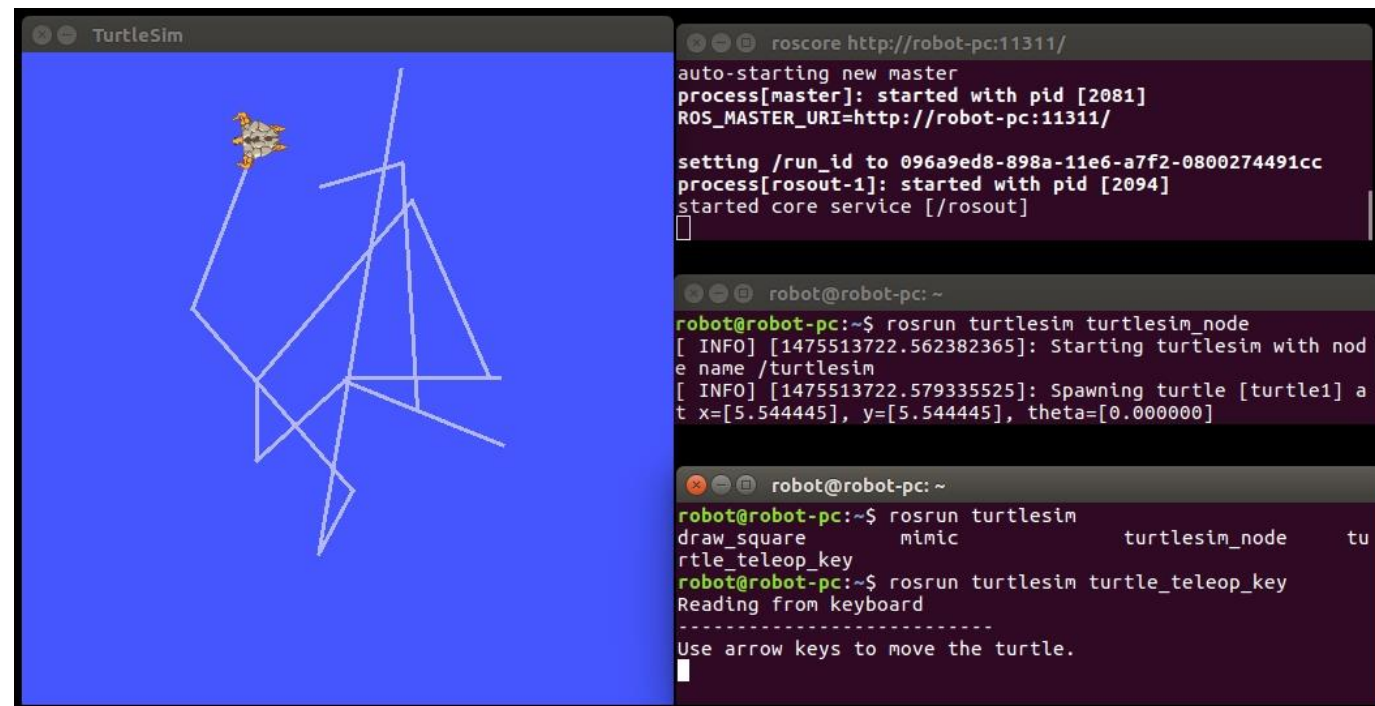
Turtlesim

Recall: Open a terminal for each command

```
> roscore
```

```
> rosrunc turtlesim turtlesim_node
```

```
> rosrunc turtlesim turtle_teleop_key
```



Turtlesim

- Questions to answer:

Which topic is the velocity command published to?

Which topic is the position information available from?

What kind of messages are used?

