
Software libre en robótica: ROS y JdeRobot

José María Cañas

josemaria.plaza@urjc.es



@RoboticsLabURJC, @JdeRobot

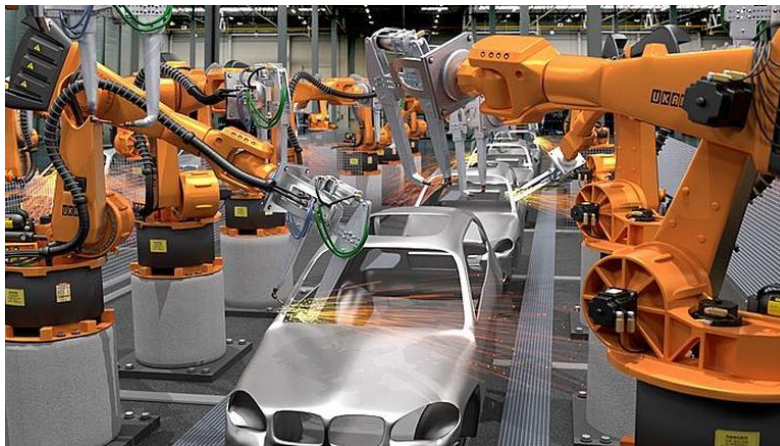
Contenidos

- La robótica es útil (y mola!)
- Software en robótica
- Un caso de (mucho) éxito: ROS
- JdeRobot

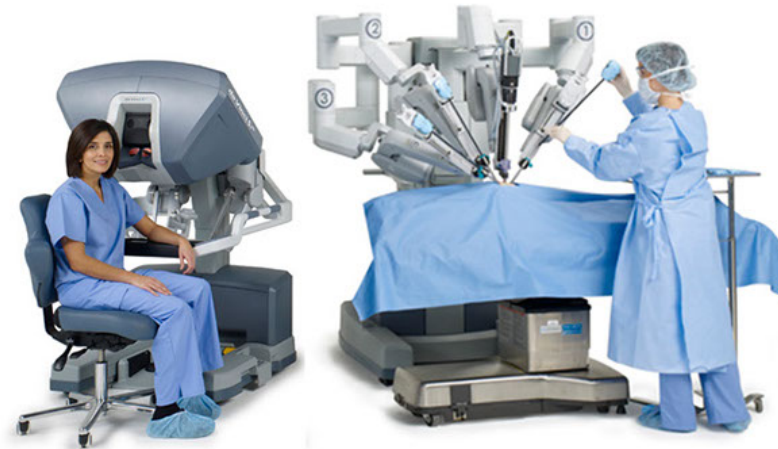
La robótica es útil (y mola!)

- Robótica ficción vs Robótica real
- *Dull, Dirty, Dangerous*
- La robótica ha salido de los laboratorios
- Aplicaciones reales, masivas

- Industria automovilística
- Coches autónomos
- Gestión de almacenes



- Hogar: aspiradoras
- Medicina
- Envasado de alimentos



¿Qué es un robot? Componentes



Sistema informático con:

- Sensores
- Actuadores
- Computador

Hay que **programarlo** para que consiga sus objetivos y sea sensible a la situación.

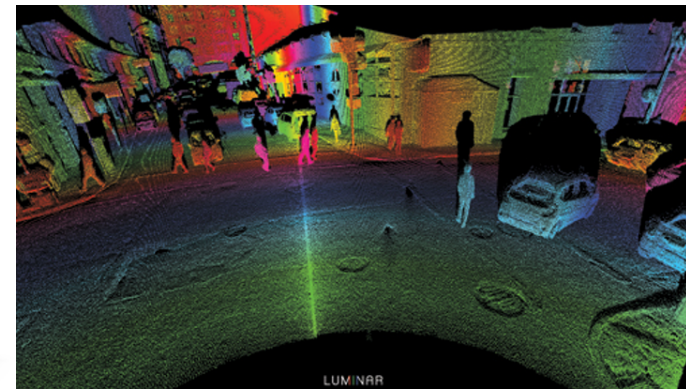
La inteligencia reside en su software

Sensores

- Cámaras, RGBD
- US, Láser, LIDAR
- Encoders

Actuadores

- Motores eléctricos
- Locomoción
- Manipulación



Software para robots

- Determina el comportamiento del robot
- Establece cómo se coordinan la percepción y la actuación
- No hay una manera universalmente aceptada de programarlos
- Lenguajes: ensamblador, C/C++, python...
- Los sistemas robóticos son sistemas complejos (el tamaño importa)

