

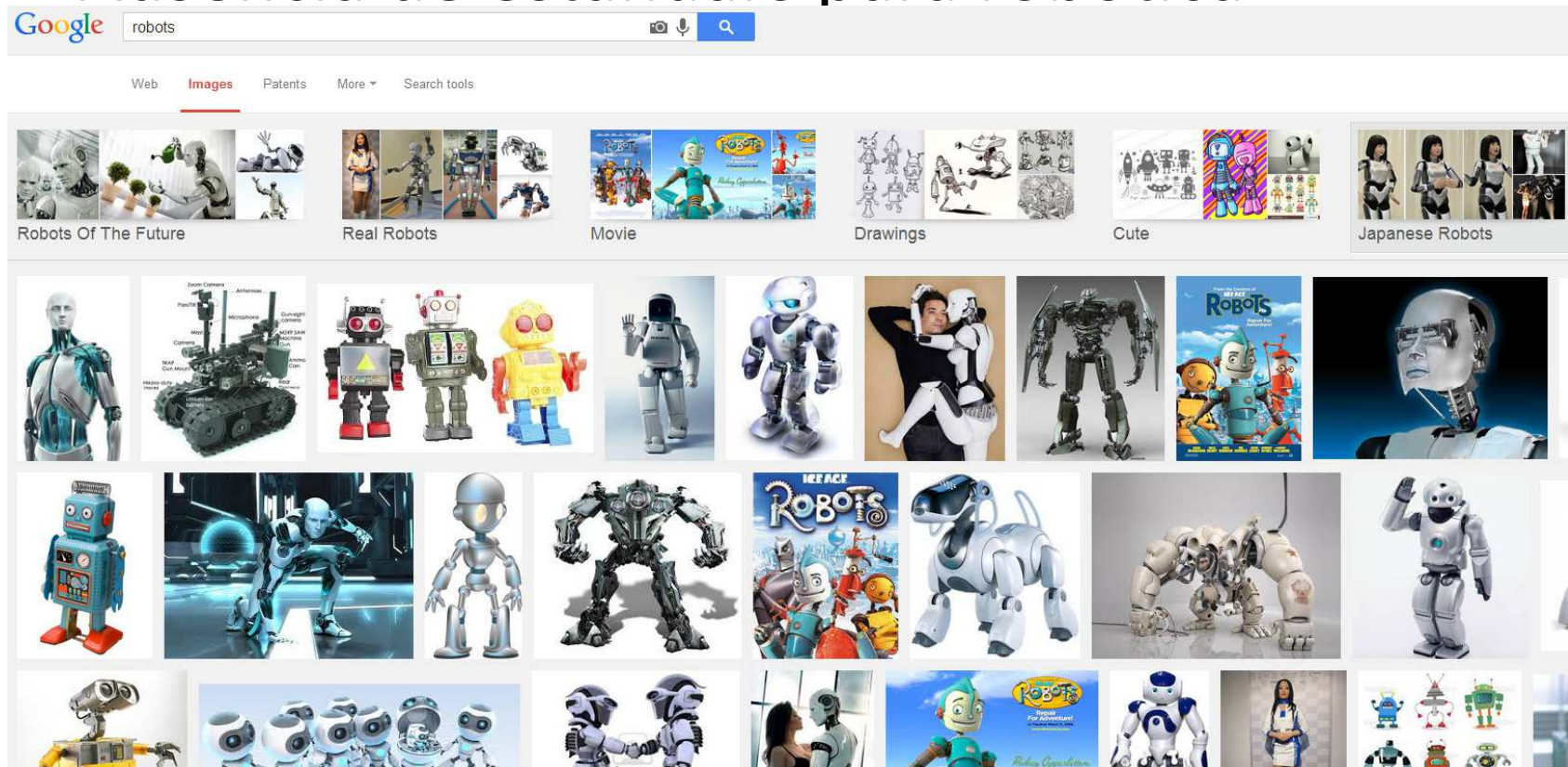
# Introducción a ROS (Robot Operating System)

Técnicas de los Sistemas Inteligentes  
Práctica1: Robótica

Algunas partes de esta presentación en <http://u.cs.biu.ac.il/~yehoshr1/89-685/>  
(C)2013 Roi Yehoshua