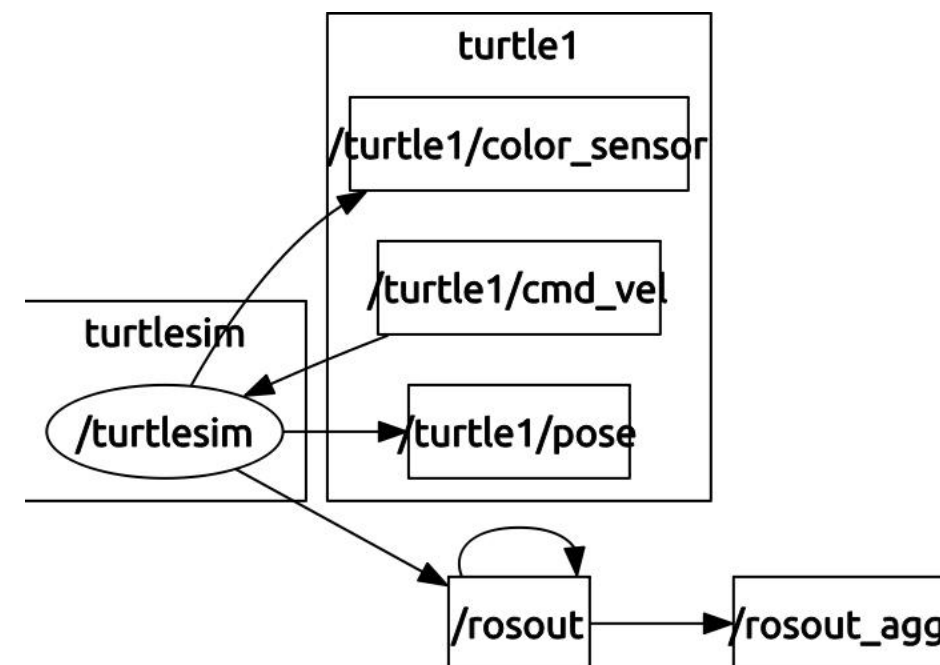
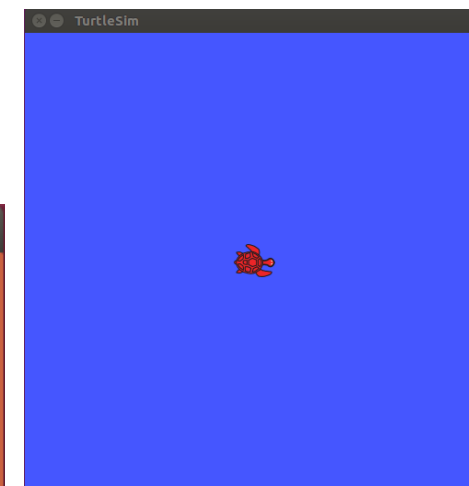
Which message packages are they from?

```
> rosrn turtlesim turtlesim_node  
> rostopic list  
> rostopic type [topic]  
> rosnode info turtlesim_node
```

Visualize node and topic

```
> rqt_graph
```

```
luc@USMB: ~  
luc@USMB:~$ rostopic list  
/rosout  
/rosout_agg  
/turtle1/cmd_vel  
/turtle1/color_sensor  
/turtle1/pose  
luc@USMB:~$
```



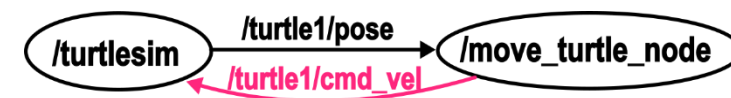
Turtlesim

Twist

- To make a turtle move in ROS we need to **publish**:

Twist messages to the topic **/turtle1/cmd_vel**

- This message has:
 - a linear component for the (x,y,z) velocities,
 - an angular component for the angular rate about the (x,y,z) axes
- Twist is part of *geometry_msgs* message package
(don't forget to add `import geometry_msgs.msg` in your code header)



```
> rostopic type /turtle1/cmd_vel  
> rosmmsg show Twist
```

```
luc@USMB:~$ rosmmsg show Twist  
[geometry_msgs/Twist]:  
geometry_msgs/Vector3 linear  
float64 x  
float64 y  
float64 z  
geometry_msgs/Vector3 angular  
float64 x  
float64 y  
float64 z
```

Example of use

```
create a Twist object — vel = Twist()  
set the linear velocity along x — vel.linear.x = 1.0  
set the angular rate about z — vel.angular.z = 0.4
```

Turtlesim

Pose

- To get a turtle position and orientation in ROS we need to **subscribe**:

to the topic **/turtle1/Pose** and read **Pose** message

- This message has:
 - a linear component for the (x,y) 2D coordinates,
 - an angular component theta about the z axes
- Pose is among others part of *turtlesim* message package
(don't forget to add `import turtlesim.msg` in your code header)



```
> rostopic type /turtle1/Pose  
> rosmmsg show Pose
```

```
luc@USMB:~$ rosmmsg show Pose  
[turtlesim/Pose]:  
float64 x  
float64 y  
float64 theta  
float64 linear_velocity  
float64 angular_velocity
```

