



# Robotique

EL 3007










**R. HAMOUCHE**

[r.hamouche@esiee.fr](mailto:r.hamouche@esiee.fr)

2023-2024

Présentation ESIEE

1



## Motivations

Des systèmes robotiques de plus en plus évolués

- Différentes applications et domaines variés
  - Humanoïde, mobilité et transport, industrie, traitement d'image, IA...
- Compétences interdisciplinaires
  - Mécanique, électrotechnique, contrôle, vision par ordinateur, informatique temps réel embarqué, réseau, traitement de signal...

WHAT ROBOTICS IS ABOUT?

<p>● <b>Robot making</b></p> <ul style="list-style-type: none"> <li>+ Design</li> <li>+ Mechatronics</li> <li>+ Materials</li> <li>+ Sensors</li> <li>+ Actuators</li> <li>+ ...</li> </ul>	<p>● <b>Robot control</b></p> <ul style="list-style-type: none"> <li>+ Cybernetics</li> <li>+ Control theory</li> <li>+ Automation</li> <li>+ Artificial Intelligence</li> <li>+ Knowledge representation</li> <li>+ Probabilistic reasoning</li> <li>+ Estimation theory</li> <li>+ Decision making</li> <li>+ Self-organization</li> </ul>	<p>● <b>Software, interfaces, testing</b></p> <ul style="list-style-type: none"> <li>+ Device drivers</li> <li>+ Inter-process communications</li> <li>+ Multi-process arbitration</li> <li>+ Interfaces for status access</li> <li>+ Status / data logging</li> <li>+ Simulation of robot(s) and environment physics</li> <li>+ Benchmarks, data analysis</li> <li>+ Safety and security</li> <li>+ Dependability</li> <li>+ ...</li> </ul>
<p>● <b>Actuation</b></p> <ul style="list-style-type: none"> <li>+ Wheels</li> <li>+ Arms</li> <li>+ Hands</li> <li>+ Joints</li> <li>+ Legs</li> <li>+ Heads</li> <li>+ Rotors, blades</li> <li>+ Motors</li> <li>+ ...</li> </ul>	<p>● <b>Sensing and perception</b></p> <ul style="list-style-type: none"> <li>+ Machine vision</li> <li>+ Speech recognition/generation</li> <li>+ Range finders</li> <li>+ Dead reckoning</li> <li>+ Localization techniques</li> <li>+ Mapping</li> <li>+ ...</li> </ul>	

29

2



# Motivations

## Complexité des systèmes robotiques

- Différentes applications et domaines variés
  - Humanoïde, mobilité et transport, industrie, traitement d'image, IA...
- Compétences interdisciplinaires
  - Mécanique, électrotechnique, contrôle, vision par ordinateur, informatique temps réel embarqué, réseau, traitement de signal...
- Besoins de réutiliser au maximum les développements et les outils existants et de pouvoir faire collaborer efficacement des équipes



Robot Operating System

ESIEE PARIS



3



# Qu'est ce que ROS ?

## Un projet collaboratif robotique à grande échelle qui fournit

- Un écosystème logiciel open source permettant de « gagner du temps » dans la mise au point d'un système robotique.
- Un ensemble de bibliothèques et outils logiciels permettant de créer des applications robotiques qui fonctionnent sur une large variété de plates-formes robotiques
- Historique
  - Développé à l'origine au début des années 2000 au Laboratoire d'intelligence artificielle de Stanford et le développement se poursuit ensuite à Willow Garage
  - Géré depuis 2013 par OSRF (Open Source Robotics Foundation)
- Norme de facto pour la programmation de robots
- Aujourd'hui, ROS utilisé par de nombreux robots, universités et entreprises.

ESIEE PARIS

4

## Les distributions ROS

Distro	Release date	Poster	Tuturtle, turtle in tutorial	EOL date
ROS Noetic Ninjemys (Recommended)	May 23rd, 2020			May, 2025 (Focal EOL)
ROS Melodic Morenia	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015			May, 2017
ROS Indigo Igloo	July 22nd, 2014			April, 2019 (Trusty EOL)
ROS Hydro Medusa	September 4th, 2013			May, 2015

ROS Groovy Galapagos	December 31, 2012			July, 2014
ROS Fuerte Turtle	April 23, 2012			...
ROS Electric Eels	August 30, 2011			...
ROS Diamondback	March 2, 2011			...
ROS C Turtle	August 2, 2010			...
ROS Box Turtle	March 2, 2010			...



5

## Déploiement de ROS

- ROS 1.0 , ROS 2.0 , ROS Industrial, ROS M
- ROS.org
  - Framework opensource foisonnant
  - Origine académique où chacun rajoute sa brique
- ROS industrial
  - Faciliter le respect des normes et la certification des machines
  - Sous-ensemble de ROS qualitatif « industry-grade »
  - Label attribués aux packages ROS respectant des bonnes pratiques et des tests
  - Etendu mondial

6

# Déploiement de ROS

ESIEE PARIS

7

## "...ROS: powering the worlds's robots !"

Exemples :

Robot	Constructeur
<a href="#">Nao</a> <sup>27</sup>	<a href="#">Aldebaran</a>
<a href="#">Pepper</a> <sup>28</sup>	<a href="#">Aldebaran</a>
PR2 <sup>29</sup>	Willow-Garage
Husky <sup>30</sup>	Clearpath Robotics
Kingfisher <sup>31</sup>	Clearpath Robotics
Turtlebot <sup>32</sup>	Willow Garage
Turtlebot 2 <sup>32</sup>	Yujin Robotics
<a href="#">Turtlebot 3</a> <sup>32</sup>	Robotis
Shadow Hand <sup>33</sup>	Shadow Robot
<a href="#">Robonaut 2</a> <sup>34</sup>	NASA
Erle-copter <sup>35</sup>	Erle Robotics
Baxter <sup>36</sup>	Rethink Robotics
Autorally <sup>37</sup>	Georgia Tech
Stan (robot voiturier) <sup>38</sup>	Stanley Robotics


Liste encore plus longues ici : <https://robots.ros.org/>




ESIEE PARIS







8

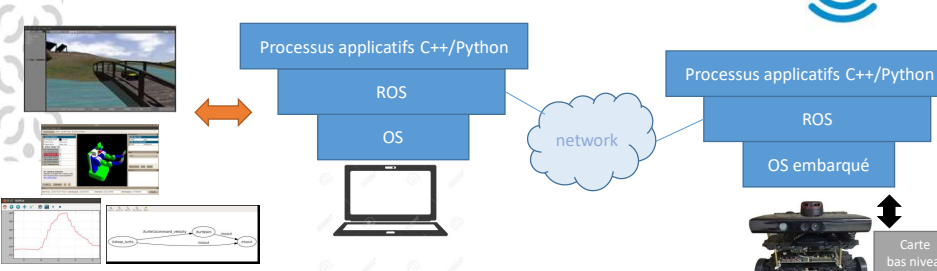
## Comment ROS est-il déployé ?

- Middleware robotique, une surcouche à l'OS
  - Linux ubuntu, windows, rasbian...
  - Ubuntu ARM, OpenEmbeddedYocto, ArchLinux...
- Framework logiciel
  - API C++ et Python, Java (+Android), C#...
  - Bibliothèques de fonctionnalités communes
  - Des outils graphiques pour le développement, le débogage et la simulation 3D

Supporté : 

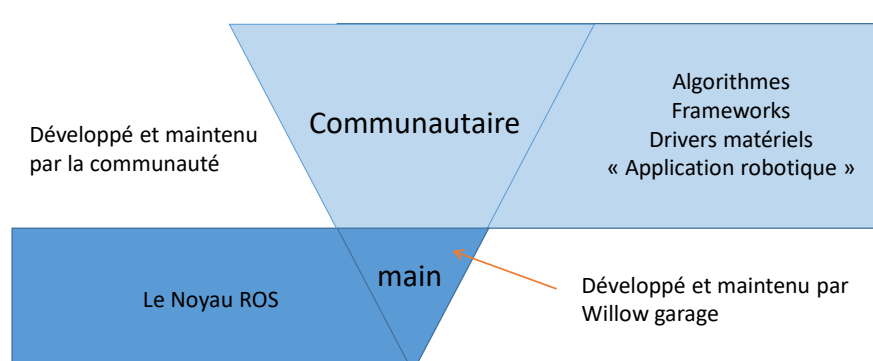
Expérimentaux :   



