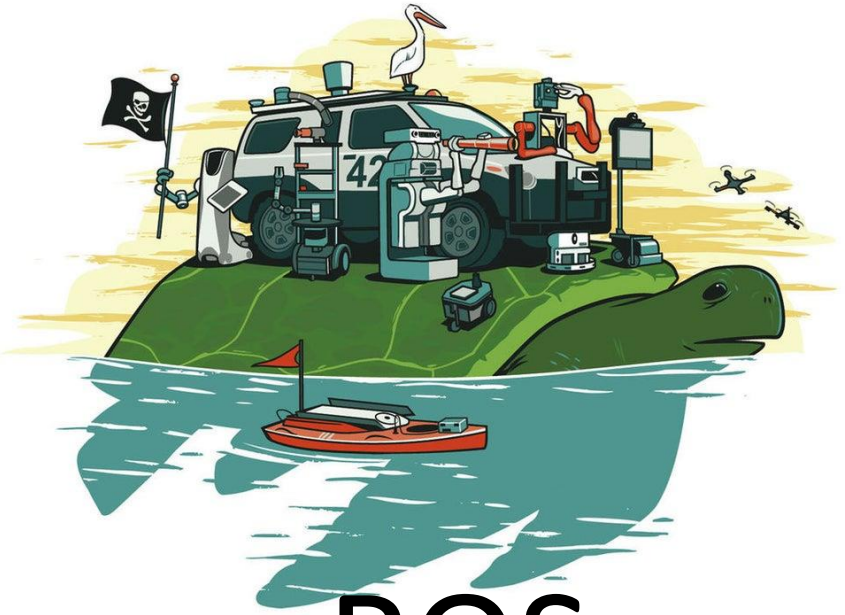
IDONiAL
CENTRO TECNOLÓGICO



Acércate a la robótica con ROS (Robot Operating System)

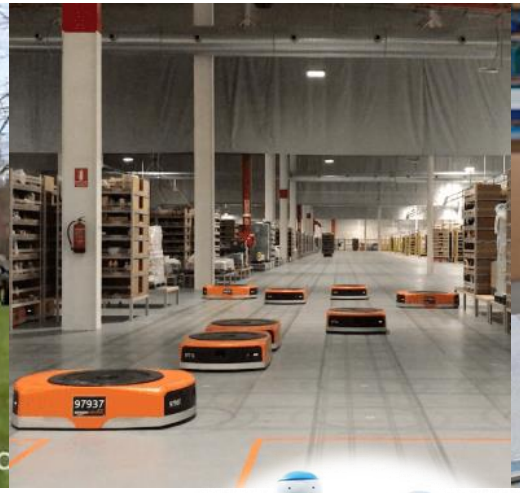
Silvia Rodríguez Jiménez

DevFest Asturias 2019

GDG Asturias

¿En qué piensas cuando oyes “robot”?





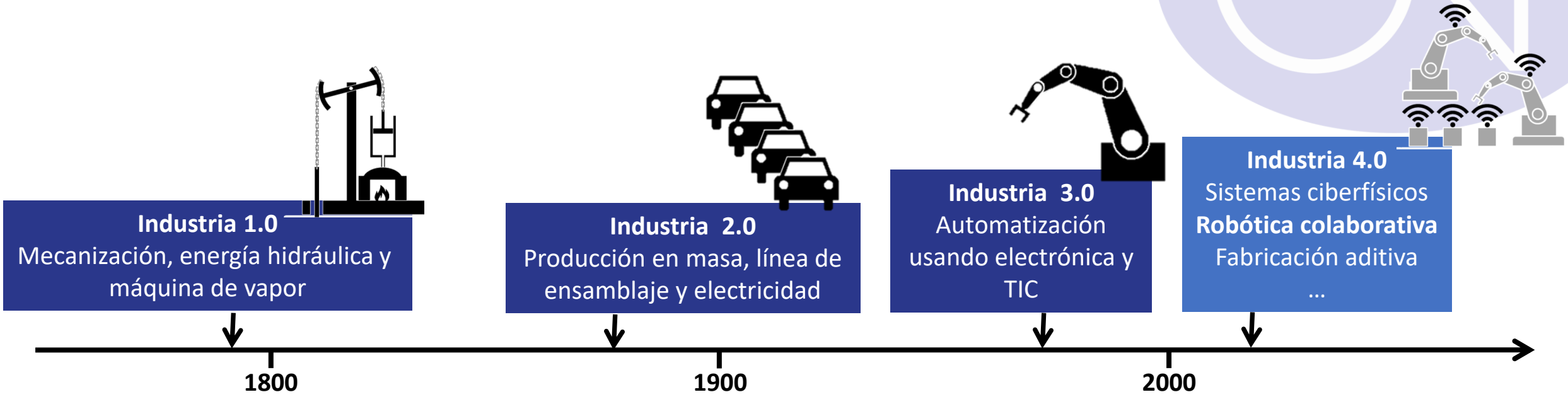
https://youtu.be/YdnJI9T-yXI
https://youtu.be/WFK1qct60nc?t=22
https://youtu.be/4MH7LSLK8Dk

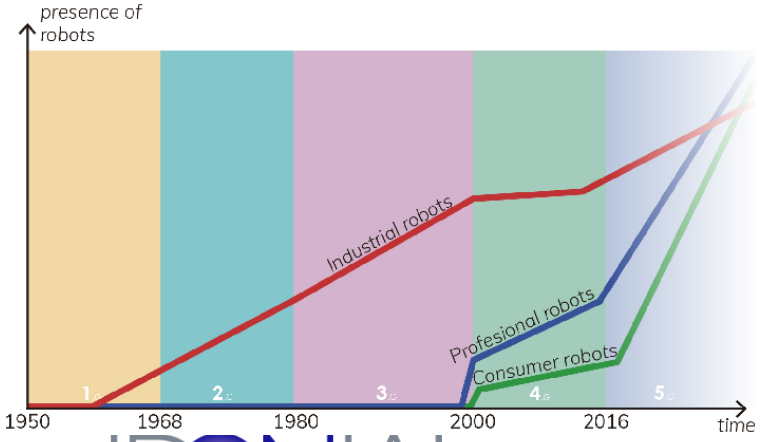



IDONIAL
CENTRO TECNOLÓGICO

[Imágenes: Xataka 2018, Marketing4ecommerce, Mechanik, Technavio, Fanuc, Droide Comunidad, PAL Robotics, Hiroshi Ishiguro Laboratory]

Robots 0.0 → Robots 4.0





0.0: Pre-robots (hasta 1950)

1.0: Manipuladores (1950-1967)

2.0: Robots sensorizados (1968-1977)

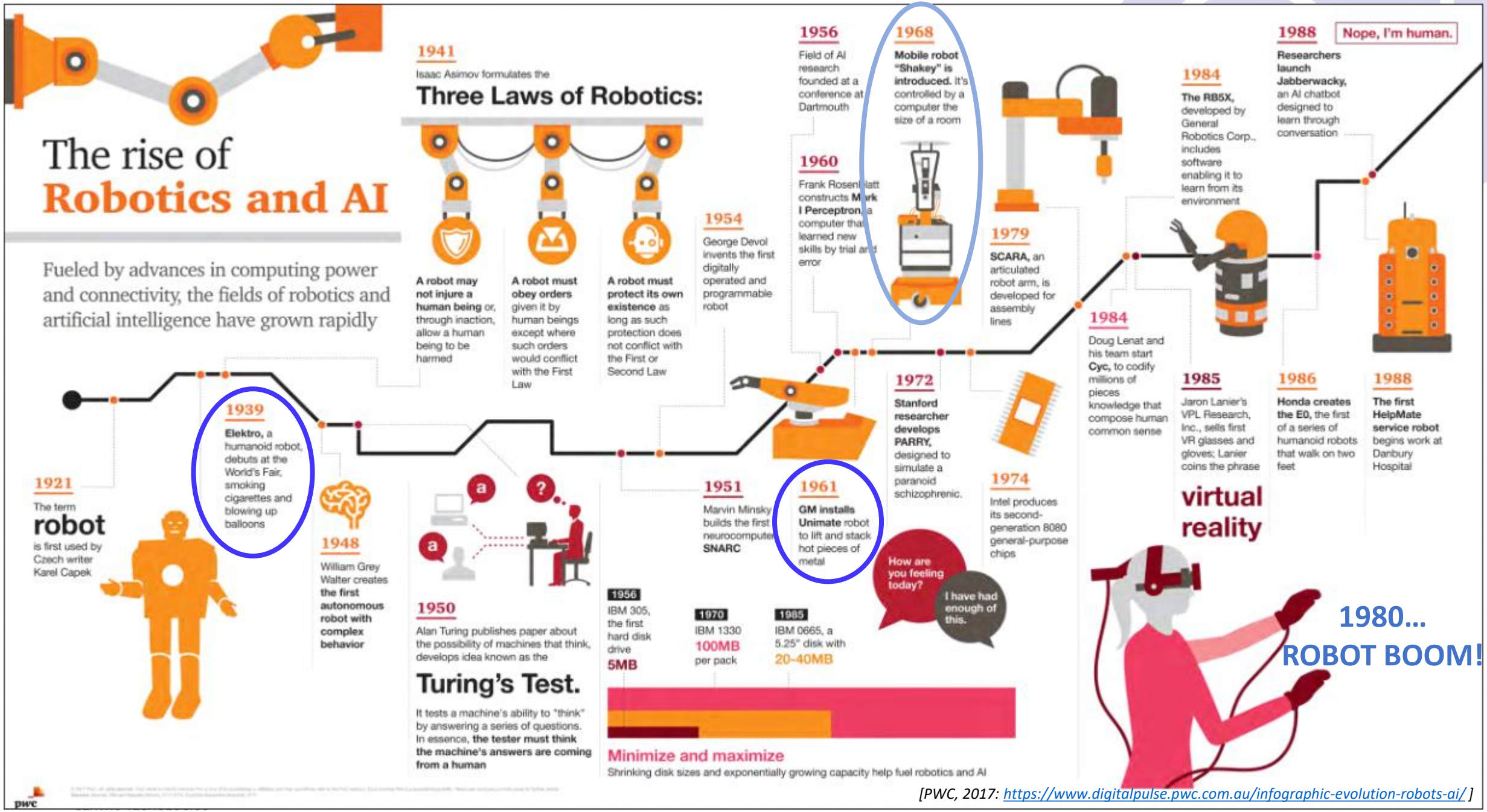
3.0: Robots industriales (1978-1999)