Example of use

```
create a Pose object — pose = Pose()  
get the x position of turtle — robot_x = pose.x  
get the y position of turtle — robot_y = pose.y
```

Turtlesim

Twist / Pose



Test moving the turtle
(command from the Terminal)

```
> rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist -- '[2.0, 0.0, 0.0]' '[0.0, 0.0, 1.8]'
```

topic

message

linear.x linear.y linear.z angular.x angular.y angular.z



Test printing out the turtle position
(command from the Terminal)

```
> rostopic echo /turtle1/pose
```

```
luc@USMB: ~  
luc@USMB:~$ rostopic echo /turtle1/pose  
x: 6.33754253387  
y: 6.65641355515  
theta: 2.55362939835  
linear_velocity: 0.0  
angular_velocity: 0.0  
---  
x: 6.33754253387  
y: 6.65641355515  
theta: 2.55362939835  
linear_velocity: 0.0  
angular_velocity: 0.0  
---
```

move_turtle_linear_node (Python)

Writing the Node

Create package

```
> cd ~/catkin_ws/src/  
> catkin_create_pkg turtlesim_tutorials rospy std_msgs
```

Edit script

```
> cd ~/catkin_ws/src/turtlesim_tutorials  
> mkdir scripts  
> sudo code move_turtle_linear_node.py
```

Make script executable

```
> cd ~/catkin_ws/src/turtlesim_tutorials/scripts  
> sudo chmod +x move_turtle_linear_node.py
```

Make package and source environment

```
> cd ~/catkin_ws  
> catkin_make  
> source ~/catkin_ws/devel/setup.bash
```

move_turtle_linear_node.py

```
#!/usr/bin/env python3  
  
import rospy  
XXXXXXXXXXXXX # import Twist message  
  
def move_turtle():  
    # Initialize node  
    XXXXXXXXXXXXXXXX  
  
    # Create a publisher to "talk" to Turtlesim  
    pub = XXXXXXXXXXXXXXXX  
  
    # Create a Twist message and add linear x values  
    vel = Twist()  
    vel.linear.x = 1.0 # Move along the x axis only  
  
    # Save current time and set publish rate at 10 Hz  
    tStart = rospy.Time.now()  
    rate = rospy.Rate(10)  
  
    # For the next 6 seconds publish vel move commands to Turtlesim  
    while rospy.Time.now() < tStart + rospy.Duration.from_sec(6):  
        XXXXXXXXXXXXXXXX # publish velocity command to Turtlesim  
        rate.sleep()  
  
if __name__ == '__main__':  
    move_turtle()
```



move_turtle_linear_node (Python)

Writing the Node

Create package

```
> cd ~/catkin_ws/src/  
> catkin_create_pkg turtlesim_tutorials rospy
```

Edit script

```
> cd ~/catkin_ws/src/turtlesim_tutorials  
> mkdir scripts  
> sudo code move_turtle_linear_node.py
```

Make script executable

```
> cd ~/catkin_ws/src/turtlesim_tutorials/scripts  
> sudo chmod +x move_turtle_linear_node.py
```

Make package and source environment

```
> cd ~/catkin_ws  
> catkin_make  
> source ~/catkin_ws/devel/setup.bash
```

move_turtle_linear_node.py

```
#!/usr/bin/env python3  
  
import rospy  
from geometry_msgs.msg import Twist # import Twist message  
  
def move_turtle():  
    # Initialize node  
    rospy.init_node('move_turtle_linear_node', anonymous=False)  
  
    # Create a publisher to "talk" to Turtlesim  
    pub = rospy.Publisher('turtle1/cmd_vel', Twist, queue_size=1)  
  
    # Create a Twist message and add linear x values  
    vel = Twist() # Creates a Twist object  
    vel.linear.x = 1.0 # Move along the x axis only  
  
    # Save current time and set publish rate at 10 Hz  
    tStart = rospy.Time.now()  
    rate = rospy.Rate(10)  
  
    # For the next 6 seconds publish vel move commands to Turtlesim  
    while rospy.Time.now() < tStart + rospy.Duration.from_sec(6):  
        pub.publish(vel) # publish velocity command to Turtlesim  
        rate.sleep()  
  
if __name__ == '__main__':  
    move_turtle()
```



move_turtle_linear_node (Python)