9

## Architecture des développements ROS



Développé et maintenu par la communauté

Communautaire

Le Noyau ROS

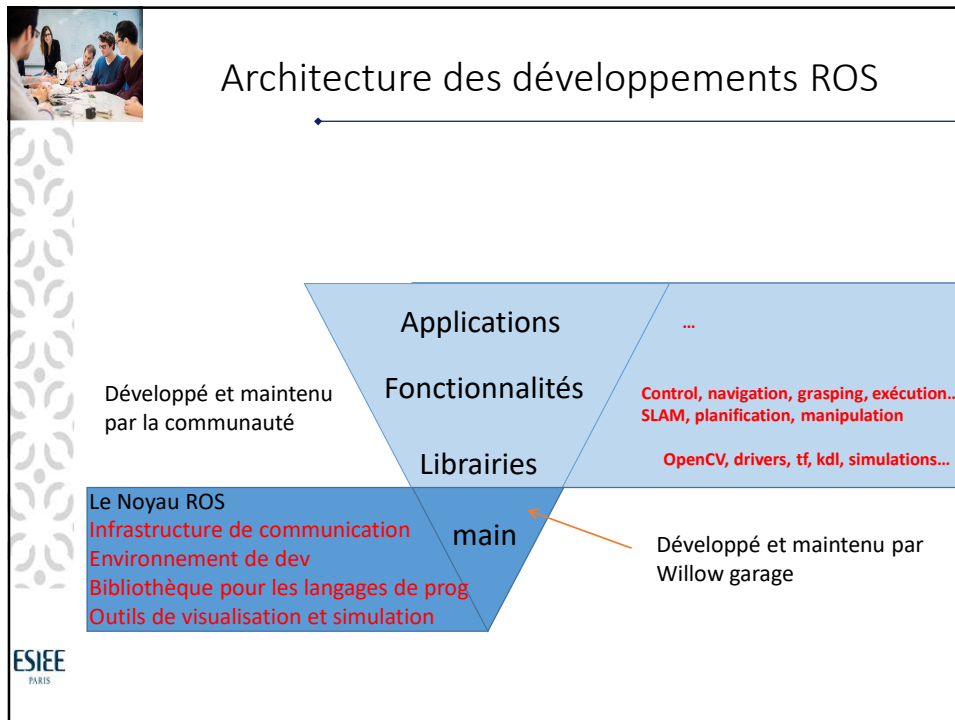
main

Algorithmes  
Frameworks  
Drivers matériels  
« Application robotique »

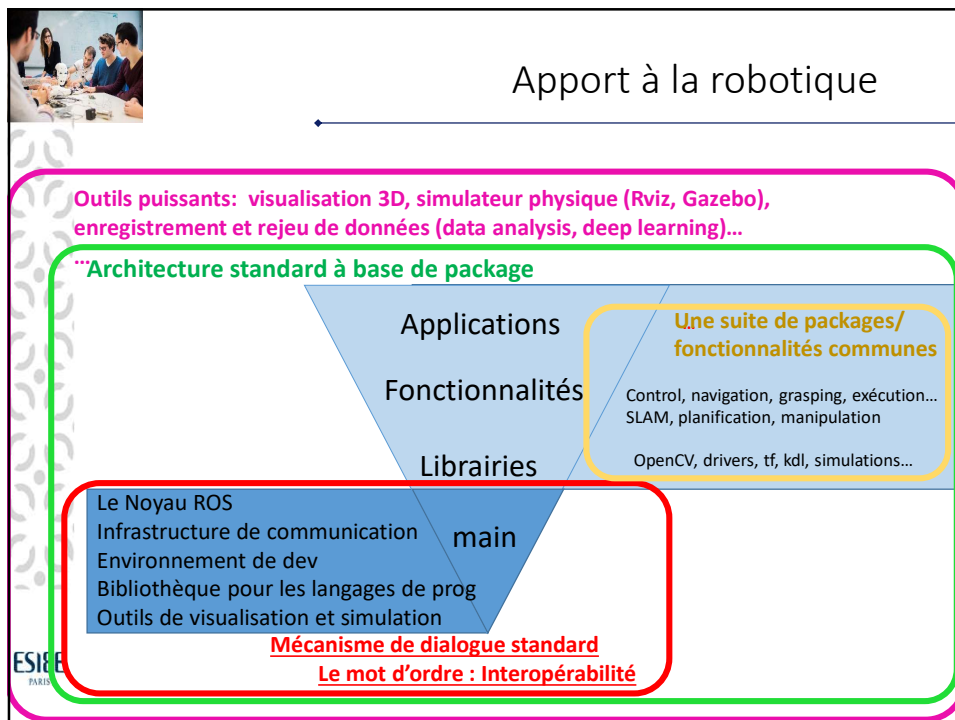
Développé et maintenu par Willow garage

ESIEE  
PARIS

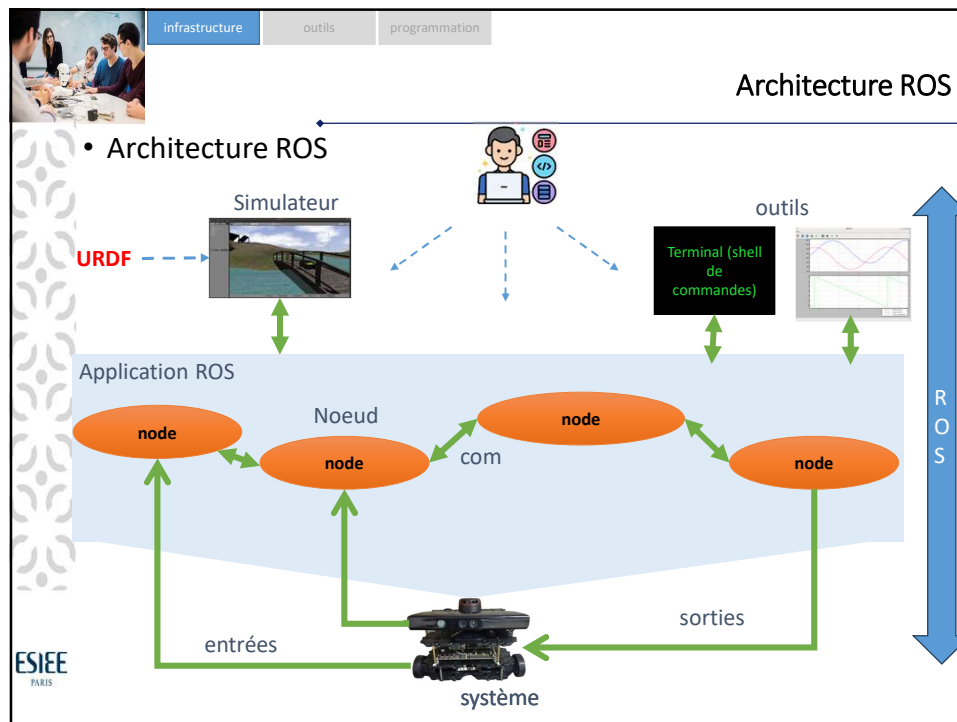
10



11



12



13

## LES NODES

ESIEE

14

14

# Un nœud ROS

- Un objet logiciel exécutant une tâche ou une opération
- Le contenu d'un nœud est développé avec un langage de programmation supporté par ROS : C++, Python, Java, Android, C#...
- Un programme possédant son propre exécutable
- Vu comme un processus dans le noyau Ros

The diagram illustrates the architecture of a ROS node. It shows a program (Prog) being executed within the ROS environment. The environment is represented by a light blue trapezoid containing five functional modules, each associated with a specific programming language or technology:

- Mesure de l'odométrie** (Odometry measurement) - C++
- Acquisition d'image** (Image acquisition) - Python
- Génération de trajectoire** (Trajectory generation) - Python
- Piloteage de moteurs** (Motor control) - Java
- Piloteage de moteurs** (Motor control) - C++

At the bottom of the diagram, a small image of a mobile robot is shown, representing the physical system that the ROS node controls.

15

infrastructure    outils    programmation

## Organisation logique des nœuds : les packages

### Architecture standard à base de package

Modularité, Réutilisabilité  
Minimisation des dépendances  
Développement multi-acteurs

The diagram illustrates a standard package-based architecture. It features a large light blue rectangular area representing the system's workspace. Within this area, there are several packages represented by blue rounded rectangles:


- Mon\_package**: A package on the left side.
- Package Nav**: A package on the right side.
- Algorithm de recherche de chemin**: An orange oval containing this text, located inside the **Package Nav**.
- Mesure de l'odométrie**: An orange oval located inside the **Mon\_package**.
- Génération de trajectoire**: An orange oval located in the center of the workspace.
- Pilotage de moteurs**: An orange oval located on the right side of the workspace.

A blue arrow points from the **Algorithm de recherche de chemin** package towards the **Génération de trajectoire** package. At the bottom center, a small image of a robotic car is shown, with a blue cone of light emanating from it, pointing upwards towards the packages, indicating its role in the system.

ESIEE  
PARIS

16





infrastructure

outils

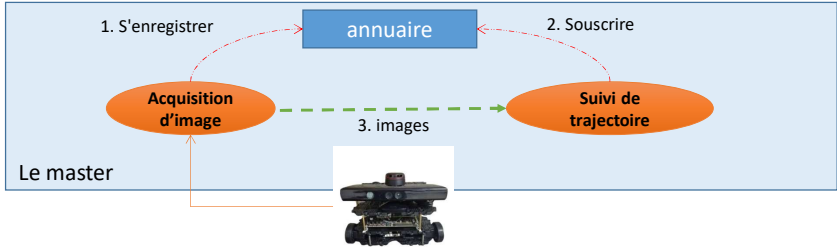
programmation

# Le Master

identification des nœuds

---

- Lors du démarrage d'un nœud, celui-ci s'identifie auprès du Master ROS
- Un système d'annuaire (*naming and registration services*) qui permet à des nœuds de se connaître et de pouvoir communiquer entre eux
  - déclarer des nœuds
  - aider les nœuds à se trouver mutuellement
  - notifier des nœuds que des changements dans le système ont été réalisés
- L'architecture utilisée par ROS s'appuie sur un réseau Point à Point (peer to peer)



```

graph LR
    subgraph "Le master"
        direction TB
        Annuaire[annuaire]
        Acq[Acquisition d'image]
        Suivi[Suivi de trajectoire]
        Acq -- "1. S'enregistrer" --> Annuaire
        Annuaire -- "2. Souscrire" --> Suivi
        Acq -.- "3. images" --> Suivi
    end
    Robot[Robot] --> Acq
    
```