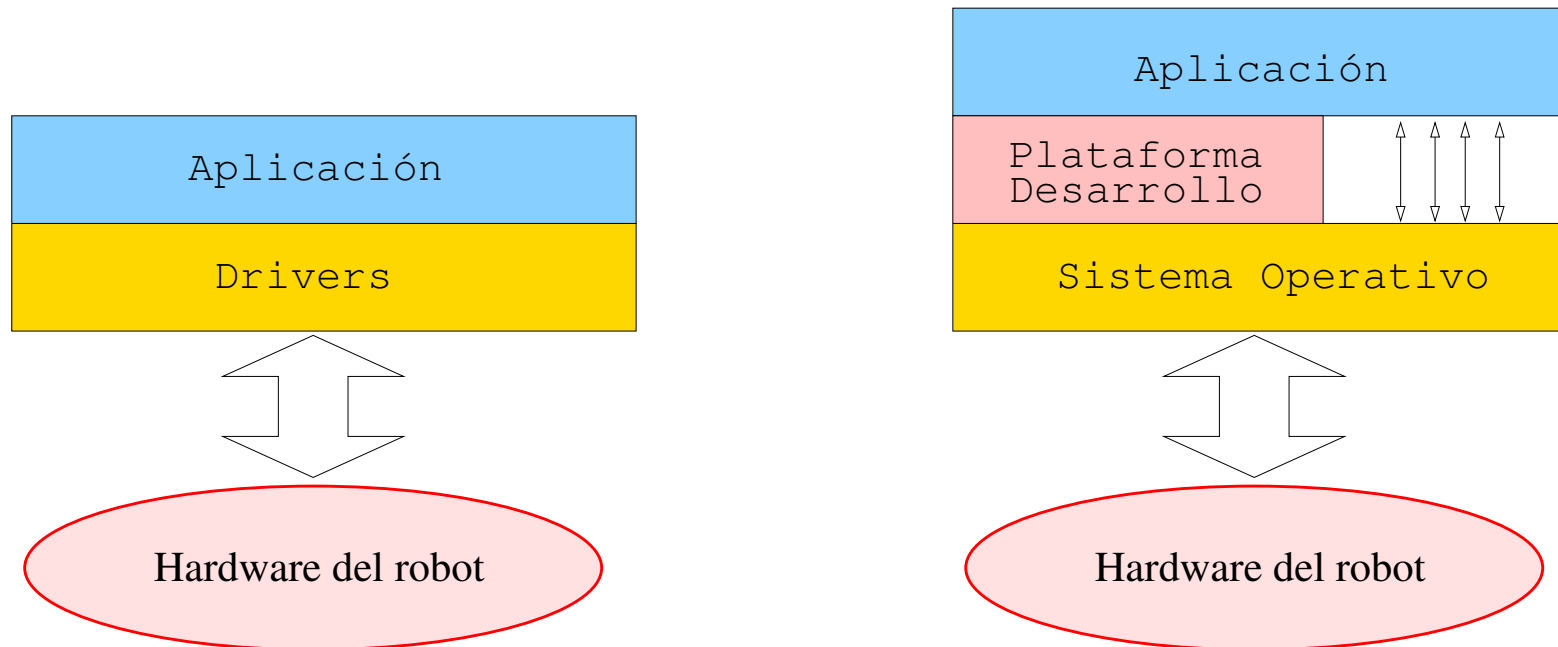
Requisitos específicos

- Vivacidad, agilidad (tiempo real)
- Multitarea (conurrencia, múltiples fuentes de actividad)
- Distribuido, comunicaciones
- Interfaz gráfica, depuración
- Expandible
- Conectado a la realidad física
- Heterogeneidad dispositivos hardware
- Encapsular funcionalidad o comportamientos es difícil

Tendencias

- Antes: cada robot su entorno de programación
- Ingeniería software: orientación a objetos, distribución
- Software orientado a componentes
- Interfaces explícitos
- Reutilizar software es difícil, pero muy ventajoso
- Plataformas software robótico