4.0: Robots inteligentes (desde 2000)

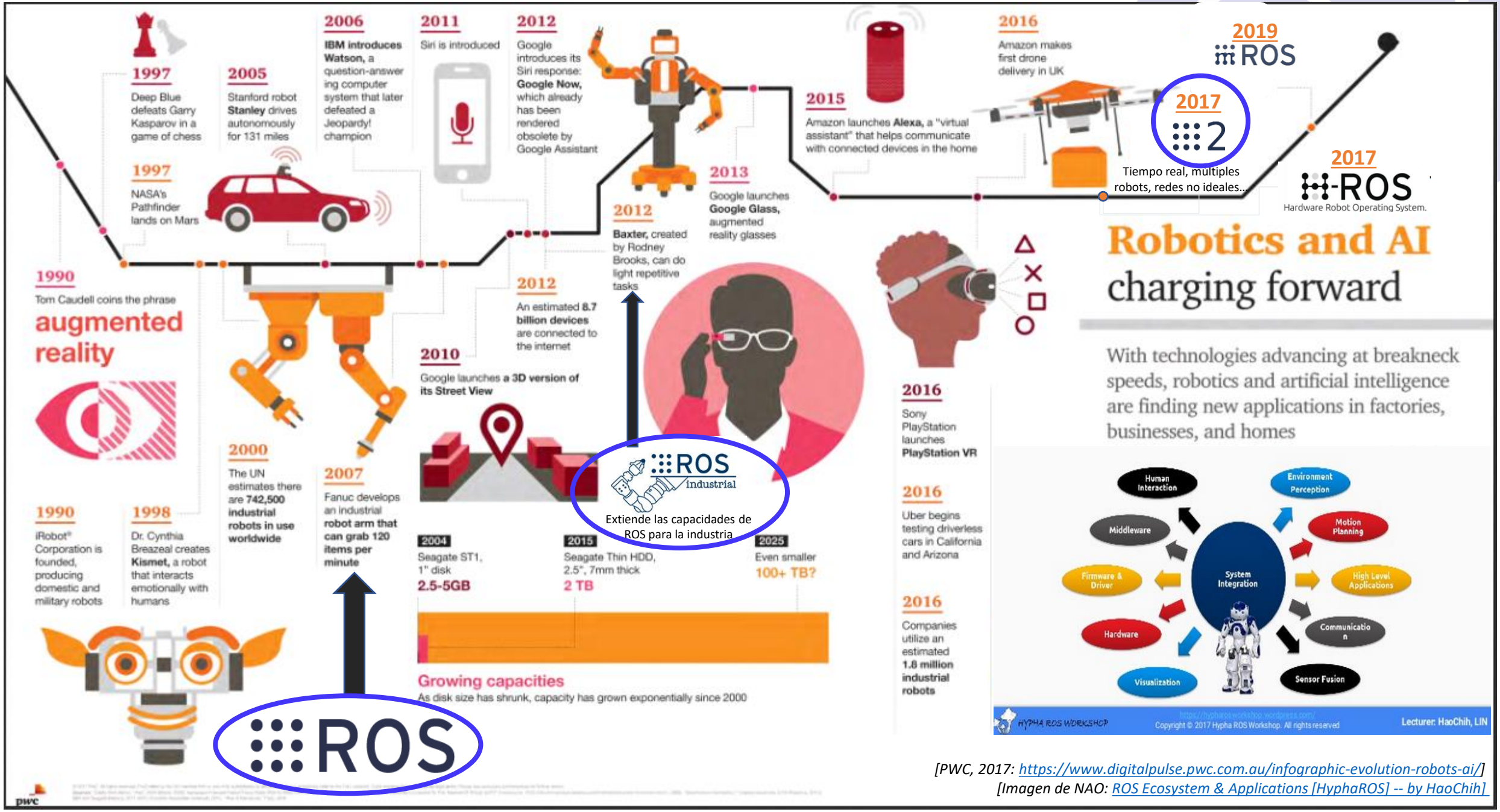
y colaborativos (desde 2008)

IDONIAL CENTRO TECNOLÓGICO

[Imagen basada en "Illustration of Industry 4.0 " de "Christoph Roser at AllAboutLean.com"]
[Gráfica de "Dissecting Robotics — historical overview and future perspectives"]



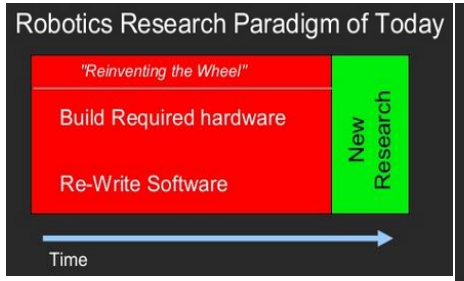
[PWC, 2017: https://www.digitalpulse.pwc.com.au/infographic-evolution-robots-ai/]



[PWC, 2017: https://www.digitalpulse.pwc.com.au/infographic-evolution-robots-ai/]
[Imagen de NAO: ROS Ecosystem & Applications [HyphaROS] -- by HaoChih]

¿Por qué surge ROS?

- Un framework de diseño unificado podría acelerar y simplificar el desarrollo de sistemas robóticos.



Enfoque científico pero aplicado

[Imagen del pitch deck de Eric Berger and Keenan Wyrobek, 2006: https://spectrum.ieee.org/automaton/robotics/robotics-software/the-origin-story-of-ros-the-linux-of-robotics]

- **Sin marco común**
- **Corto periodo de vida**
- **Incapacidad para comparar resultados**
- **Incompatibilidades SW y HW**

[Imagen de Jorge Cham, Ph.D. Comics: http://www.willowgarage.com/blog/2010/04/27/reinventing-wheel]