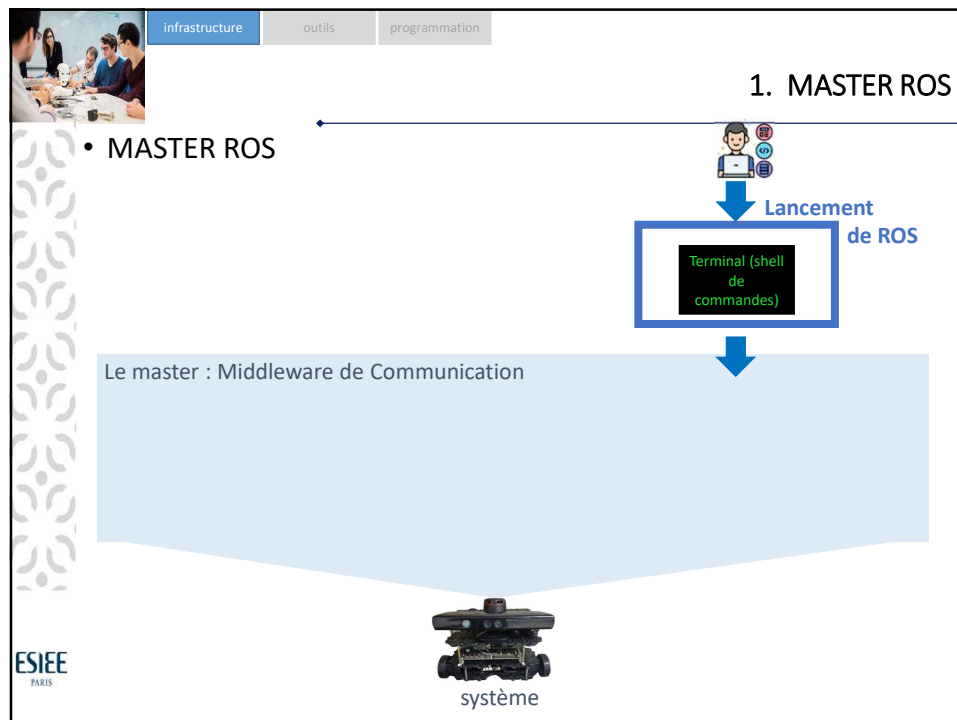
17

## LANCEMENT DES NODES

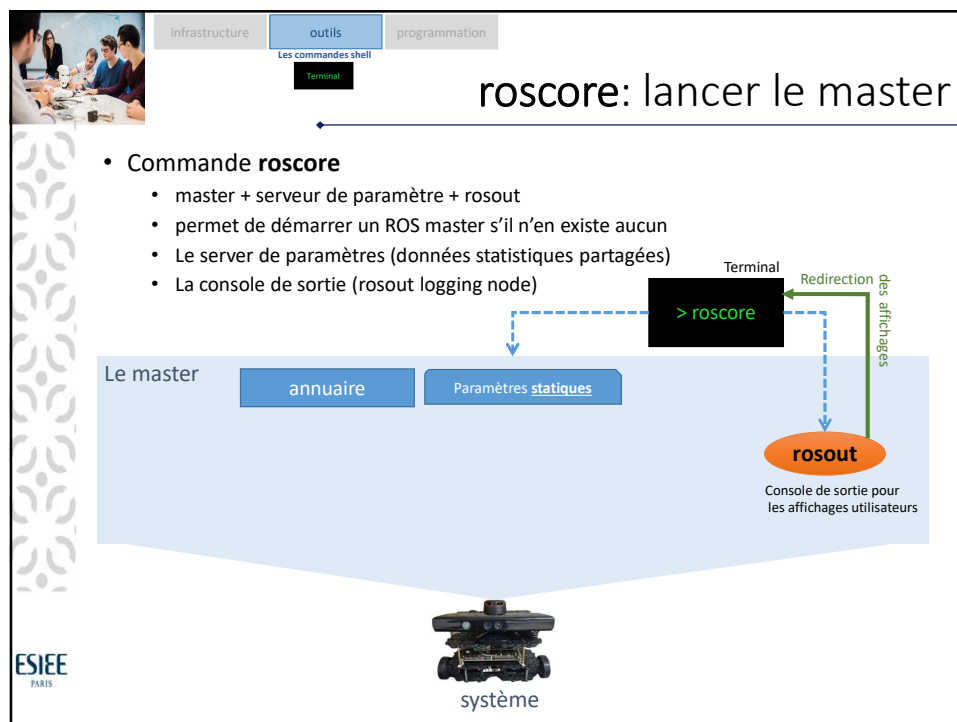
ESIEE

18


18



19



20



infrastructure
outils
programmation

Les commandes shell  
Terminal

## roscore: lancer le master

- Roscore = master + serveur de paramètre + rosout

```

#574  +
... logging to /home/user/.ros/log/83f7b25a-a7ae-11ed-b280-0242ac1a000
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://4_xterm:45585/
ros_comm version 1.12.14

SUMMARY
=====
PARAMETERS
 * /rostdistro: kinetic
 * /rosversion: 1.12.14

NODES
auto-starting new master
process[master]: started with pid [1210]
ROS_MASTER_URI=http://4_xterm:11311/

setting /run_id to 83f7b25a-a7ae-11ed-b280-0242ac1a0007
process[rosout-1]: started with pid [1236]
started core service [/rosout]
  
```

21




infrastructure
outils
programmation

Les commandes shell  
Terminal


## roslaunch: lancer un noeud

- Lancer un node : roslaunch
  - lancer un fichier exécutable/node d'un package


**Lancement de NODES**

Terminal  

```
roslaunch mon_pkg node1
```



Console de sortie pour  
les affichages utilisateurs

**Le master**  

annuaire

Paramètres statiques

mon\_pkg  

node1

rosout

- Lancer un nœud + changer son nom
  - roslaunch mon\_pkg node1 \_\_name:=une\_autre\_nom

22



infrastructure

outils

programmation

Les commandes shell  
Terminal

## roslaunch: lancer un nœud

---

DEMO

```

#574 #2842 +
user:~$ roslaunch m2wr_motion_control pilotage_moteurs
[ INFO] [1675860777.196146991]: Node Pilotage des moteurs : lancement OK !
[ INFO] [1675860777.219039468]: Pret pour recevoir des consignes


```

m2wr\_motion\_control

Pilotage\_moteurs



23



infrastructure

outils

programmation

Les commandes shell  
Terminal

## roslaunch: lancer un nœud

---

DEMO

```

#574 #2842 +
user:~$ roslaunch m2wr_motion_control pilotage_moteurs
[ INFO] [1675860777.196146991]: Node Pilotage des moteurs : lancement OK !
[ INFO] [1675860777.219039468]: Pret pour recevoir des consignes

```

```

#2525 +
user:~$ roslaunch m2wr_motion_control mesure_odometrie
[ INFO] [1675860936.897726986]: Node Mesure odometrie : lancement OK !


```

m2wr\_motion\_control



pilotage\_moteurs

m2wr\_motion\_control

mesure\_odometrie



24

infrastructure  
Les commandes shell  
**Terminal**

**outils**  
Les commandes shell  
**Terminal**

programmation


## rostopic list et info

---

- **rostopic**
  - Obtenir la liste des nœuds actifs
    - `rostopic list`
  - Obtenir des informations sur un nœud
    - `rostopic info /[nom_du_nœud]`

Terminal  
`rostopic list`

↗



**lister les NODES existants**

Le master  

annuaire

Paramètres statiques



↖

mon\_pkg  
**node1**

**rosout**

- TUTO : <http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes>

25

infrastructure  
Les commandes shell  
**Terminal**

**outils**  
Les commandes shell  
**Terminal**

programmation

## rostopic list

---

DEMO

#2525    #4267    +  
 user:~\$ rostopic list  
 /mesure\_odometrie  
 /pilotage\_moteurs  
 /rosout  
 user:~\$

m2wr\_motion\_control  

pilotage\_moteurs

m2wr\_motion\_control  

mesure\_odometrie

26

infrastructure    outils    programmation

Les commandes shell

Terminal

## roscnode list et info via rqt\_graph

Version graphique

rqt\_graph

The screenshot shows the rqt\_graph window with two nodes connected by a bidirectional arrow. The nodes are labeled `/pilotage_moteurs` and `/mesure_odometrie`.

