
Robótica del siglo XXI y educación

José María Cañas Plaza

josemaria.plaza@gmail.com



PROSEGUR, 22 de octubre de 2020

Contenidos

- Introducción
 - Robótica, una tecnología en auge
 - ¿Qué es un robot?
 - Software es importante en robótica
- Educación robótica para niños e iniciación
- Educación robótica superior, profesionales

Introducción

Robótica, una tecnología en auge

Robótica ficción vs Robótica real



- Cada vez más aplicaciones y productos robóticos útiles en la sociedad

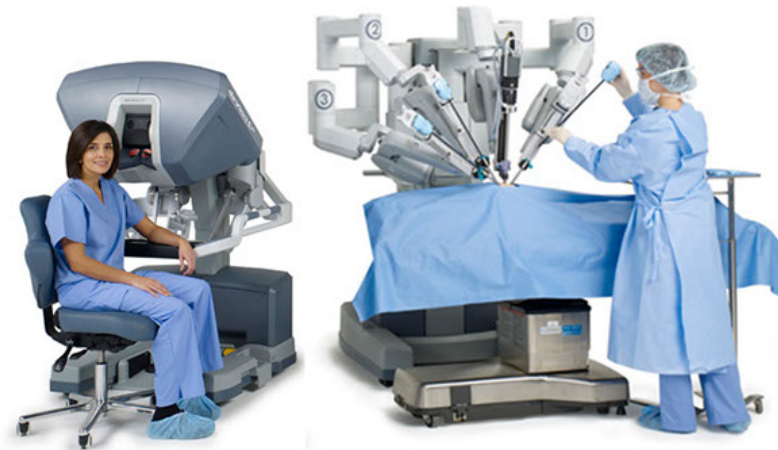
Aplicaciones reales

- *Dull, Dirty, Dangerous*
- ¡La robótica ha salido de los laboratorios!
- Aplicaciones reales, masivas

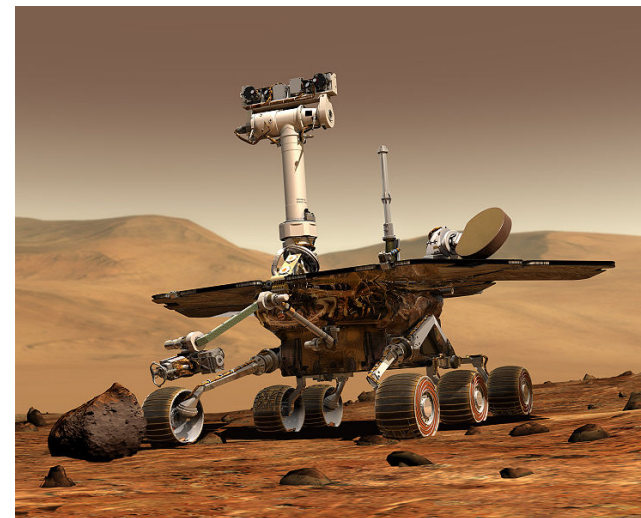
- Industria automovilística
- Coches autónomos
- Gestión de almacenes



- Hogar: aspiradoras
- Medicina
- Envasado de alimentos



- Entretenimiento: LEGO...
- Usos militares: PackBot...
- Espacio: Opportunity...
- Prestige, limpieza centrales nucleares



¿Qué es un robot?



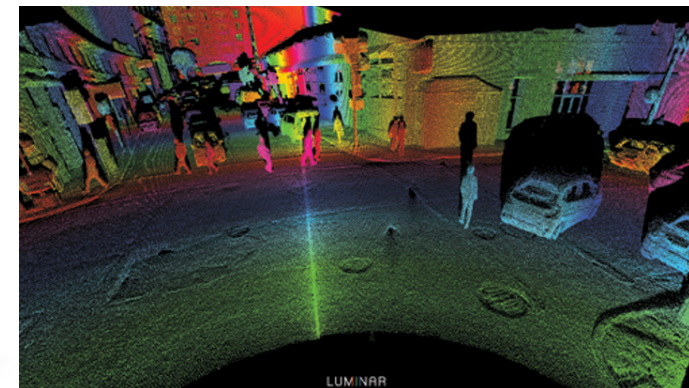
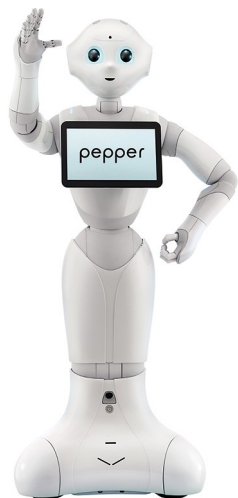
- *robot = hardware + software*
- (Hardware) Sistema informático con:
 - Sensores
 - Actuadores
 - Computadoras
- (Software) Hay que **programarlo** para que consiga sus objetivos y sea sensible a la situación.
- **La inteligencia reside en su software**

SENSORES

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

ACTUADORES

- Motores eléctricos
- Locomoción
- Manipulación



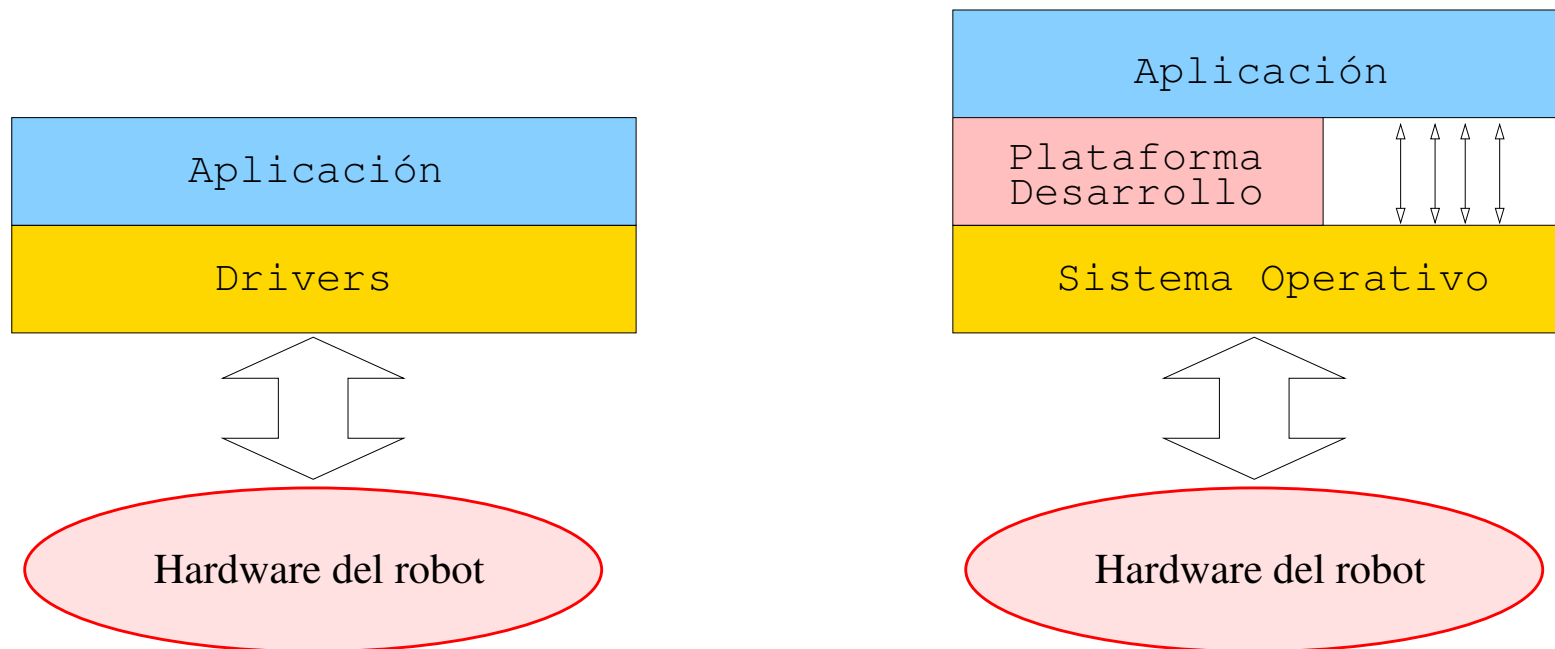
Software para robots

- 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++ (de bajo y alto nivel)
- **Heterogeneidad**
 - Dispositivos hardware
 - Encapsular funcionalidad
- Requisitos específicos
- Sistemas operativos y plataformas
- Simuladores

Requisitos específicos

- Vivacidad, agilidad (tiempo real)
- Multitarea (conurrencia, múltiples fuentes de actividad)
- Distribuido, comunicaciones
- Interfaz gráfica, depuración
- Expandible
- Conectado a la realidad física
- Hardware heterogéneo
- Reutilizar software es difícil

Sistemas operativos y plataformas

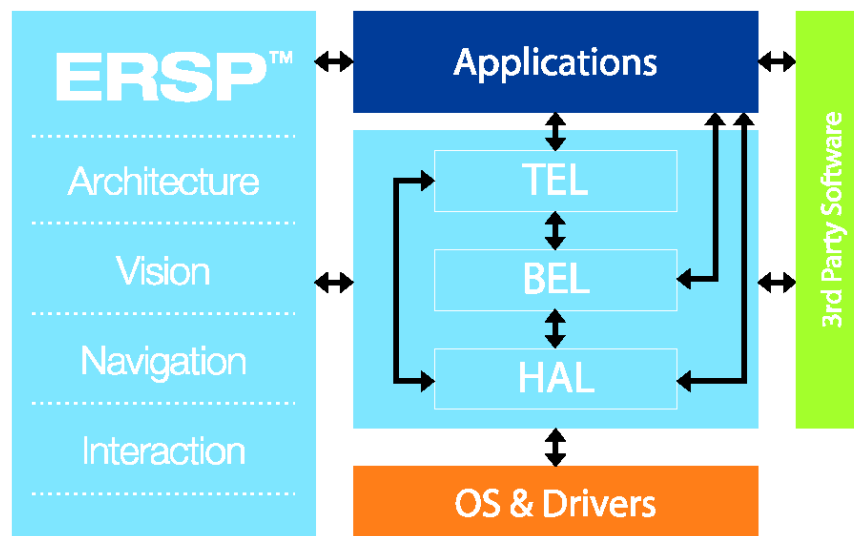


- Procesadores empotrados (robots pequeños) o PC (medianos-grandes).
- Sistemas operativos: dedicados o generalistas
- *Middleware* para simplificar la creación de aplicaciones robóticas

- Empezar de cero cada vez, cada robot su entorno de programación
- Reutilizar software es difícil
- Encapsular funcionalidad o comportamientos es difícil
- Tendencia a software orientado a componentes
- Tendencia a interfaces explícitos

¿Qué proporciona una plataforma sw para robots?

