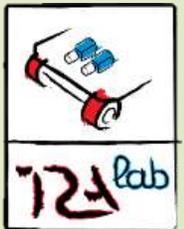




www.ros.org

Introduzione a Robotic Operating System (ROS)

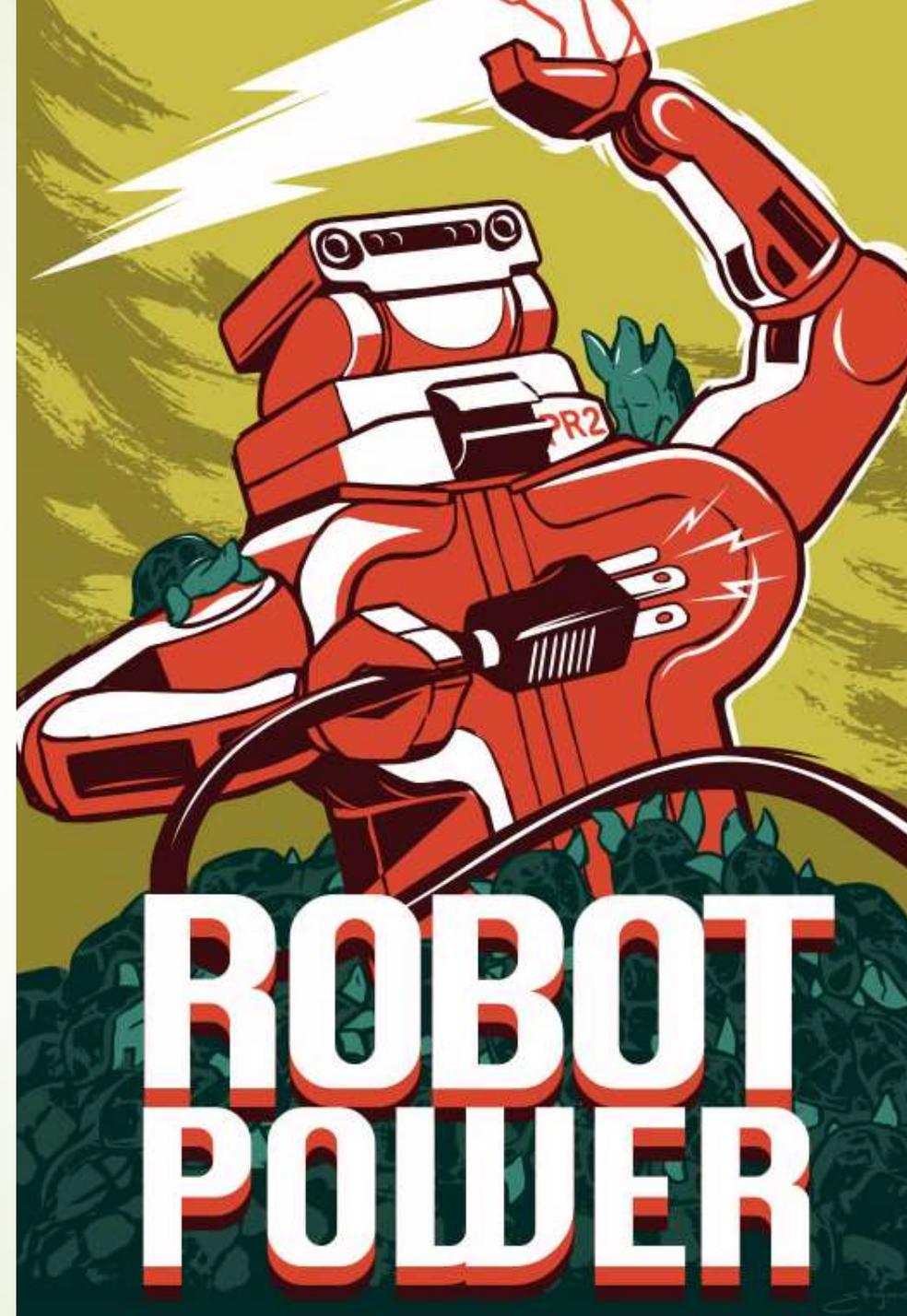
IRALAB Research Group
www.ira.disco.unimib.it



Augusto Luis Ballardini - ballardini@disco.unimib.it
Pietro Colombo - p.colombo45@campus.unimib.it

Indice

- Che cosa è ROS
- Perché un «sistema operativo» per robot?
- Come installare ROS e verificare che «tutto funzioni»
- Concetti principali, nodi messaggi e servizi
- Strumenti per la gestione dei nodi, dei messaggi e dei servizi
- DEMO: Visualizzazione e Simulazione



Cosa è ROS



“The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS has what you need for your next robotics project”

And it's all open source.