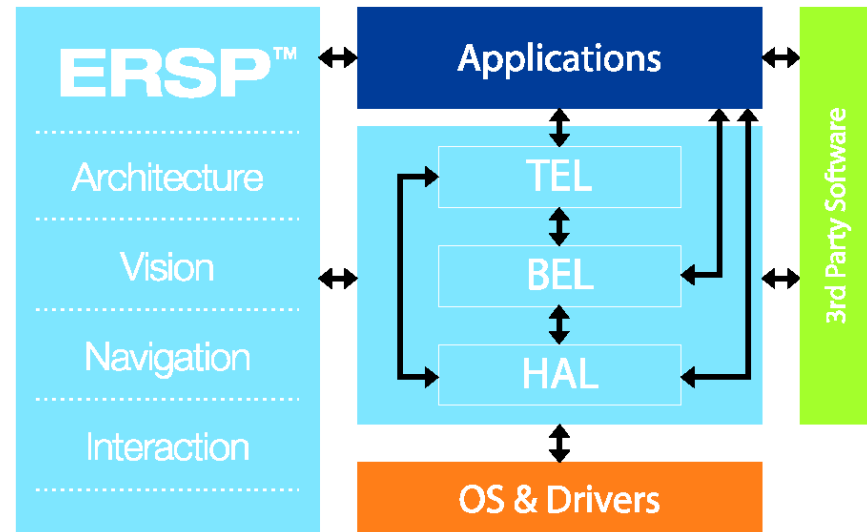
Plataformas de software robótico



- Procesadores empotrados (robots pequeños) o PC (medianos-grandes).
- Sistemas operativos: dedicados o generalistas
- *Middleware* para simplificar la creación de aplicaciones robóticas

¿Qué proporciona una plataforma?

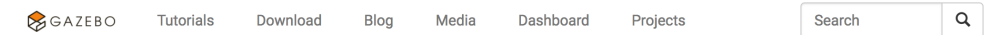
- Abstracción hardware (HAL)
- Arquitectura software
- Funcionalidades de uso común
- Herramientas



- Comerciales, investigación, software libre
- ROS, Urbi, YARP, Orca, OROCOS, Player/Stage, Claraty, MSRS