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



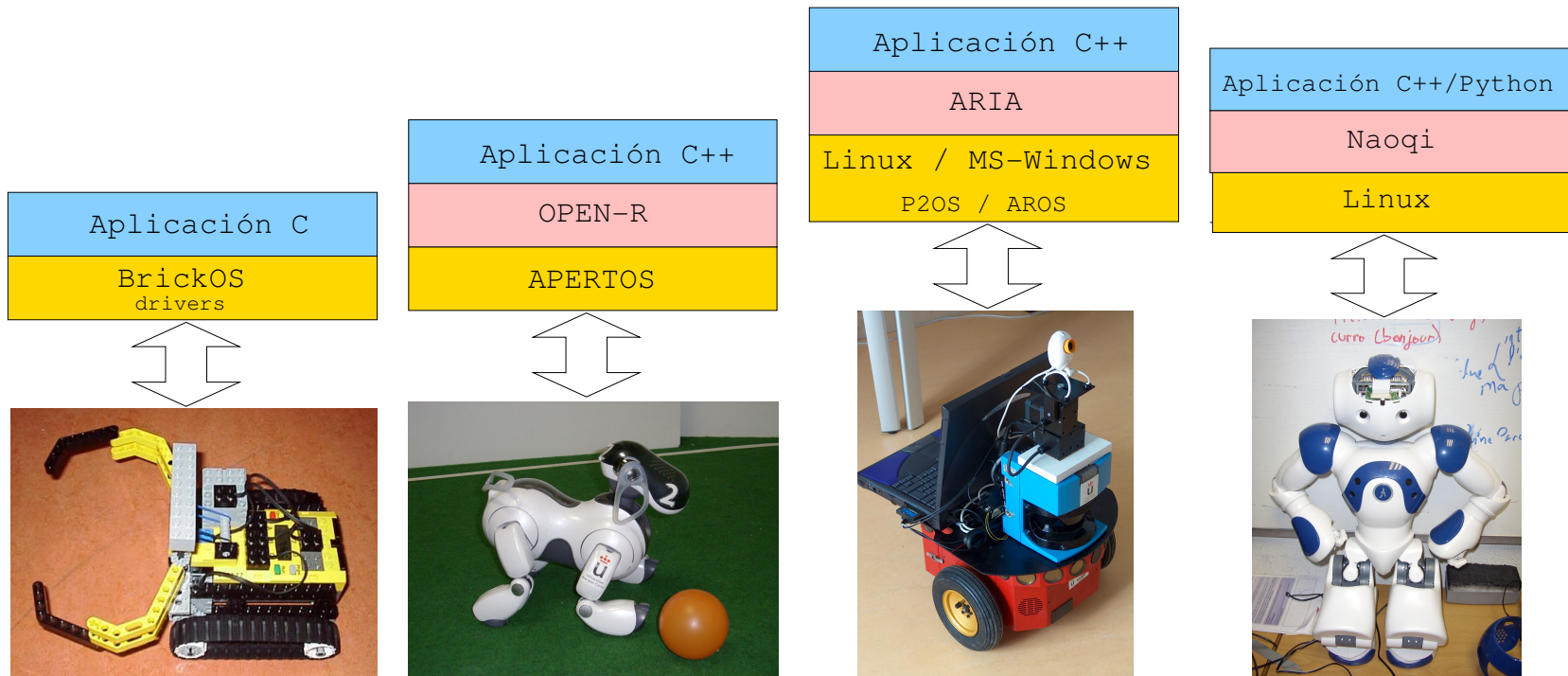
- Comerciales, investigación, software libre
- Ingeniería software: orientación a objetos, distribución
- ROS, Orca, Carmen, OROCOS, ERSP, Player/Stage, Clarity, etc.

Robot Operating System (ROS)



- *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

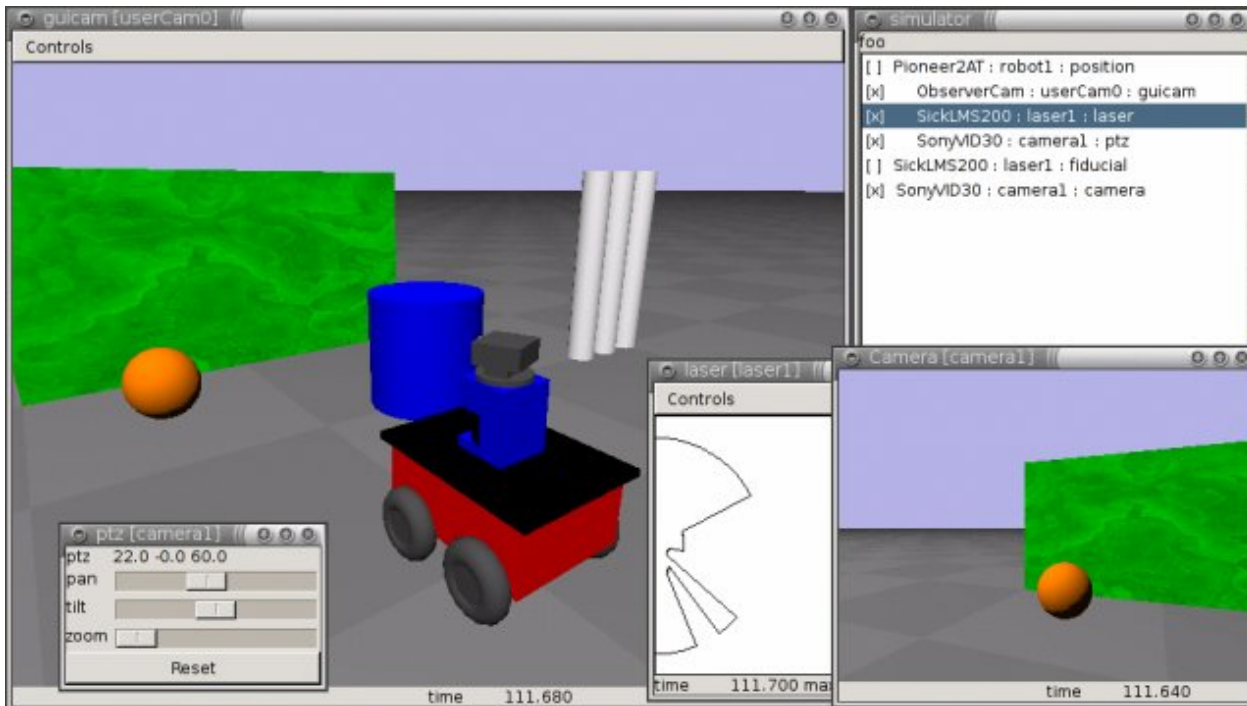
Ejemplos de plataformas sw



Simuladores

- Madurar algoritmos
 - Comodidad trabajar sin robot
 - Las caídas no duelen
 - Mundo, sensores y actuadores
 - OpenGL (OGRE) para imágenes
 - Motor físico: ODE (*Open Dynamics Engine*), Bullet...
-
- Gazebo, CoppeliaSim, Webots, Stage, MORSE