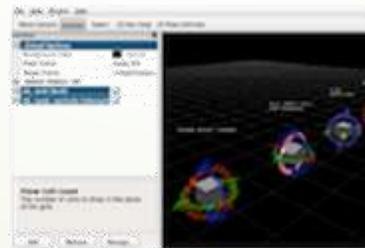
from
ros.org

- E' un sistema di componenti atti a semplificare la creazione di piattaforme robotiche.



«Middleware»
di
comunicazion

+



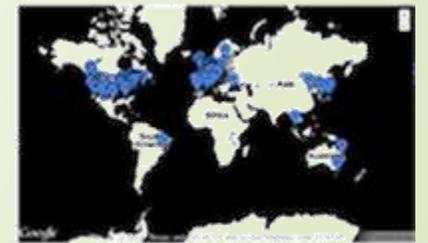
«Strumenti»
Visualizzazione
e Simulazione

+



«Capacità»
Driver/Algoritmi

+



E' un progetto
collaborativo!

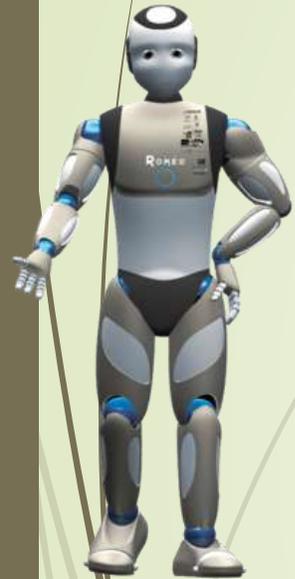
Cosa è ROS



Cosa è ROS



Cosa è ROS



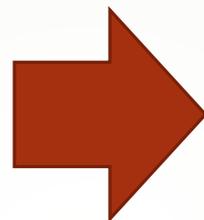
«ROS E' UN MIDDLEWARE NATO PER VELOCIZZARE IL PROCESSO DI SVILUPPO DEI ROBOT»

Cenni Storici e note legali

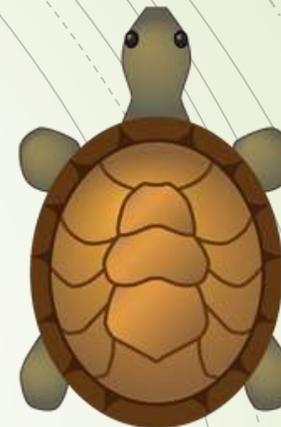
2007/2008



STANFORD UNIVERSITY
Personal Robotics Program



2010



⋮ Box Turtle

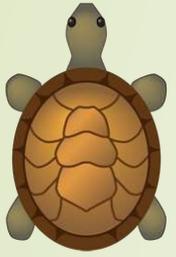
- Primi passi: lo sviluppo di ROS è iniziato nel laboratorio di intelligenza artificiale dell'univ. di Stanford.
- Il team di sviluppo venne poi supportato dall'incubatore "Willow Garage", è il periodo in cui vennero creati i PR2
- E' stato sviluppato con l'idea di essere "**the Linux of Robotics**".
- ROS CORE ha una licenza di tipo BSD 3-clausole che permette il riutilizzo del codice in applicazioni commerciali e closed-source.

<http://www.ros.org/is-ros-for-me>

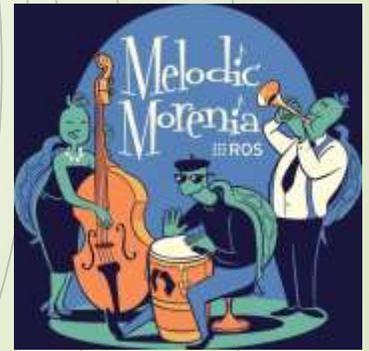
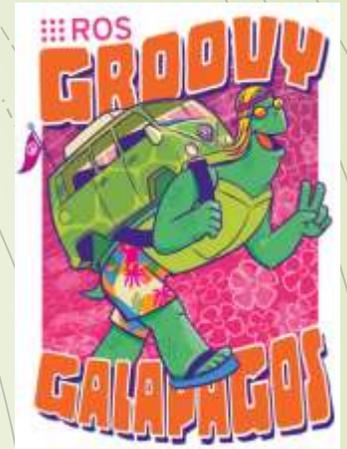
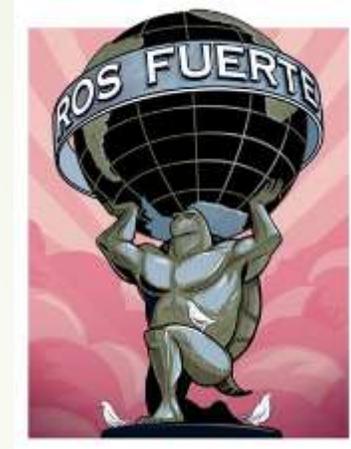
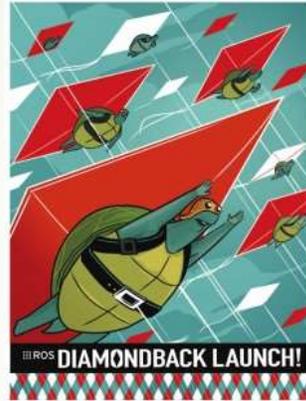
<https://opensource.org/licenses/BSD-3-Clause>

<https://spectrum.ieee.org/automaton/robotics/robotics-software/the-origin-story-of-ros-the-linux-of-robotics>

Dal 2010 ad oggi ...