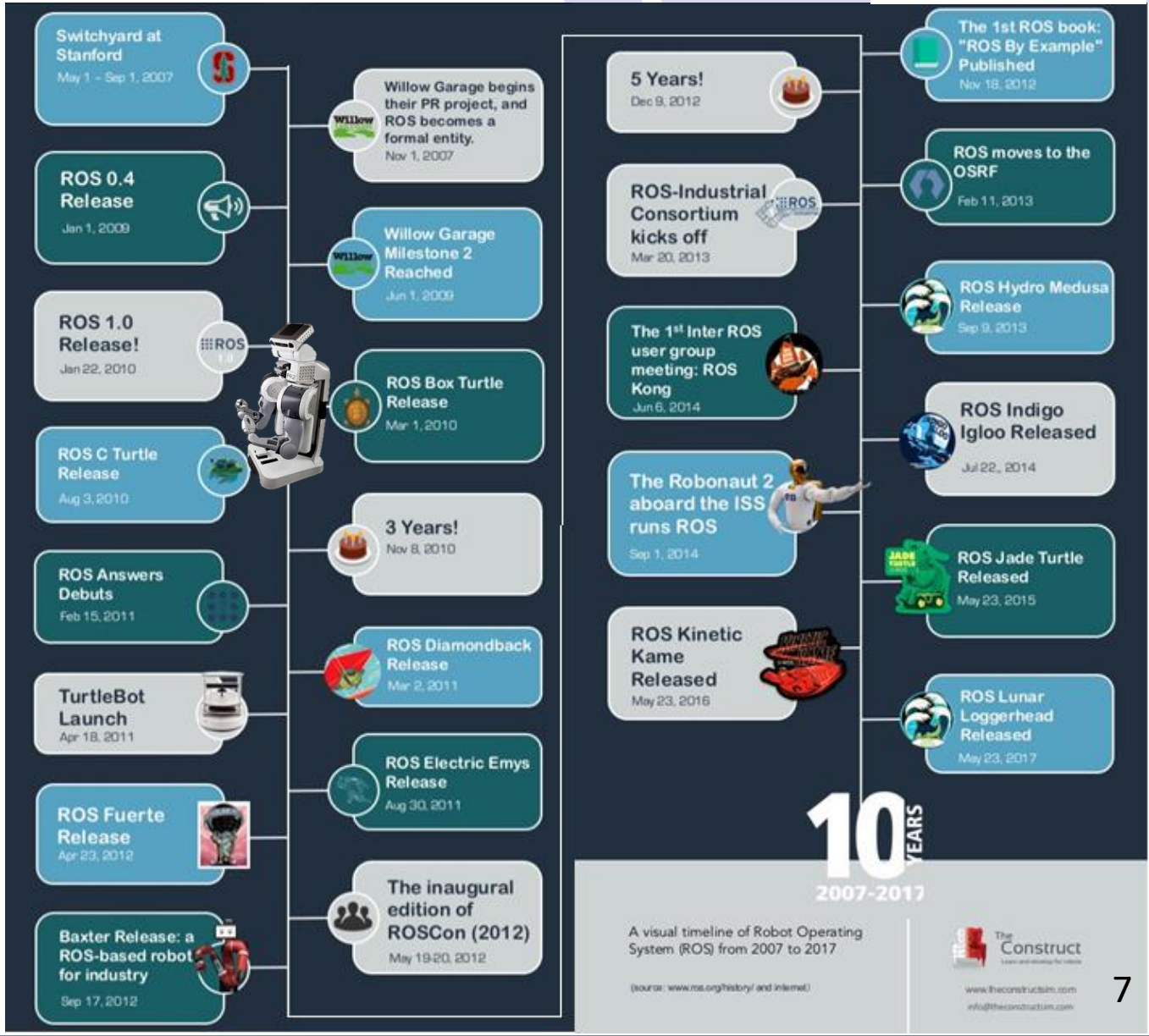
IDONIAL CENTRO TECNOLÓGICO

ROS

- **Open Source** (BSD)
- **Meta-sistema operativo para el desarrollo de sistemas robóticos**: herramientas, bibliotecas y convenciones. No real-time
- Reusabilidad de código, ubicuidad, escalabilidad, fomentar desarrollo SW robótico de forma colaborativa…
- Orientado para UNIX

Distro	Release date	Poster	*Tuturtle*, turtle in tutorial	EOL date
ROS Noetic Ninjemys	May, 2020 (planned, see Upcoming Releases)	TBA	TBA	May, 2025 (planned)
ROS Melodic Morenia (**Recommended**)	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)

http://wiki.ros.org/Distributions



IDONIAL
CENTRO TECNOLÓGICO

ROS, no es solo un framework

- *Plumbing*: infraestructura de mensajes “publish / subscribe”.
- Herramientas: configurar, debugging, visualizar, etc.
- Capacidades: extensa colección de librerías.
- Ecosistema: comunidad en todo el mundo.

[Imagen de https://www.ros.org/about-ros/]

ROS, no es solo un framework

- *Plumbing*: infraestructura de mensajes “publish / subscribe”.
- Herramientas: configurar, debugging, visualizar, etc.
- Capacidades: extensa colección de librerías.
- Ecosistema: comunidad en todo el mundo.

IDONIAL
CENTRO TECNOLÓGICO

[Imagen de https://www.ros.org/about-ros/]

Instalación y setup

- Instalar Ubuntu:
  - **ROS Kinetic** soporta Wily (Ubuntu 15.10), Xenial (Ubuntu 16.04) y Jessie (Debian 8)
  - **ROS Melodic** está dirigido principalmente a Ubuntu 18.04 (Bionic), aunque otros sistemas Linux, así como Mac OS X, Android y Windows son compatibles en diversos grados.
- http://wiki.ros.org/Distributions →http://wiki.ros.org/kinetic
- http://wiki.ros.org/kinetic/Installation: $ sudo apt-get install <name>
  - ¡Importante! https://answers.ros.org/question/325039/apt-update-fails-cannot-install-pkgs-key-not-working/

IDONIAL
CENTRO TECNOLÓGICO

Compilación de paquetes

- **catkin**: sistema de compilación oficial de ROS http://wiki.ros.org/catkin/conceptual_overview
- catkin_make o catkin_build: con catkin_build se consigue un entorno aislado. Son incompatibles.
  - catkin_make: http://wiki.ros.org/catkin
  - catkin_build: https://catkin-tools.readthedocs.io/en/latest/installing.html
- Crear el espacio de trabajo: $ mkdir -p ~/catkin_ws/src
- Build:
  - catkin_make: $ cd ~/catkin_ws/; $ catkin_make
  - catkin_build: $ catkin init; $ catkin build

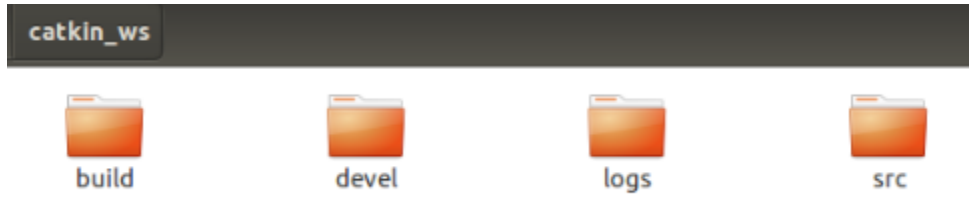
```
$ gedit ~/.bashrc

source /opt/ros/kinetic/setup.bash
source /home/youruser/catkin_ws/devel.setup.bash
export
ROS_PACKAGE_PATH=/home/youruser/catkin_ws/src:/opt/ros/kinetic/sh
are
```

IDONIAL CENTRO TECNOLÓGICO

Sistema de archivos

- **ROS workspace**: carpeta para organizar los archivos de proyecto de ROS.
- **catkin workspace**: espacio de trabajo de ROS donde se utiliza catkin como herramienta de compilación. Carpetas:
  - **build**: archivos binarios compilados.
  - **devel**: archivos de configuración para el entorno ROS del proyecto y todos los ejecutables binarios de su espacio src.
  - **logs**: archivos de log
  - **src**: contiene el código fuente. Esta es la carpeta de trabajo que contiene el software organizado por “ROS packages”.



IDONIAL
CENTRO TECNOLÓGICO

ROS package

- Crear un paquete: http://wiki.ros.org/catkin/Tutorials/CreatingPackage
  - $ cd catkin_ws/src
  - $ catkin_create_pkg <name_package> <deps>
- Se creará una carpeta name_package con **CMakeList.txt** y **package.xml**:
  - scripts: scripts de python.
  - src: archivos C++.
  - srv: servicios.
  - msg: mensajes.
  - action: acciones.
  - launch: archivos launch.

```
cmake_minimum_required(VERSION 2.8.3)
project(sias_lane_follower)

## Compile as C++11, supported in ROS Kinetic and newer
# add_compile_options(-std=c++11)

## Find catkin macros and libraries
## if COMPONENTS list like find_package(catkin REQUIRED COMPONENTS xyz)
## is used, also find other catkin packages
find_package(catkin REQUIRED COMPONENTS
  # cv_bridge
  # opencv2
  roscpp
  rospy
  sensor_msgs
  std_msgs
)
```

```
<!--
  Use doc_depend for packages you need only for building documentation:
-->
<!--   <doc_depend>doxygen</doc_depend> -->
<buildtool_depend>catkin</buildtool_depend>
<build_depend>cv_bridge</build_depend>
<build_depend>opencv2</build_depend>
<build_depend>roscpp</build_depend>
<build_depend>rospy</build_depend>
<build_depend>sensor_msgs</build_depend>
<build_depend>std_msgs</build_depend>
<build_depend>message_generation</build_depend>
```

Si necesitas más dependencias:
$ rosdep install <package_name>

Node

- **Master** (roscore) organiza el registro de nodos, monitoriza tráfico…
- **Node**: unidad de software de procesamiento de datos o información (Python, C++…):
  - Publisher: genera información.
  - Suscriber: recibe información. Usa “topic callback functions” para procesar la información recibida.
  - Los nodos se comunican con *topics*. Comunicación unidireccional de muchos a mucho pero a veces es necesario recibir información sólo cuándo se solicite (*services* y *actions*).

Recuerda hacer tu nodo ejecutable
$ roscd devfest/scripts
$ chmod +x program.py
$ catkin build

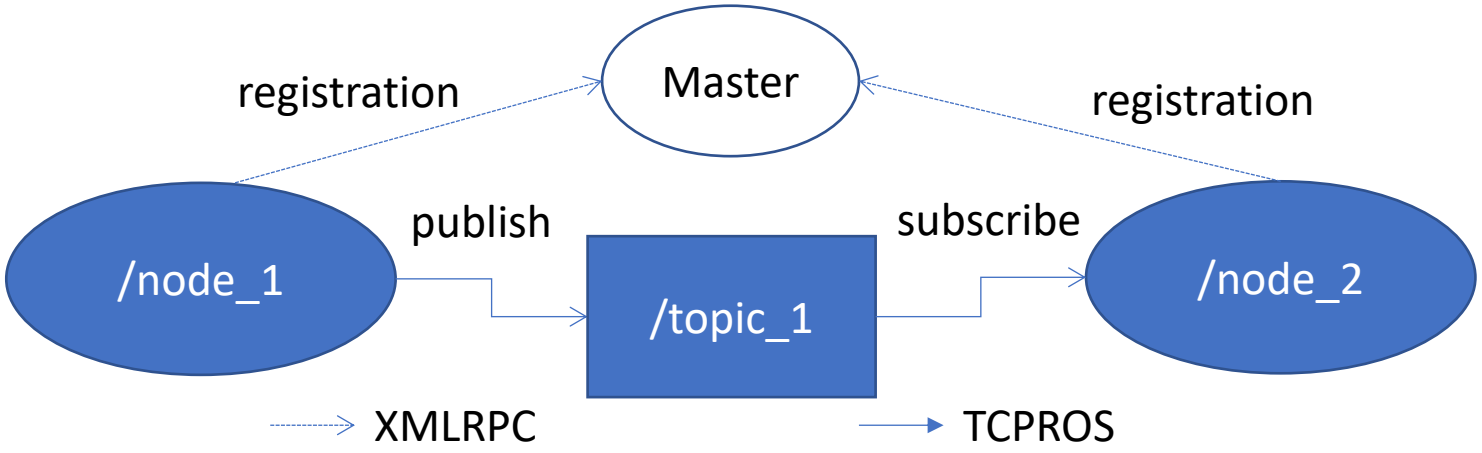
IDONIAL
CENTRO TECNOLÓGICO

Topics

- **Topics**: entidad para transportar información entre nodos.
  - La información se organiza como una estructura de datos (*message type*, .msg): recopilación de diferentes tipos de datos, como string, int, float, etc.
  - Se identifica con un "nombre" y un "tipo“ (estándar o personalizado).

