無人機智慧系統開發與實作
System Development and Implementation of Drone Intelligence

ROS

http://www.ros.org/

ROS

Robot Operating System(ROS)

用於開發機器人軟體的靈活框架(包含一系列的工具、套件和協定)。

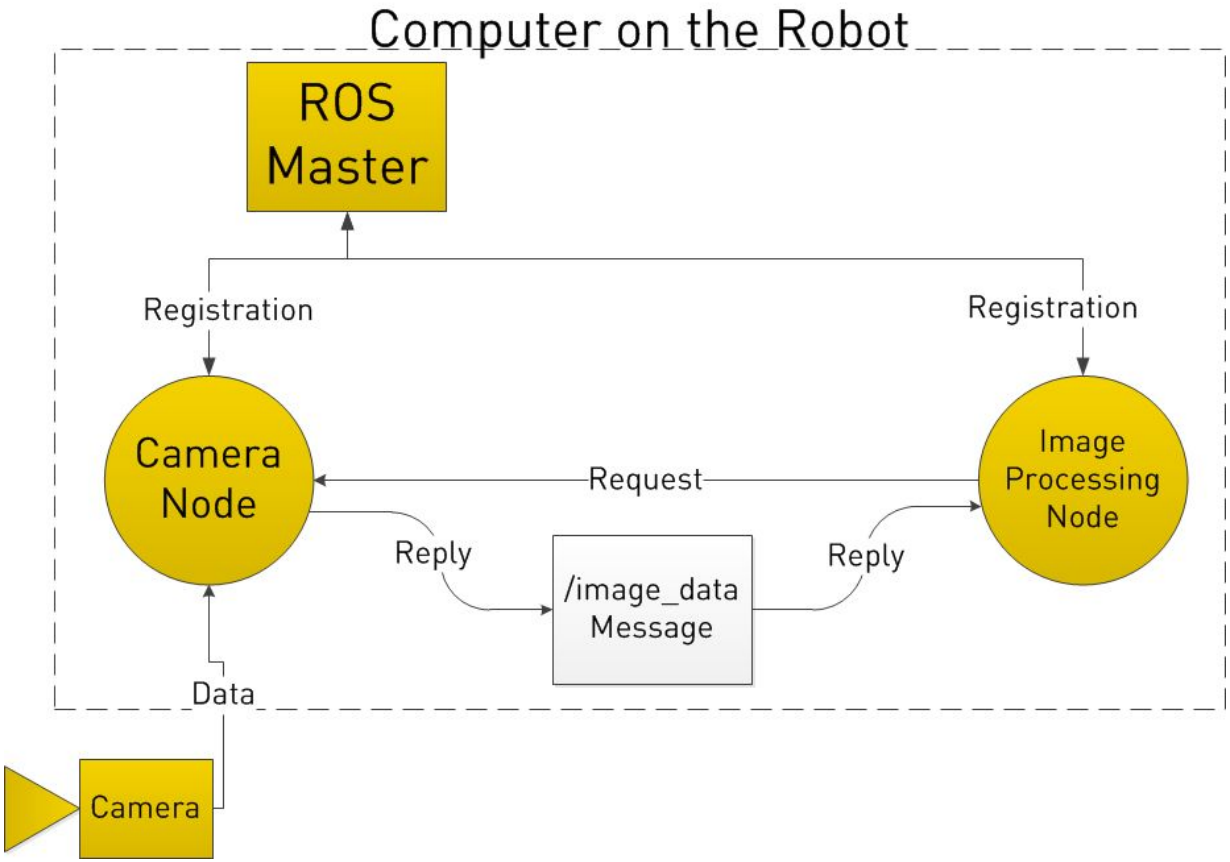
目的: 開發穩定且通用的機器人軟體是一件困難的事情，而ROS主要希望簡化此一開發過程，讓各個機器人軟體可以快速整合。

例子: 有三間實驗室的專家希望共同開發機器人，第一間實驗室有繪製室內地圖的專家，第二間實驗室有專家使用地圖進行導航，第三間實驗室有視覺處理專家，可以識別室內雜亂的小物體。透過ROS，可以快速地讓這三間實驗室整合他們開發的成果於機器人上，並進行測試。