Box Turtle



Select Your Platform

Supported:



Ubuntu Wily amd64 i386
Xenial amd64 i386 armhf arm64



Debian Jessie amd64 arm64

Source installation

Experimental:



OS X (Homebrew)



Gentoo



OpenEmbedded/Yocto

Unofficial Installation Alternatives:



Single A single line command to install
line install ROS Kinetic on Ubuntu



Installazione

- ROS segue un programma di distribuzione simile a quella di Ubuntu, release periodiche più LTS
- La piattaforma di sviluppo principale è basata su Ubuntu/Debian. Esistono distribuzioni sperimentali (non testate ufficialmente) es. su OS-X, Gentoo ed anche nel Windows 10 - Subsystem for Linux (WSL).
- La maggior parte dei pacchetti standard sono disponibili al download mediante PPA (*soliti comandi apt-get install ...*)
- Supporto per architetture ARM, supportato ufficialmente su RaspberryPi.



Installazione (su Ubuntu)

<http://wiki.ros.org/melodic/Installation/Ubuntu>

1. Configura sources.list e aggiorna le apt-keys del Sistema Operativo

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'  
sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-key  
421C365BD9FF1F717815A3895523BAEED01FA116
```

2. apt-update

3. apt-get install

1. Desktop full install includes ROS, rqt, rviz, robot-generic libraries, 2D/3D simulators and 2D/3D perception
2. Desktop install includes ROS, rqt, rviz, and robot-generic libraries
3. ROS-Base includes ROS, build, and communication libraries. No GUI tools.

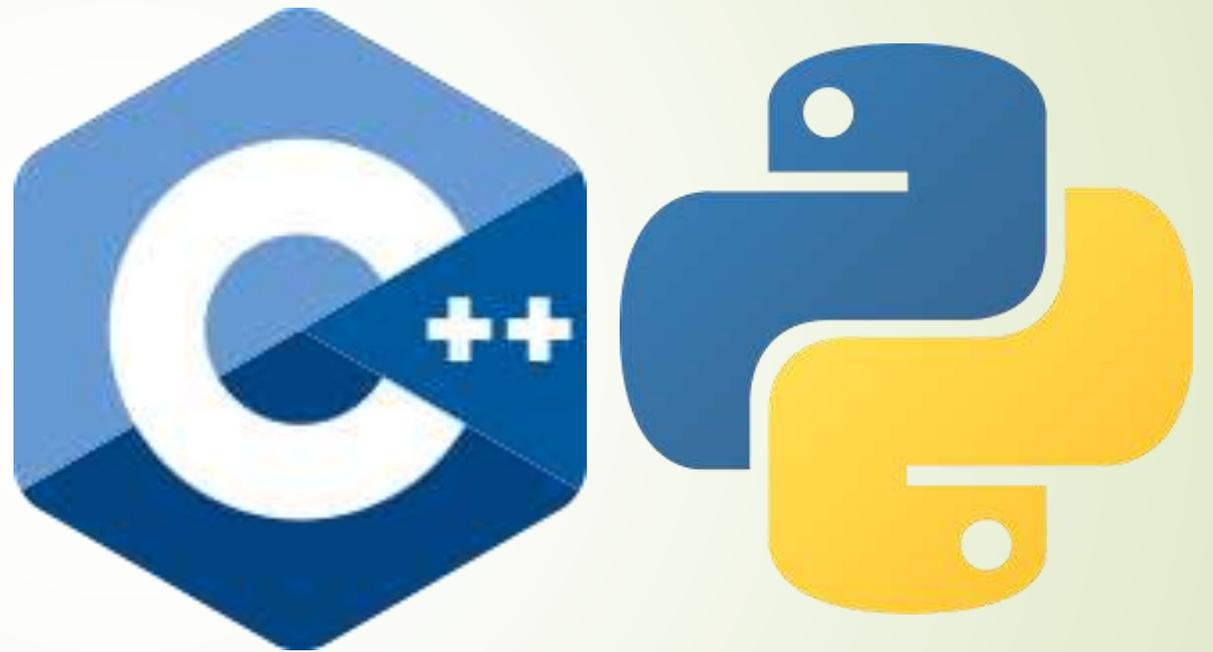
4. Eseguire i comandi

```
sudo rosdep init  
rosdep update
```

5. Per compilare i pacchetti è necessario avere installato nel sistema l'ambiente necessario per la compilazione. Si esegue con