# El problema

- Ausencia de estándares para robótica



# El problema

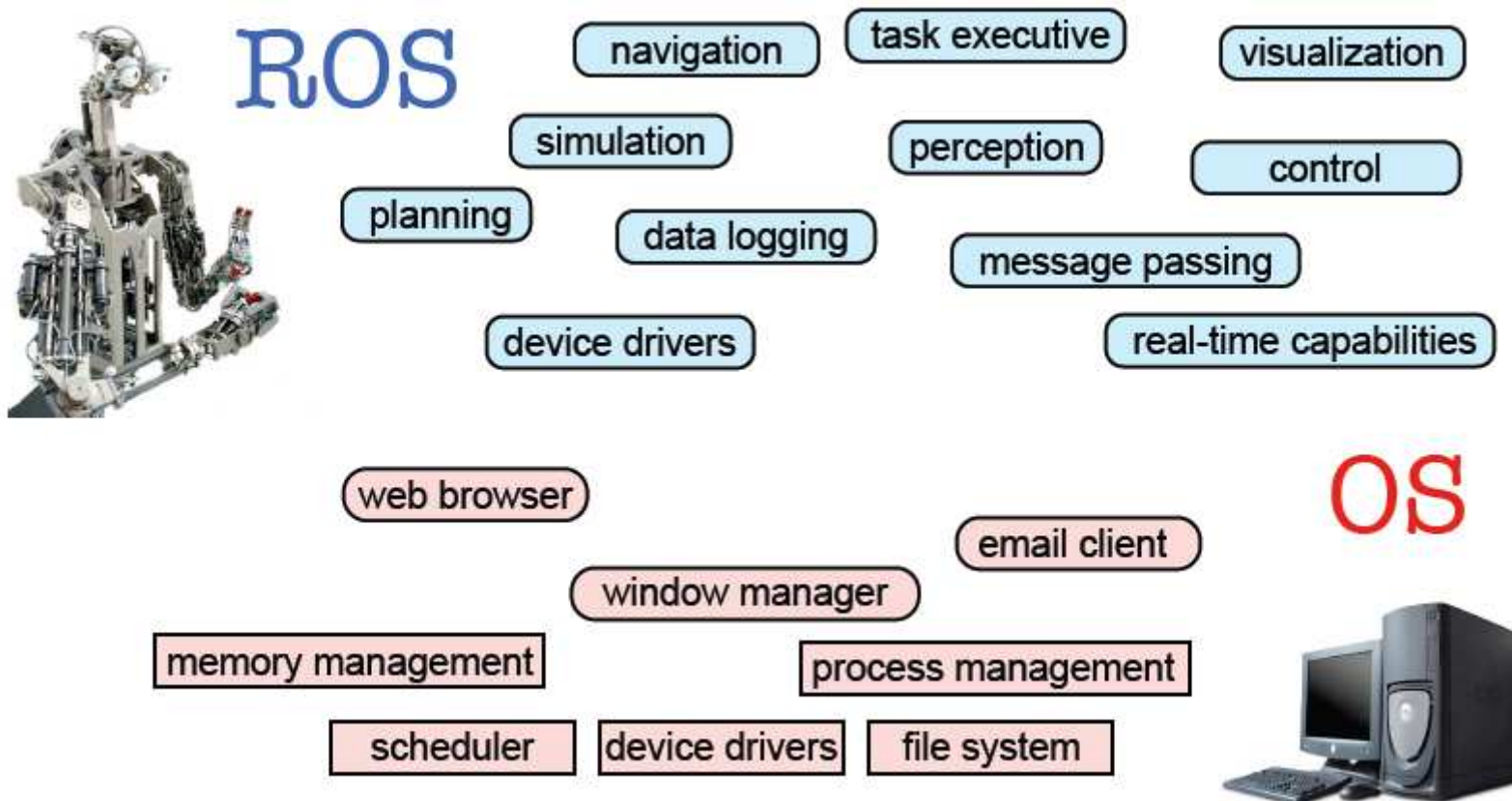
- Ausencia de estándares para robótica



# ¿Qué es ROS?

- Un **sistema operativo para robots** de código abierto (<http://wiki.ros.org/>) (<http://www.ros.org>)
- Objetivo fundamental:
  - Soportar **reutilización de código** en el desarrollo e investigación sobre robótica.
- Originalmente desarrollado en 2007 en el Stanford Artificial Intelligence Laboratory (<https://ai.stanford.edu/>)
- Su desarrollo continúa en "Willow Garage" (<http://www.willowgarage.com/>)
  - Institución para la investigación y desarrollo
  - Incubadora de empresas sobre robótica.
- Versión actual: ROS Kinetic (Ubuntu 15.10 y superiores), hasta 2020.

# Características principales de ROS





# Características principales de ROS

- **Abstracción** del hardware del robot y de comunicaciones por red.
- Facilita el **control de dispositivos a bajo nivel**.
- **Orientado a procesos:**
  - Una aplicación en ROS es siempre un conjunto de componentes
  - Independientes, débilmente acoplados.
  - Implementados como procesos (similares a hebras).
- Comunicación entre procesos basada **en paso de mensajes:**
  - Middleware para gestionar comunicación inter-proceso.
- Funcionalidades comunes de **robots pre-implementadas**
- **Gestión de paquetes:**
  - El código fuente de los componentes se organiza por paquetes
- **Extendido entre una enorme comunidad internacional, muy bien documentado.**

# Robots que usan ROS.

<http://wiki.ros.org/Robots>



[Fraunhofer IPA Care-O-bot](#)



[Videre Erratic](#)



[TurtleBot](#)



[Aldebaran Nao](#)



[Lego NXT](#)



[Shadow Hand](#)



[Willow Garage PR2](#)



[iRobot Roomba](#)



[Robotnik Guardian](#)



[Merlin miabotPro](#)



[AscTec Quadrotor](#)



[CoroWare Corobot](#)



[Clearpath Robotics Husky](#)




[Clearpath Robotics Kingfisher](#)



[Festo Didactic Robotino](#)

# Wiki de ROS

- <http://wiki.ros.org/>
- Instalación: <http://wiki.ros.org/ROS/Installation>
- Tutoriales: <http://wiki.ros.org/ROS/Tutorials>
- Vídeo-Tutoriales:  
<https://www.youtube.com/playlist?list=PLDC89965A56E6A8D6>
- Cheat Sheet (Hoja Resumen):   
<http://www.tedusar.eu/files/summerschool2013/ROScheatsheet.pdf>

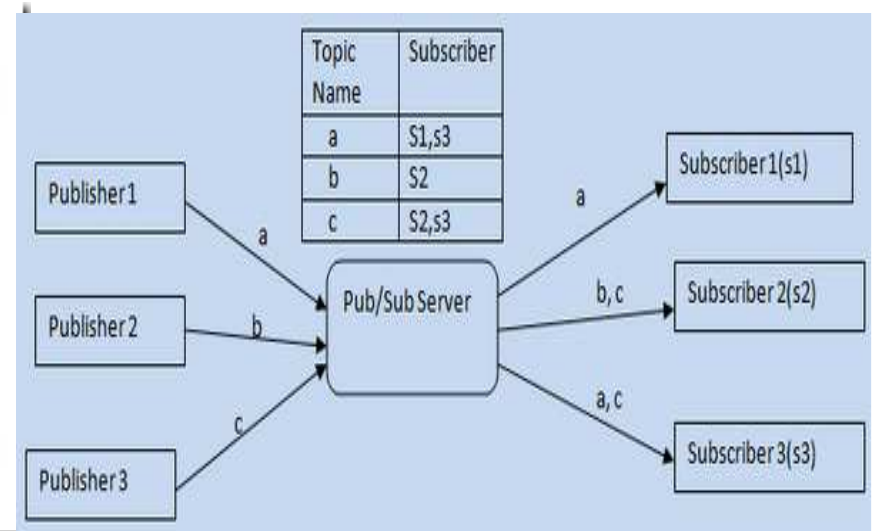
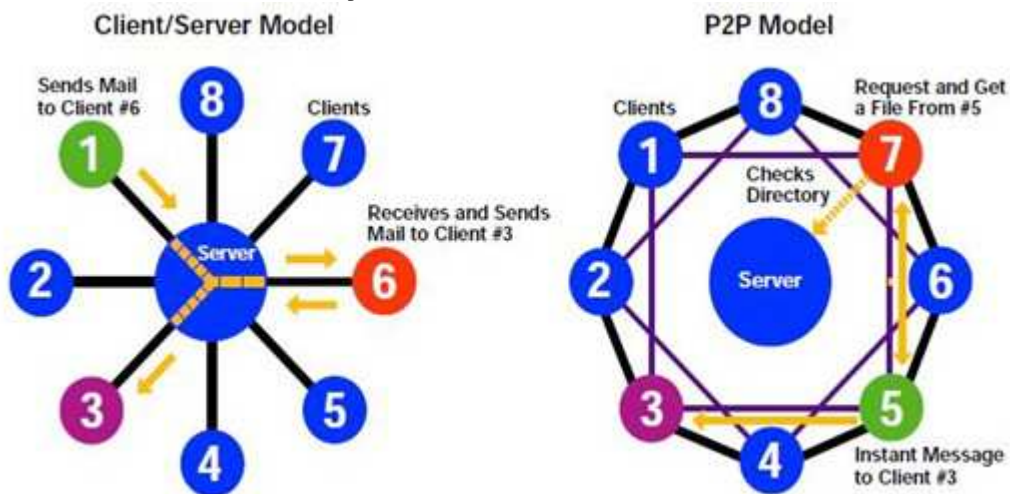