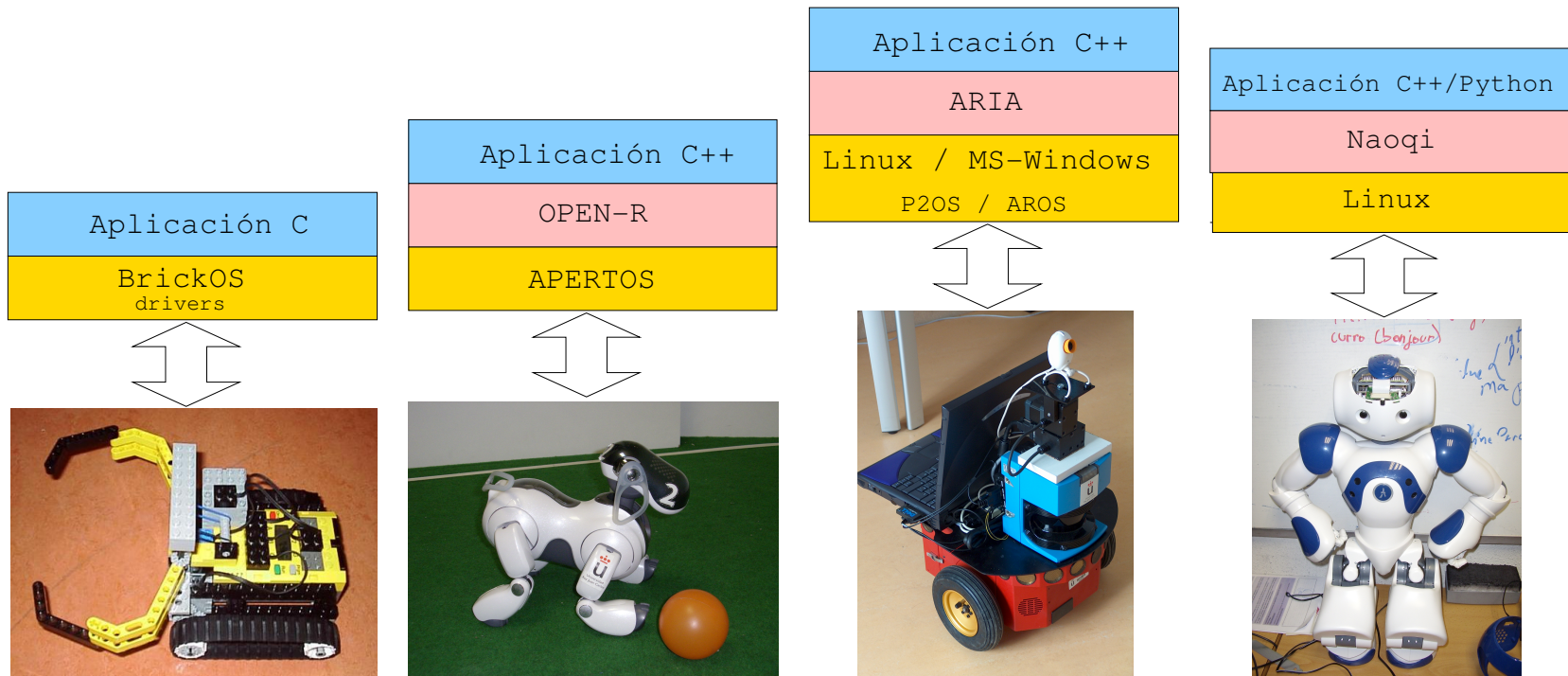
Simuladores

- Madurar algoritmos
- Comodidad trabajar sin robot
- Las caídas no duelen
- Mundo, sensores y actuadores
- Motor físico: ODE



- Gazebo, V-REP, Stage, Webots, MORSE

Ejemplos



Robot Operating System (ROS)

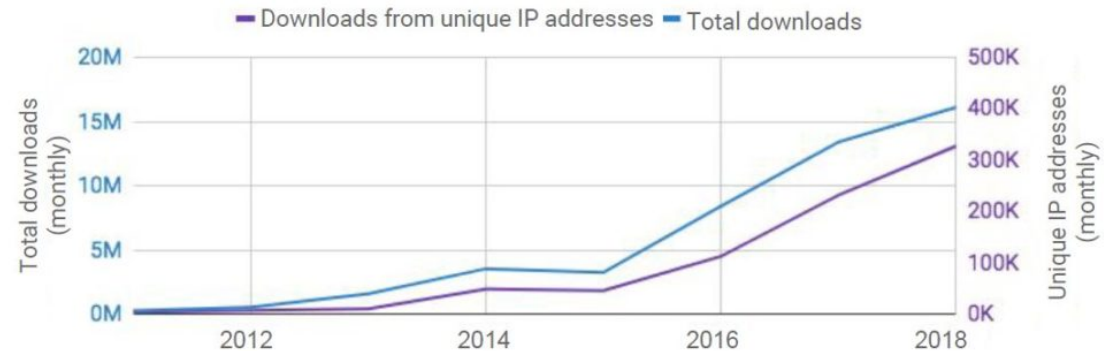
⋮ ROS.org

- *Middleware* para software robótico
- Evitar la reinención de la rueda: comunicaciones, drivers...
- Gratis y software libre: <https://ros.org>
- Colección de paquetes
- C/C++, Python
- Comunidad enorme (usuarios, desarrolladores, soporte...)
- **Standard de facto en robótica de servicios**
- Sobre Linux principalmente

Un poco de historia (2006-2019)

- Stanford (-2008)
Personal Robotics Program
- WillowGarage (-2014)
Gazebo, Turtlebot
- OSRF (-2017)
- OpenRobotics (-today)