Si creas un nuevo mensaje, acuérdate de añadirlo en la sección correspondiente de CMakeLists.txt

```
## Generate messages in the 'msg' folder
# add_message_files(
#   FILES
#   Message1.msg
#   Message2.msg
# )
```



Launch

- **Launch file**: grupo de múltiples nodos de ROS en un solo archivo (formato XML). Lanza varios nodos (argumentos, parámetros…).

```
-<launch>
  <!-- roslaunch arguments -->
 -<node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="screen">
    <param name="video_device" value="/dev/video0"/>
    <param name="image_width" value="1920"/>
    <param name="image_height" value="1080"/>
    <param name="framerate" value="30"/>
    <param name="video_device" value="/dev/video0"/>
    <param name="pixel_format" value="mjpeg"/>
    <param name="autofocus" value="false"/>
    <param name="focus" value="infinite"/>
    <param name="autoexposure" value="false"/>
    <param name="exposure" value="8"/>
    <param name="auto_white_balance" value="false"/>
    <param name="auto_brigthness" value="false"/>
    <param name="brigthness" value="10"/>
    <param name="camera_frame_id" value="usb_cam"/>
  -<!--
      <param name="io_method" value="mmap"/>  mmap/red/userptr
    -->
  </node>
 -<node name="image_view" pkg="image_view" type="image_view" respawn="false" output="screen">
    <remap from="image" to="/usb_cam/image_raw"/>
    <param name="autosize" value="false"/>
  </node>
</launch>
```

IDONIAL
CENTRO TECNOLÓGICO

Servicios y acciones

- **Servicios**: ejecuta la solicitud en un “service callback”.
  - Son bloqueantes. Comunicación “request-response”.
  - Definido por dos campos (.srv): request y response.
  - Hay 2 tipos de nodos: cliente y servidor.
- **Acciones**: ejecución no bloqueante mediante goalCallback
  - Sistema “request-response” más general.
  - Definido por tres campos (.action): goal, result y feedback.
  - Hay 2 tipos de nodos: cliente y servidor.
  - Funciones:
    - Generar acciones manualmente: $ rosrun actionlib_msgs genaction.py <path_to_action_file>
    - Mostrar contenido de un mensaje: $ rosmsg show <stack_name>_msgs/<ActionMessage>

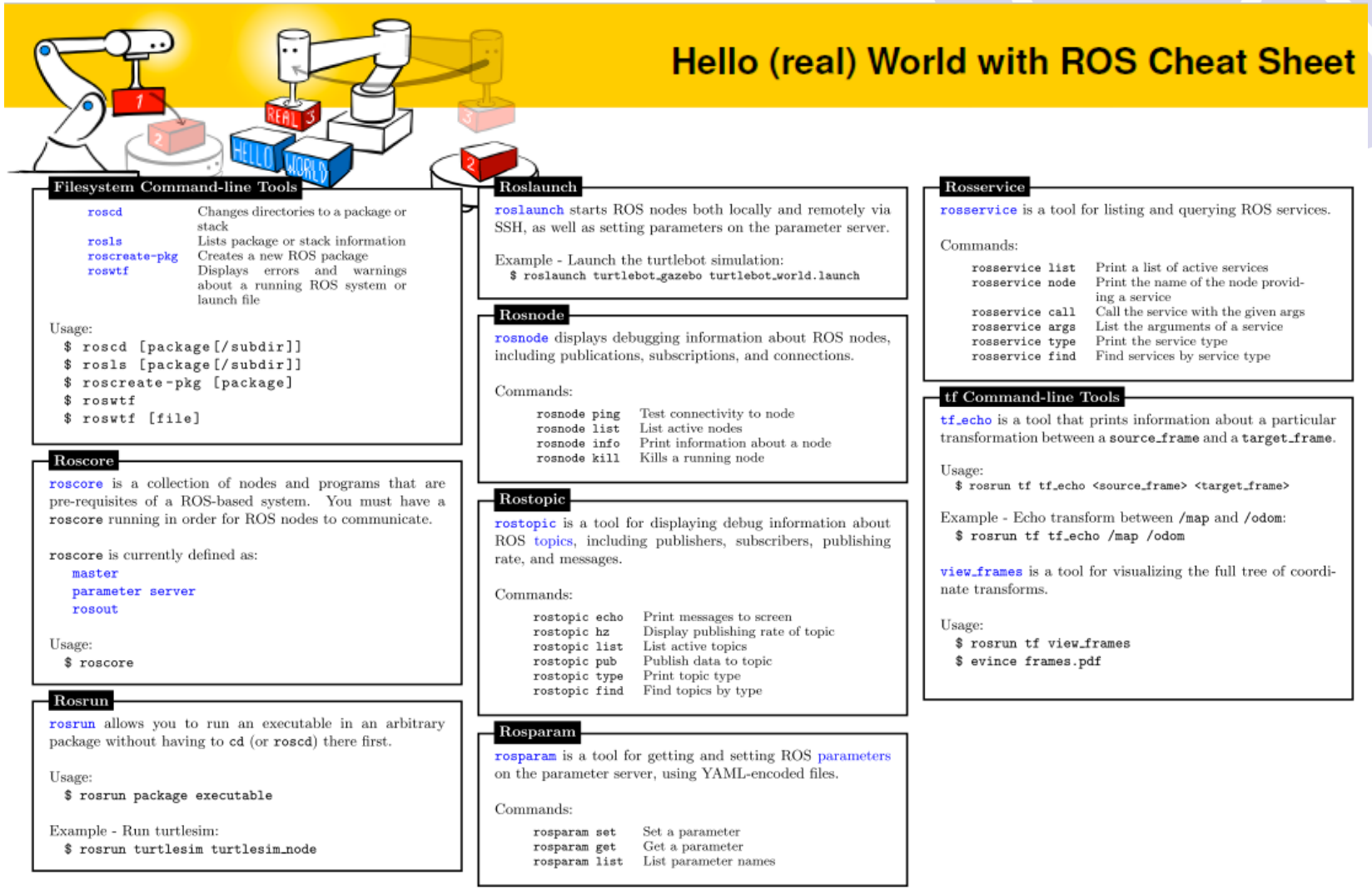
```
## Generate services in the 'srv' folder
# add_service_files(
#   FILES
#   Service1.srv
#   Service2.srv
# )

## Generate actions in the 'action' folder
# add_action_files(
#   FILES
#   Action1.action
#   Action2.action
# )
```

Si utilizas acciones, añade actionlib y actualiza package.xml

Si creas un servicio o una acción, acuérdate de añadirlo en la sección correspondiente de CMakeLists.txt

ROS Cheat Sheet



Hello (real) World with ROS Cheat Sheet

Filesystem Command-line Tools

roscd	Changes directories to a package or stack
rosls	Lists package or stack information
roscreate-pkg	Creates a new ROS package
roswtf	Displays errors and warnings about a running ROS system or launch file

Usage:

```
$ roscd [package[/subdir]]
$ rosls [package[/subdir]]
$ roscreate-pkg [package]
$ roswtf
$ roswtf [file]
```

Roscore

roscore is a collection of nodes and programs that are pre-requisites of a ROS-based system. You must have a roscore running in order for ROS nodes to communicate.

roscore is currently defined as:

- master
- parameter server
- rosout

Usage:

```
$ roscore
```

Rosrun

rosrun allows you to run an executable in an arbitrary package without having to cd (or roscd) there first.

Usage:

```
$ rosrun package executable
```

Example - Run turtlesim:

```
$ rosrun turtlesim turtlesim_node
```

Roslaunch

roslaunch starts ROS nodes both locally and remotely via SSH, as well as setting parameters on the parameter server.

Example - Launch the turtlebot simulation:

```
$ roslaunch turtlebot_gazebo turtlebot_world.launch
```

Rosnode

rosnode displays debugging information about ROS nodes, including publications, subscriptions, and connections.

Commands:

rosnode ping	Test connectivity to node
rosnode list	List active nodes
rosnode info	Print information about a node
rosnode kill	Kills a running node

Rostopic

rostopic is a tool for displaying debug information about ROS topics, including publishers, subscribers, publishing rate, and messages.

Commands:

rostopic echo	Print messages to screen
rostopic hz	Display publishing rate of topic
rostopic list	List active topics
rostopic pub	Publish data to topic
rostopic type	Print topic type
rostopic find	Find topics by type

Rosparam

rosparam is a tool for getting and setting ROS parameters on the parameter server, using YAML-encoded files.

Commands:

rosparam set	Set a parameter
rosparam get	Get a parameter
rosparam list	List parameter names

Rosservice

rosservice is a tool for listing and querying ROS services.