ROS concept

Nodes
Messages
Topics
Master

rosout
roscore



http://wiki.ros.org/ROS/Tutorials

Install(linux)

version: ROS Kinetic **ONLY** supports Wily (Ubuntu 15.10), Xenial (Ubuntu 16.04) and Jessie (Debian 8) for debian packages.

```
1.sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list'
2. sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key
421C365BD9FF1F717815A3895523BAEEB01FA116
3. sudo apt-get update
4. sudo apt-get install ros-kinetic-desktop-full

5. sudo rosdep init
6. rosdep update
7. echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
8. source ~/.bashrc
9. sudo apt install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

http://wiki.ros.org/kinetic/Installation/Ubuntu

Install(mac & windows)

max > OS X (Homebrew)

https://fitsir.me/2017/02/14/build-ros-kinetic-on-osx/

windows (Windows 10 (version 1703) or newer)

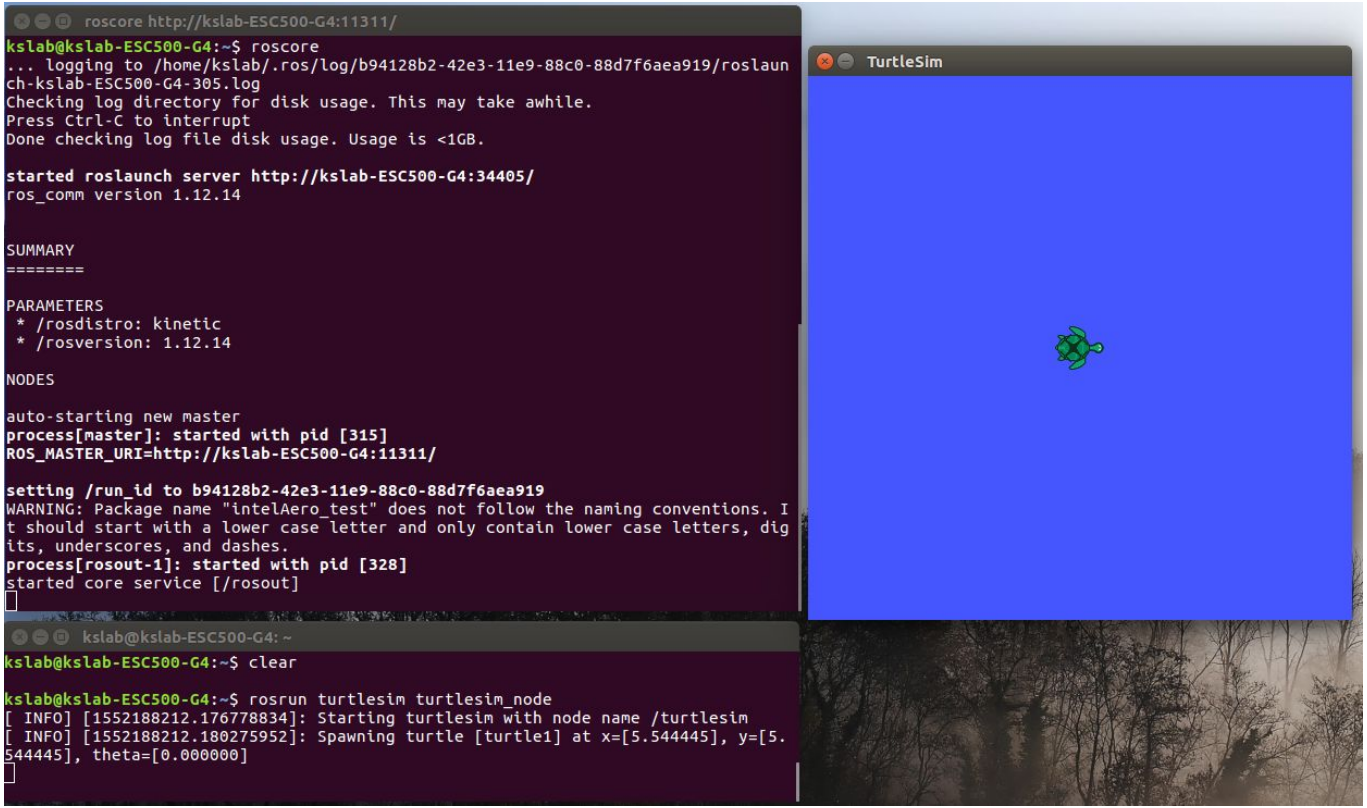
http://wiki.ros.org/Installation/Windows

https://janbernloehr.de/2017/06/10/ros-windows

Turtlesim