ROS package downloads (2011–2018)
Source: ROS metrics report

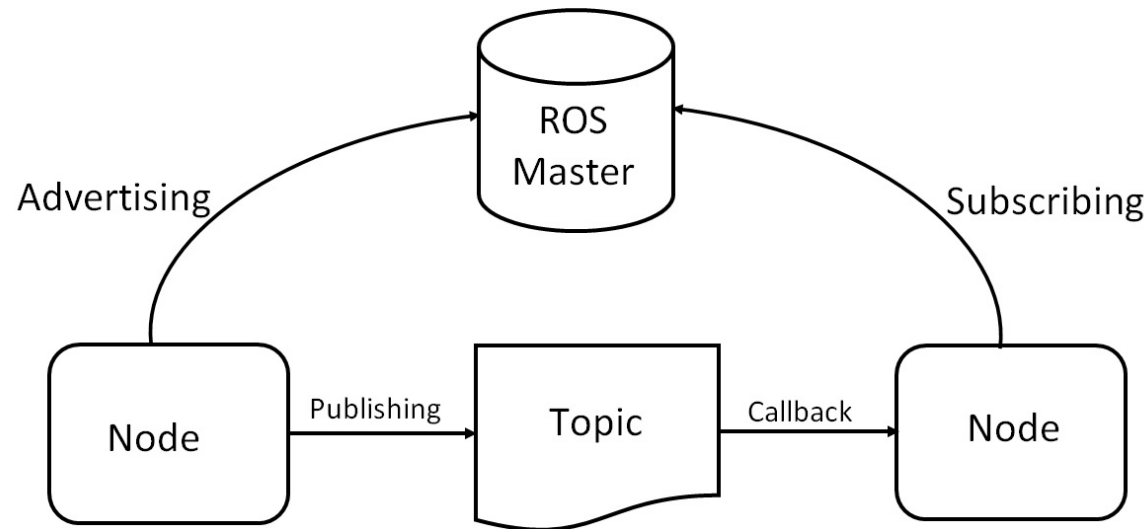


Características

- Aplicaciones robóticas distribuidas en nodos que se comunican
- Standard robot messages
- Drivers
- Herramientas
- Capacidades
- Aumenta la interoperación y reutilización de sw robótico

Comunicaciones

- *Topics*: publicación-suscripción, asíncronos y anónimos
- *Services*: RPC, bloqueantes
- *Actions*: interrumpibles



Herramientas

- ROSbags, recording and playback
- RViz, visualizador 3D
- Rqt-graph, grafo de cómputo