```
sudo apt-get install python-rosinstall python-rosinstall-generator python-wstool build-essential
```

ROS e linguaggio di sviluppo



Creazione del primo Workspace e verifica dell'installazione

- ROS Build System è: catkin
- catkin è basato su Cmake e una serie di script python
- «catkin» è il successore di «roscbuild». Catkin fornisce una serie di script per «catkinizzare» un pacchetto originariamente scritto per «roscbuild»

```
$ mkdir -p ~/catkin_ws/src  
$ cd ~/catkin_ws/  
$ catkin_make  
$ source devel/setup.bash
```



CMake

ROS – Concetti Principali

- Nodes: A node is an executable that uses ROS to communicate with other nodes.
- Messages: ROS data type used when subscribing or publishing to a topic.
- Topics: Nodes can *publish* messages to a topic as well as *subscribe* to a topic to receive messages.
- Master: Name service for ROS (i.e. helps nodes find each other)
- rosout: ROS equivalent of stdout/stderr
- roscore: Master + rosout + parameter server (parameter server will be introduced later)

roscore is a command-line tool for printing information about ROS Nodes.

roscore ping test connectivity to node

roscore list list active nodes

roscore info print information about node

roscore machine list nodes running on a particular machine or list machines

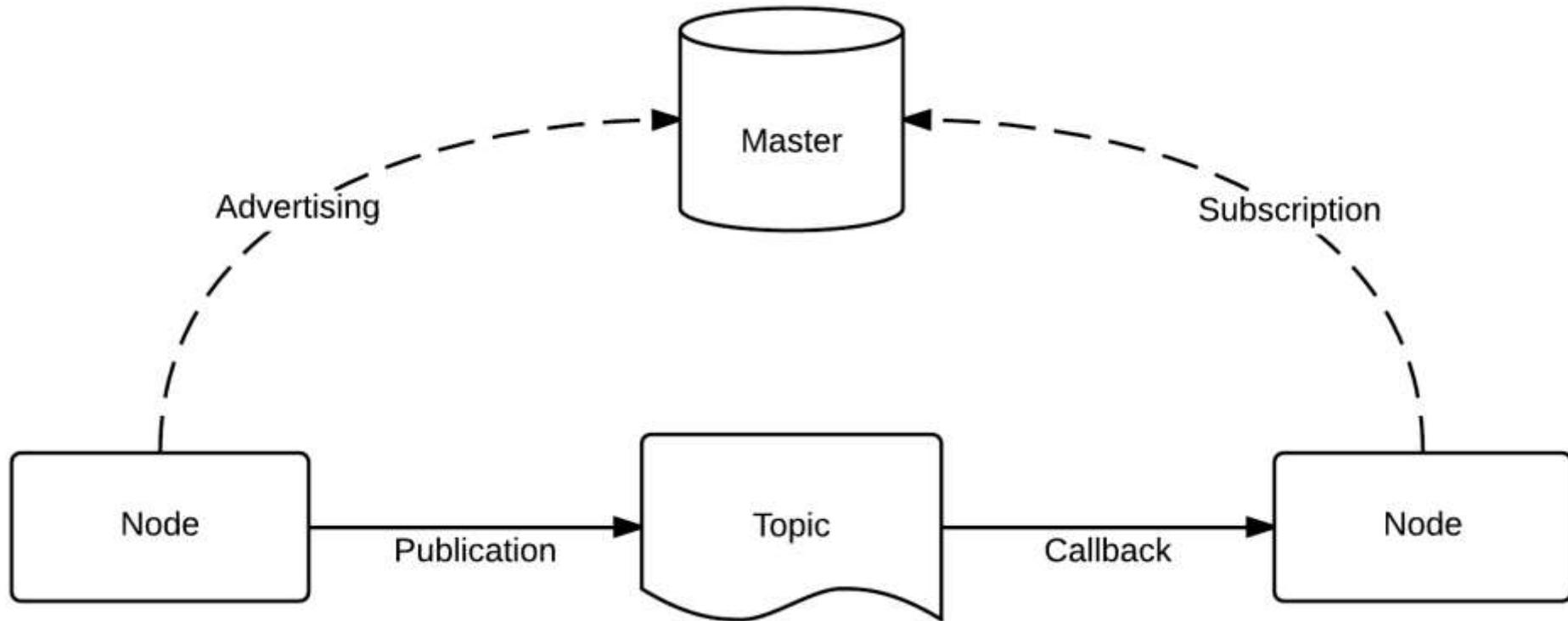
roscore kill kill a running node

roscore cleanup purge registration information of unreachable nodes

ROS – Concetti Principali (**NODO**)

NODO = programma/eseguibile

ROS consente lo scambio di messaggi in una **RETE** di nodi.



ROS – Concetti Principali (TOPICS)

Visualizzare i nodi ed i topic in modo grafico



Diagnostica sui TOPIC *rostopic*, a command-line tool for printing information about ROS Topics.

<code>rostopic bw</code>	display bandwidth used by topic
<code>rostopic delay</code>	display delay of topic from timestamp in header
<code>rostopic echo</code>	print messages to screen
<code>rostopic find</code>	find topics by type
<code>rostopic hz</code>	display publishing rate of topic
<code>rostopic info</code>	print information about active topic
<code>rostopic list</code>	list active topics
<code>rostopic pub</code>	publish data to topic
<code>rostopic type</code>	print topic or field type

ROS – Concetti Principali (**MESSAGGI**)

In ROS due nodi comunicano tra di loro attraverso topic, mediante i quali vengono inviati i messaggi. I messaggi sono tipizzati.

rosmg is a command-line tool for displaying information about ROS Message types.

Commands:

```
rosmg show    Show message description
rosmg info    Alias for rosmg show
rosmg list    List all messages
rosmg md5     Display message md5sum
rosmg package List messages in a package
rosmg packages List packages that contain messages
```

Type rosmg <command> -h for more detailed usage

ROS – Concetti Principali (ROSCORE)

In ROS due nodi comunicano tra di loro attraverso topic, mediante i quali vengono inviati i messaggi. I messaggi sono tipizzati.

`roscore` will start up a ROS Master, a ROS Parameter Server and a rosout logging node

Options:

- h, --help show this help message and exit
- p PORT, --port=PORT master port. Only valid if master is launched
- v verbose printing
- w NUM_WORKERS, --numworkers=NUM_WORKERS
override number of worker threads
- t TIMEOUT, --timeout=TIMEOUT
override the socket connection timeout (in seconds).

ROS – Componenti principali #1

- *Il «core» di ROS è costituito da un sistema per lo scambio di messaggi mediante uno schema architetturale di tipo **Publish/Subscribe**.*
- **La comunicazione tra NODI è quindi svolta in modo asincrono**
la comunicazione sincrona è consentita mediante l'utilizzo dei servizi, che implementano una architettura RPC
- **Alcuni nodi pubblicano messaggi, altri ricevono messaggi (o entrambi)**