ESIEE PARIS

27

infrastructure    outils    programmation

## Roslaunch

<node>

- Lancer plusieurs nœuds, roslaunch
  - lit un fichier xml qui contient
    - un ensemble de nœuds à démarrer pour exécuter une tâche donnée.
    - S'il n'existe pas de master (roscore), alors roslaunch en crée un.
  - roslaunch [package] [fichier.launch]

app.launch

```
<launch>
  <node name = "node1" pkg = "mon_pkg" type = "node1" output = "screen" />
  <node name = "node2" pkg = "mon_pkg" type = "node2" output = "screen" />
</launch>
```

Terminal

```
roslaunch mon_pkg app.launch
```

Console de sortie pour les affichages utilisateurs

Le master

annuaire

Paramètres statiques

mon\_pkg1

node1

node2

rosout

screen

screen

ESIEE PARIS

28

infrastructure
outils
programmation

Les commandes shell
Terminal

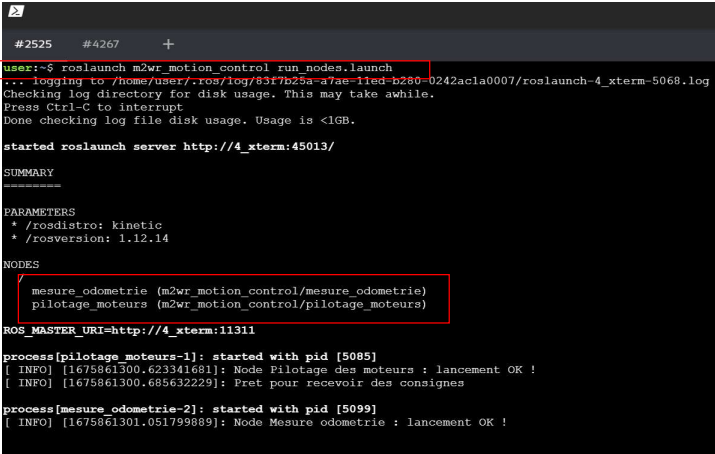
## roslaunch: lancer plusieurs nodes

DEMO

```

1 <launch>
2
3 <node name = "pilotage_moteurs" pkg = "m2wr_motion_control" type = "pilotage_moteurs" output = "screen" />
4 <node name = "mesure_odometrie" pkg = "m2wr_motion_control" type = "mesure_odometrie" output = "screen" />
5
6 [/launch]

```



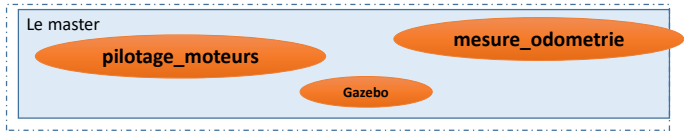
The terminal shows the execution of `roslaunch m2wr_motion_control run_nodes.launch`. It logs the process of checking log directories and disk usage. It then starts a `roslaunch` server on `http://4_xterm:45013/`. The summary shows parameters for `/rodistro: kinetic` and `/rosversion: 1.12.14`. The nodes listed are `mesure_odometrie` and `pilotage_moteurs`. The ROS master URI is `http://4_xterm:11311`. The logs confirm that both nodes started successfully with their respective PIDs.

29

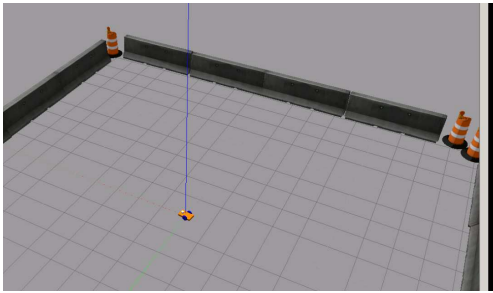
infrastructure
outils
programmation

## Un nœud ROS sous forme d'outils de simulation -- Gazebo

DEMO




The diagram shows a light blue box labeled 'Le master' containing three orange ovals: 'pilotage\_moteurs', 'mesure\_odometrie', and 'Gazebo'. This represents the ROS master managing the simulation environment.



The image shows a 3D simulation of a robot (a small yellow cube) in a Gazebo environment. The robot is on a grey grid floor, with a black wall and orange traffic cones in the background.

Robot simulé + Environnement 3D

30

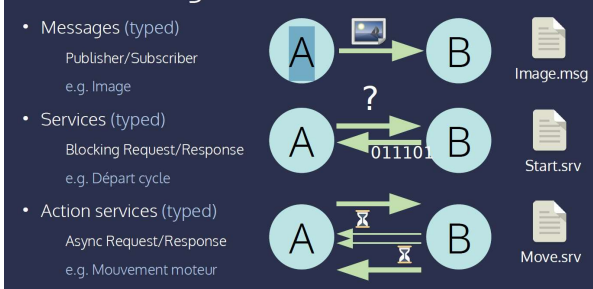


infrastructure
outils
programmation

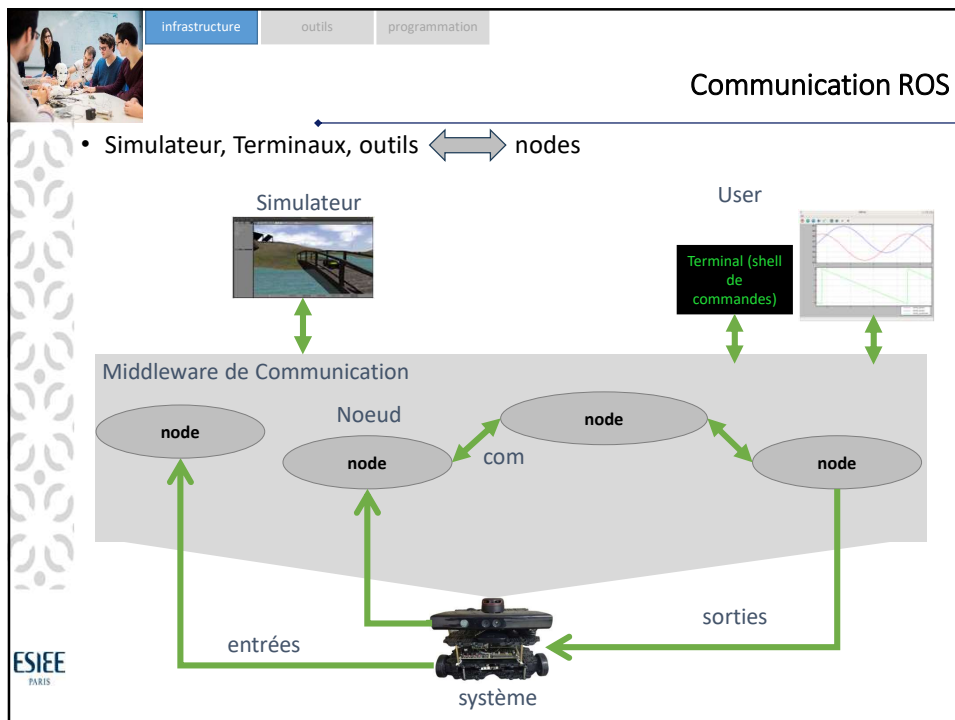
## Mécanismes de communication

- Les nœuds communiquent avec eux via le master à travers les mécanismes suivants:
  - Les topics (communication asynchrone) (E3)
  - Les services (communication synchrone) (E4-SE)
  - Les actions (processus asynchrone) (E4-SE)

- Messages (typed)  
Publisher/Subscriber  
e.g. Image
- Services (typed)  
Blocking Request/Response  
e.g. Départ cycle
- Action services (typed)  
Async Request/Response  
e.g. Mouvement moteur



31



32



# LES TOPICS

ESIEE

33

33



infrastructure
outils
programmation

## Mécanismes de communication

- Les nœuds communiquent avec d'autres nœuds via
  - • Les topics (communication asynchrone)
  - Les services (communication synchrone)
  - Les actions (processus asynchrone)

- Messages (typed)  
Publisher/Subscriber  
e.g. Image
- Services (typed)  
Blocking Request/Response  
e.g. Départ cycle
- Action services (typed)  
Async Request/Response  
e.g. Mouvement moteur



34

infrastructure

outils

programmation

# Les Topics

Principes

Les nœuds communiquent à travers des topics  
Un topic transporte des données typés

**Node**

/topic\_name  
msg:type

**Node**

**Node**

/position

X = 1.0  
Y = 2.5

**Node**

**Node**

/flux

 Image

**Node**

35

infrastructure

outils

programmation

# Les Topics

Principes

## Mécanisme Publisher / Subscriber

Communication asynchrone entre des nœuds

Un publisher

**Publisher**

pub « bonjour »

Topic : /discussion  
Type : chaîne de caractère

master annuaire

Un publisher, un subscriber

**Publisher**

pub « bonjour »

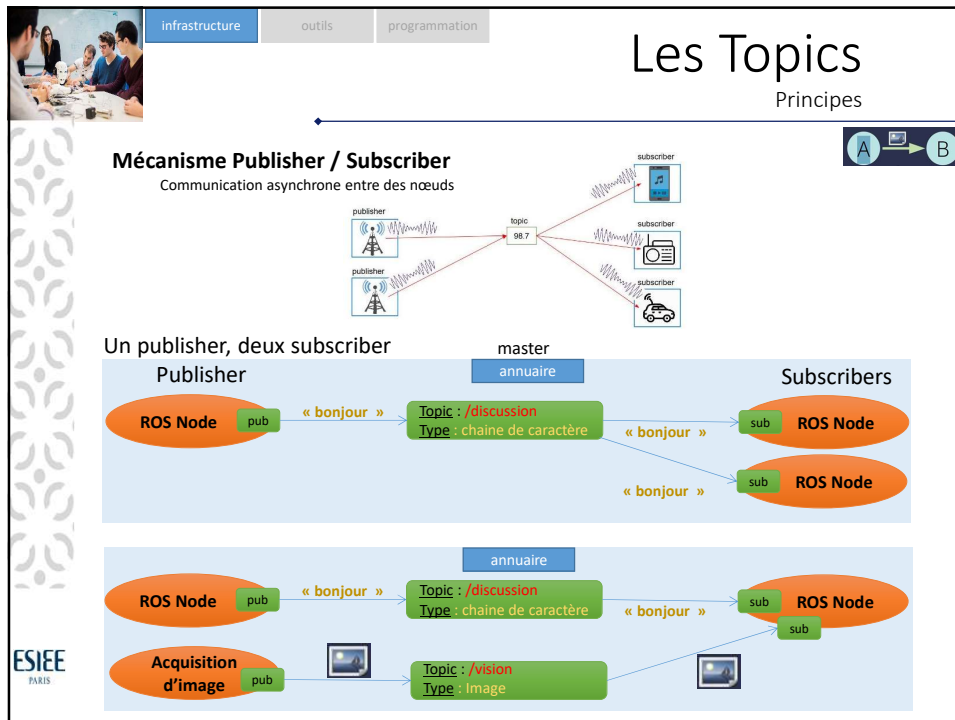
Topic : /discussion  
Type : chaîne de caractère