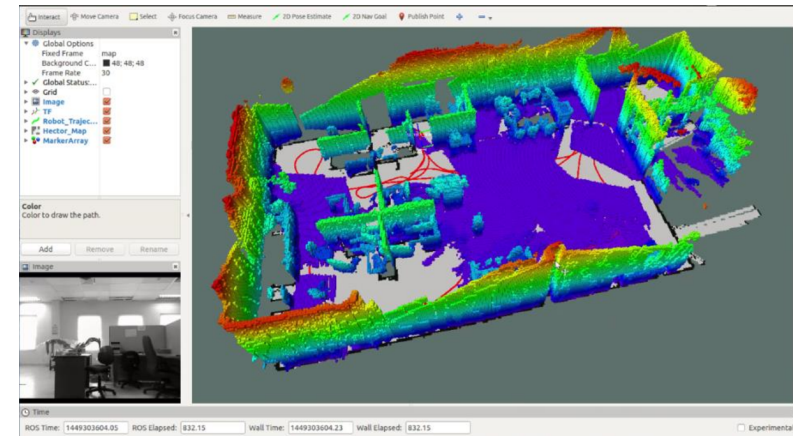
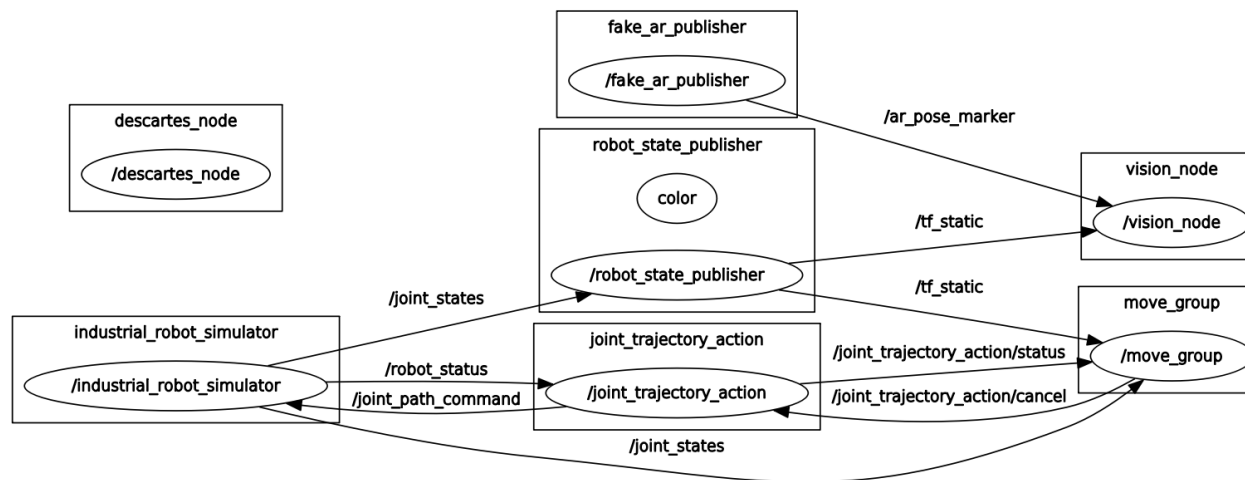
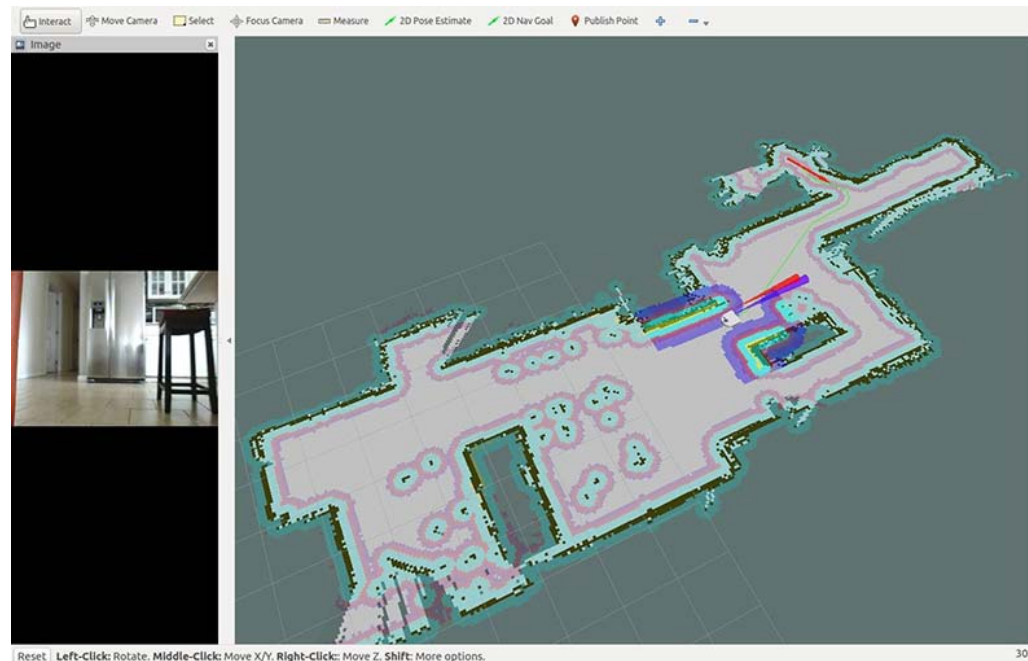


Figura 7. Visualiza 2D & 3D espacial por using RVIZ



Capacidades

- Nodos (*stacks*) con implementación de algoritmos punteros
- Localización
- Construcción de mapas
- Navegación



Tendencias

- ROS-Industrial
- ROS2: DDS
 - security
 - real-time
 - no single point of failure (roscore)
- Ignition simulator
 - cloud



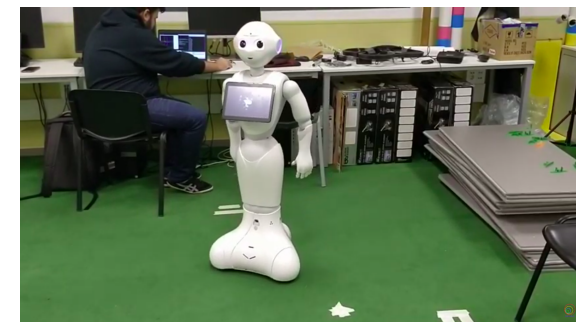
JdeRobot



- Gratis y software libre: <https://jderobot.org>
- Nació en la URJC
- C/C++, Python, JavaScript
- Linux
- De plataforma a toolkit
- Adaptación a ROS, de drivers ICE a drivers ROS