Run the Node

start roscore

run the turtlesim_node

run your node !

```
luc@USMB: ~/catkin_ws
roscore http://USMB:11311/80x17

luc@USMB:~$ roscore
... logging to /home/luc/.ros/log/6feca35e-6c8a-11ea-84e5-0800270a6f6f/roslaunch-USMB-5814.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://USMB:43511/
ros_comm version 1.14.3

SUMMARY
=====

PARAMETERS
* /rostdistro: melodic
* /rosversion: 1.14.3

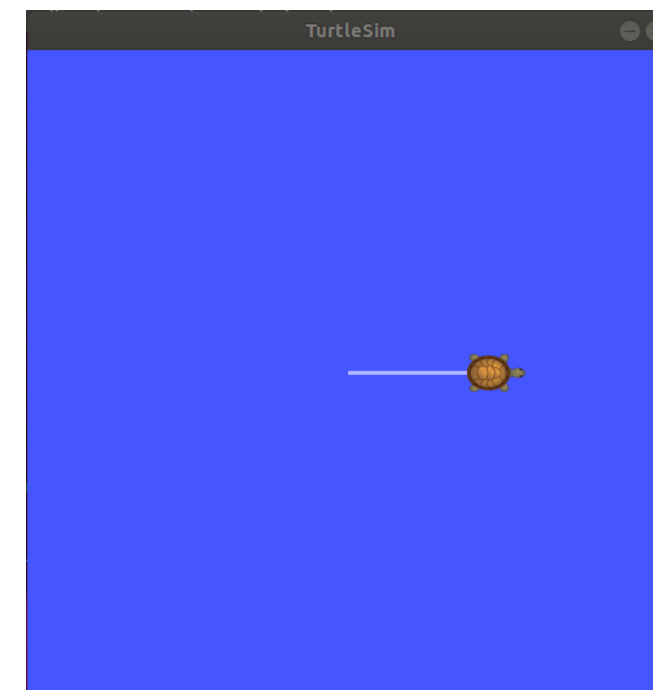
luc@USMB: ~ 80x11

luc@USMB:~$ rosrn turtlesim turtlesim_node
[ INFO] [1584915251.651160880]: Starting turtlesim with node name /turtlesim
[ INFO] [1584915251.654312984]: Spawning turtle [turtle1] at x=[5,544445], y=[5,544445], theta=[0,000000]
[ WARN] [1584915522.357794677]: Oh no! I hit the wall! (Clamping from [x=11,096445, y=5,544445])
[ WARN] [1584915522.373602824]: Oh no! I hit the wall! (Clamping from [x=11,104889, y=5,544445])
[ WARN] [1584915522.388809895]: Oh no! I hit the wall! (Clamping from [x=11,104889, y=5,544445])
[ WARN] [1584915522.405520633]: Oh no! I hit the wall! (Clamping from [x=11,104889, y=5,544445])

luc@USMB: ~/catkin_ws 80x9

luc@USMB:~/catkin_ws$ rosrn turtlesim_tutorials move_turtle_linear_node.py
```

move_turtle_linear_node.py



move_turtle_command_node (Python)

Adding command line arguments

move_turtle_command_node.py

```
#!/usr/bin/env python3

import rospy
from geometry_msgs.msg import Twist

# Handling command line arguments
import sys # Python sys module to get the command-line arguments
           # inside our code

def move_turtle_command(lin_vel, ang_vel):
    rospy.init_node('move_turtle_command', anonymous=False)

    pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)
    rate = rospy.Rate(10) # 10hz

    vel = Twist() # creates a Twist object

    while not rospy.is_shutdown():

        # Adding linear and angular velocity to the message
        vel.linear.x = lin_vel
        vel.linear.y = 0
        vel.linear.z = 0

        vel.angular.x = 0
        vel.angular.y = 0
        vel.angular.z = ang_vel
```

Display information in the Console

```
rospy.loginfo("Linear Vel = %f: Angular Vel =%f",lin_vel,ang_vel)

#Publishing Twist message
pub.publish(vel)

rate.sleep()

if __name__ == '__main__':
    #Providing linear and angular velocity through command line
    move_turtle_command(float(sys.argv[1]),float(sys.argv[2]))
```