« bonjour »

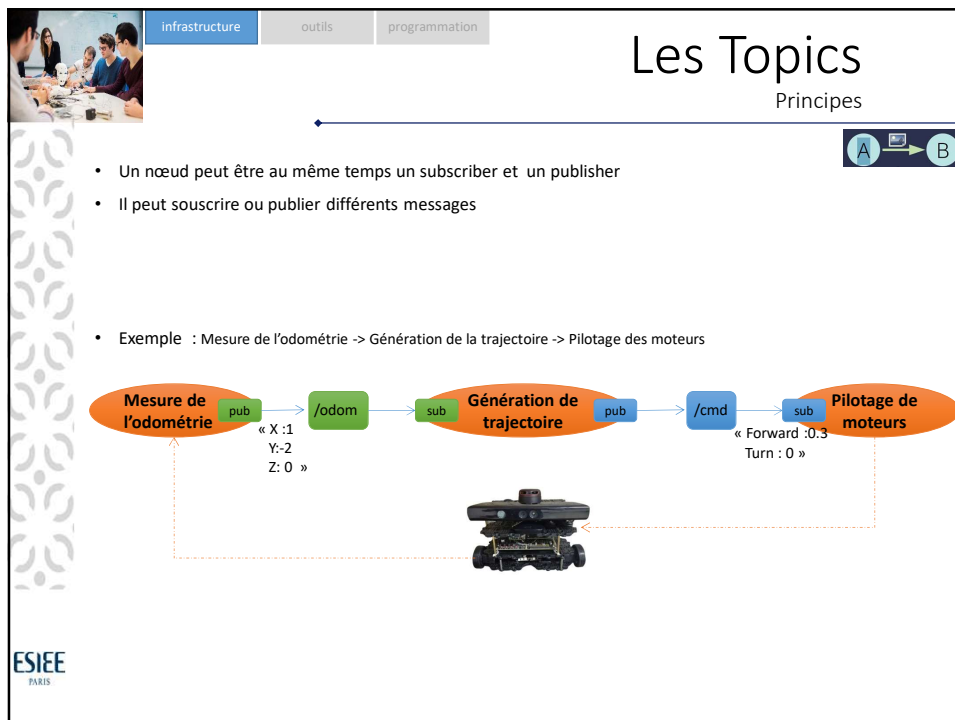
**Subscriber**

master annuaire


36



37



38



infrastructure

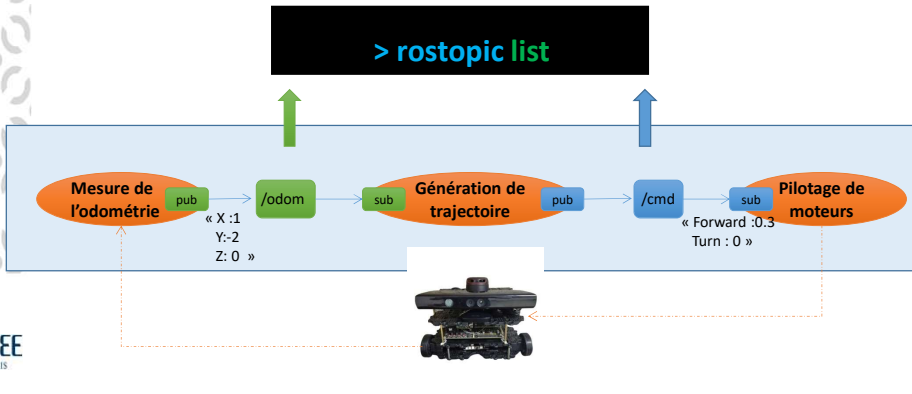
outils  
Les commandes shell

programmation


## rostopic list

---

- Analyser l'application pour connaître les données publiées sur un topic
- Connaître l'ensemble des topics actuellement abonnés et publiés
  - rostopic list**



39



infrastructure

outils

programmation

## rostopic list

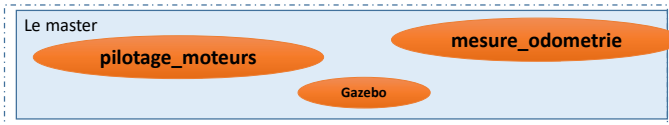
---

### DEMO

```

user:~$ rostopic list
/avancer1s
/clock
/cmd_vel
/direction
/odometrie
/pos
/rosout
/rosout_agg
/stop
/tf

```



40



infrastructure

outils  
Les commandes shell

programmation


# rostopic info

---

- Analyser l'application pour connaître les infos (types de messages) publiées sur un topic
- Connaître l'ensemble des topics actuellement abonnés et publiés
  - rostopic list
- Connaître les topics d'un noeud
  - rostopic info /[nom\_du\_node]



41



infrastructure

outils

programmation

# rostopic info

---

## DEMO

```

user:~$ rostopic info /avancer1s
Type: std_msgs/Empty

Publishers: None

Subscribers:
* /pilotage_moteurs (http://4_xterm:37533/)

```

```

user:~$ rostopic info /direction
Type: std_msgs/Int32

Publishers: None

Subscribers:
* /pilotage_moteurs (http://4_xterm:37533/)


```


/avancer1s

Type = Empty

/direction

Type = Int32





42

infrastructure

outils

programmation

# Les messages ROS

messages standards "primitives"

Un message est une structure de données utilisée pour la communication entre des nœuds (via des topics ou les services)

[http://wiki.ros.org/std\\_msgs](http://wiki.ros.org/std_msgs)

Bool, String, Int8, Int16, Int32, Int64, Float32, Float64  
Time, Duration  
Empty

**std\_msgs::**

String	string data
Char	char data
Bool	bool data
Int32	int32 data
...	

```

graph LR
    subgraph Publishers
        direction TB
        RN1[ROS Node] -- pub --> T1[Topic : /discussion  
Type : String]
        AC[Acquisition capteur] -- pub --> T2[Topic : /mesure  
Type : Float64]
    end
    subgraph Subscribers
        direction TB
        RN2[ROS Node] -- sub --> T1
        C[Contrôleur] -- sub --> T2
    end
    T1 -- "data = « bonjour »" --> RN2
    T2 -- "data = 1.45" --> C
  
```

43

infrastructure

outils

programmation

# rostopic pub

Les commandes shell

Terminal

- Publier une donnée vers un topic : **le terminal devient un nœud publisher**
  - `rostopic pub /[nom_du_topic] [type message] [valeur]`

Penser à utiliser la touche **<tab>** pour la complétion automatique

```
rostopic pub /avancer1s std_msgs/Empty "{}"
```

```


graph LR
    P1[pub] -- "Type = Empty" --> T1[/avancer1s/]
    T1 -- "Type = Empty" --> S1[sub]
    S1 --- PM[pilotage_moteurs]
  
```

```
rostopic pub /direction std_msgs/Int32 "data: 1"
```

```

graph LR
    P2[pub] -- "Int32" --> T2[/direction/]
    T2 -- "Int32" --> S2[sub]
    S2 --- PM
  
```

44



infrastructure
outils
programmation

Les commandes shell
Terminal

## rostopic pub périodique

- Publier une donnée vers un topic : **le terminal devient un nœud publisher**
  - Utiliser l'option « -r » pour préciser la fréquence d'émission en Hz
  - `rostopic pub -r [frequence] /[nom_du_topic] [type message] [valeur]`

`rostopic pub -r 1 /avancer1s std_msgs/Empty "{}"`

pub


Envoi de message  
sur le topic  
à chaque 1 sec

/avancer1s


}

sub

pilotage\_moteurs



45



infrastructure
outils
programmation

Les commandes shell
Terminal

## rostopic echo

- lire les données d'un topic: **le terminal devient un nœud subscriber**
  - `rostopic echo /[nom_du_topic]`

pub


mesure\_odometrie

/odometrie

sub

rostopic pub /odometrie

```
position:
x: 0.0159372600429
y: 0.215760457056
z: 0.099999995679
```



46

infrastructure

outils  
Les commandes shell

programmation

rostopic

Terminal

- Analyser l'application pour connaître les données publiées sur un topic
- Connaître l'ensemble des topics actuellement abonnés et publiés
  - `rostopic list`
- Connaître les topics d'un noeud
  - `rostopic info /[nom_du_node]`
- Publier une donnée vers un topic. -
  - `rostopic pub /[nom_du_topic] [type message] [valeur]`
- Publier périodiquement une donnée vers un topic
  - Utilisez l'option « -r » pour préciser la fréquence d'émission en Hz
  - `rostopic pub -r [frequence] /[nom_du_topic] [type message] [valeur]`
- Afficher en temps réel les données qui sont publiées dans un topic particulier.
  - `rostopic echo /[nom_du_topic]`

Penser à utiliser la touche <tab> pour la complétion automatique

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal

Terminal


Terminal

Terminal

Terminal

Terminal





infrastructure

outils

programmation

# Les messages ROS

messages "structurés"