```
sudo apt-get install
ros-$(rosversion
-d)-turtlesim
```

```
roscore
rosrun turtlesim
turtlesim_node
```



Turtlesim

`rosnode list`

```
kslab@kslab-ESC500-G4:~$ rosnode list
/rosout
/turtlesim
```

`rostopic list`

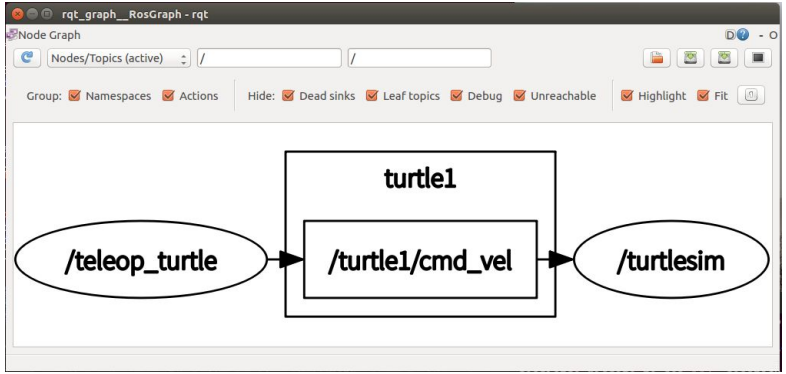
```
kslab@kslab-ESC500-G4:~$ rostopic list
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

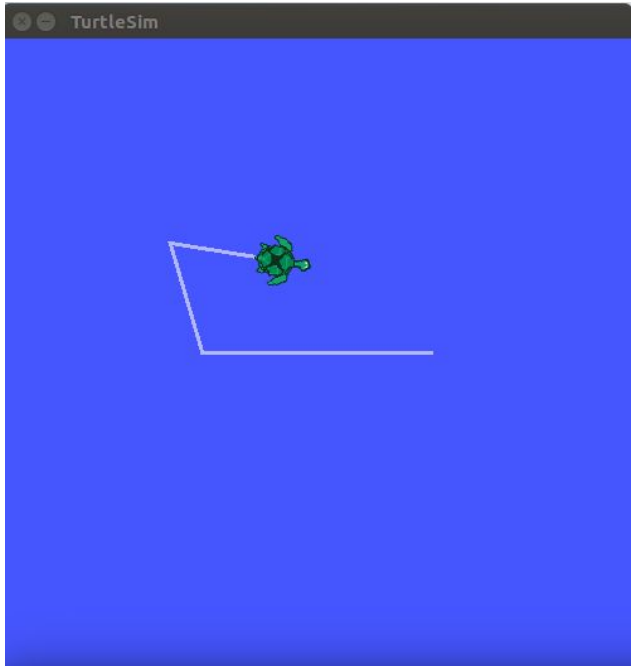
Turtle keyboard teleoperation

```
rosrun turtlesim turtle_teleop_key
```

```
kslab@kslab-ESC500-G4:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle.
```