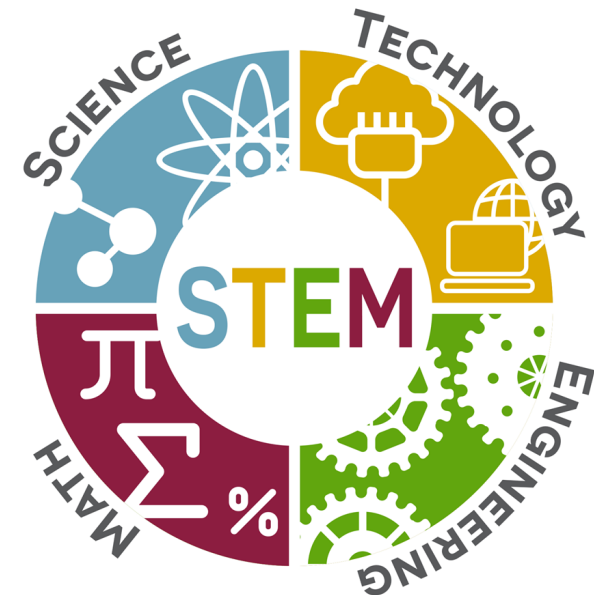


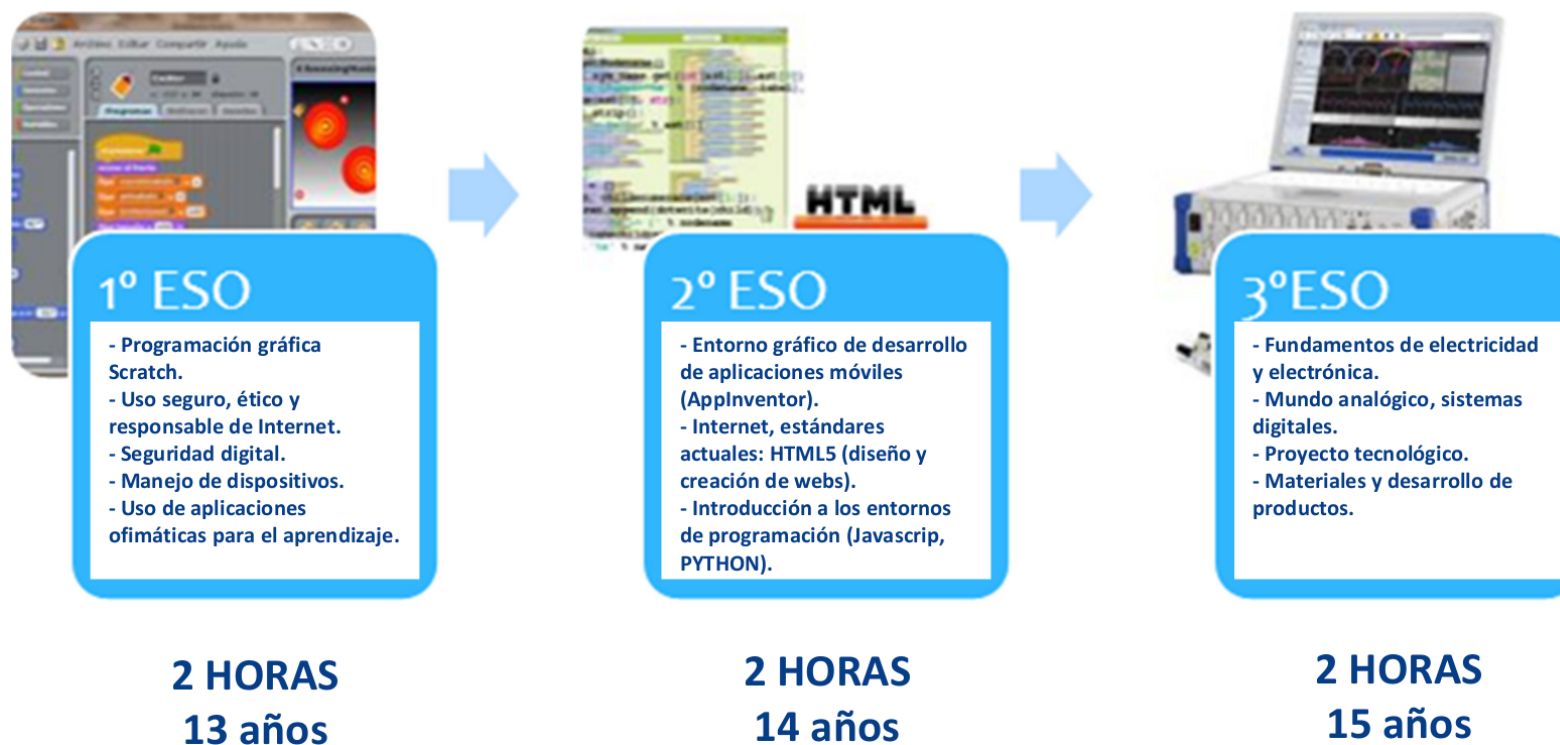


Educación robótica para niños e iniciación

Demanda creciente de formación en robótica

- Nociones básicas útiles para todos en XXI
- Introducción divertida a la tecnología
- Disciplinas **STEM**
Science, Technology, Engineering, Math
- + Permite docencia por proyectos
- + Permite *gamificación*
- + Facilita trabajo en equipo
- Pensamiento Computacional





- Currículo oficial Enseñanza Secundaria (Com.Madrid, Galicia...)
 - 1º, 2º, 3º ESO “Tecnología, Programación y robótica” (2h/sem)
 - 4º ESO “Tecnología” (3h/sem)
- Actividades extraescolares

Competiciones robóticas infantiles



- First LEGO League
- Robocup Junior
- VEX
- RoboCampeones

Robots educativos

LEGO MindStorms EV3

- Comercial LEGO
- Fácilmente adquirible
- Robusto y estable
- Control en posición
- Control en velocidad



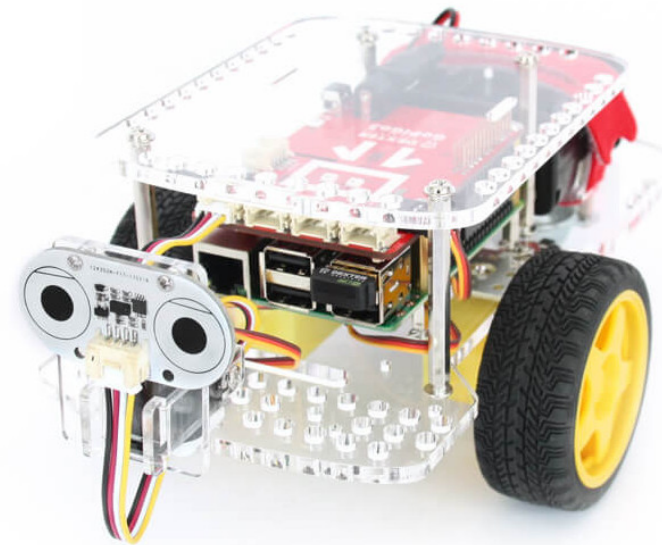
Robot mBot

- Comercial Makeblock
- Barato, fácilmente adquirible
- Robusto
- Arduino
- Motores
- Ultrasonidos, Infrarrojos
- mBlock, lenguaje Arduino



Robot GoPiGo3

- Comercial, Dexter industries
- Abierto, construible
- RaspberryPI
- Motores con encoders
- Ultrasonidos, Infrarrojos
- (PiCam cámara)



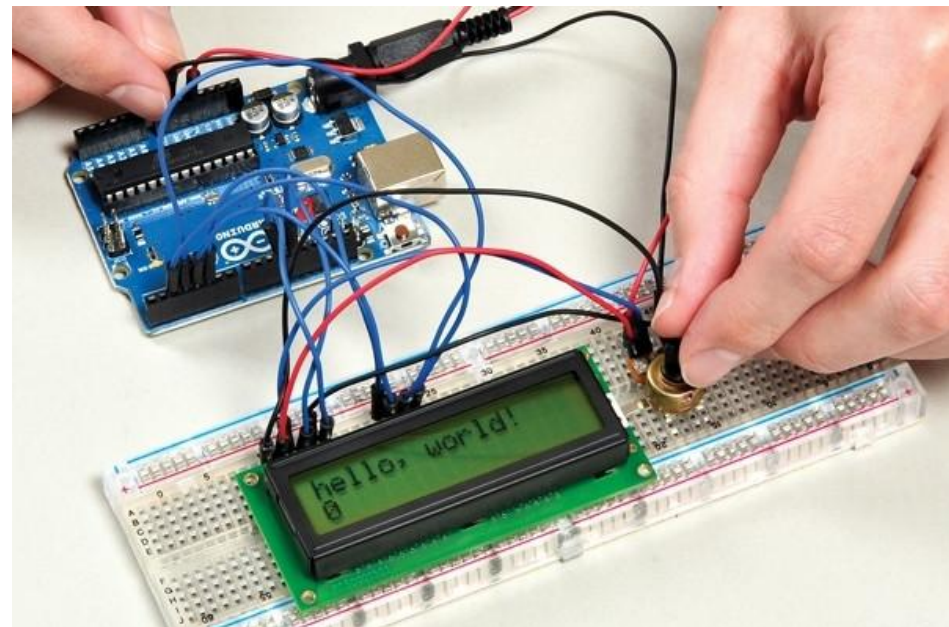
Drone Tello

- Comercial DJI+Intel
- Barato, fácilmente adquirible
- Robusto y estable
- Cámara
- Control en posición
- Control en velocidad



Microcontrolador Arduino

- Sensores, Motores
- Pulsadores
- Pantalla
- Electrónica
- Programación
- Lenguaje Arduino