# Filosofía de ROS

- Modularidad y P2P (peer-to-peer).
- Independiente del lenguaje.
- Ligero
- Free & Open-Source

# Modularidad/P2P

- ROS consiste en un conjunto de procesos ejecutándose (como hebras)
  - Independientes
  - Pueden ejecutarse en diferentes hosts.
  - Conectados siguiendo una tipología peer-to-peer.
- No hay servidor central



# Independencia del lenguaje

- Varias interfaces clientes:
  - Estables: **roscpp (C++)**, rospy (Python), roslisp (LISP)
  - Experimentales: rosjava, roscs
  - Contribuciones: roserial, roshack, ipc-bridge (MATLAB),...
- Nivel común para paso de mensajes
  - Interface Definicion Language (IDL).

# Ligero

- Desarrollo al estilo de librerías
  - todo desarrollo ocurre en librerías "standalone" con dependencias mínimas sobre ROS.
- ROS reutiliza código de otros proyectos de código abierto
  - simuladores
  - sistemas de navegación de robots
  - algoritmos de visión de OpenCV
  - ...

# Abierto y libre

- Código fuente está públicamente disponible.
- Herramientas contribuidas bajo una variedad de licencias (abiertas y cerradas).
- Favorece reutilización del código, y construcción basada en comunidad.

# **CONCEPTOS ESENCIALES DE ROS**

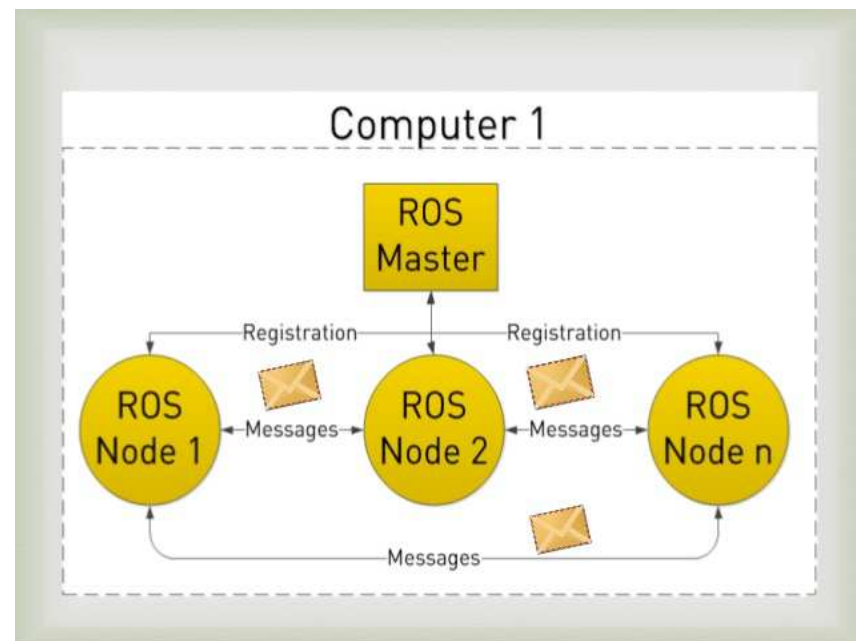


# Conceptos esenciales

- Nodes (<http://wiki.ros.org/Nodes>)
- Topics (<http://wiki.ros.org/Topics>)
- Messages (<http://wiki.ros.org/Messages>)
- Services (<http://wiki.ros.org/Services>)
- Bags (<http://wiki.ros.org/Bags>)
- ROS Master (<http://wiki.ros.org/Master>)
- Parameters ([Parameter Server](#))

# Nodos Ros (Nodes)

- Programas ejecutables para un propósito concreto:
  - p.e. drivers de sensores, drivers de actuadores, navegador local, navegador global.
- Diseño modular
  - Los nodos se compilan, ejecutan y gestionan individualmente
  - Es típico dividir la funcionalidad del software en módulos
  - Cada módulo puede ejecutarse como un nodo, o como múltiples nodos.
- Un sistema de control de robot en ROS comprende varios nodos.