[https://wiki.ros.org/common\\_msgs](https://wiki.ros.org/common_msgs)

**Exemple: geometry\_msgs / Twist**

geometry\_msgs/Twist

Vector3 linear

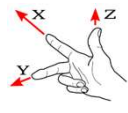
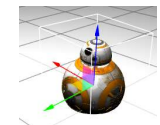
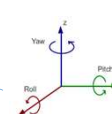
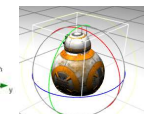
Vector3 angular


Vector3

float64 x

float64 y

float64 z



49



infrastructure

outils

programmation

# Les outils ROS

topics

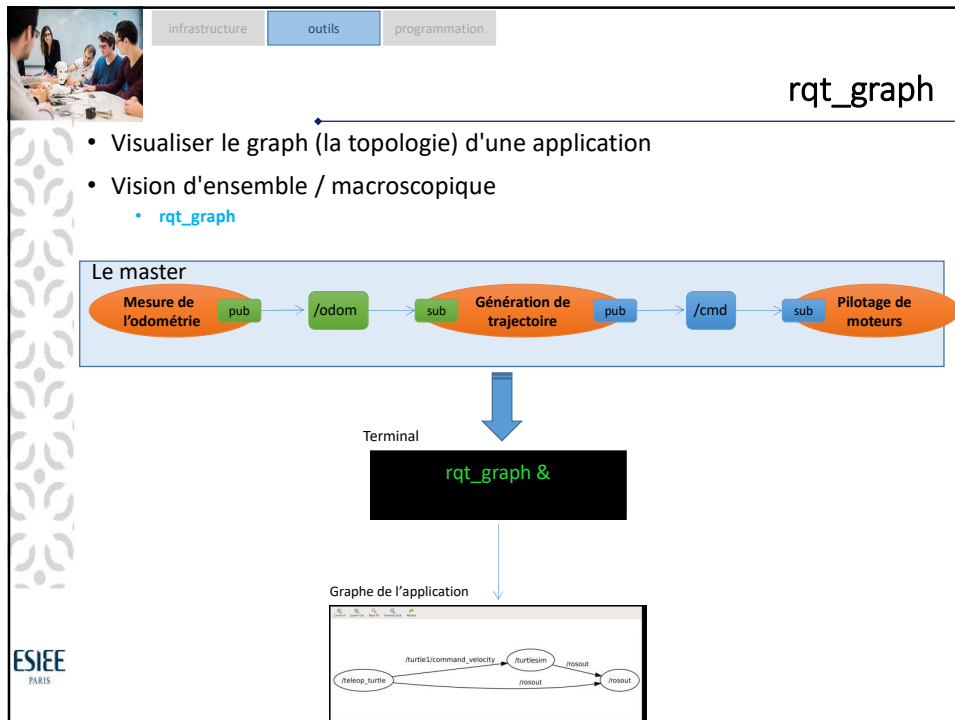




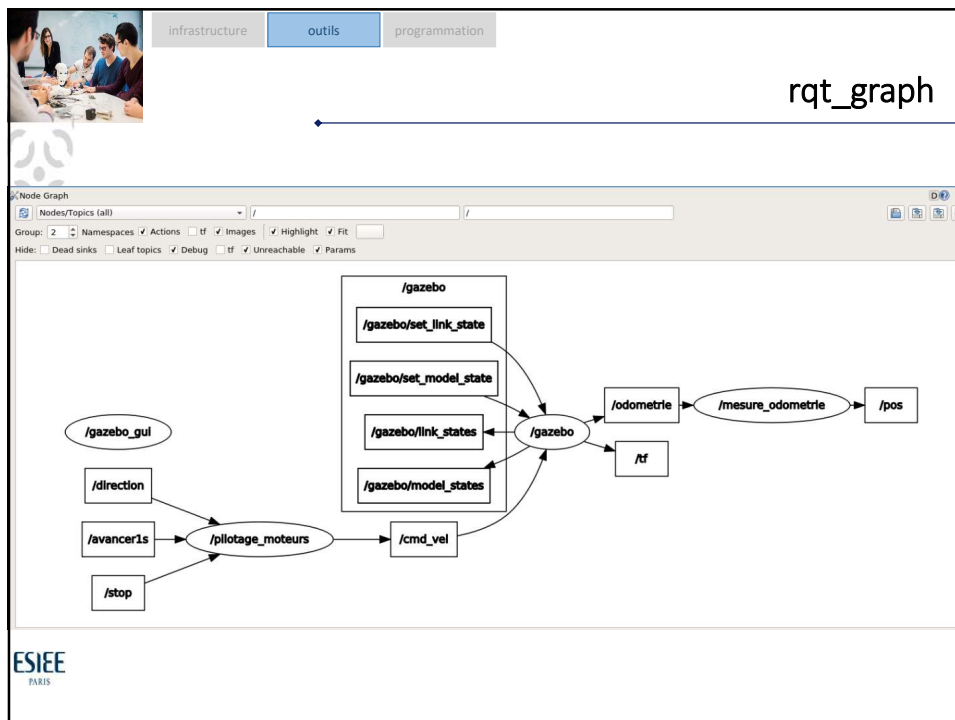




50



51



52

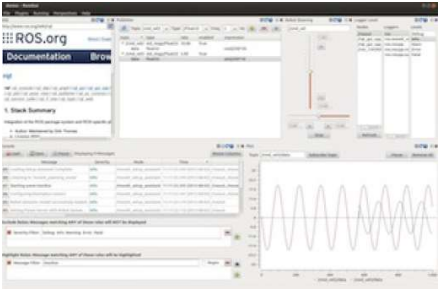
infrastructure

outils

programmation

rqt

- Une interface d'affichage et de contrôle composée de plugins.
  - Construction incrémentale d'une vision centralisé du comportement des nœuds et des message échangés



**Le master**

annuaire

Paramètres statiques

mon\_pkg1

node1

node2

→

rosout

53

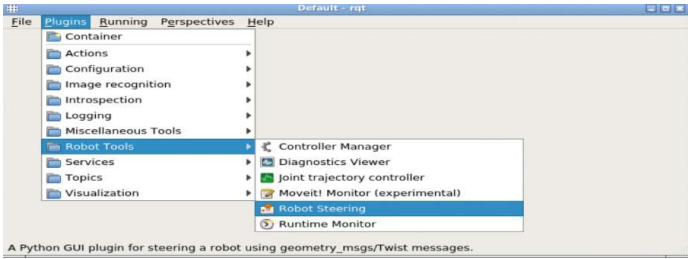
infrastructure

outils

programmation

rqt: robot steering

- Une interface d'affichage et de contrôle composée de plugins.
  - Construction incrémentale d'une vision centralisé du comportement des nœuds et des message échangés
  - **Robot steering** : Piloter manuellement les mouvements d'un robot



**Le master**

annuaire

Paramètres statiques

mon\_pkg1

node1

node2

→

rosout

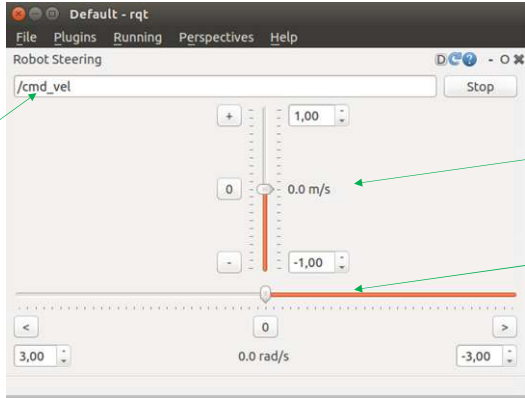
54

infrastructure
outils
programmation

## rqt: robot steering

- Une interface d'affichage et de contrôle composée de plugins.
  - Construction incrémentale d'une vision centralisé du comportement des nœuds et des message échangés
- Robot steering**: Piloter manuellement les mouvements des robots

```
linear:
  x: 0.2
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0
```



Vitesse linéaire

Vitesse angulaire

Le nom du topic de commande

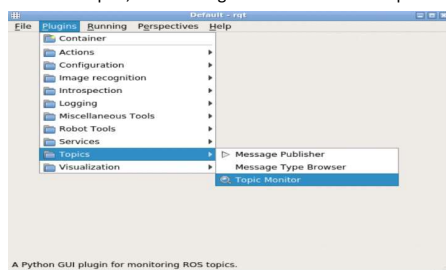
ESIEE PARIS

55

infrastructure
outils
programmation

## rqt: topic monitor

- Une interface d'affichage et de contrôle composée de plugins.
  - Construction incrémentale d'une vision centralisé du comportement des nœuds et des message échangés
- Topics monitor**: Visualiser les topics, leur message et les valeurs en temps réel



**Le master**

annuaire

Paramètres statiques

mon\_pkg1

node1

node2

rosout

↑

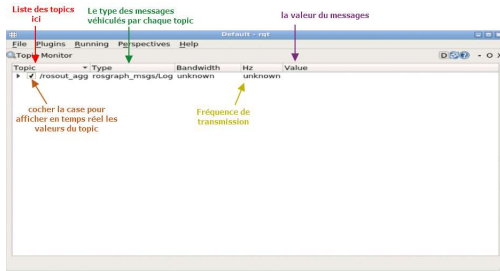
ESIEE PARIS

56

infrastructure
outils
programmation

## rqt: topic monitor

- Une interface d'affichage et de contrôle composée de plugins.
  - Construction incrémentale d'une vision centralisé du comportement des nœuds et des message échangés
- Topics monitor** : Visualiser les topics, leur message et les valeurs en temps réel