*ROS forza la pubblicazione di messaggi definiti secondo il Message Description Language. ROS fornisce un vasto supporto di messaggi predefiniti e supporta la creazione di messaggi «custom». I **servizi** sono definiti come particolari tipi di messaggio che includono una «risposta»*
- **ROS permette la registrazione e riproduzione di messaggi**
*tutti i messaggi che transitano in rete possono venire registrati in un unico file chiamato «bag-file» usando il comando **rosbag**. Questo consente ad esempio di verificare il comportamento di un algoritmo con dei dati pre-registrati e quindi senza «tornare sul robot»*

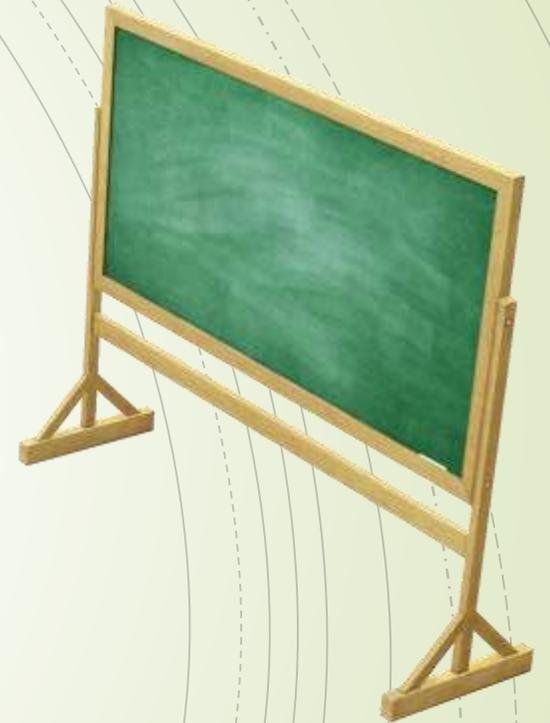
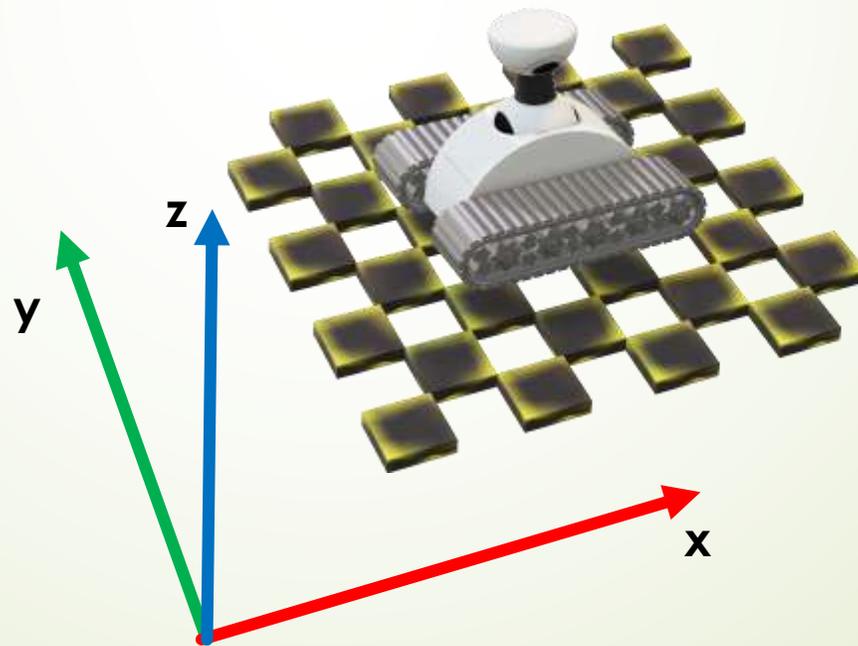
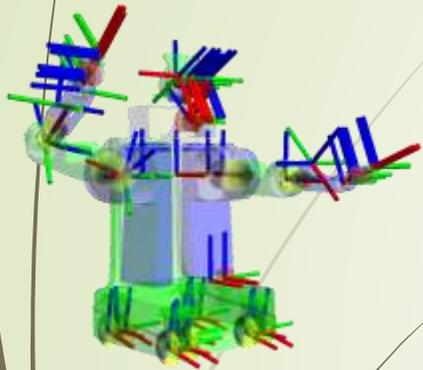
ROS – Componenti principali #2

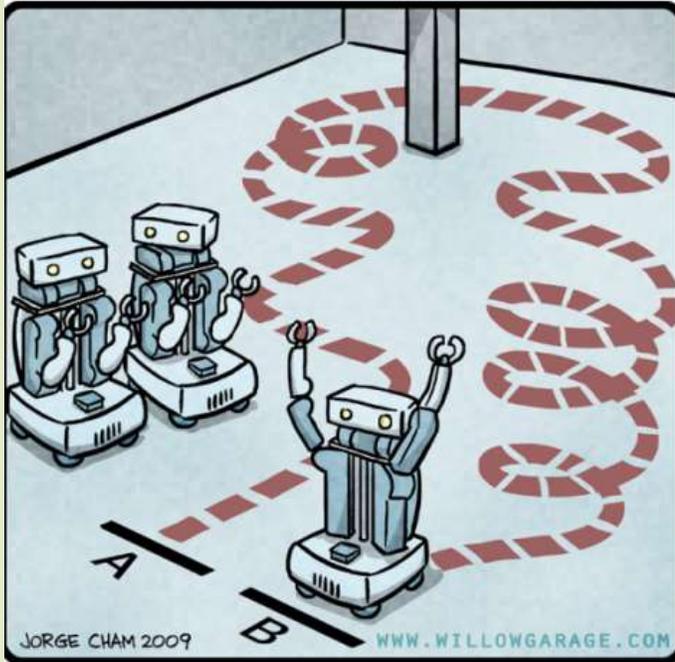
- Il nodo «**ROSCORE**» fornisce all'utente un «**PARAMETER SERVER**» consente di salvare dei parametri «globali» che possono essere letti da tutti i nodi
- Fornisce strumenti normalmente utili in qualsiasi piattaforma robotica:
 - gestione di sistemi di coordinate (nodo **TF**)
 - librerie per calcoli geometrici (basati sulla libreria **Eigen**)
 - strumenti di visualizzazione grafica
 - strumenti di simulazione
 - strumenti di diagnostica (partendo da **roswtf**)
- Il «**ROS Computational Graph**» consiste in una serie di nodi che si scambiano informazioni mediante messaggi e servizi. E' quindi formato da:

ROSCORE + NODI + PARAMETER SERVER + TOPICS + MESSAGES + SERVICES