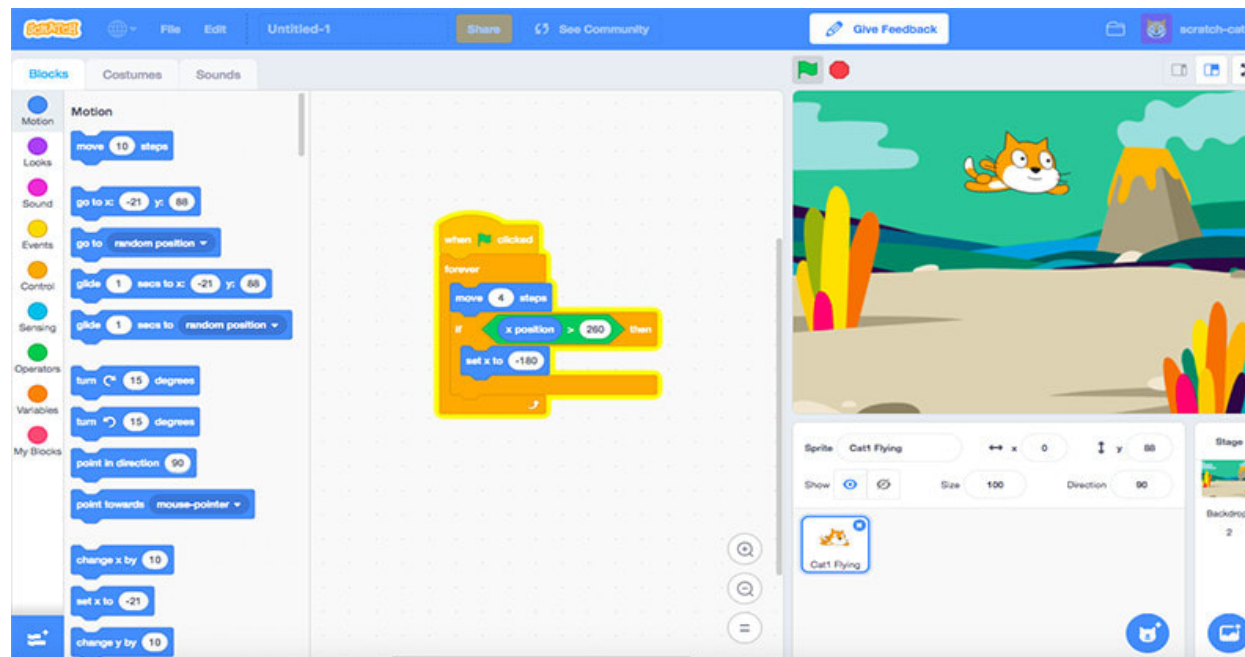


Plataformas educativas infantiles

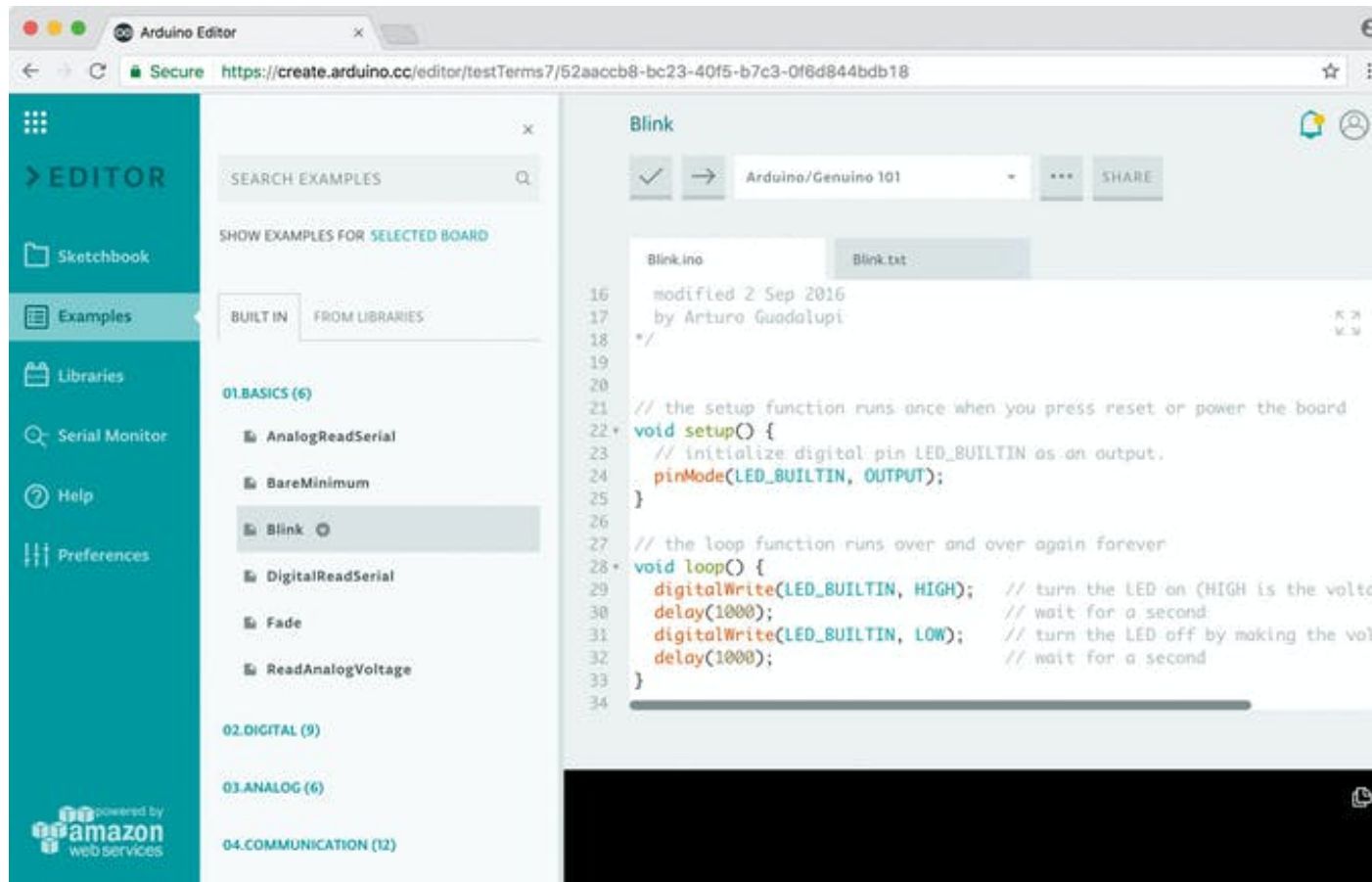
- **Permiten programar los robots**
 - Cada fabricante ofrece el suyo
 - Tiene que ser fácil, para niños
 - Tiene que ser estimulante
 - Lenguajes gráficos o de texto
- *LEGO*
 - *VEX*
 - *Arduino web-IDE*
 - *mBlock IDE*
 - *Scratch*
 - *OpenRoberta Lab*

Scratch

- MIT, <https://scratch.mit.edu>
- Abierto, gratuito
- Bloques visuales
- Muy intuitivo
- Conceptos básicos de pensamiento computacional
- Gato en una pantalla 2D
- Bloques de control de flujo

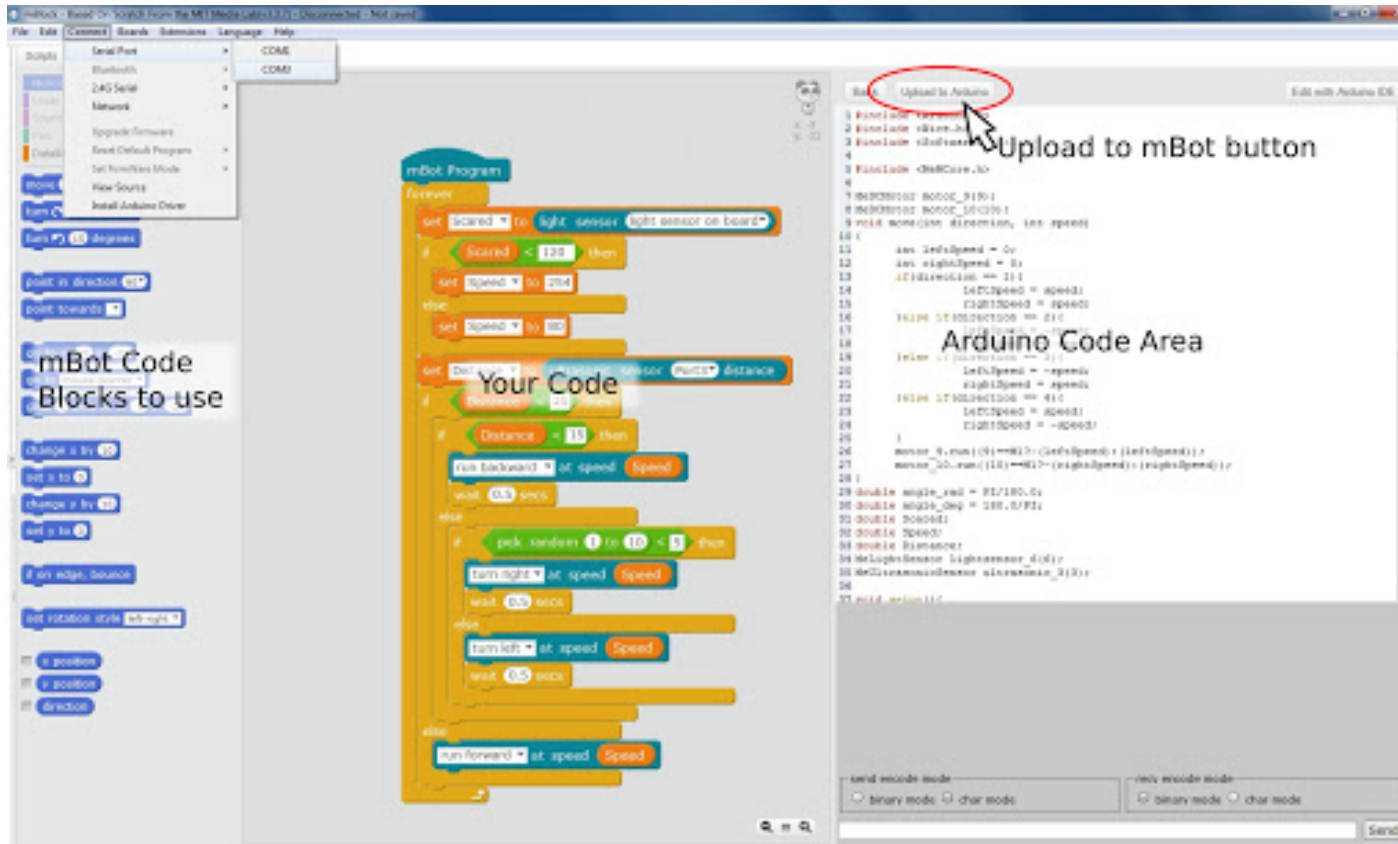


Arduino web-IDE



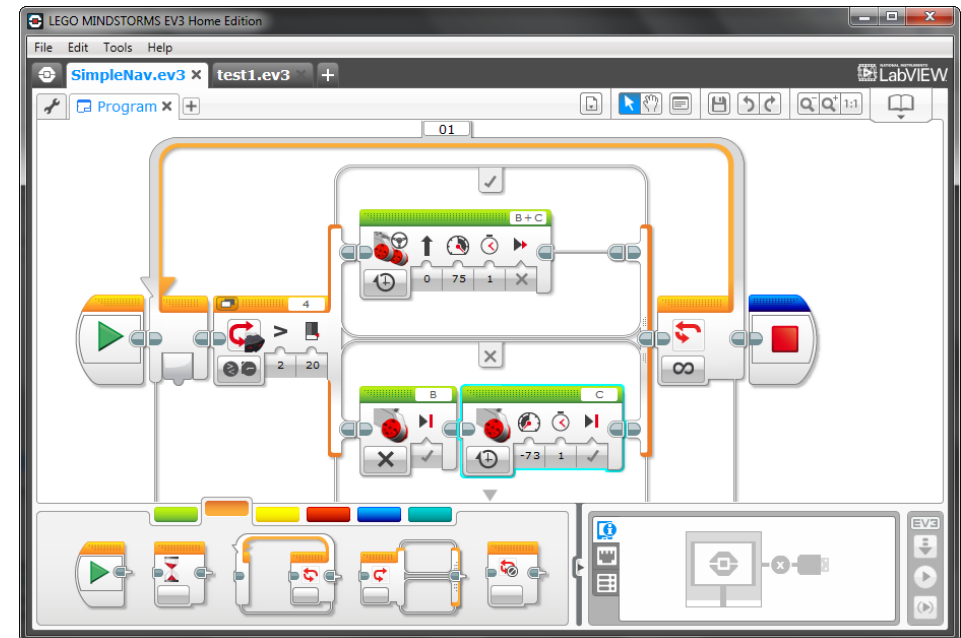
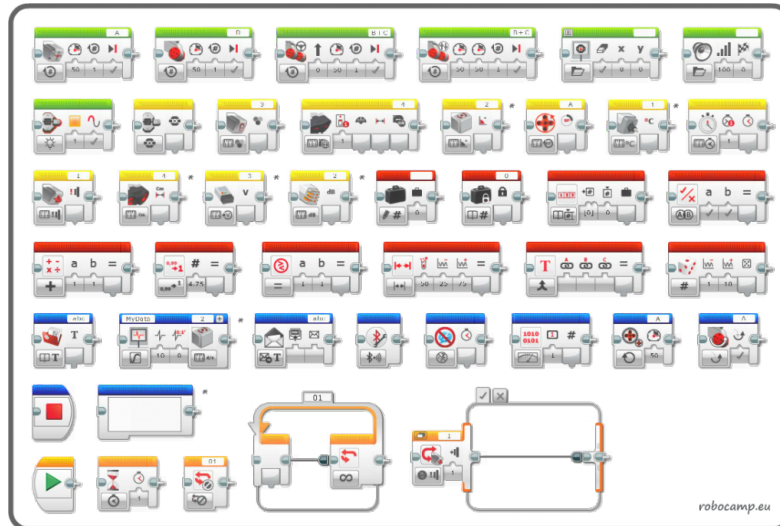
- Versión local y versión web <https://create.arduino.cc/>

mBlock IDE



LEGO EV3 Home Edition

- Almacén bloques visuales
- Intuitivo
- Se descarga en robot LEGO EV3
- Bloques de sensores, de actuadores