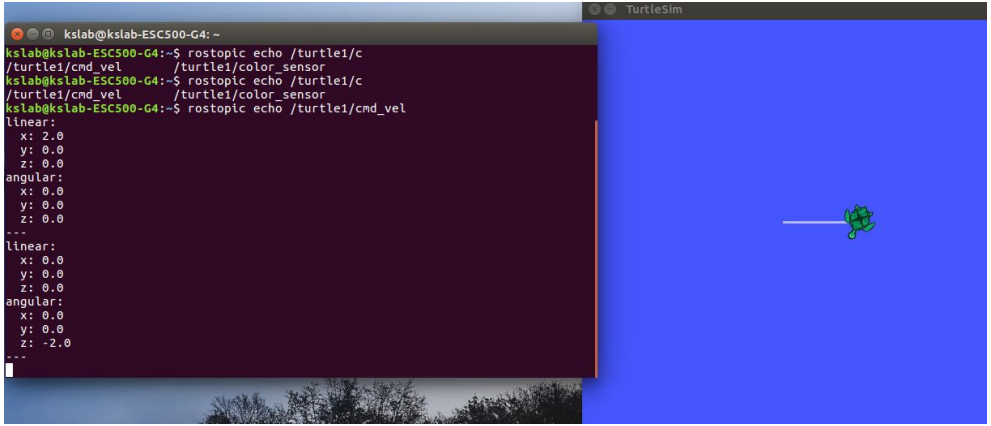
```
rosrun rqt_graph rqt_graph
```

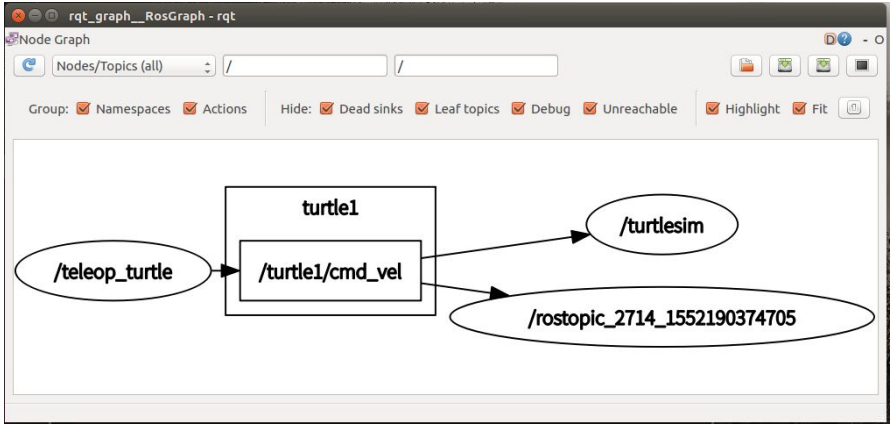




Turtlesim

```
rostopic echo /turtle1/cmd_vel
```





Turtlesim

```
rostopic type /turtle1/cmd_vel
```

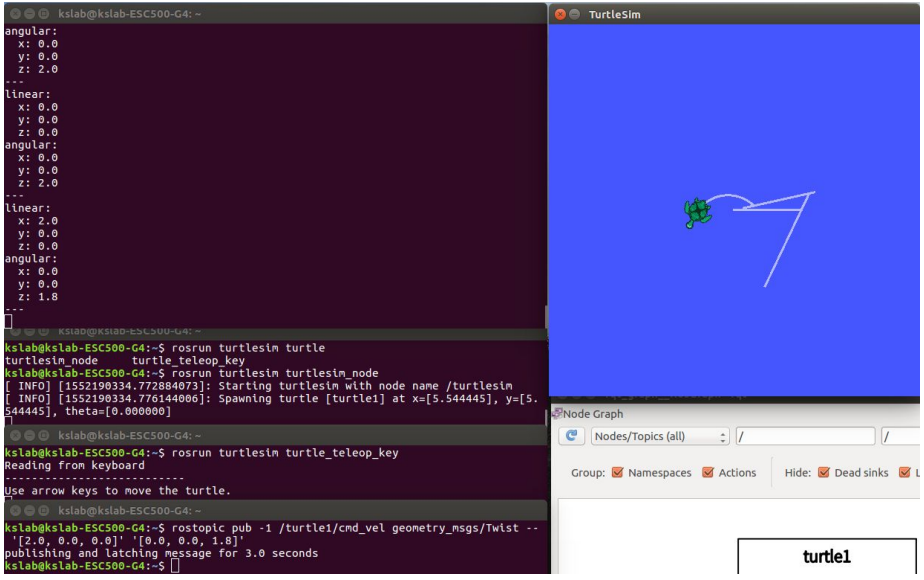
```
kslab@kslab-ESC500-G4:~$ rostopic type /turtle1/cmd_vel
geometry_msgs/Twist
```

```
rosmsg show geometry_msgs/Twist
```

```
kslab@kslab-ESC500-G4:~$ rosmsg show geometry_msgs/Twist
WARNING: Package name "intelAero_test" does not follow the naming conventions. I
t should start with a lower case letter and only contain lower case letters, dig
its, underscores, and dashes.
geometry_msgs/Vector3 linear
  float64 x
  float64 y
  float64 z
geometry_msgs/Vector3 angular
  float64 x
  float64 y
  float64 z
```