Commands:

rosservice list	Print a list of active services
rosservice node	Print the name of the node providing a service
rosservice call	Call the service with the given args
rosservice args	List the arguments of a service
rosservice type	Print the service type
rosservice find	Find services by service type

tf Command-line Tools

tf_echo is a tool that prints information about a particular transformation between a source_frame and a target_frame.

Usage:

```
$ rosrun tf tf_echo <source_frame> <target_frame>
```

Example - Echo transform between /map and /odom:

```
$ rosrun tf tf_echo /map /odom
```

view_frames is a tool for visualizing the full tree of coordinate transforms.

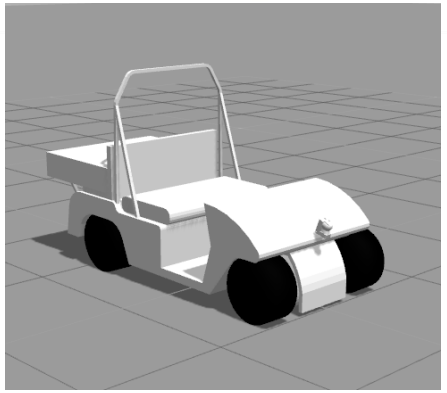
Usage:

```
$ rosrun tf view_frames
$ evince frames.pdf
```

[Cheat Sheet del curso "Hello (Real) World with ROS – Robot Operating System": https://www.edx.org/es/course/hello-real-world-with-ros-robot-operating-system-2]

Construir tu propio entorno robótico

- **URDF**: formato de ROS para describir el layout de un robot. Tipo XML.
  - Formado por links y joints (fixed, revolute, continuous, prismatic, planar, floating)
- **XACRO**: URDF macro para generar URDF.
  - Permite más modularidad y reusar código
  - Convertir a URDF: $ rosrun xacro /path/to/robot.xacro > robot.urdf
- Suele haber un link que une todo: base_link o world.
- Si quieres modificar un modelo:
  - Añadir componente a archivo.urdf.xacro para indicar la posición
  - En /urdf/name_componente se añade el name_componente.urdf.xacro
  - Se añade el .stl o .dae a la carpeta de meshes



http://wiki.ros.org/Robots/RBCAR

Sistema multi computadora

- Necesitas configurar cada máquina indicando cómo conectarse al ROS_MASTER:
  - Conexión a la misma red
  - Añade en /etc/hosts: IP_PC_MASTER <name_PC_MASTER>
  - En el terminal: export ROS_MASTER_URI=http://name_PC_MASTER:11311
- Necesitas que el PC_MASTER sepa qué máquinas se van a conectar:
  - Añade en /etc/hosts: IP_PC <name_PC>

IDONIAL
CENTRO TECNOLÓGICO

ROS, no es solo un framework

- Plumbing: infraestructura de mensajes “publish / subscribe”.
- Herramientas: configurar, debugging, visualizar, etc.
- Capacidades: extensa colección de librerías.
- Ecosistema: comunidad en todo el mundo.

IDONIAL
CENTRO TECNOLÓGICO

[Imagen de https://www.ros.org/about-ros/]

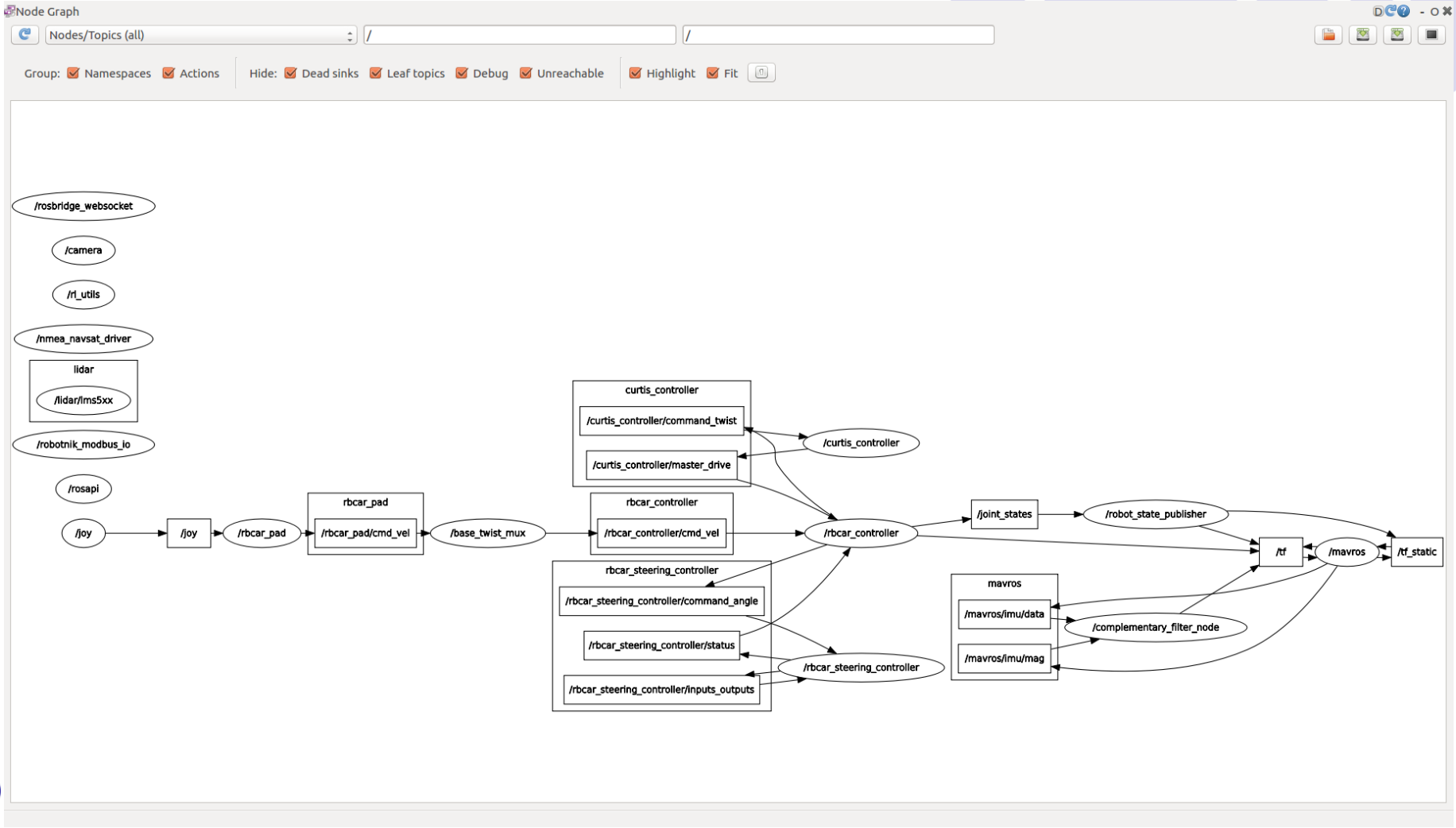
Tools

- Herramientas estándar de Linux: compiladores, debuggers, loggers, IDEs.
  - Por ejemplo: https://ros-qtc-plugin.readthedocs.io/en/latest/_source/How-to-Install-Users.html
- Múltiples lenguajes: C++, Python, Lisp, Java, C#, etc.
- Librerías estándar: Boost, MySQL, XML…
- Gráficas, diagnóstico, simulación / visualización…:
  - 10 useful ROS tools (https://ros.guru/2018/02/12/10-useful-ros-tools/): rqt_console, rqt_logger_level, rqt_launchtree, rqt, Message Thief, status panel, **rviz**, rosnode info, rqt_top, roswtf
  - **Gazebo** (simulación gráfica 3D), rqt_graph, rosbag, tf, Openrace…