Run the node

```
> rosrun turtlesim_tutorials move_turtle_command_node.py 0.5 0.2
```

arguments
linear.x angular.z



move_turtle_printout_node (Python)

Adding the turtle position print out

move_turtle_printout_node.py

```
#!/usr/bin/env python3

import rospy
XXXXXXXXXXXX # import Twist message
XXXXXXXXXXXX # import Pose message

import sys

# callback for topic /turtle1/Pose
def pose_callback(XXXXX):
    XXXXXXXX # printout in the console the pose of turtle1

def move_turtle_printout(lin_vel,ang_vel):
    rospy.init_node('move_turtle_printout', anonymous=False)

    pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)

    # Creating new subscriber. Topic name: /turtle1/pose
    #                               Callback name: pose_callback
    XXXXXXXXXXXXXXXXXXXX

    rate = rospy.Rate(10) # 10hz

    vel = Twist()
```

```
while not rospy.is_shutdown():
    vel.linear.x = lin_vel
    vel.linear.y = 0
    vel.linear.z = 0

    vel.angular.x = 0
    vel.angular.y = 0
    vel.angular.z = ang_vel

    rospy.loginfo("Linear Vel = %f: Angular Vel
    =%f",lin_vel,ang_vel)

    pub.publish(vel)

    rate.sleep()

if __name__ == '__main__':
    # Providing linear and angular velocity through command line
    move_turtle_printout(float(sys.argv[1]),float(sys.argv[2]))
```



move_turtle_printout_node (Python)

Adding the turtle position print out

move_turtle_printout_node.py

```
#!/usr/bin/env python3

import rospy
from geometry_msgs.msg import Twist      # import Twist message
from turtlesim.msg import Pose           # import Pose message

import sys

# callback for topic /turtle1/Pose
def pose_callback(pose):
    rospy.loginfo("Robot X = %f : Y=%f : Z=%f\n",pose.x,pose.y,pose.theta)

def move_turtle_printout(lin_vel,ang_vel):
    rospy.init_node('move_turtle_printout', anonymous=False)

    pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)

    # Creating new subscriber. Topic name: /turtle1/pose
    #                               Callback name: pose_callback
    rospy.Subscriber('/turtle1/pose', Pose, pose_callback)

    rate = rospy.Rate(10) # 10hz

    vel = Twist()
```

```
while not rospy.is_shutdown():
    vel.linear.x = lin_vel
    vel.linear.y = 0
    vel.linear.z = 0

    vel.angular.x = 0
    vel.angular.y = 0
    vel.angular.z = ang_vel

    rospy.loginfo("Linear Vel = %f: Angular Vel
    =%f",lin_vel,ang_vel)

    pub.publish(vel)

    rate.sleep()

if __name__ == '__main__':
    # Providing linear and angular velocity through command line
    move_turtle_printout(float(sys.argv[1]),float(sys.argv[2]))
```



move_turtle_feedback_node (Python)

Adding the position feedback

move_turtle_feedback_node.py

```
#!/usr/bin/env python3

import rospy
from geometry_msgs.msg import Twist
from turtlesim.msg import Pose

import sys

robot_x = 0

# callback for topic /turtle1/Pose
def pose_callback(pose):
    global robot_x
    rospy.loginfo("Robot X = %f\n", pose.x)
    robot_x = pose.x

def move_turtle(lin_vel, ang_vel, distance):

    global robot_x

    rospy.init_node('move_turtle', anonymous=False)
    pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size=10)
    rospy.Subscriber('/turtle1/pose', Pose, pose_callback)
    rate = rospy.Rate(10) # 10hz
    vel = Twist()
```

```
while not rospy.is_shutdown():

    vel.linear.x = lin_vel
    vel.linear.y = 0
    vel.linear.z = 0
    vel.angular.x = 0
    vel.angular.y = 0
    vel.angular.z = ang_vel

    # Checking the robot distance is greater than the commanded distance
    # If it is greater, stop the node
    if(robot_x >= distance):
        rospy.loginfo("Robot Reached destination")
        rospy.logwarn("Stopping robot")

        break

    pub.publish(vel)
    rate.sleep()

if __name__ == '__main__':
    #Providing linear and angular velocity through command line
    move_turtle(float(sys.argv[1]), float(sys.argv[2]),
float(sys.argv[3]))
```