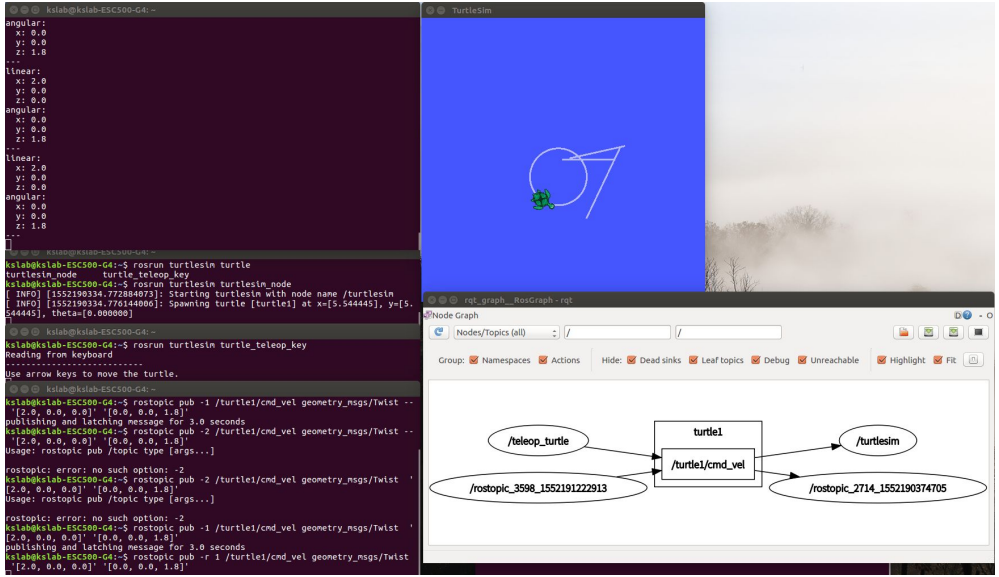
Turtlesim

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



Turtlesim

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



Turtlesim with rospy: your first ros programming

1. build your workspace

```
sudo apt-get install build-essential python-rosdep python-catkin-tools
mkdir -p ~/catkin_ws/src && cd ~/catkin_ws/src
catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
cd ~/catkin_ws/
catkin_make
. ~/catkin_ws/devel/setup.bash
cd ~/catkin_ws/src/beginner_tutorials
mkdir scripts
cd scripts
```

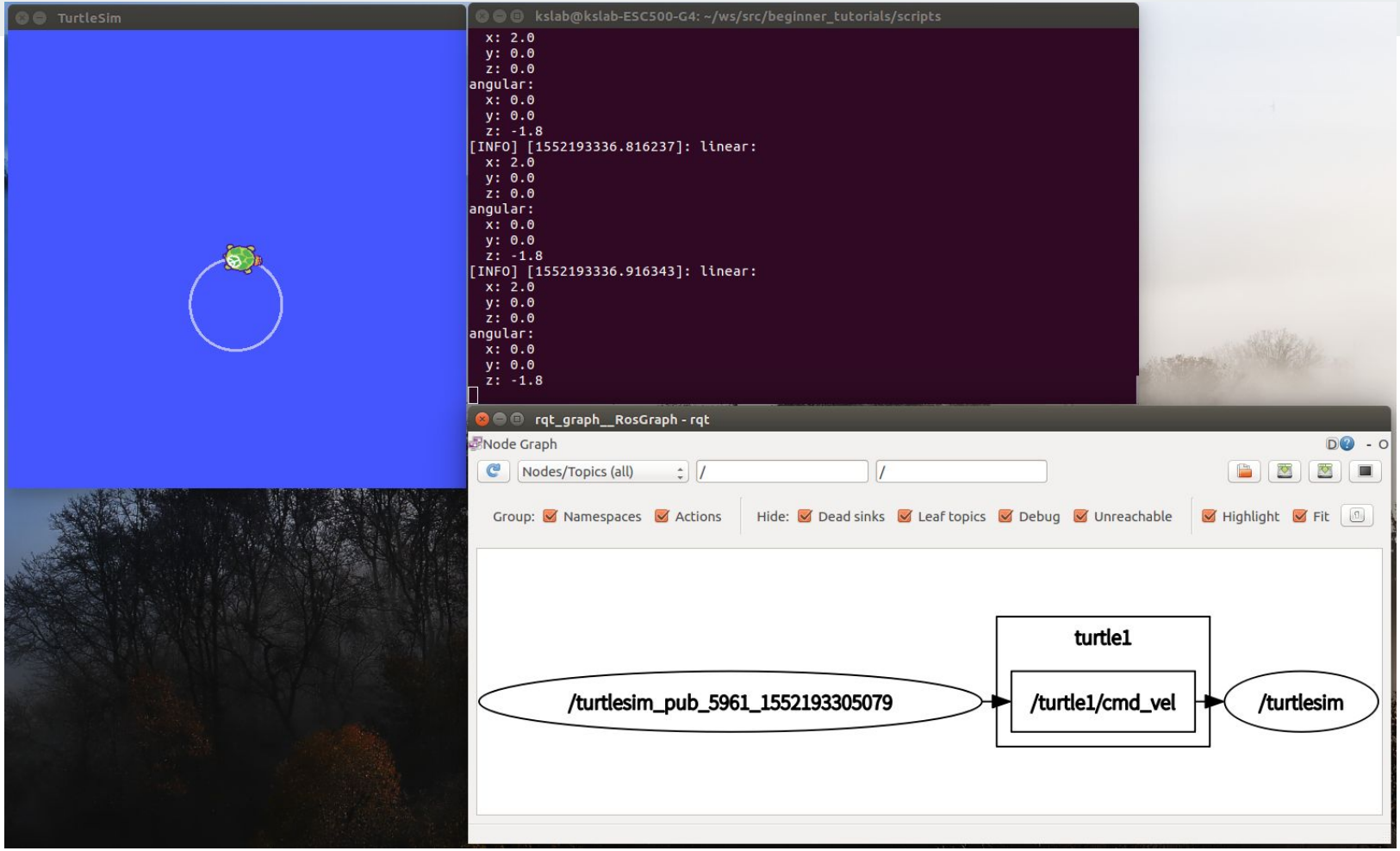
Turtlesim with rospy: your first ros programming