IDONIAL CENTRO TECNOLÓGICO

rqt_graph

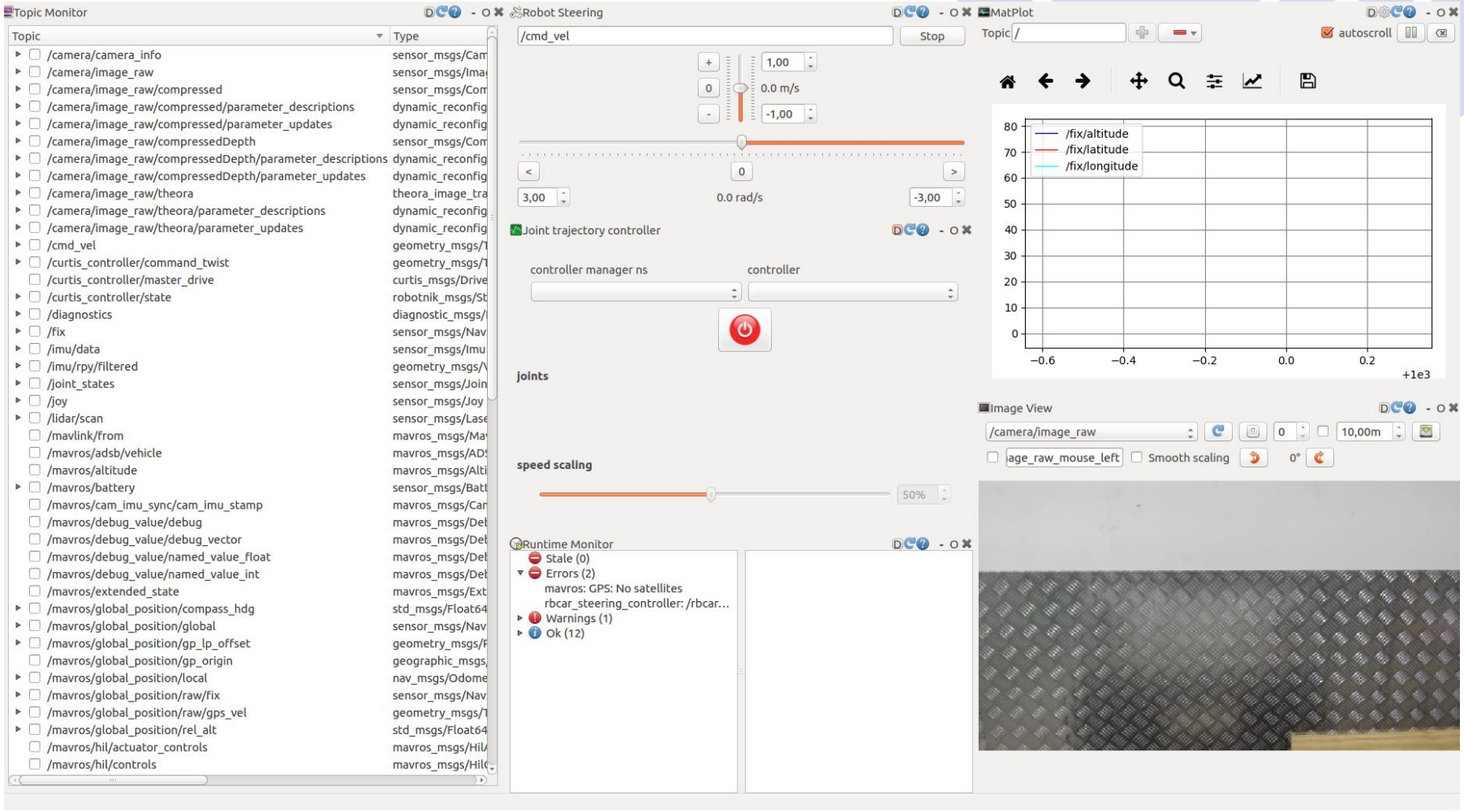
http://wiki.ros.org/rqt_graph



[Imagen del coche: https://www.robotnik.es/robots-moviles/rb-car/]

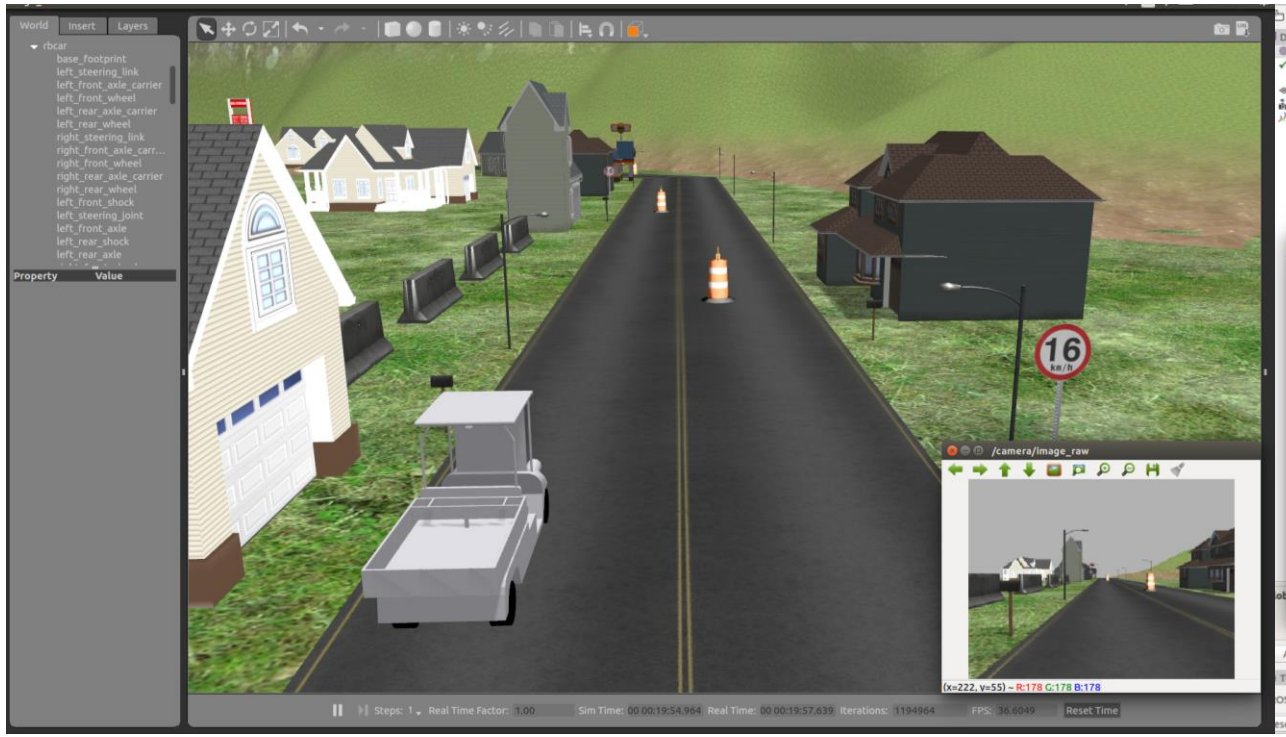
rqt

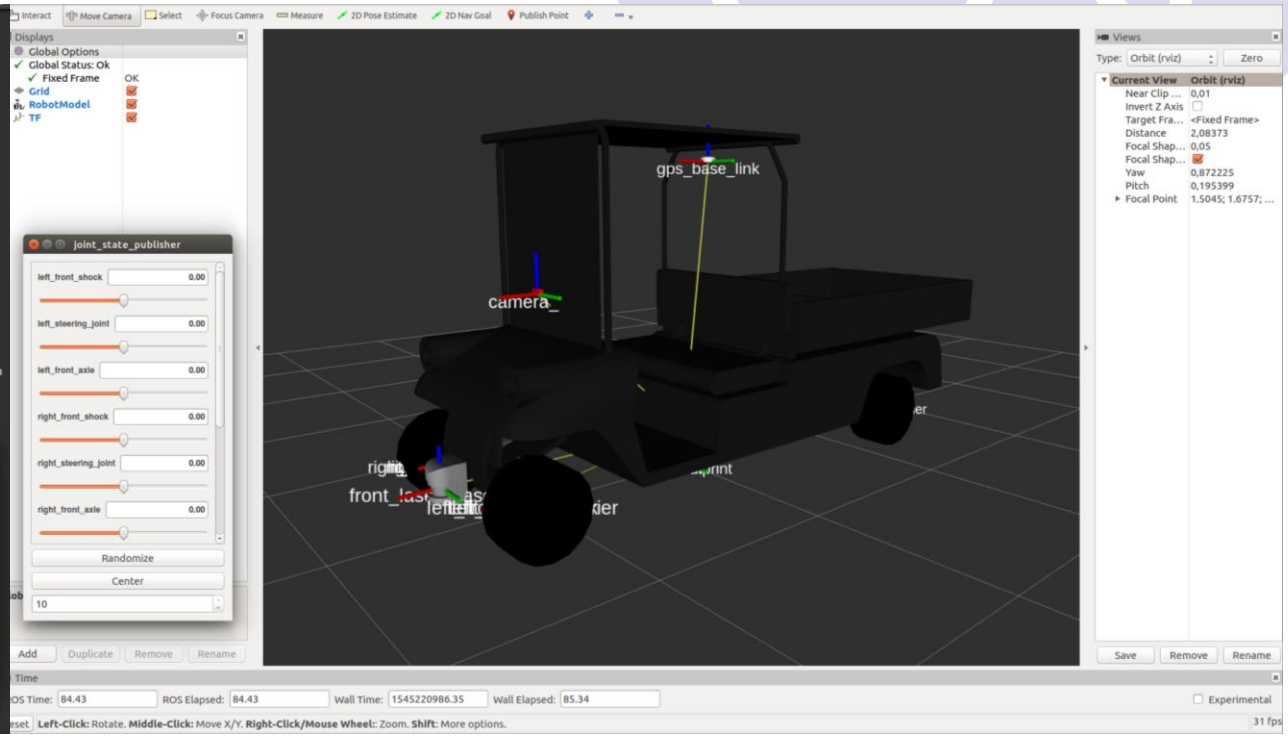
http://wiki.ros.org/rqt



[Imagen del coche: https://www.robotnik.es/robots-moviles/rb-car/]

Gazebo y rviz (I)



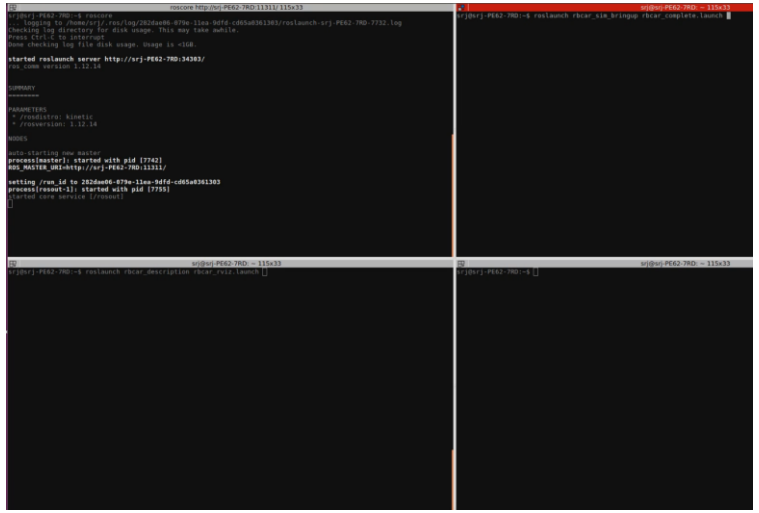


También puedes grabar los ROS topics que recibe el nodo roscore y luego reproducirlos:

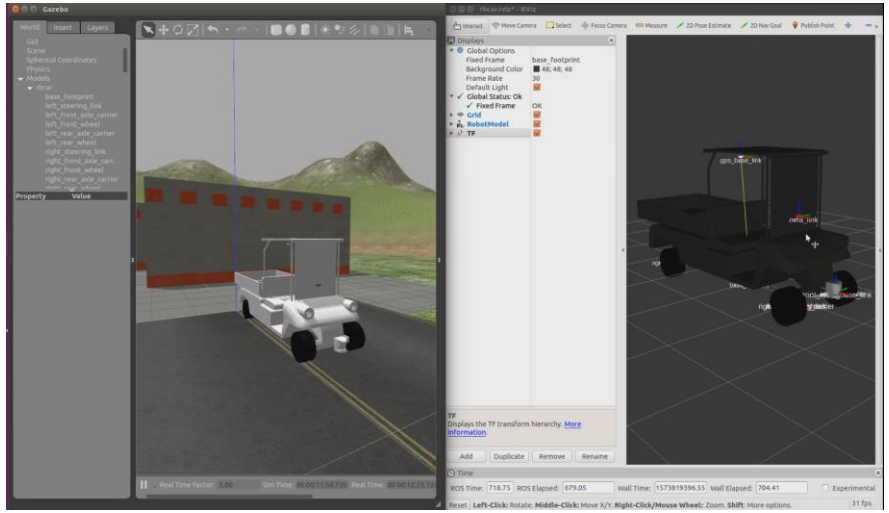
```
$rosbag record -a
$ rosbag play archive_rosbag.bag
```

IDONIAL
CENTRO TECNOLÓGICO

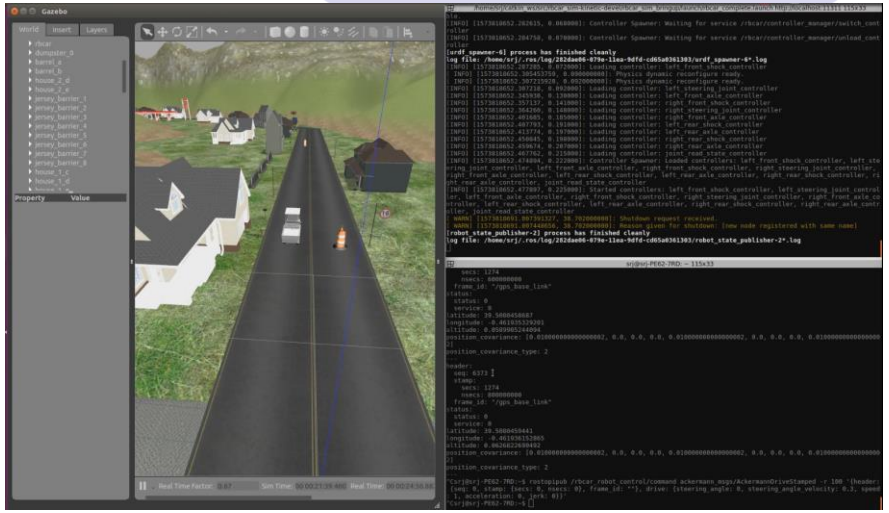
Gazebo y rviz (II)



Video 1: lanzar gazebo y rviz



Video 2: rviz

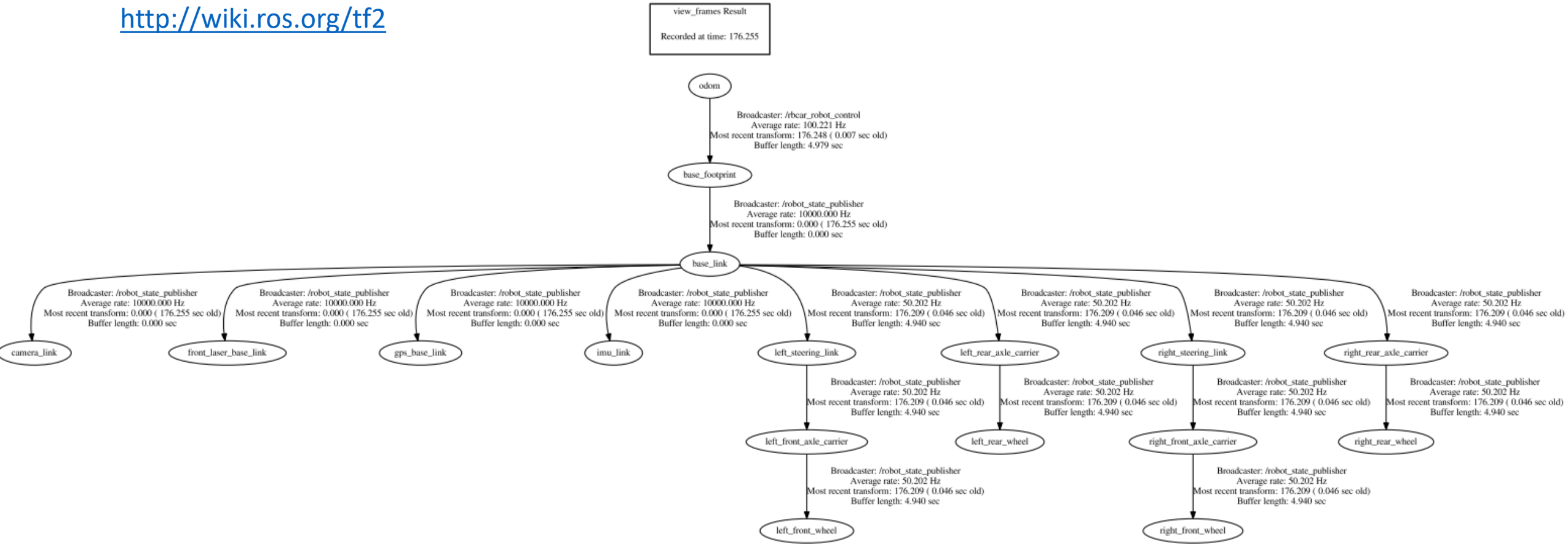


Video 3: Gazebo

IDONIAL
CENTRO TECNOLÓGICO

tf

http://wiki.ros.org/tf
http://wiki.ros.org/tf2



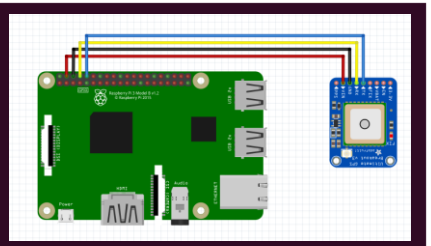
$ rosrun tf view_frames

Ejemplo con Raspberry Pi3

- GPS: Publisher y Subscriber

$ rosrun robot_nav NavSatFix_publisher.py
$ rostopic echo /navsat

```
---
header:
  seq: 36
  stamp:
    secs: 1505784946
    nsecs: 693011999
  frame_id: world
status:
  status: 1
  service: 0
latitude: 4338.9201
longitude: 11619.7482
altitude: 2651.902972
position_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
position_covariance_type: 0
---
```



Message from the NavSatFix Publisher

CMakeList.txt:
Find_package(catkin REQUIRED
COMPONENTS
 roscpp
 rospy
 sensor_msg)

Package.xml:
<buildtool_depend>catkin</buildtool_depend>
<build_depend>roscpp</build_depend>
<build_depend>rospy</build_depend>
<build_depend> sensor _msg</build_depend>

sensor_msgs/NavSatFix Message

File: sensor_msgs/NavSatFix.msg

Raw Message Definition

```
# Navigation Satellite fix for any Global Navigation Satellite System
#
# Specified using the WGS 84 reference ellipsoid

# header.stamp specifies the ROS time for this measurement (the
#        corresponding satellite time may be reported using the
#        sensor_msgs/TimeReference message).
#
# header.frame_id is the frame of reference reported by the satellite
#        receiver, usually the location of the antenna.  This is a
#        Euclidean frame relative to the vehicle, not a reference
#        ellipsoid.
Header header

# satellite fix status information
NavSatStatus status

# Latitude [degrees]. Positive is north of equator; negative is south.
float64 latitude

# Longitude [degrees]. Positive is east of prime meridian; negative is west.
float64 longitude

# Altitude [m]. Positive is above the WGS 84 ellipsoid
# (quiet NaN if no altitude is available).
float64 altitude

# Position covariance [m^2] defined relative to a tangential plane
# through the reported position. The components are East, North, and
# Up (ENU), in row-major order.
#
# Beware: this coordinate system exhibits singularities at the poles.

float64[9] position_covariance

# If the covariance of the fix is known, fill it in completely. If the
# GPS receiver provides the variance of each measurement, put them
# along the diagonal. If only Dilution of Precision is available,
# estimate an approximate covariance from that.

uint8 COVARIANCE_TYPE_UNKNOWN = 0
uint8 COVARIANCE_TYPE_APPROXIMATED = 1
uint8 COVARIANCE_TYPE_DIAGONAL_KNOWN = 2
uint8 COVARIANCE_TYPE_KNOWN = 3

uint8 position_covariance_type
```