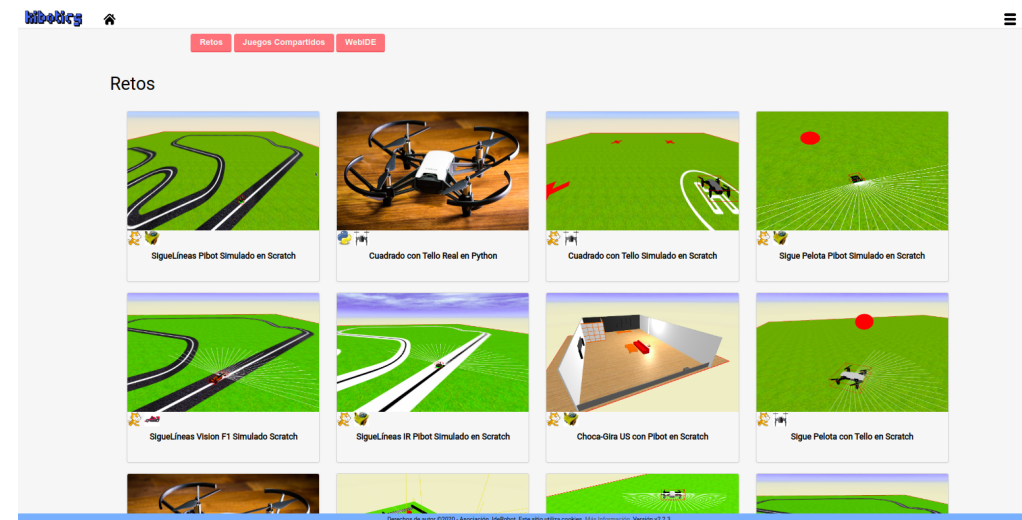
OpenRoberta Lab

- <https://lab.open-roberta.org/>
- Iniciativa educativa alemana, Fraunhofer IAIS
- Abierto, gratuito
- Soporta varios robots
- Lenguaje visual NEPO
- **Simulador 2D**



Plataforma Kibotics

- **En línea**, navegador web
- Primaria, secundaria y bachillerato
- Énfasis en programación
Lenguajes **Scratch** y **Python**
- **Robots físicos** y **simulados**
mBot, drone, PiBot, visión...
- **Contenidos educativos**
- **Interacción social**, foro



Plataforma web: <http://kibotics.org>

- No hay que instalar nada
- Desde cualquier sitio, a cualquier hora :-)
- Multiplataforma: Linux, Windows, Tablets, MacOS, Android

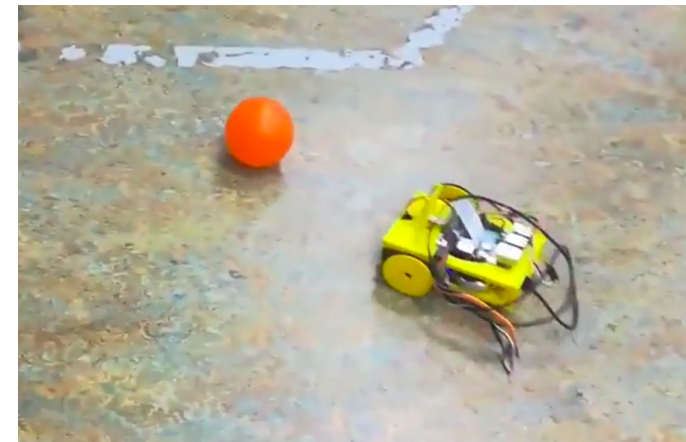
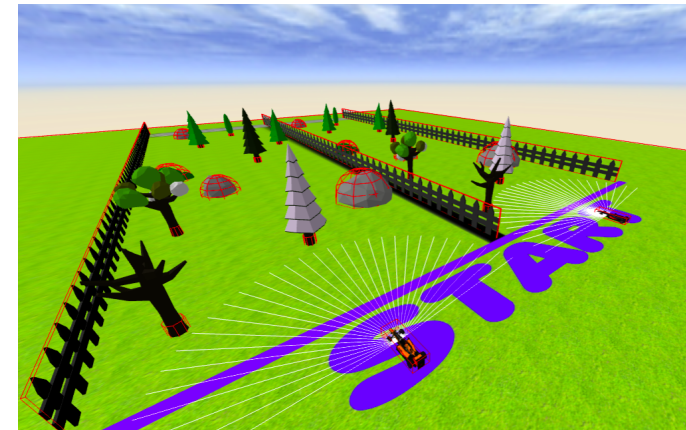


Ventajas: cuándo y dónde quieras

- Incluye evaluación automática
- Completamente en línea
- Los alumnos pueden acceder también fuera de las horas de clase
- Pueden acceder desde clase y también desde su casa
- Se puede enseñar robótica sin (o con menos) robots físicos
- Ahorro de costes

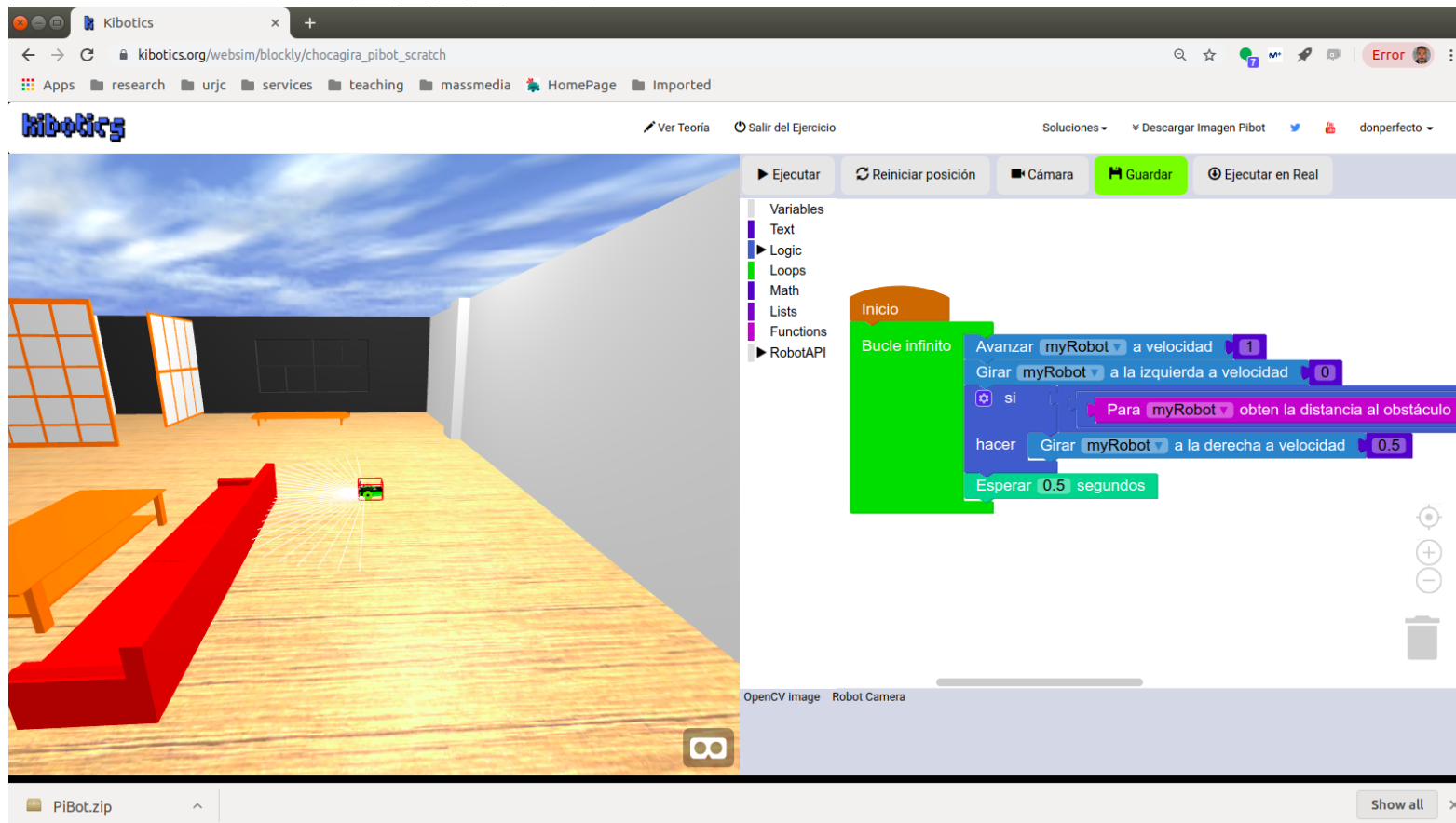
Contenidos educativos

- Adaptados a currícula oficiales
- *Learn by doing*
- ¡Divertidos! *Gamificación*
- Complejidad dosificada
- Ejercicios prácticos
 - Choca-gira con US
 - Sigue-líneas con IR
 - Sigue objeto con visión ...
- Lecciones de teoría