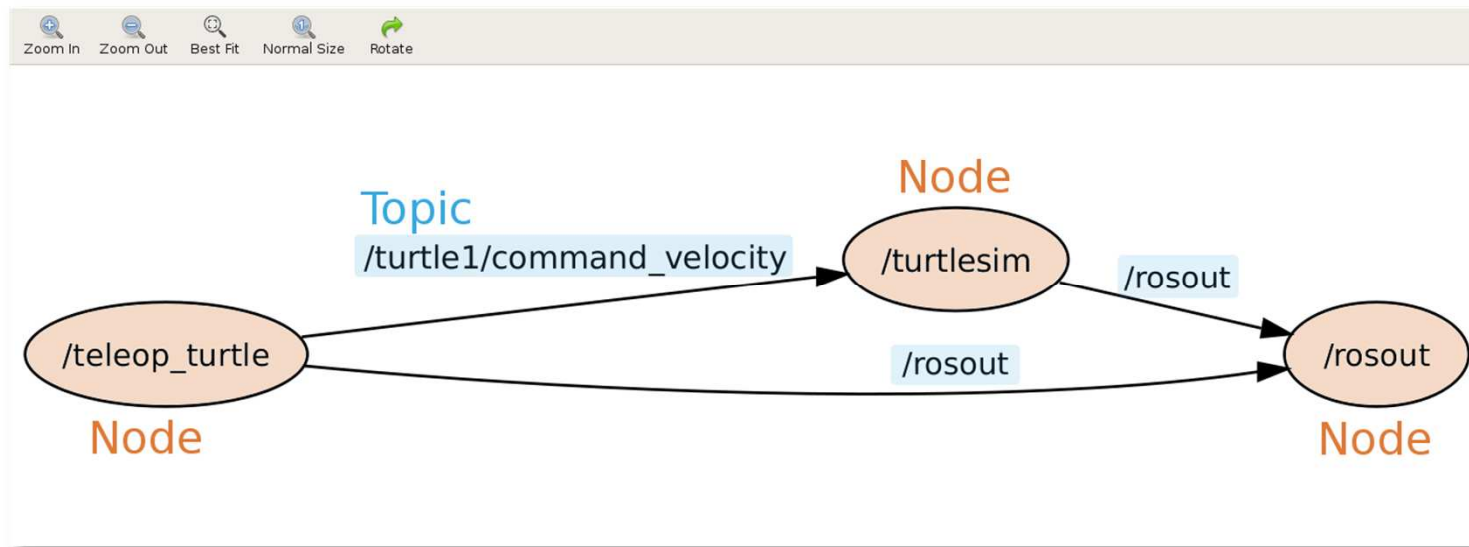


# Nodos ROS: Librería cliente

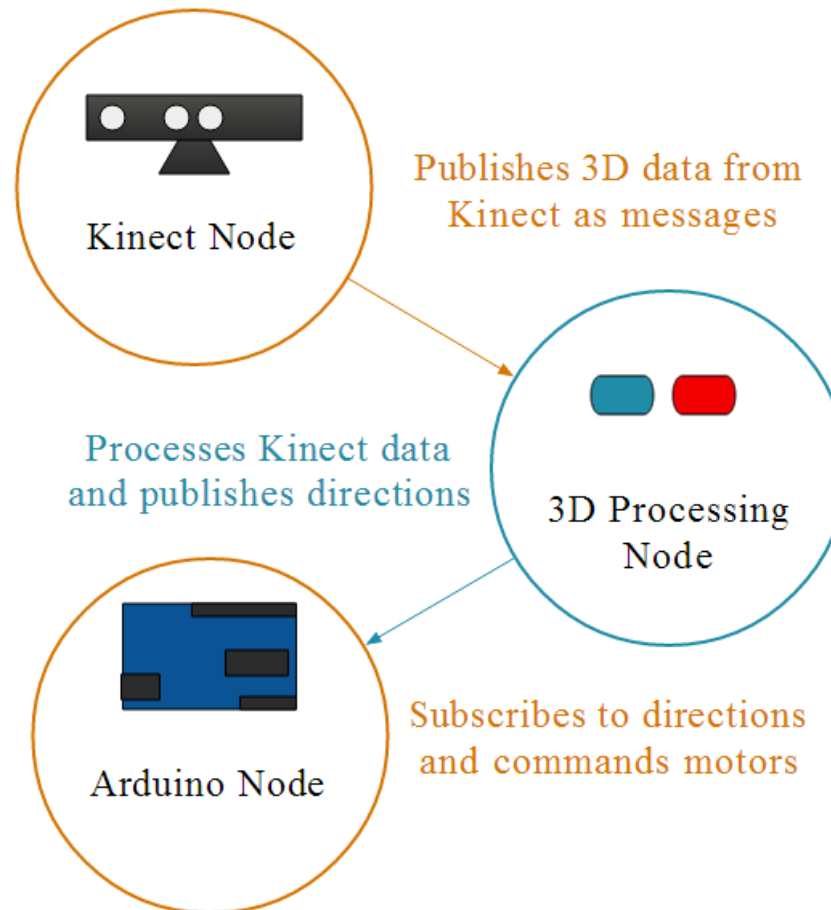
- ¿Cómo implemento un Nodo ROS?. Usando una librería cliente: colección de código que facilita el trabajo del programador.
- La librería cliente permite
  - Implementar nodos, publicar y subscribirse a topics, enviar/recibir mensajes, escribir y llamar a servicios y usar el servidor de parámetros.
- Principales librerías:
  - roscpp = C++ client library
  - rospy = python client library

# Tópicos ROS (Topics)

- Tópico:
  - representa una tipo de cadena de datos para comunicación asíncrona.
  - Ejemplos: imagen de una cámara, posición de un robot, lectura de un sensor, estado de un actuador,...).
- **Permite la Comunicación Asíncrona entre nodos:** Los nodos se comunican entre ellos mediante la publicación de mensajes bajo un tópico dado.
- Cada nodo puede subscribirse a o publicar uno o varios topics.
  - La publicación se entiende como Broadcasting 1-a-N.



# Ros Topics



# Ros Messages

- Estructuras de datos fuertemente tipadas para comunicación entre nodos.
- Los mensajes pueden incluir:
  - Tipos primitivos (int, float, bool, ...)
  - Arrays de primitivos.
  - Estructuras y arrays anidados (como C structs)



# Mensajes ROS (Messages)

- La estructura de un mensaje se define en un fichero de texto siguiendo una sintaxis propia de ROS
- ROS tiene una librería de mensajes predefinida muy útil
- Ejemplo:  
geometry\_msgs/Twist.msg

Vector3 linear Vector3 angular
-----------------------------------

## **MyMessage.msg**

```
# this is a very useful comment!  
float64 myDouble  
string myString  
float64[] myArrayOfDouble
```

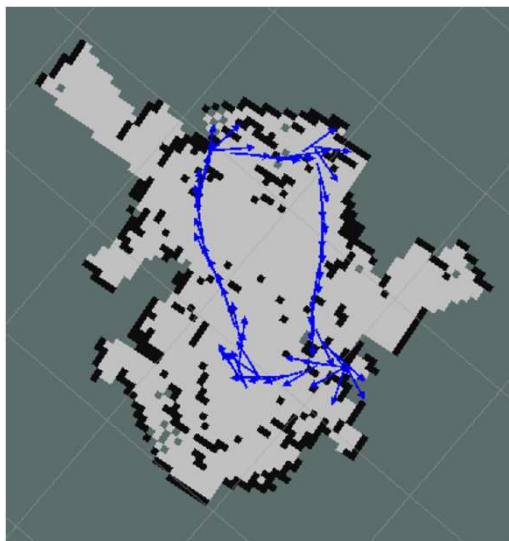
- Cada fichero .msg tiene su correspondiente clase asociada en C++.
- <http://wiki.ros.org/Messages>