Tipici componenti ROS usati in ogni robot

Il nodo **TF** (sviluppato da *Tully Foote*) permette di tenere traccia di sistemi di riferimento diversi nel tempo





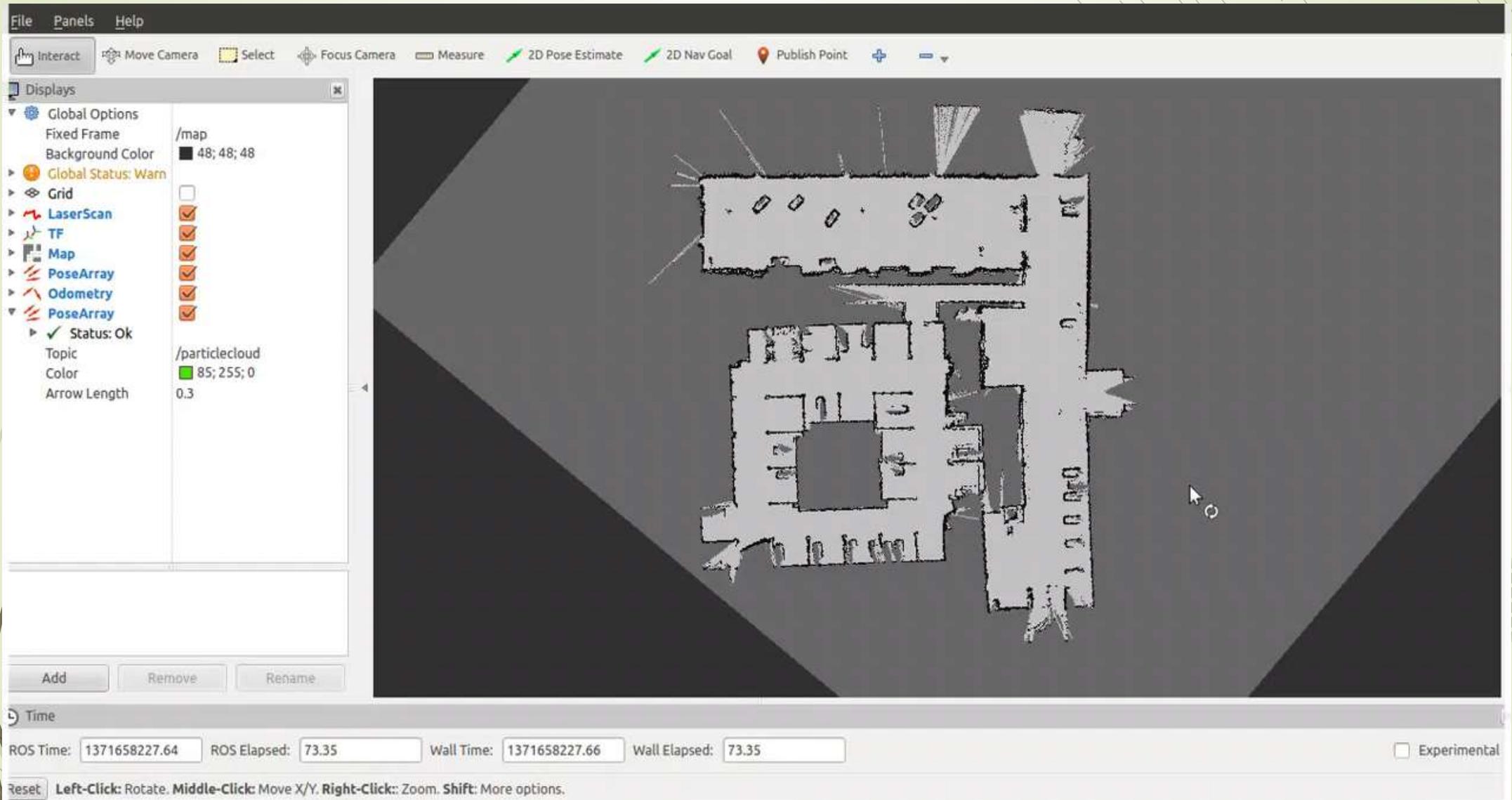
"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

Tipici componenti ROS usati in ogni robot

- **NAVIGATION STACK** contiene routine per
 - Localizzazione
 - Pianificazione del moto
- Di default è pensato per **robotica indoor**
- **Estensibile** "facilmente" (una volta che si capisce lo schema generale) ad esempio:
 - Cinematica veicoli Ackerman
 - Pianificazione moto OpenStreetMap
 - Line detector, Curbs detector, Dynamic Obstacle ecc... per vedere demo andate dagli iralabbers in atrio U7



RVIZ – ROS visualization tool



ROS + Gazebo

- ROS è compatibile con il simulatore **GAZEBO**
- Viene installato nella versione «desktop_full», altrimenti scaricabile da <http://gazebo.org>
- Sono disponibili svariati «**modelli**» di robot in ambienti differenti
- E' estensibile in linguaggio **URDF** «*Unified Robot Description Format*»

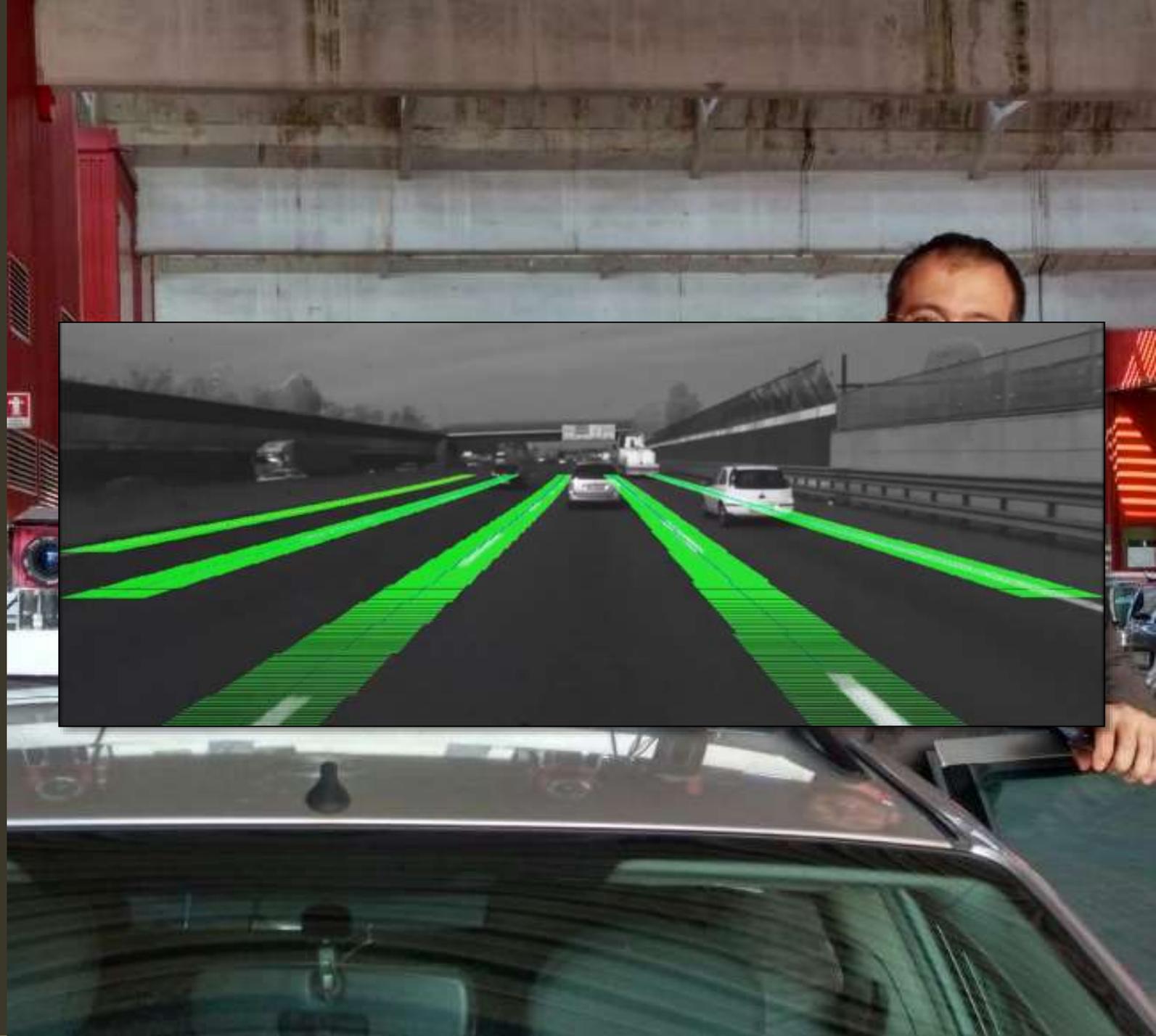


<https://www.youtube.com/c/GazeboSim>



ROSBAG

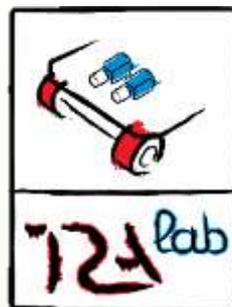
- ▶ Lavorare in robotica richiede spesso interventi «sull'hardware»
- ▶ ROS fornisce uno strumento per «acquisire dati» e «riprodurli»