Lenguajes de programación: Scratch y Python

- **Scratch**: lenguaje visual, fácil de aprender

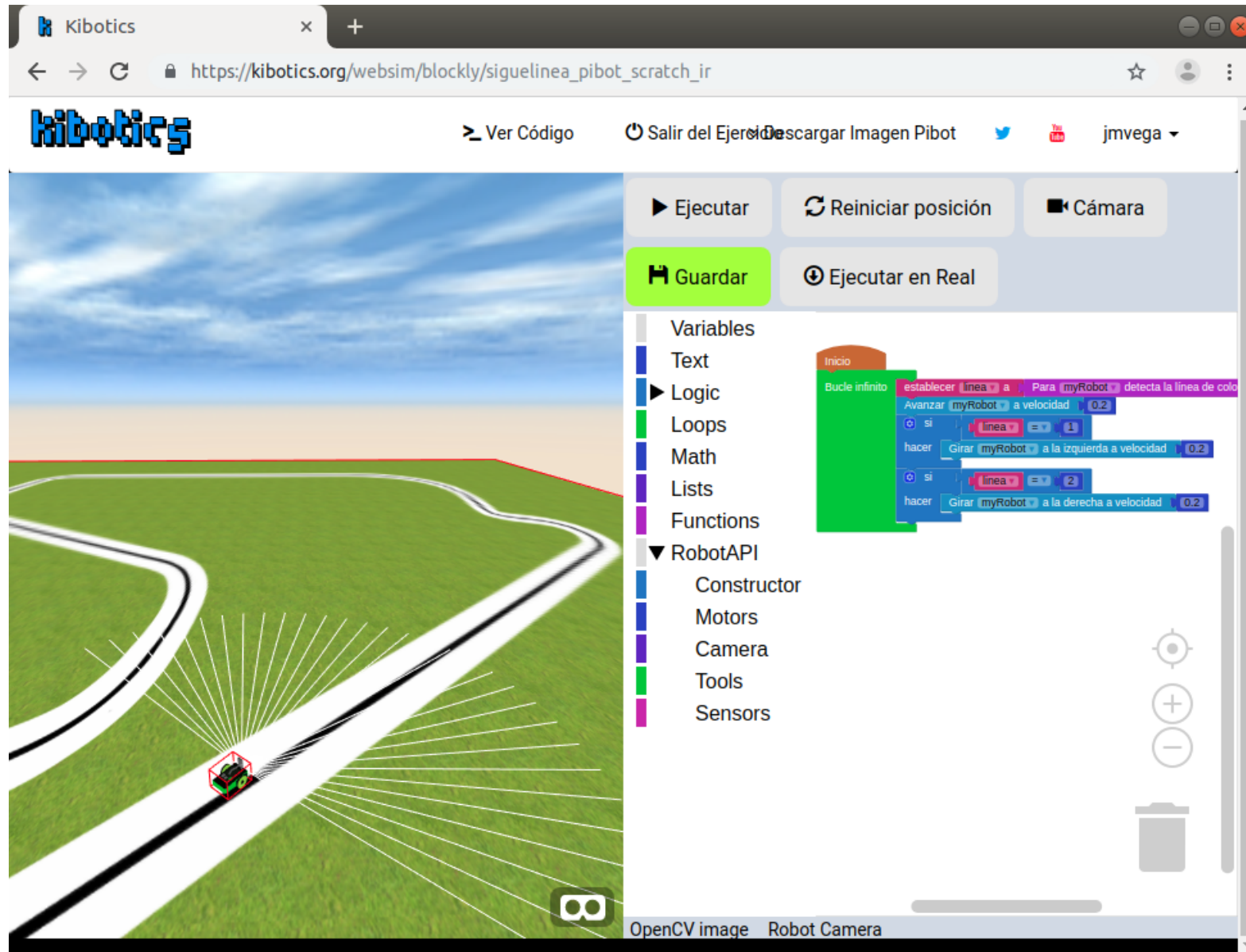


The screenshot shows the Kibotics web interface. The browser address bar displays `kibotics.org/websim/blockly/chocagira_pibot_scratch`. The interface includes a 3D simulation environment on the left with a robot on a wooden floor, a red sofa, and a blackboard. On the right, a Scratch-style blockly editor is visible, containing the following script:

```

Inicio
Bucle infinito
  Avanzar myRobot a velocidad 1
  Girar myRobot a la izquierda a velocidad 0
  si
  hacer
    Girar myRobot a la derecha a velocidad 0.5
  Esperar 0.5 segundos
  Para myRobot obten la distancia al obstáculo
  
```

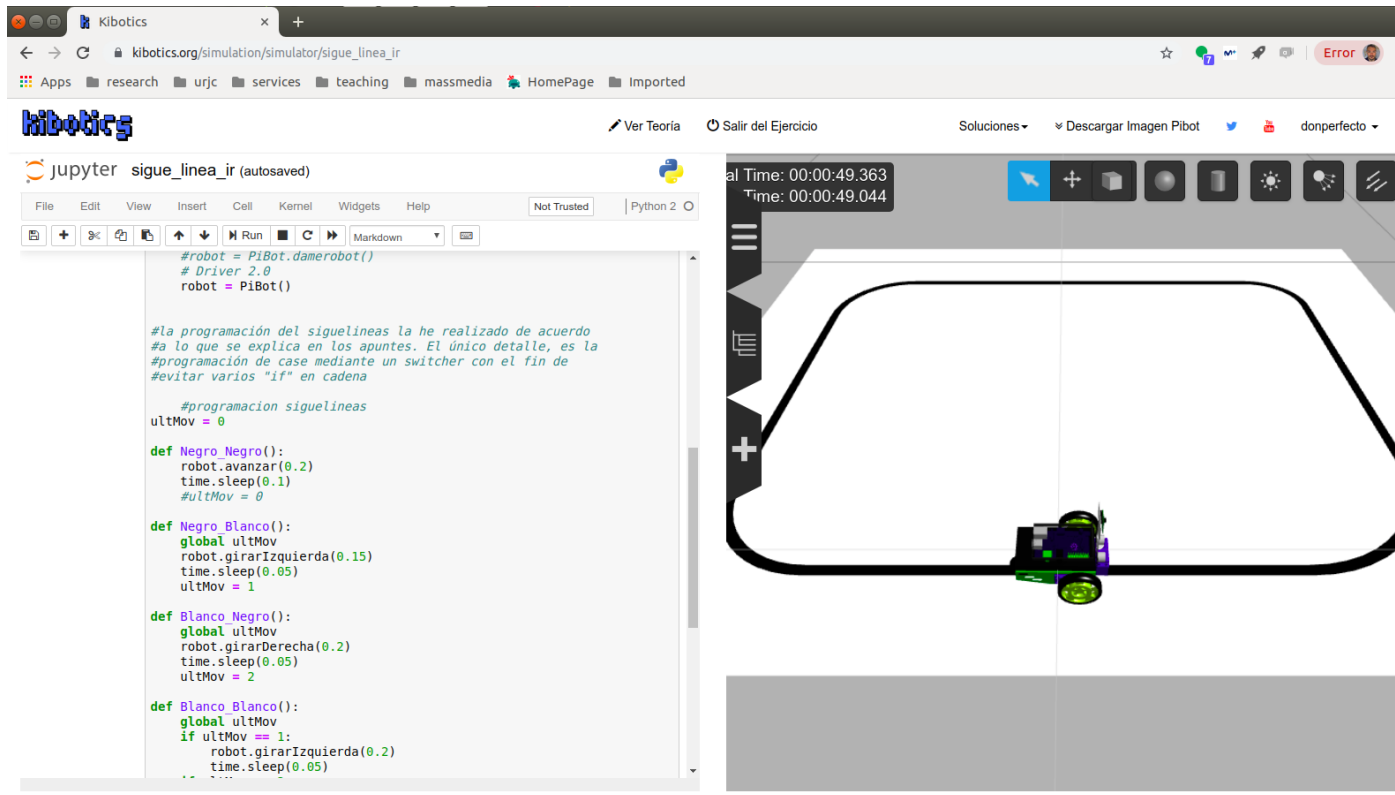
At the bottom of the interface, there is a file named `PiBot.zip` and a `Show all` button.



The screenshot displays the Kibotics web application interface. At the top, the browser address bar shows the URL `https://kibotics.org/websim/blockly/siguelinea_pibot_scratch_ir`. The main interface is divided into several sections:

- Navigation and Controls:** Includes buttons for "Ejecutar" (Execute), "Reiniciar posición" (Reset position), "Cámara" (Camera), "Guardar" (Save), and "Ejecutar en Real" (Execute in Real).
- Block-based Programming (Blockly):** A central workspace with a "Inicio" (Start) block and a "Bucle infinito" (Infinite loop) block. The code logic is as follows:
 - establecer línea a Para myRobot detecta la línea de color (Set line to Para myRobot detects the line of color)
 - Avanzar myRobot a velocidad 0.2 (Advance myRobot at speed 0.2)
 - si línea == 1 (if line == 1)
 - hacer Girar myRobot a la izquierda a velocidad 0.2 (do Girar myRobot to the left at speed 0.2)
 - si línea == 2 (if line == 2)
 - hacer Girar myRobot a la derecha a velocidad 0.2 (do Girar myRobot to the right at speed 0.2)
- Left Panel:** A category menu with options: Variables, Text, Logic, Loops, Math, Lists, Functions, RobotAPI (expanded to show Constructor, Motors, Camera, Tools, Sensors).
- 3D Simulation:** A central 3D view of a green field with a white track. A small robot is positioned on the track, with white lines radiating from it, representing its field of view or sensor range.
- Bottom Right:** A "Robot Camera" view showing an "OpenCV image" of the robot's perspective.