# Servicios ROS (Services)

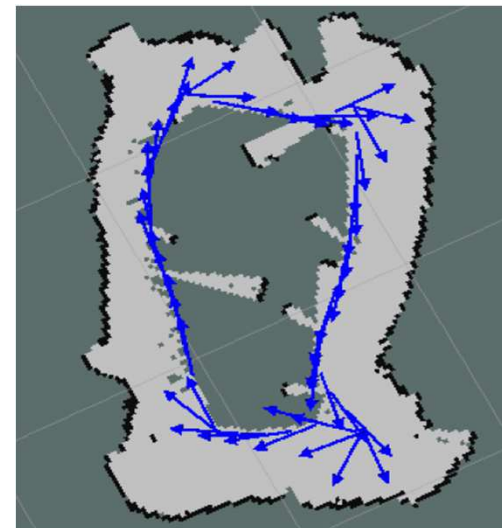
- Cada nodo puede llamar a o suministrar uno o más servicios.
- Usados para hacer transacciones/llamadas síncronas entre nodos
  - RPC: Remote Procedure Calls
- Modelo cliente/servidor
  - comunicación 1 a 1, basada en peticiones-respuestas.
- Roles de los servicios:
  - Llevar a cabo un cálculo remoto (ante una petición).
  - Disparar una funcionalidad/comportamiento concretos.
- P.e: el paquete "explore" provee un servicio llamado "explore\_map" que permite a un usuario externo preguntar por el mapa actual.

# Ros Bags

- Bags almacenan en un fichero datos de mensajes de ROS.
- Usados para reproducir, off-line, el funcionamiento previo de un conjunto de nodos.
- Ejemplo:
  - Reproducir varias veces la trayectoria de un robot reconociendo un mapa.
  - Cambiando en cada experimento la configuración de cómo interpretar sensores.

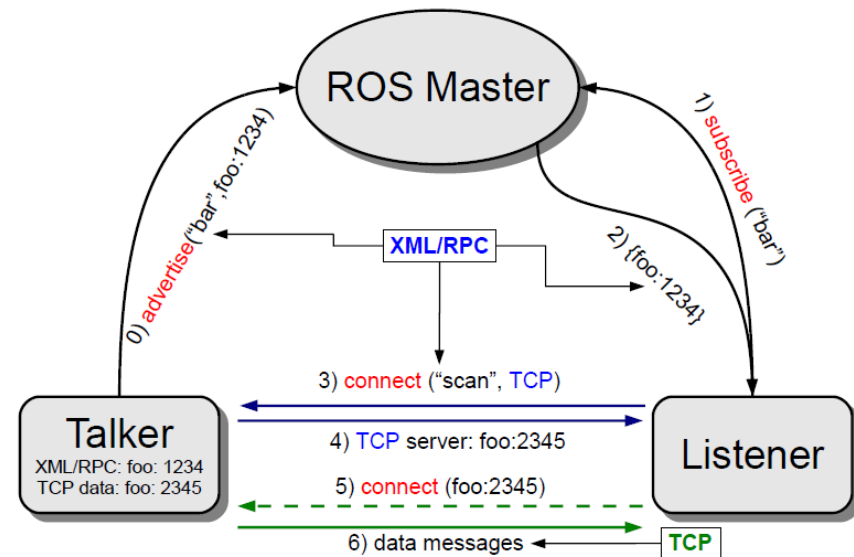


Varios  
Experimentos  
despues ....



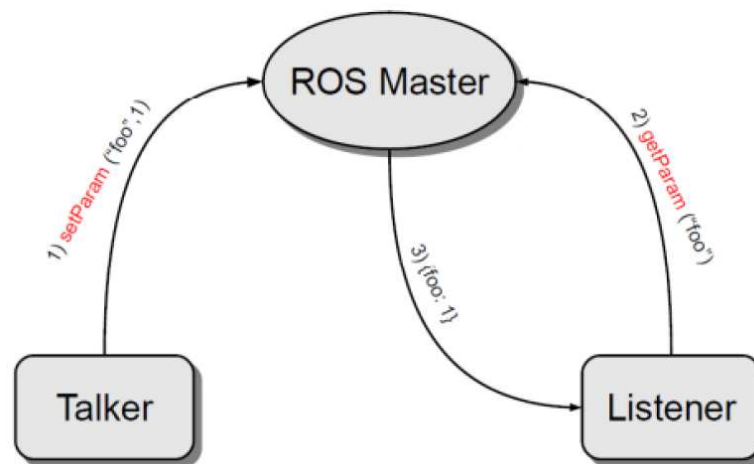
# ROS Master

- Permite que los nodos se puedan localizar unos a otros.
- Ofrece servicios de denominación y registro para
  - nodes, topics, services,...

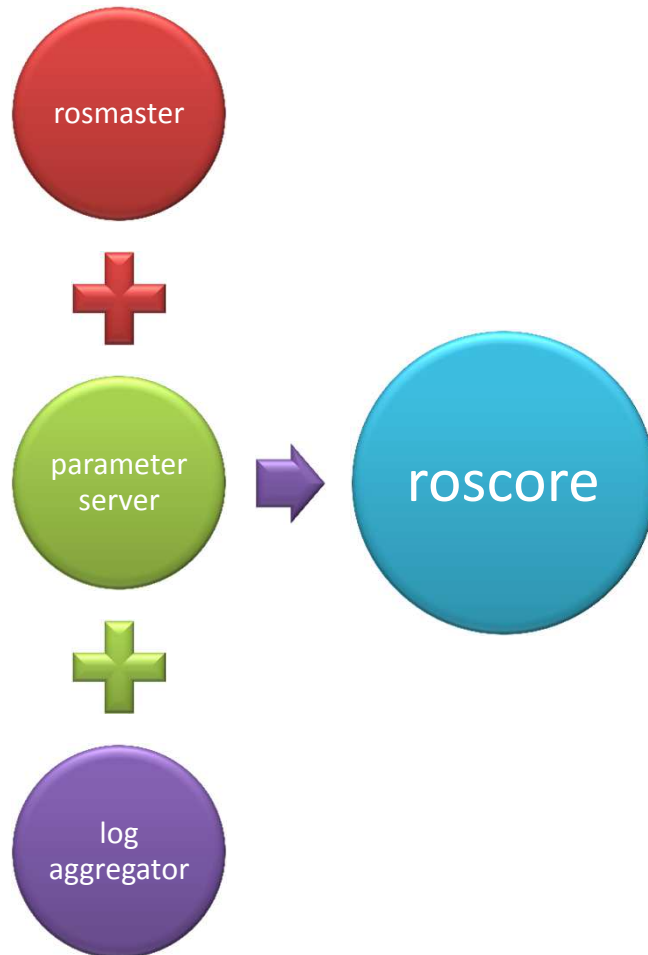


# Servidor de parámetros (Parameter Server)

- Cada nodo puede **asignar** u **obtener** valores de uno o varios parámetros
  - Parámetros públicos , modificables por otros nodos
  - Parámetros privados, modificables por el propio nodo.
- Parameter server: Es un diccionario compartido, accesible via APIS de red.
- Usado para datos estáticos, no binarios, como parámetros de configuración.
- Se ejecuta dentro del ROS master.