2. create turtlesim_pub.py

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

def turtle_pub():
  #The queue_size argument is New in ROS hydro and limits the amount of queued messages if any subscriber is not receiving them fast enough
  pub = rospy.Publisher('/turtle1/cmd_vel', Twist, queue_size = 10)
  rospy.init_node('turtlesim_pub', anonymous=True)
  rate = rospy.Rate(10)
  while not rospy.is_shutdown():
    msg = Twist()
    msg.linear.x = 2.0
    msg.angular.z = -1.8
    rospy.loginfo(msg)
    pub.publish(msg)
    rate.sleep()

if __name__ == '__main__':
  try:
    turtle_pub()
  except rospy.ROSInterruptException:
    pass
```


TurtleSim
kslab@kslab-ESC500-G4: ~/ws/src/beginner_tutorials/scripts
x: 2.0
y: 0.0
z: 0.0
angular:
x: 0.0
y: 0.0
z: -1.8
[INFO] [1552193336.816237]: linear:
x: 2.0
y: 0.0
z: 0.0
angular:
x: 0.0
y: 0.0
z: -1.8
[INFO] [1552193336.916343]: linear:
x: 2.0
y: 0.0
z: 0.0
angular:
x: 0.0
y: 0.0
z: -1.8
rqt_graph__RosGraph - rqt
Node Graph
Nodes/Topics (all)
Group: Namespaces Actions
Hide: Dead sinks Leaf topics Debug Unreachable
Highlight Fit
turtle1
/turtlesim_pub_5961_1552193305079
/turtle1/cmd_vel
/turtlesim

Turtlesim with rospy: your first ros programming

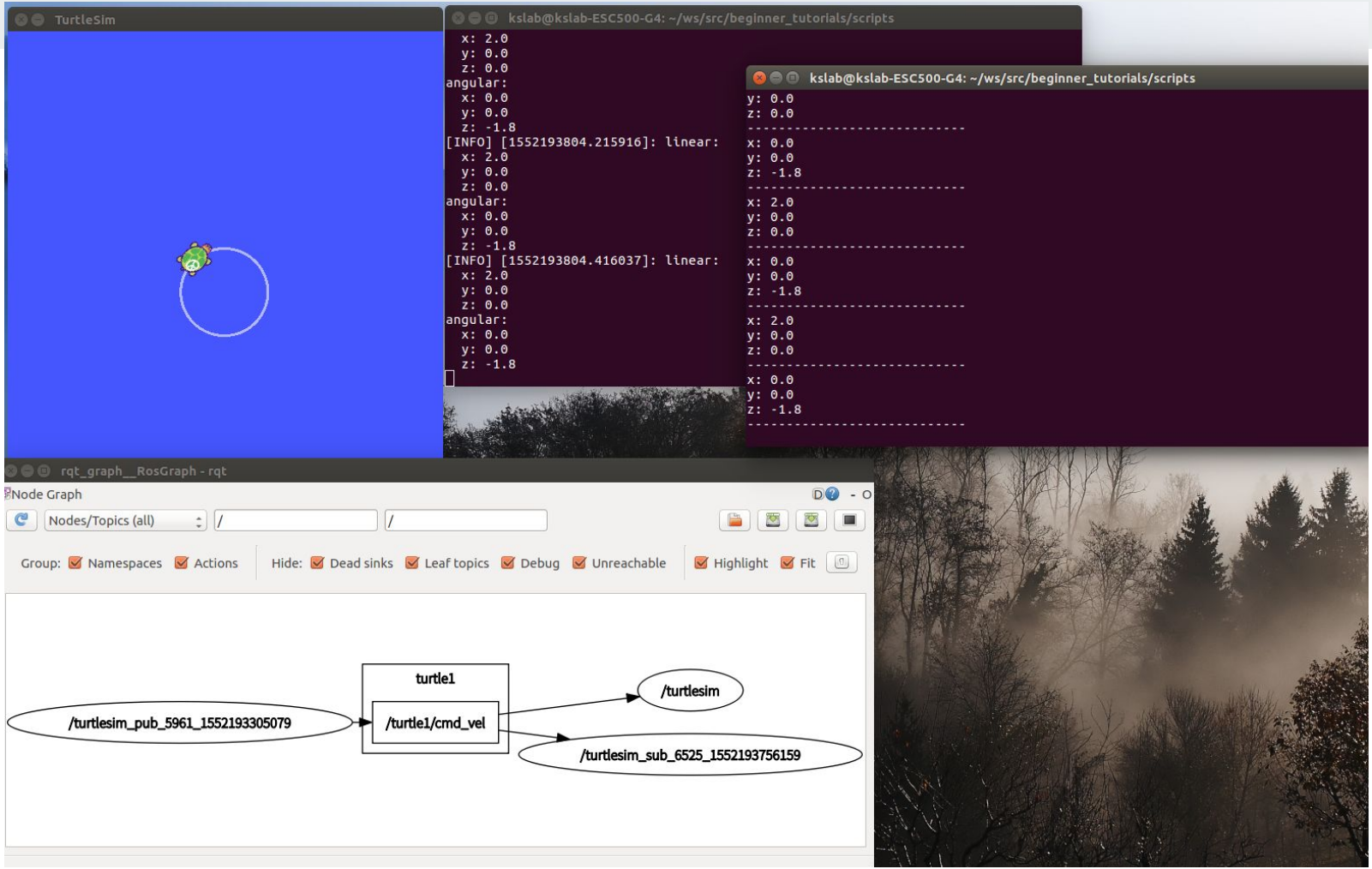