- **Python**: lenguaje de texto, sencillo pero potente
- Se usa también en la universidad y trabajo



The screenshot displays the Kibotics web interface. On the left, a Jupyter notebook titled 'sigue_linea_ir (autosaved)' contains Python code for controlling a robot. The code includes comments in Spanish and several function definitions for line following logic.

```

#robot = PiBot.damerobot()
# Driver 2.0
robot = PiBot()

#La programación del siguelineas la he realizado de acuerdo
#a lo que se explica en los apuntes. El único detalle, es la
#programación de case mediante un switcher con el fin de
#evitar varios "if" en cadena

#programacion siguelineas
ultMov = 0

def Negro_Negro():
    robot.avanzar(0.2)
    time.sleep(0.1)
    #ultMov = 0

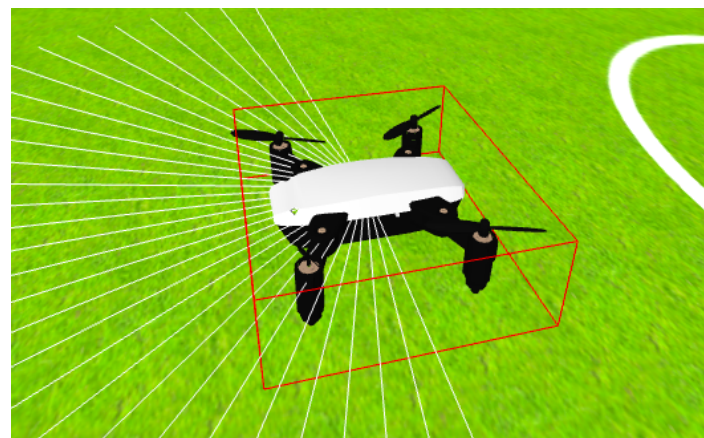
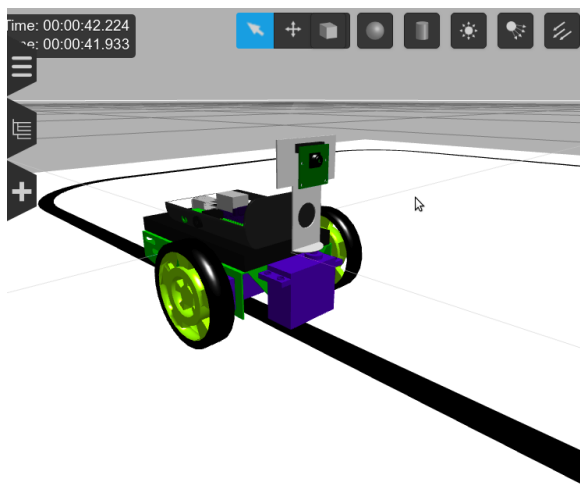
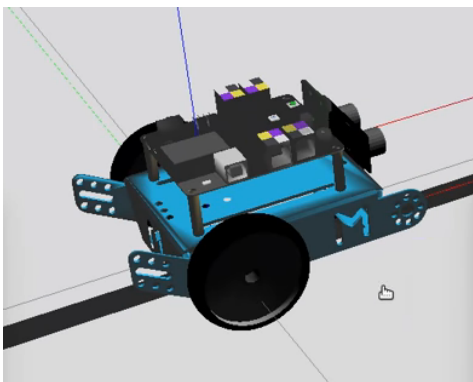
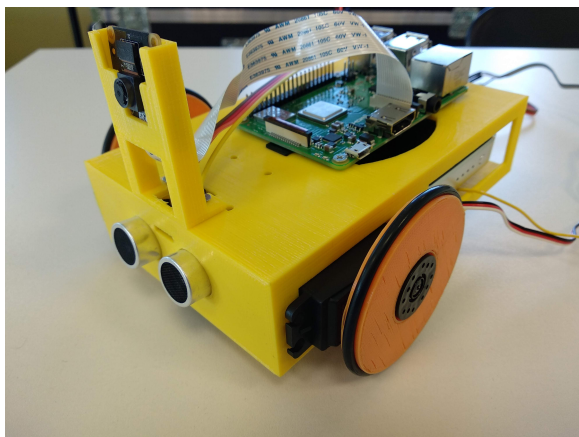
def Negro_Blanco():
    global ultMov
    robot.girarIzquierda(0.15)
    time.sleep(0.05)
    ultMov = 1

def Blanco_Negro():
    global ultMov
    robot.girarDerecha(0.2)
    time.sleep(0.05)
    ultMov = 2

def Blanco_Blanco():
    global ultMov
    if ultMov == 1:
        robot.girarIzquierda(0.2)
        time.sleep(0.05)
    ..
    ..
    
```

On the right, a 3D simulation shows a small robot on a track. The track is a black line on a white background, forming a rectangular shape with rounded corners. The robot is positioned on the track, and the simulation interface includes a toolbar with various controls and a timer showing 'Time: 00:00:49.363' and 'Time: 00:00:49.044'.

Múltiples robots soportados: mBot, drones,...



Casos de éxito

- Escuela de Pensamiento Computacional INTEF (2019)
- Ayuntamiento de Fuenlabrada (2019, 2020)
- Empresa Logix5 (2019)
- Universidad Rey Juan Carlos (2019)
- IES Martinez Uribarri (Salamanca, curso 2019-2020)
- Comunidad de Madrid (2020)
- ...

Educación superior en robótica, profesionales

Demanda creciente de formación en robótica

- Nuevas aplicaciones, productos y servicios robóticos
- Cada vez más ofertas laborales en este ámbito
- Ingenierías, profesionales
- Investigación
- *robot = hardware + software*
- *hardware = sensores + actuadores + computadoras*

Universidades

- Doctorado
- Másteres en las escuelas de ingeniería
 - Ing. Industrial, Ing. Informática, Ing. Telecomunicación...
 - Electrónica, Mecánica, Programación, Inteligencia Artificial
- Grados específicos
 - U.Alicante: Grado en Robótica
 - U.Rey Juan Carlos: Grado en Robótica Software

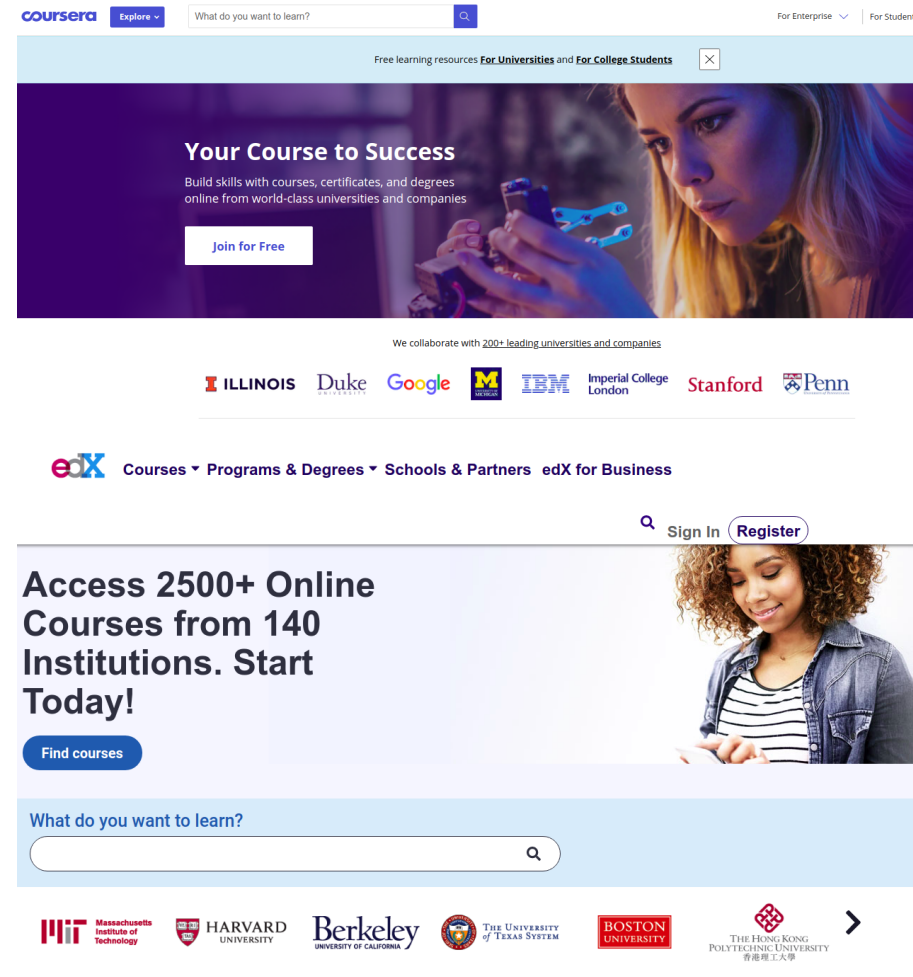
URJC: GRADO INGENIERÍA EN ROBÓTICA SOFTWARE

- <https://www.urjc.es/estudios/grado/3099-ingenieria-de-robotica-software>
- Fundamentos genéricos ingeniería
- Fundamentos específicos de robótica
- Dominios de aplicación

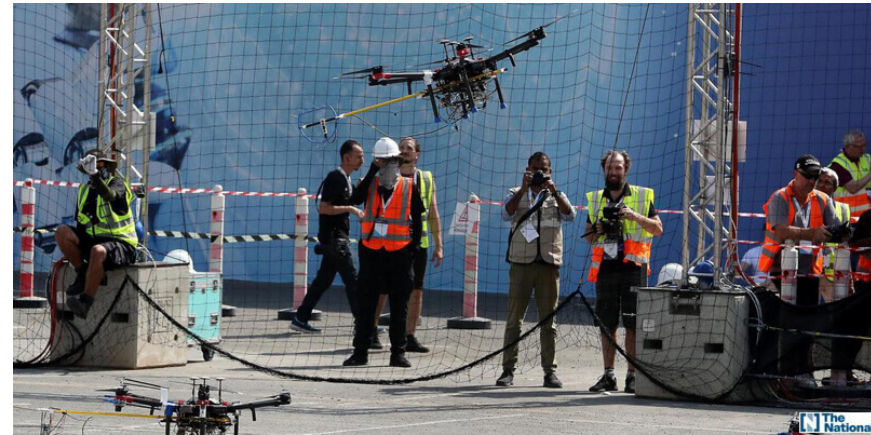
Curso 1º (60 ECTS)	Curso 2º (60 ECTS)	Curso 3º (60 ECTS)	Curso 4º (60 ECTS)
Emprendimiento e innovación en Robótica	Probabilidad y estadística	Robótica móvil	Aprendizaje Automático y Profundo
Álgebra	Fundamentos de Redes de Ordenadores	Sistemas distribuidos y concurrentes	Mecatrónica
Fundamentos físicos de la ingeniería	Diseño Software	Inteligencia Artificial	Robótica de Servicios
Fundamentos de la Programación	Sensores y actuadores	Redes de ordenadores para robots y máquinas	Robótica aérea
Arquitectura de computadores	Idioma Moderno	Ingeniería de Control	RAC
Cálculo	Deontología y normativa en robótica	Planificación y Sistemas Cognitivos	TFG
Evolución y futuro de la Robótica	Ampliación de matemática aplicada	Sistemas empotrados y de tiempo real	Practicas
Algoritmos y estructuras de datos	Arquitecturas SW para robots	Robótica Industrial	
Electrónica digital	Fundamentos de automática	Visión artificial	
Laboratorio de Sistemas	Sistemas Operativos	Modelado y Simulación de robots	