# Roscore

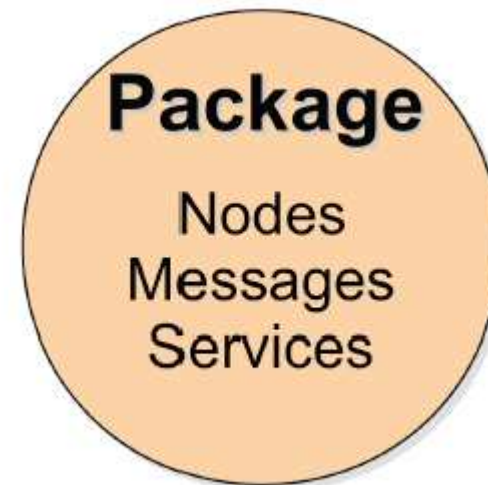


- Rosmaster
  - directorio para anunciadores,subscriptores, servicios
  - no es un nodo de comunicación central
- Parameter Server
  - repositorio centralizado de parámetros, acceso para todos los nodos
- Log aggregator
  - se subscribe al topic `"/out"`
  - almacenar salida en ficheros (log)



# Paquetes ROS

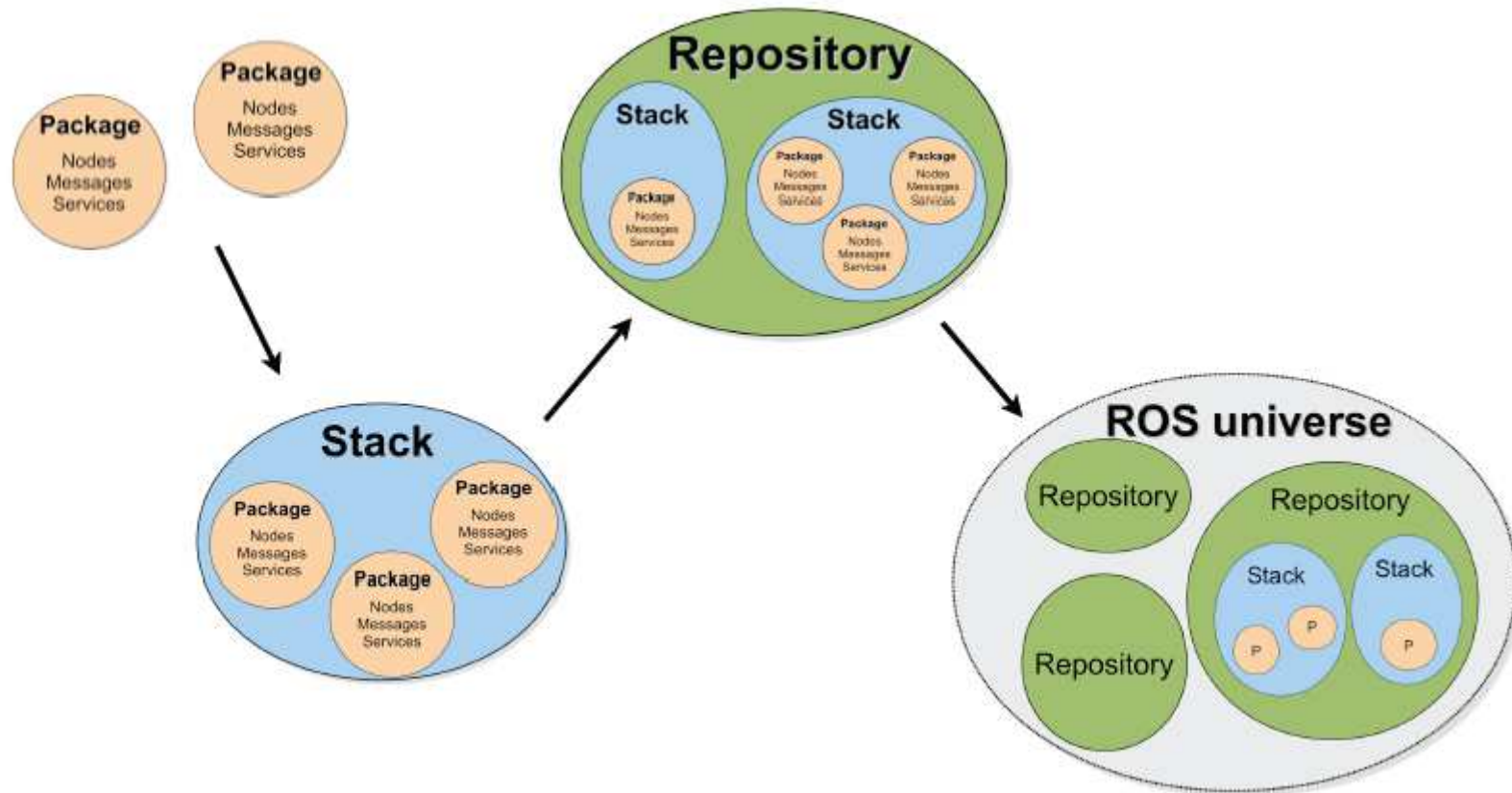
- El software en ROS se organiza en paquetes.
- Un paquete
  - contiene (el código fuente y los ejecutables de) uno o más nodos
  - provee una interfaz para ROS.
- la herramienta para manejo de paquetes se llama **catkin**  
<http://wiki.ros.org/catkin>



# Ejemplos de paquetes

- **explore** - frontier-based exploration.
- **move base** - implements the action of movement to a destination location
- **gmapping** - provides laser-based SLAM (Simultaneous Localization and Mapping) using a grid map.
- **stageros** - implements two-dimensional robot simulation using Stage.
- **gazebo\_ros** – 3D simulator.