Le master

annuaire  
Paramètres statiques

mon\_pkg1  

node1

node2

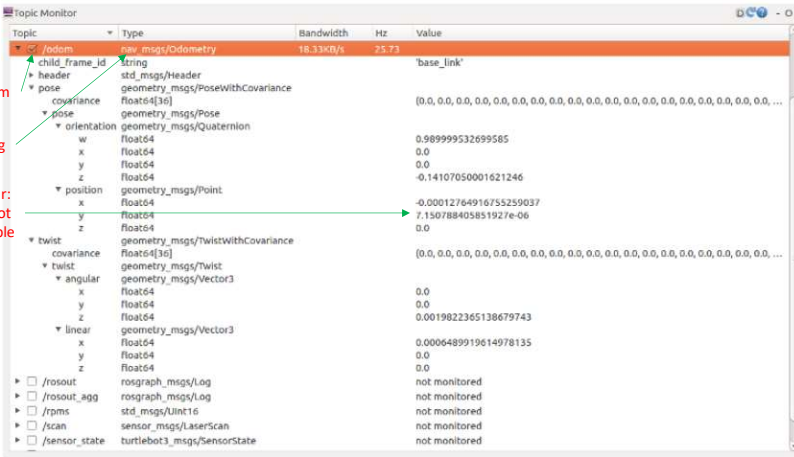
rosout

57

infrastructure
outils
programmation

## rqt: topic monitor

- Une interface d'affichage et de contrôle composée de plugins.
  - Construction incrémentale d'une vision centralisé du comportement des nœuds et des message échangés
- Topics monitor** : Visualiser les topics, leur message et les valeurs en temps réel




Topic : /odom

Type des msg

Valeur: Position du robot par exemple

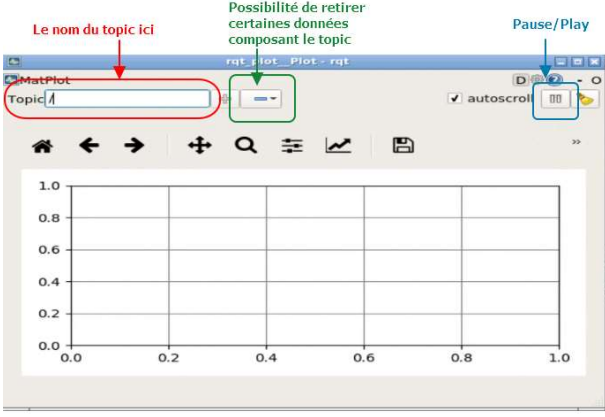
58




infrastructure
outils
programmation

## rqt: rqt\_plot

- Une interface d'affichage et de contrôle composée de plugins.
  - Construction incrémentale d'une vision centralisé du comportement des nœuds et des message échangés
- rqt\_plot**: Visualiser la courbe des messages en temps réel



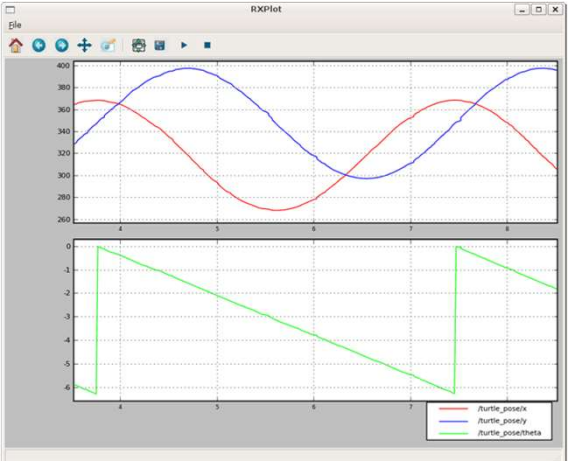
59



infrastructure
outils
programmation

## rqt\_plot

- Traceur de courbes
- Données issues des messages de topics



60

# LES SERVICES

ESIEE

61

61



infrastructure
outils
programmation

## Le Master


ses mécanismes de communication

- Les nœuds communiquent avec d'autres nœuds via
  - Les topics (communication asynchrone)
  - ➔ • Les services (communication synchrone)
  - Les actions (processus asynchrone)

- Messages (typed)
  - Publisher/Subscriber
  - e.g. Image
- Services (typed)
  - Blocking Request/Response
  - e.g. Départ cycle
- Action services (typed)
  - Async Request/Response
  - e.g. Mouvement moteur



62



infrastructure

outils

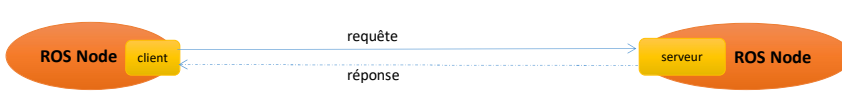
programmation

# Les services

## principes

A
B


- Le service est un mécanisme de communication synchrone uniquement entre deux nœuds identifiés comme
  - Fournisseur/serveur : le nœud qui fournit le service
  - client : le nœud qui consomme le service




```

graph LR
    subgraph Client [ROS Node client]
        direction TB
        C1[client]
    end
    subgraph Server [ROS Node serveur]
        direction TB
        S1[serveur]
    end
    C1 -- requête --> S1
    S1 -. réponse .-> C1
      
```

- Exemples
  - Effectuer des calculs à distance
  - Demander la carte de navigation ou une partie de la carte à un "serveur de carte"
  - Demander et recevoir des informations sur l'état d'un autre robot



63



infrastructure

outils

programmation


# Les services

## le processus

A
B


Un processus à trois étapes également

- phase 1: publier
  - Un nœud *Serveur* se déclare au Master pour lui indiquer qu'il souhaite publier un service en précisant un nom et un message
- phase2: chercher
  - Un nœud *Client* se déclare au Master pour lui indiquer qu'il souhaite consommer un service avec un nom donné
  - La signature du service à consommer est déjà connue par le *client*. La signature du service désigne le nom du service et les paramètres d'entrées et de sorties définis dans des messages.
- phase 3 : consommer
  - Une connexion est établie entre le Serveur et le Client via l'intermédiaire du Master. Le nœud Client invoque le service en respectant sa signature



64





infrastructure

outils

programmation

Les commandes shell

Terminal

# rosservice

---

- Lister et appeler les services d'un noeud
- Découvrir les services disponibles
  - rosservice list list of services
  - rosservice args /[nom\_service] print service arguments
  - rosservice info /[nom\_service] print information about service
- Appeler un service
  - rosservice call /[nom\_service] [arguments\_service]

TUTO : <http://wiki.ros.org/ROS/Tutorials/UnderstandingServicesParams>

65



infrastructure

outils

programmation

Les commandes shell

Terminal

# Les outils ROS

---



Lister les services

Appeler un service

...

66

infrastructure
outils
programmation

## rqt: service caller

- Une interface d'affichage et de contrôle composée de plugins.
  - Construction incrémentale d'une vision centralisé du comportement des nœuds et des message échangés
- **Service caller** : appeler un service



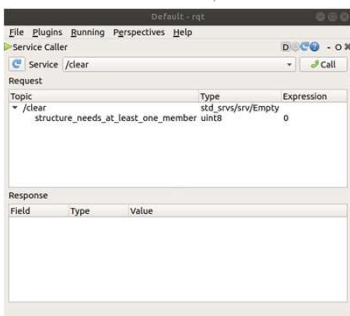
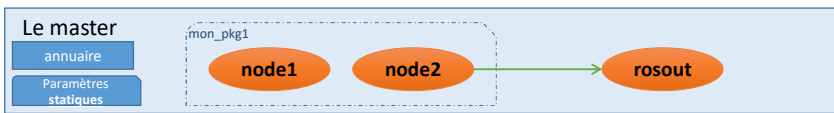

ESIEE PARIS

67

infrastructure
outils
programmation

## rqt: service caller

- Une interface d'affichage et de contrôle composée de plugins.
  - Construction incrémentale d'une vision centralisé du comportement des nœuds et des message échangés
- **Service caller** : appeler un service

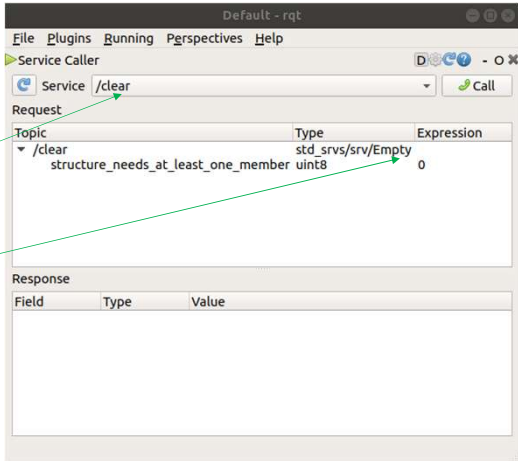
ESIEE PARIS

68

infrastructure
outils
programmation

## rqt: service caller

- Une interface d'affichage et de contrôle composée de plugins.
  - Construction incrémentale d'une vision centralisé du comportement des nœuds et des message échangés
- **Service caller** : appeler un service