3. create turtlesim_sub.py

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

def callback(data):
  print(data.linear)
  print("----------------------------")
  print(data.angular)
  print("----------------------------")

def turtle_sub():
  rospy.init_node('turtlesim_sub', anonymous=True)
  rospy.Subscriber("/turtle1/cmd_vel", Twist, callback)
  # spin() simply keeps python from exiting until this node is stopped
  rospy.spin()

if __name__ == '__main__':
  turtle_sub()
```


TurtleSim
kslab@kslab-ESC500-G4: ~/ws/src/beginner_tutorials/scripts
x: 2.0
y: 0.0
z: 0.0
angular:
x: 0.0
y: 0.0
z: -1.8
[INFO] [1552193804.215916]: linear:
x: 2.0
y: 0.0
z: 0.0
angular:
x: 0.0
y: 0.0
z: -1.8
[INFO] [1552193804.416037]: linear:
x: 2.0
y: 0.0
z: 0.0
angular:
x: 0.0
y: 0.0
z: -1.8
kslab@kslab-ESC500-G4: ~/ws/src/beginner_tutorials/scripts
y: 0.0
z: 0.0
---
x: 0.0
y: 0.0
z: -1.8
---
x: 2.0
y: 0.0
z: 0.0
---
x: 0.0
y: 0.0
z: -1.8
---
x: 2.0
y: 0.0
z: 0.0
---
x: 0.0
y: 0.0
z: -1.8
---
rqt_graph__RosGraph - rqt
Node Graph
Nodes/Topics (all)
/
/
Group: Namespaces Actions
Hide: Dead sinks Leaf topics Debug Unreachable
Highlight Fit
turtle1
/turtle1/cmd_vel
/turtlesim_pub_5961_1552193305079
/turtlesim
/turtlesim_sub_6525_1552193756159