- ▶ **ROSBAG** registra su un singolo file tutti i messaggi che transitano tra i nodi ROS, consentendo una successiva riproduzione



Ciò che tutti (forse) aspettavano ... DEMO TIME 😊

1. Download ed installazione **ROS** (cenni)
2. Configurazione nuovo workspace e verifica dell'installazione
3. Creazione di un nodo (**listener**)
4. Compilazione del nodo e test (public. messaggio con «**rostopic pub**»)
5. Uso di un nodo «driven» per **Velodyne**. Visualizzazione dati su **RVIZ**.
6. Attacco del driver su sistema di riferimento. **TF** + **RVIZ** + **Velodyne**
7. Simulazione con **GAZEBO**
8. Short Video about **ROS**
9. Infine, Q&A session

Q & A time!



<http://www.ira.disco.unimib.it/>
<https://www.instagram.com/iralabdisco/>



CREDITS



- All the images and videos are from ros.org and/or from the IRALAB team.
 - The presentation is inspired following the excellent work done by Mike Anderson <https://www.youtube.com/watch?v=yWtGUk3PBms&t=2987s>
 - Download ROS from ros.org !
Powerpoint presentation available upon request
-



- Tutte le immagini e video sono stati recuperati da ros.org oppure forniti dal gruppo IRALAB
- La presentazione trae ispirazione dall'eccellente lavoro di Mike Anderson <https://www.youtube.com/watch?v=yWtGUk3PBms&t=2987s>
- Scaricate ROS da ros.org!

augusto.ballardini@disco.unimib.it
trigal@gmail.com

LINUX DAY MILANO 2018

Presentazione Powerpoint disponibile su richiesta