Le nom du service

Sans argument

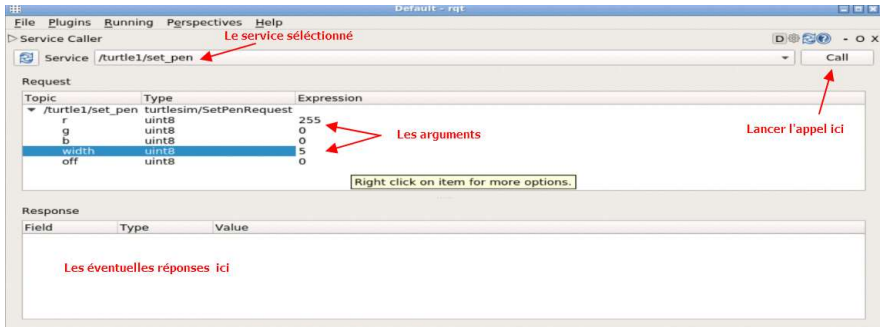
ESIEE PARIS

69

infrastructure
outils
programmation

## rqt: service caller

- Une interface d'affichage et de contrôle composée de plugins.
  - Construction incrémentale d'une vision centralisé du comportement des nœuds et des message échangés
- **Service caller** : appeler un service



Le service sélectionné

Les arguments

Lancer l'appel ici


Les éventuelles réponses ici

ESIEE PARIS

70

# EXEMPLE

71



infrastructure
outils
programmation

Terminal

## Exemple

- Lancer avec un fichier launch le talker et listener : **run.launch**

```

<launch>
  <node name="listener" pkg="roscpp_tutorials" type="listener" output="screen"/>
  <node name="talker" pkg="roscpp_tutorials" type="talker" output="screen"/>
</launch>

```

- Exécuter le fichier launch depuis le dossier où il se trouve

> **roslaunch run.launch**

```

student@ubuntu:~/catkin_ws$ roslaunch roscpp_tutorials talker_listener.launch
... logging to /home/student/.ros/log/794321aa-e950-11e6-95db-000c297bd368/rosl
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://ubuntu:37592/

SUMMARY
=====
PARAMETERS
 * /roslistro: indigo
 * /rosversion: 1.11.20


NODES
 /
  listener (roscpp_tutorials/listener)
  talker (roscpp_tutorials/talker)

auto-starting new master
process[master]: started with pid [5772]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 794321aa-e950-11e6-95db-000c297bd368
process[roscout-1]: started with pid [5785]
started core service [/roscout]
process[listener-2]: started with pid [5788]
process[talker-3]: started with pid [5795]
[ INFO] [1466044252.537001350]: hello world 0
[ INFO] [1466044252.638886394]: hello world 1
[ INFO] [1466044252.738279674]: hello world 2
[ INFO] [1466044252.838279674]: hello world 3

```

72



infrastructure

outils

programmation

Les commandes shell

Terminal

## Exemple

---

- Lancer roscore

```

student@ubuntu:~/catkin_ws$ roscore
... logging to /home/student/.ros/log/6c1852aa-e961-11e6-8543-000c297bd368/ros
launch-ubuntu-6696.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://ubuntu:34089/
ros_comm version 1.11.20

SUMMARY
=====
PARAMETERS
* /rostdistro: indigo
* /rosversion: 1.11.20


NODES
auto-starting new master
process[master]: started with pid [6708]
ROS_MASTER_URI=http://ubuntu:11311/

setting /run_id to 6c1852aa-e961-11e6-8543-000c297bd368
process[roscpp_tutorials-1]: started with pid [6721]
started core service [/roscout]

```



73



infrastructure

outils

programmation

Les commandes shell

Terminal

## Exemple

---

- Lancer le nœud talker

```


> rosrunc roscpp_tutorials talker

```


```

student@ubuntu:~/catkin_ws$ rosrunc roscpp_tutorials talker
[ INFO] [1486051708.424661519]: hello world 0
[ INFO] [1486051708.525227845]: hello world 1
[ INFO] [1486051708.624747612]: hello world 2
[ INFO] [1486051708.724826782]: hello world 3
[ INFO] [1486051708.825928577]: hello world 4
[ INFO] [1486051708.925379775]: hello world 5
[ INFO] [1486051709.024971132]: hello world 6
[ INFO] [1486051709.125450960]: hello world 7
[ INFO] [1486051709.225272747]: hello world 8
[ INFO] [1486051709.325389210]: hello world 9

```



74



infrastructure
outils
programmation

Les commandes shell
Terminal

## Exemple


- Lire les nœuds actifs
 

```
> rosnodet list
```

```
student@ubuntu:~/catkin_ws$ rosnodet list
/rosout
/talker
```
- Afficher les informations du nœud talker
 


```
> rosnodet info /talker
```

```
student@ubuntu:~/catkin_ws$ rosnodet info /talker
-----
Node [/talker]
Publications:
 * /chatter [std_msgs/String]
 * /rosout [rosgaph_msgs/Log]
Subscriptions: None
Services:
 * /talker/get_loggers
 * /talker/set_logger_level
```



```
graph LR
    talker((talker)) -- pub -->|std_msgs/String| chatter[chatter]
```

75



infrastructure
outils
programmation

Les commandes shell
Terminal

## Exemple

- Lire les informations du topic /chatter
 

```
> rostopic info /chatter
```


```
student@ubuntu:~/catkin_ws$ rostopic info /chatter
Type: std_msgs/String
Publishers:
 * /talker (http://ubuntu:39173/)
Subscribers: None
```
- Lire le type des messages du topic /chatter
 

```
> rostopic type /chatter
```

```
student@ubuntu:~/catkin_ws$ rostopic type /chatter
std_msgs/String
```
- Afficher le contenu des messages du topic /chatter
 


```
> rostopic echo /chatter
```

```
student@ubuntu:~/catkin_ws$ rostopic echo /chatter
data: hello world 11874
---
data: hello world 11875
---
data: hello world 11876
```



```
graph LR
    talker((talker)) -- pub -->|std_msgs/String| chatter[chatter]
    chatter --> rostopic_echo([rostopic echo])
```

76



infrastructure

outils  
Les commandes shell

programmation

Terminal

## Exemple

- Lire les informations du topic /chatter
 

```
> rostopic info /chatter
```

```
student@ubuntu:~/catkin_ws$ rostopic info /chatter
Type: std_msgs/String
Publishers:
 * /talker (http://ubuntu:39173/)
Subscribers: None
```
- Lire le type des messages du topic /chatter
 

```
> rostopic type /chatter
```

```
student@ubuntu:~/catkin_ws$ rostopic type /chatter
std_msgs/String
```
- Afficher le contenu des messages du topic /chatter
 


```
> rostopic echo /chatter
```

```
student@ubuntu:~/catkin_ws$ rostopic echo /chatter
data: hello world 11874
data: hello world 11875
data: hello world 11876
```
- Déterminer la fréquence d'émission des messages du topic /chatter
 

```
> rostopic hz /chatter
```

```
student@ubuntu:~/catkin_ws$ rostopic hz /chatter
subscribed to [/chatter]
average rate: 9.991
  min: 0.099s max: 0.101s std dev: 0.00076s window: 10
average rate: 9.996
  min: 0.099s max: 0.101s std dev: 0.00069s window: 20
```

77



infrastructure

outils  
Les commandes shell

programmation

Terminal

## Exemple

- Lancer un nœud listener du package roscpp\_tutorials
 

```
> rosrunc roscpp_tutorials listener
```

```
student@ubuntu:~/catkin_ws$ rosrunc roscpp_tutorials listener
[ INFO ] [1486953802.204104598]: I heard: [hello world 19548]
[ INFO ] [1486953802.304538827]: I heard: [hello world 19549]
[ INFO ] [1486953802.403853395]: I heard: [hello world 19550]
[ INFO ] [1486953802.504438133]: I heard: [hello world 19551]
[ INFO ] [1486953802.604297608]: I heard: [hello world 19552]
```
- Vérifier la liste des nœuds
 

```
> rosnode list
```

```
student@ubuntu:~/catkin_ws$ rosnode list
/listener
/rosout
/talker
```
- Vérifier les informations du topic /chatter
 


```
> rostopic info /chatter
```

```
student@ubuntu:~/catkin_ws$ rostopic info /chatter
Type: std_msgs/String
Publishers:
 * /talker (http://ubuntu:39173/)
Subscribers:
 * /listener (http://ubuntu:34664/)
```

```

graph LR
    talker([talker]) -- pub --> chatter[/chatter/]
    chatter -- "std_msgs/String" --> listener([listener])
    listener -- sub --> chatter
  
```

78



infrastructure
outils
programmation

Les commandes shell
Terminal

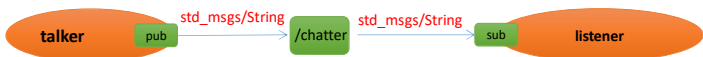
## Exemple

---


- Mettre fin à l'exécution du nœud talker
 

Ctrl+C

 dans le terminal du talker



79



infrastructure
outils
programmation

Les commandes shell
Terminal

## Exemple


---

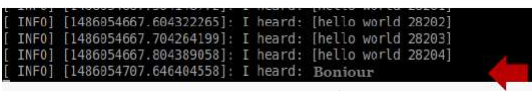
- Mettre fin à l'exécution du nœud talker
 

Ctrl+C

 dans le terminal du talker
- Depuis un terminal, générer des messages sur le topic /chatter
 

```
> rostopic pub /chatter std_msgs/String
"data: 'bonjour' "
```





80





# TD turtlesim

<https://perso.esiee.fr/~hamouchr/el3007/td/>

Présentation ESIEE