# Sistema de paquetes en ROS



Taken from Sachin Chitta and Radu Rusu (Willow Garage)

# Repositorios de paquetes.

- Colecciones de paquetes y stacks.
- Muchos repositorios (> 50):
  - Stanford, CMU, Leuven, USC, ...
- La mayoría alojados en GitHub
- <http://wiki.ros.org/RecommendedRepositoryUsage/CommonGitHubOrganizations>

# Distribuciones ROS

- <http://wiki.ros.org/Distributions>
- La última distribución es Kinetic (creada desde Mayo de 2016 soporte hasta Mayo 2021)
  - Ubuntu Wily (15.10) y Ubuntu Xenial (16.04)  
<http://wiki.ros.org/kinetic/Installation/>
- JADE es una distribución intermedia con corto plazo de soporte. lanzada en Mayo de 2015. Recomendada para las últimas versiones de Ubuntu.
- Indigo Lanzada en Septiembre de 2014 es la recomendada para Ubuntu 14.\*
- Usa un sistema de paquetes llamado
  - CATKIN

# ROS Installation

- Follow the instructions at:
  - <http://wiki.ros.org/indigo/Installation/Ubuntu>
  - <http://wiki.ros.org/kinetic/Installation/Ubuntu>
  - Easy installation, takes about 20 mins
  - Note: Installation takes 2,442MB space of your HD
- To test your installation, follow the ROS tutorials:  
<http://wiki.ros.org/ROS/Tutorials>



# ROS Installation

```
rolyeho@ubuntu: ~  
texlive-latex-extra texlive-latex-extra-doc texlive-latex-recommended  
texlive-latex-recommended-doc texlive-luatex texlive-pictures  
texlive-pictures-doc texlive-pstricks texlive-pstricks-doc tipa tk8.5  
tk8.5-dev tk8.5-lib tk8.6 ttf-liberation ttf-marvosym uuid-dev  
x11proto-composite-dev x11proto-core-dev x11proto-damage-dev  
x11proto-dri2-dev x11proto-fixes-dev x11proto-glx-dev x11proto-input-dev  
x11proto-kb-dev x11proto-randr-dev x11proto-render-dev  
x11proto-scrnsaver-dev x11proto-xext-dev x11proto-xf86vidmode-dev  
x11proto-xinerama-dev xorg-sgml-doctools xtrans-dev zlib1g-dev  
The following packages will be upgraded:  
libasound2 libcurl3 libdbus-1-3 libdrm-intel1 libdrm-nouveau2 libdrm-radeon1  
libdrm2 libegl1-mesa libegl1-mesa-drivers libgcrypt11 libgl1-mesa-glx  
libglapi-mesa libgnutls26 libldap-2.4-2 libpython2.7 libpython2.7-minimal  
libpython2.7-stdlib libqt4-dbus libqt4-declarative libqt4-designer  
libqt4-help libqt4-network libqt4-opengl libqt4-script libqt4-scripttools  
libqt4-sql libqt4-sql-sqlite libqt4-svg libqt4-test libqt4-xml  
libqt4-xmlpatterns libqtcore4 libqtgui4 libssl1.0.0 libtiff5 libx11-6  
libx11-xcb1 libxcb-dri2-0 libxcb-glx0 libxcb-render0 libxcb-shm0 libxcb1  
libxcursor1 libxext6 libxf86vm3 libxi6 libxinerama1 libxml2 libxrandr2  
libxrender1 libxt6 libxxf86vm1 python2.7 python2.7-minimal qdbus  
55 upgraded, 791 newly installed, 0 to remove and 217 not upgraded.  
Need to get 1,055 MB of archives.  
After this operation, 2,442 MB of additional disk space will be used.  
Do you want to continue [Y/n]?
```

# Setting Up ROS Environment

- **Importante!!**
- Add the following line to your bash startup file (typically ~/.bashrc):  

```
source /opt/ros/hydro/setup.bash
```
- Configura valores de variables del entorno ROS. Añadir esta línea garantiza que cada terminal se abrirá con los valores del entorno actualizados.

# Basic ROS Commands

- **roscore, rosrun, rosnod**
- **roscore** – a collection of nodes and programs that are pre-requisites of a ROS-based system
- roscore is defined as:
  - master
  - parameter server
  - rosout
- Usage:
  - \$roscore

# Basic ROS Commands

- **roslaunch** – allows you to run an executable in an arbitrary package without having to cd (or roscd) there first
- Usage:
  - \$roslaunch package executable
- Example
  - Run turtlesim
    - \$roslaunch turtlesim turtlesim\_node

# Basic ROS Commands

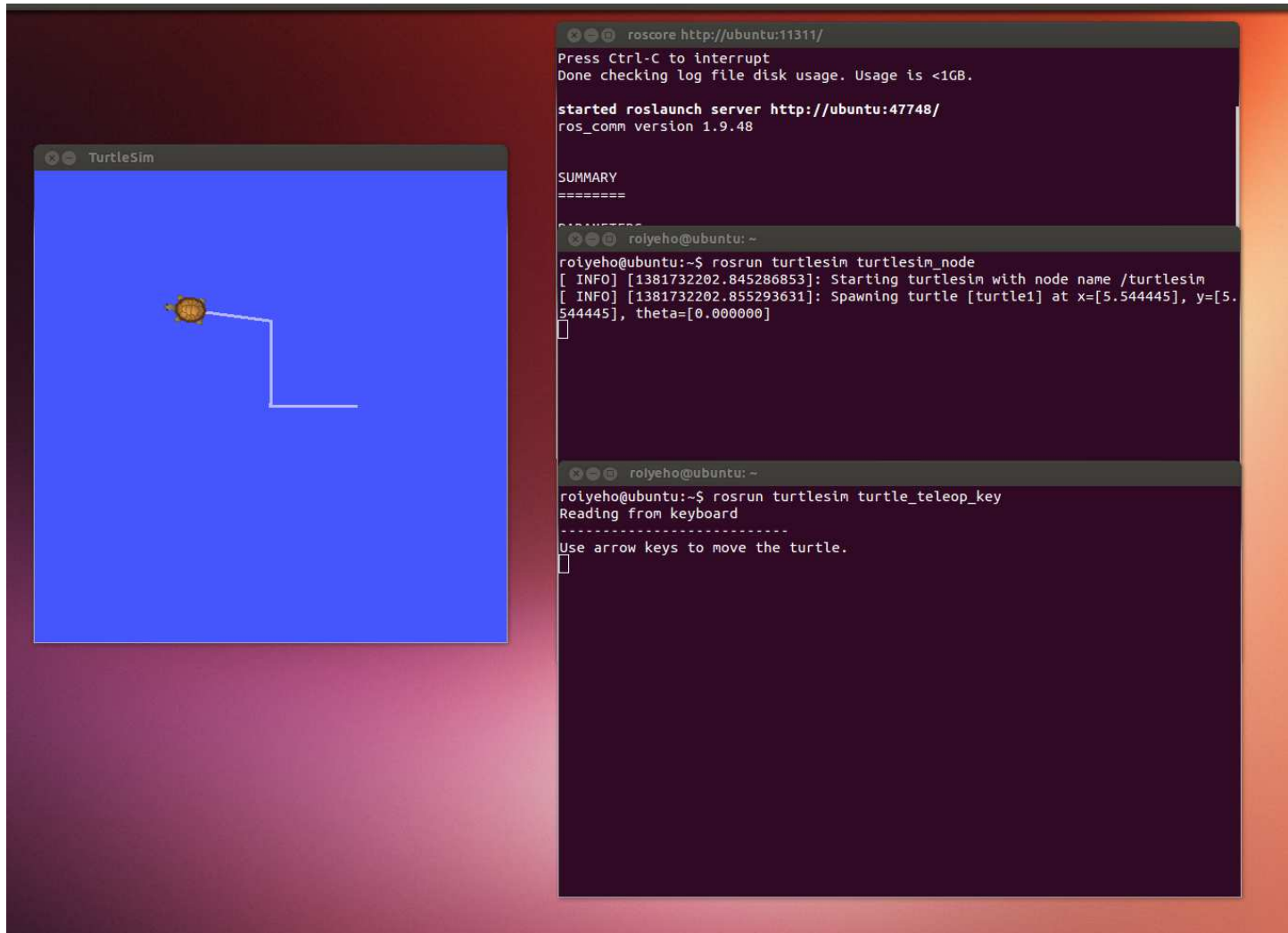
- **rostopic** – Displays debugging information about ROS nodes, including publications, subscriptions and connections
- **Commands:**