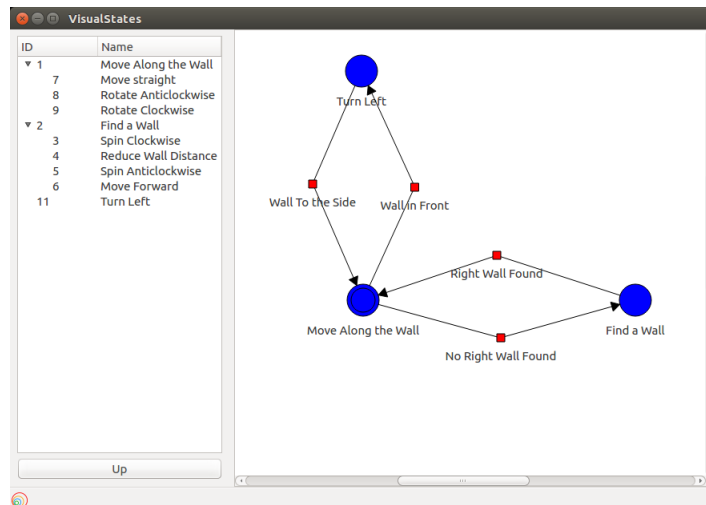
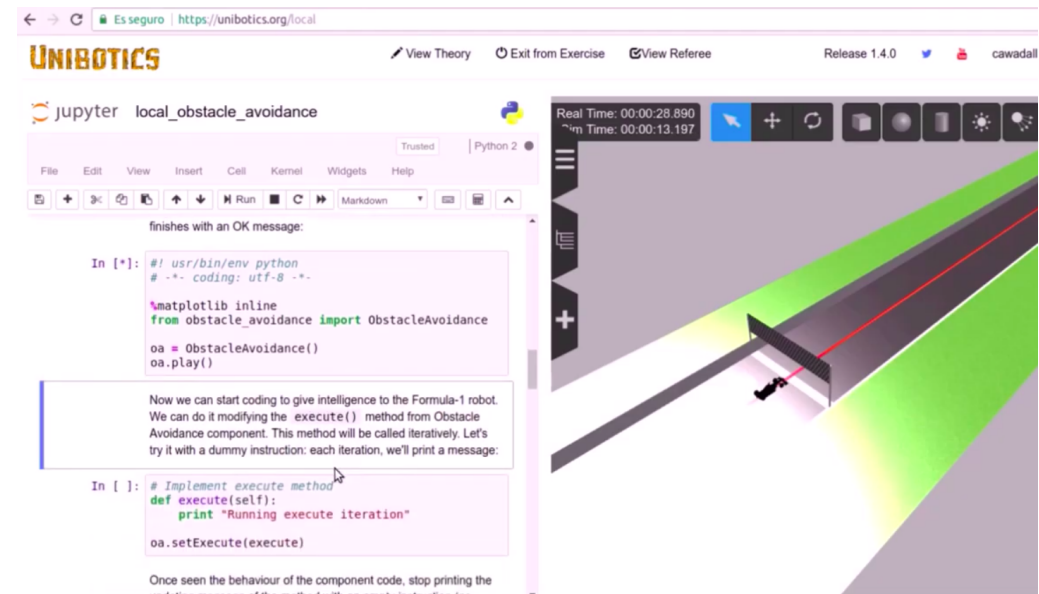
Proyectos

- Herramientas de programación de robots
- *DeepLearning*, redes neuronales
percepción y comportamientos
- FPGAs en robótica
- VisualSLAM
- **Educación en robótica**
RoboticsAcademy, Unibotics
- Drones