Further References

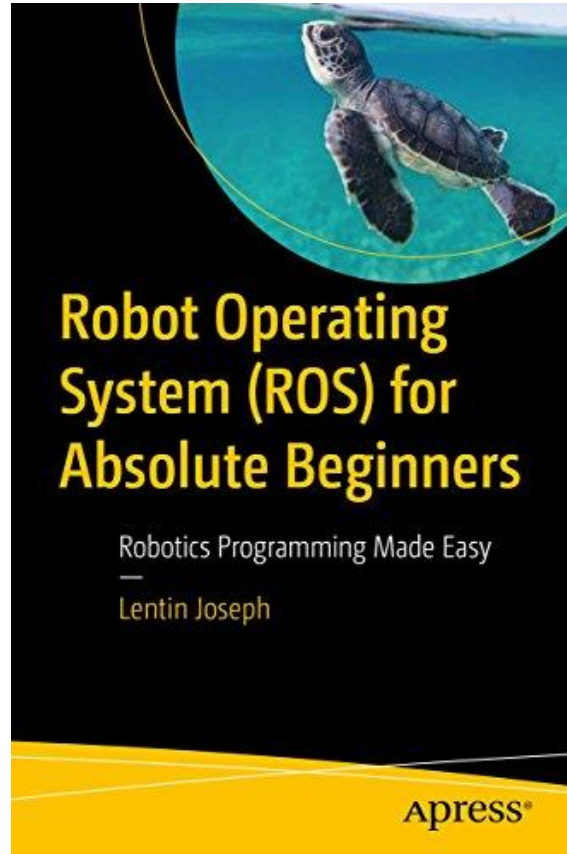
- **ROS Turtlesim tutorials**

- wiki.ros.org/turtlesim/Tutorials/

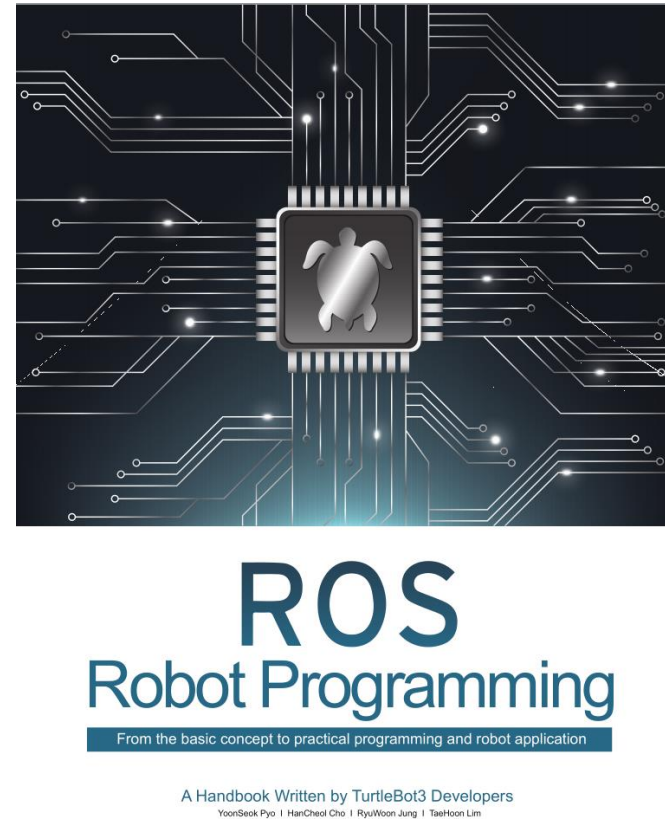
- **ROS Cheat Sheet**

- <https://www.clearpathrobotics.com/ros-robot-operating-system-cheat-sheet/>
- https://kapeli.com/cheat_sheets/ROS.docset/

Relevant books and sources



Chapter 5



Chapter 10

Contact Information

Université Savoie Mont Blanc

Polytech' Annecy Chambéry
Chemin de Bellevue
74940 Annecy
France

<https://www.polytech.univ-savoie.fr>

Lecturer



Luc Marechal (luc.marechal@univ-smb.fr)



SYMME Lab (Systems and Materials for Mechatronics)

SYMME