Command	
\$rostopic list	List active nodes
\$rostopic ping	Test connectivity to node
\$rostopic info	Print information about a node
\$rostopic kill	Kill a running node
\$rostopic machine	List nodes running on a particular machine

# Demo - Turtlesim

- In separate terminal windows run:
  - `roscore`
  - `roslaunch turtlesim turtlesim_node`
  - `roslaunch turtlesim turtle_teleop_key`

# Demo - Turtlesim

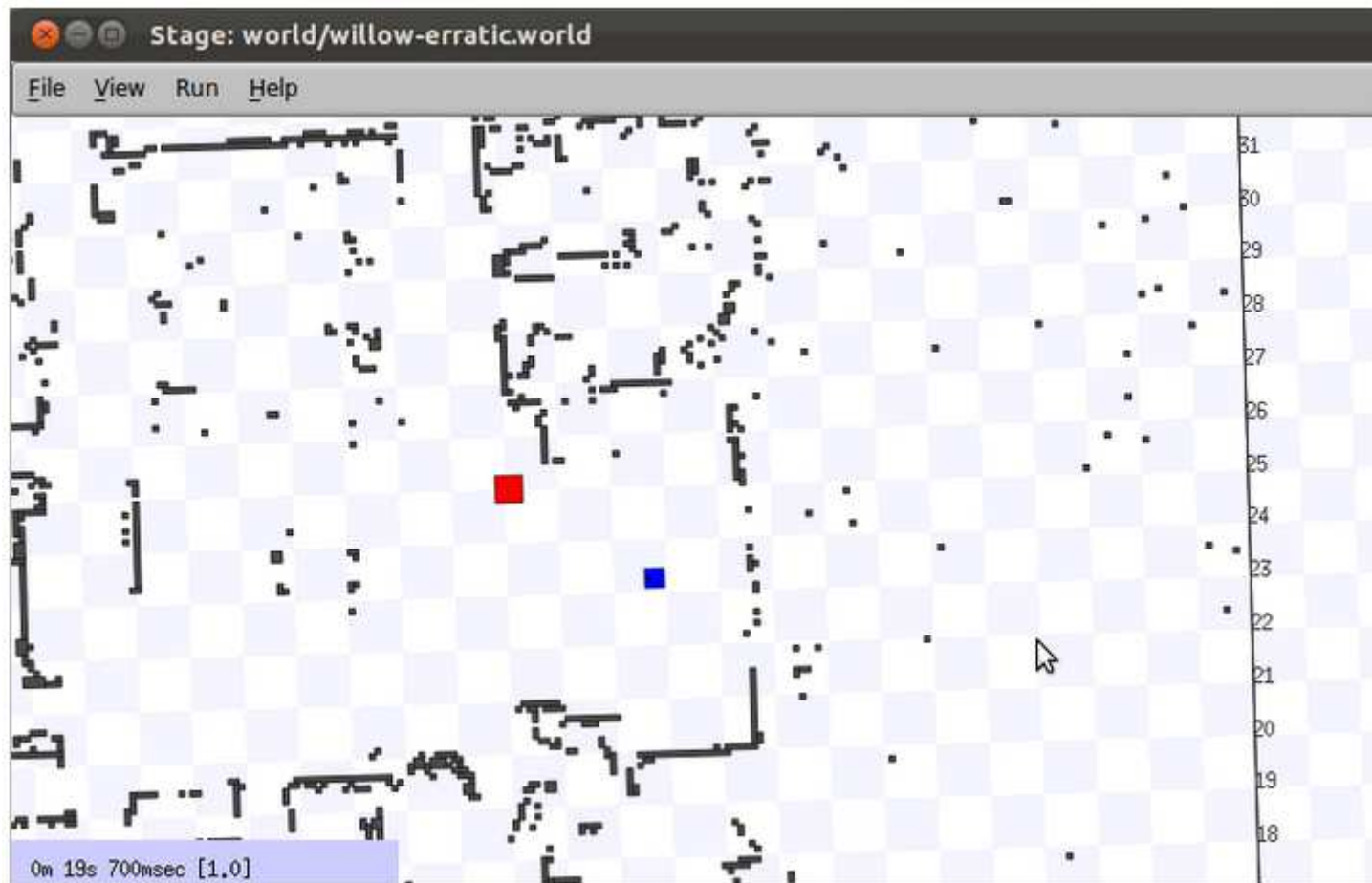


# ROS Simulators

- Stage - 2D Simulator
- Gazebo - 3D Simulator
- En las prácticas usaremos Stage (ver "stageros package" en [wiki.ros.org](http://wiki.ros.org)).



# Stage Simulator



# More Information about ROS

- ROS Cheat Sheet
  - Manual rápido de todos los comandos básicos de ROS.
  - <http://u.cs.biu.ac.il/~veredm/89-689/ROScheatsheet.pdf>
- ROS Tutorial Videos
  - <http://www.youtube.com/playlist?list=PLDC89965A56E6A8D6>

# Para casa durante esta semana

- Instalar ROS
- Leer y realizar todos los "Beginner Level tutorials"
  - <http://wiki.ros.org/ROS/Tutorials>