Algunos productos

- VisualStates
- Unibotics
- Conducción autónoma

The screenshot shows the Unibotics web interface. The top navigation bar includes 'View Theory', 'Exit from Exercise', 'View Referee', 'Release 1.4.0', and 'cawadall'. The main content area displays a Jupyter notebook titled 'local_obstacle_avoidance'. The notebook contains the following code:

```

finishes with an OK message:

In [*]: #!/usr/bin/env python
# -*- coding: utf-8 -*-

%matplotlib inline
from obstacle_avoidance import ObstacleAvoidance

oa = ObstacleAvoidance()
oa.play()

Now we can start coding to give intelligence to the Formula-1 robot.
We can do it modifying the execute() method from Obstacle
Avoidance component. This method will be called iteratively. Let's
try it with a dummy instruction: each iteration, we'll print a message:

In [ ]: # Implement execute method
def execute(self):
    print "Running execute iteration"

oa.setExecute(execute)

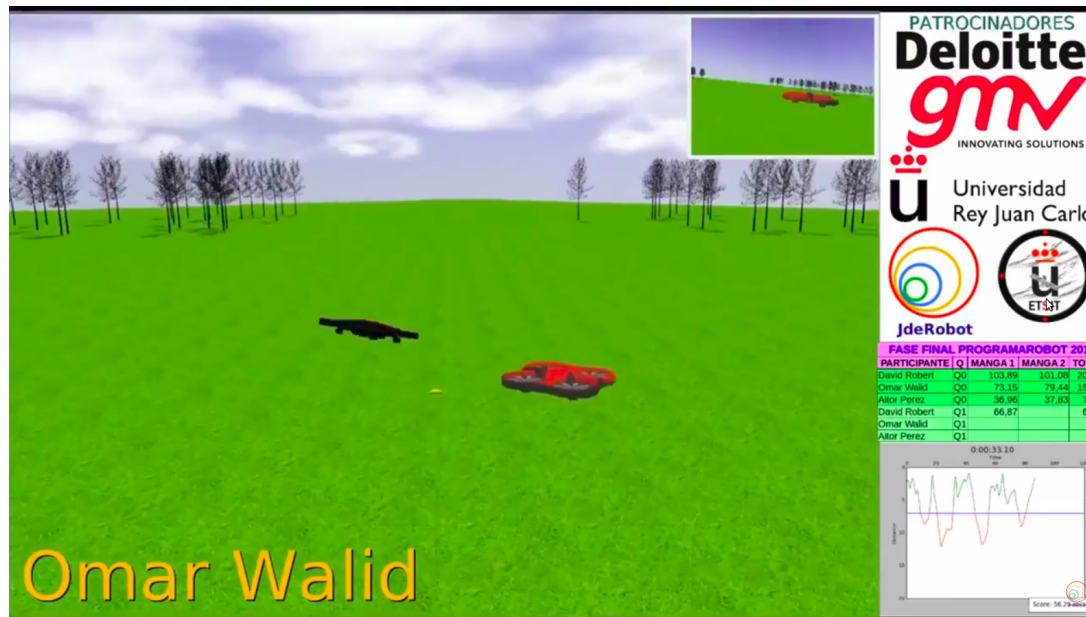
Once seen the behaviour of the component code, stop printing the
update message of the method with an empty instruction (a

```

On the right side of the notebook, there is a 3D visualization of a robot in a corridor, with a red laser line indicating its path and a green area representing the robot's field of view.

Actividades

- RoboticsClub
- Competiciones: IROS 2018
- Google Summer of Code 2015,2017,2018,2019



The screenshot displays a 3D virtual environment for a robot competition. The main area shows a green field with a line of trees in the background and two robots (one black, one red) on the ground. An inset window shows a top-down view of the robots. On the right side, there are logos for sponsors: Deloitte, gmV, Universidad Rey Juan Carlos, and JdeRobot. Below the logos is a table titled 'FASE FINAL PROGRAMAROBOT 2017' showing scores for participants in two mangas.

PARTICIPANTE	Q	MANGA 1	MANGA 2	TOTAL
David Robert	Q0	103,99	101,08	204,97
Omar Walid	Q0	73,15	79,44	152,59
Alfon Perez	Q0	36,96	37,83	74,79
David Robert	Q1	66,87		66,87
Omar Walid	Q1			0,00
Alfon Perez	Q1			0,00

At the bottom of the interface, there is a graph showing a fluctuating line and a timer displaying '0:00:33.30'. The name 'Omar Walid' is written in large orange letters at the bottom left of the screen.

Infraestructura de desarrollo

- Muuuuchas líneas de código (<https://github.com/JdeRobot>)
- De svn a gitlab y github: incidencias y parches
- Documentación: de mediawiki a github pages
- CI-CD, jenkins...
- Slack, Hangouts
- YouTube channel
- Twitter @JdeRobot