NavSatFix_publisher.py

```python
#!/usr/bin/env python
import rospy
import serial
import pynmea2
from sensor_msgs.msg import NavSatFix


s = serial.Serial('/dev/ttyS0', 9600)
if s.isOpen() is False:
    sys.exit(1)


def gps_talker():
    pub = rospy.Publisher('navsat', NavSatFix, queue_size=10)
    rospy.init_node('gps_talker', anonymous=True)
    msg = NavSatFix()
    seq = 0
    while not rospy.is_shutdown():
        line = s.readline()
        data = pynmea2.parse(line)
        if line[1:5] == 'GPRM':
            seq += 1
            msg.header.seq = seq
            msg.header.stamp = rospy.Time.now()
            msg.header.frame_id = 'world'
            if data.status == 'A':
                msg.status.status = int(1)
            else:
                msg.status.status = int(0)
            msg.status.service = int(0)
            msg.latitude = float(data.lat)
            msg.longitude = float(data.lon)
            # data.true_course=142.46
            # data.spd_over_grnd
        if line[1:5] == 'GPGG':
            seq += 1
            msg.header.seq = seq
            msg.latitude = float(data.lat)
            msg.longitude = float(data.lon)
            msg.altitude = float(data.altitude)*3.28084
        pub.publish(msg)


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

[Imágenes de: https://medium.com/@ericmaggard/building-an-autonomous-car-using-a-1-10th-scale-rc-car-part-2-38aa8ee9abed, http://docs.ros.org/melodic/api/sensor_msgs/html/msg/NavSatFix.html]

ROS, no es solo un framework

- Plumbing: infraestructura de mensajes “publish / subscribe”.
- Herramientas: configurar, debugging, visualizar, etc.
- Capacidades: extensa colección de librerías.
- Ecosistema: comunidad en todo el mundo.

IDONIAL
CENTRO TECNOLÓGICO

[Imagen de https://www.ros.org/about-ros/]

Capacidades

- Algoritmos del estado del arte
- Rápido desarrollo desde la comunidad
- Paquetes de ROS: gran colección de paquetes con funcionalidades: movilidad, manipulación, percepción, etc.
- Más de 2000 paquetes open-source disponibles
- Reutilización

[Imagen de ROS Ecosystem & Applications [HyphaROS] -- by HaoChih]

ROS, no es solo un framework

- Plumbing: infraestructura de mensajes “publish / subscribe”.
- Herramientas: configurar, debugging, visualizar, etc.
- Capacidades: extensa colección de librerías.
- Ecosistema: comunidad en todo el mundo.

IDONIAL
CENTRO TECNOLÓGICO

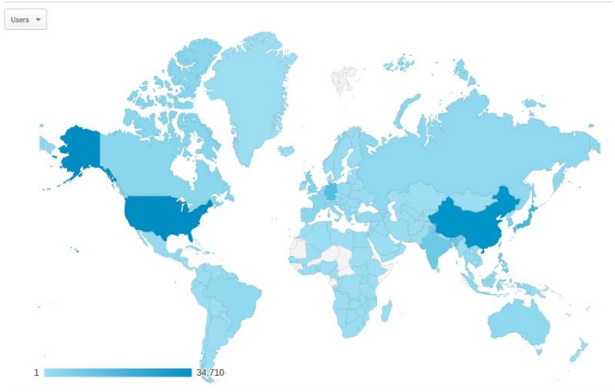
[Imagen de https://www.ros.org/about-ros/]

Ecosistema

- On-line:
  - Wiki: http://wiki.ros.org/es
  - Answers: https://answers.ros.org/questions/
  - ROS discourse Forum: https://discourse.ros.org/
  - …
- Off-line:
  - ROSCon: https://roscon.ros.org/
  - SIG meetups, tutorial, local events…

1.	United States	34,710	(19.08%)
2.	China	31,946	(17.56%)
3.	Japan	15,518	(8.53%)
4.	Germany	12,711	(6.99%)
5.	India	8,400	(4.62%)
6.	Philippines	7,235	(3.98%)
7.	South Korea	6,790	(3.73%)
8.	United Kingdom	4,325	(2.38%)
9.	Taiwan	4,233	(2.33%)
10.	France	3,725	(2.05%)
11.	Canada	3,354	(1.84%)
12.	Spain	2,955	(1.62%)
13.	Singapore	2,842	(1.56%)
14.	Italy	2,744	(1.51%)
15.	Russia	2,465	(1.35%)
16.	Indonesia	2,461	(1.35%)
17.	Australia	2,436	(1.34%)
18.	Brazil	2,231	(1.23%)
19.	Hong Kong	2,147	(1.18%)
20.	Turkey	1,928	(1.06%)
21.	Netherlands	1,511	(0.83%)
22.	Thailand	1,437	(0.79%)
23.	Poland	1,335	(0.73%)
24.	Switzerland	1,242	(0.68%)
25.	Vietnam	1,125	(0.62%)

wiki.ros.org visitor locations:



Source: Google Analytics
Site: wiki.ros.org in July 2018

6

Ecosistema: ROS Robots

ROS 10 CELEBRATING TEN YEARS

https://robots.ros.org/

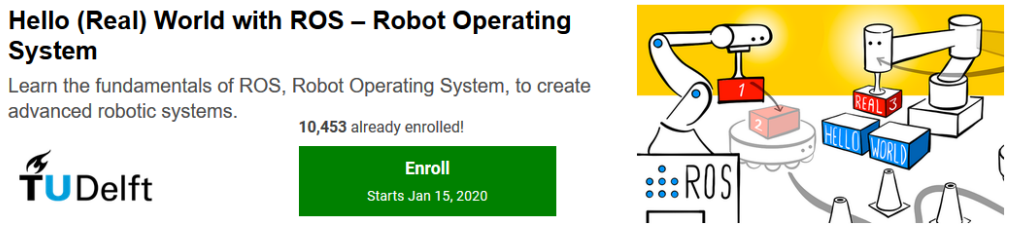
Find a robot by category:

Aerial | Component | Ground | Manipulator | Marine

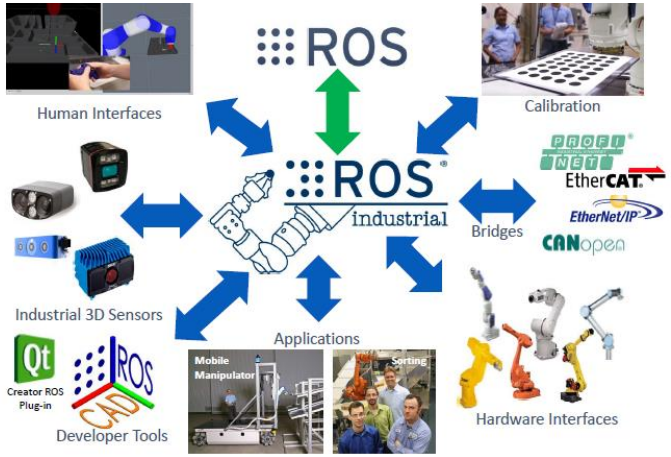
[Imagen de "ROS and mROS: How to accelerate the development of robot systems and integrate embedded devices", 2019]

Si quieres saber más…

- ROS: https://www.ros.org/
  - https://www.edx.org/course/hello-real-world-with-ros-robot-operating-system-2



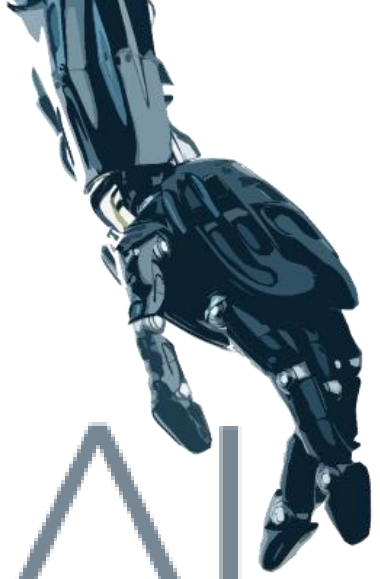
  - Programming Robots with ROS *(O'Reilly Book)*
  - ROS for Beginners: Basics, Motion, and OpenCV *(Udemy course)*
- ROS 2: https://index.ros.org/doc/ros2/
- ROS-I: https://rosindustrial.org/
- ROSin: https://rosin-project.eu/
- ROSCon: https://roscon.ros.org/2019/



IDONIAL CENTRO TECNOLÓGICO

[Imagen de ROS-I, 2018: "Enabling Facility-Level Interoperability Between Robot Teamsand Machine Cell Devices",]
[PSA for ROS users, 2019: https://ubuntu.com/blog/psa-for-ros-users-some-things-to-know-as-python-2-approaches-eol]

¡Muchas gracias!



IDONIAL
CENTRO TECNOLÓGICO

YouTube
Fundación Idonial

@idonialCT

Idonial Centro Tecnológico

@idonialTech

www.idonial.com

CONTACTO

D Parque Científico y Tecnológico de Gijón - Zona INTRA
Avda. Jardín Botánico 1345 - Edificio “Antiguo secadero de tabacos”
33203 Gijón, Asturias
T 984 390 060
E silvia.rodriguez@idonial.com
https://es.linkedin.com/in/silviarj
@srod_jim

IDONIAL
CENTRO TECNOLÓGICO