Formación no reglada

- Internet
- Masivo, en línea
- Cursos
- MOOCs:
 - Coursera
 - EDx
- Empresas:
 - Udemy
 - Udacity
 - TheConstruct



Competiciones robóticas internacionales



- Robocup: soccer, home, rescue...
- DARPA Robotics Challenge
- MBZIRC: drones...
- Program-a-Robot

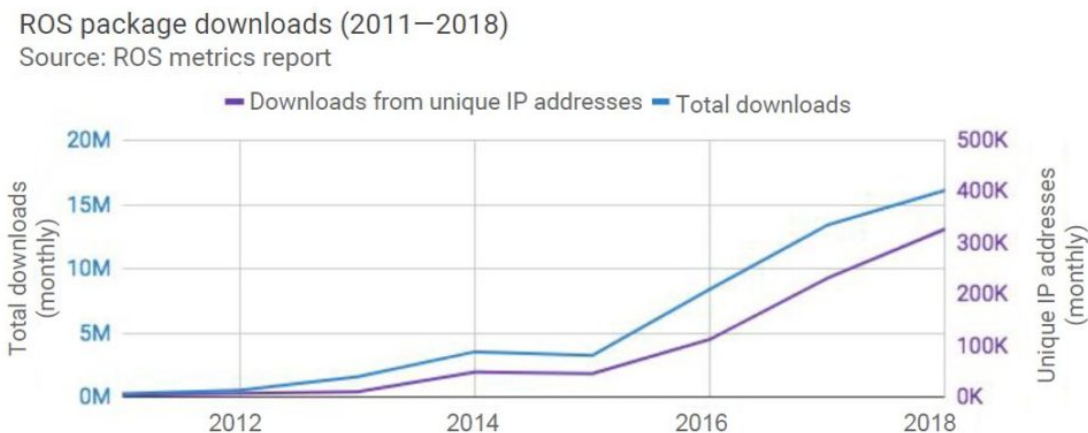
Plataforma Robot Operating System (ROS)

⋮ 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)

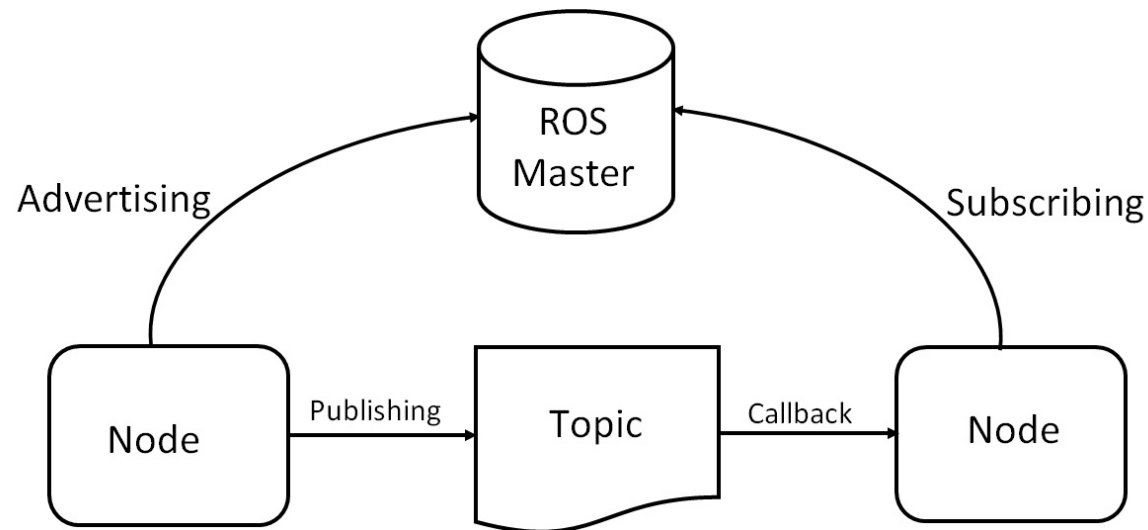


CARACTERÍSTICAS

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

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

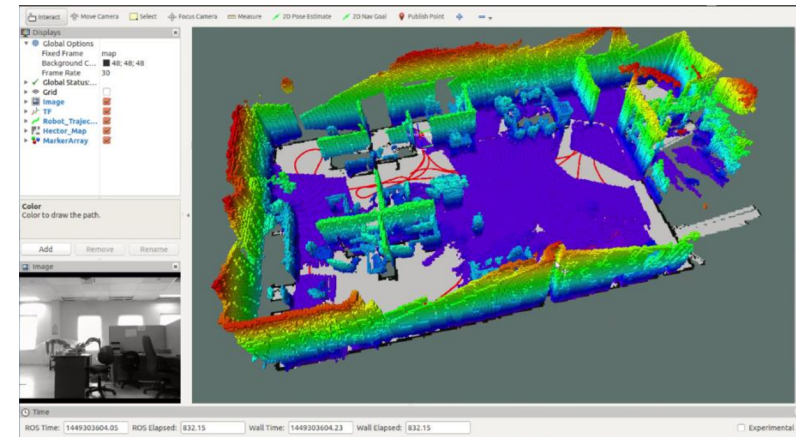
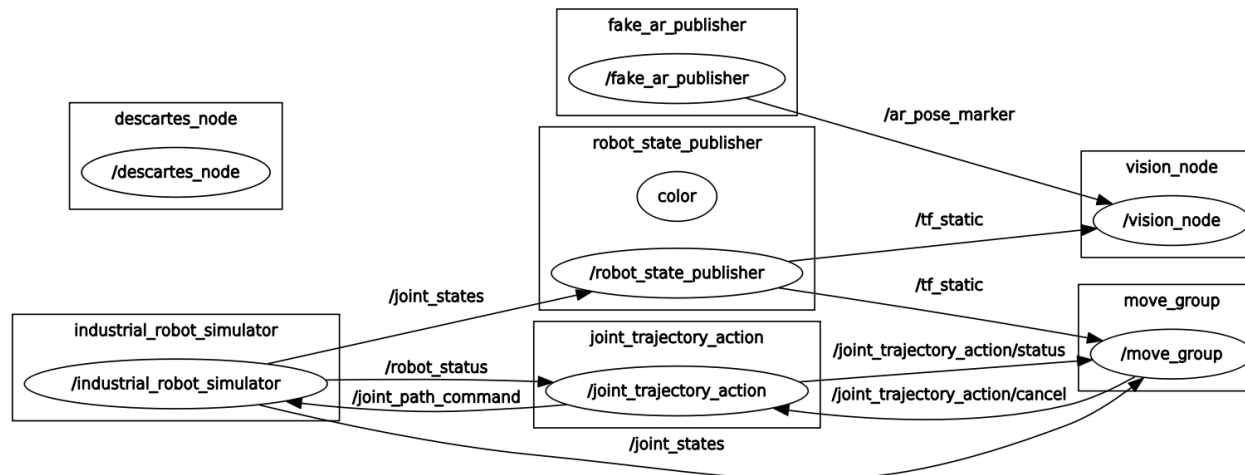
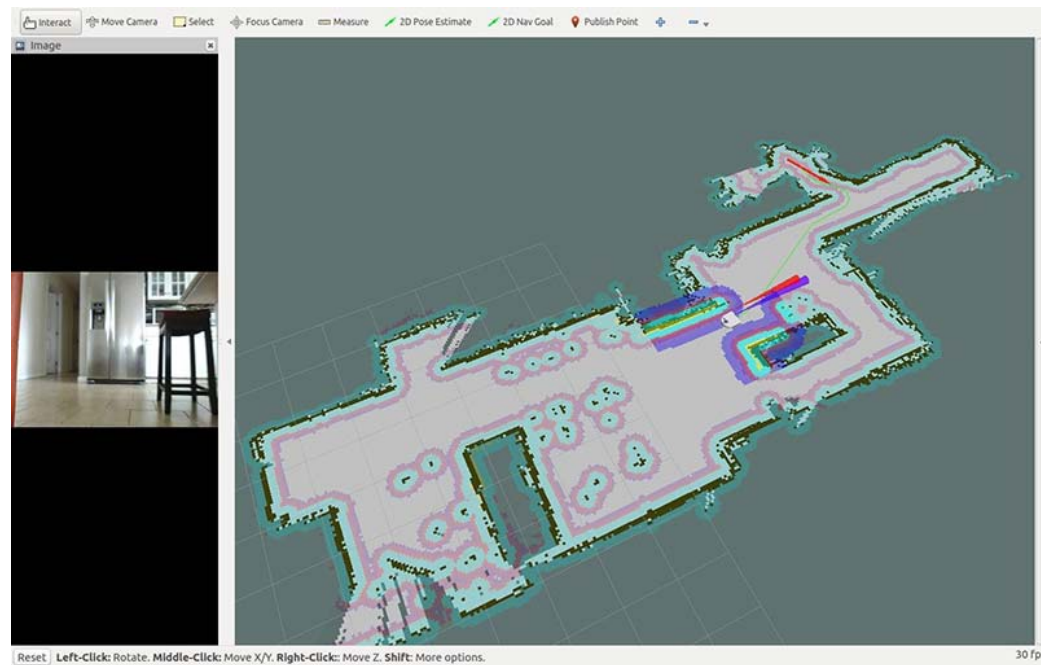


Figura 7. Visualiza 2D & 3D espacial, usa RVIZ



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)
 - Multiplataforma
Linux, Windows, MacOS
- Ignition simulator
 - cloud



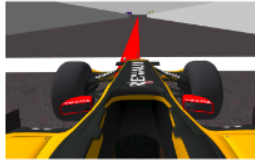

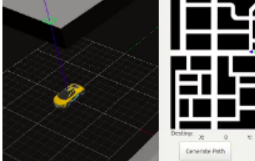



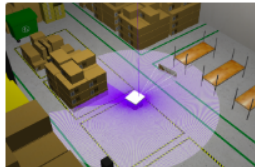
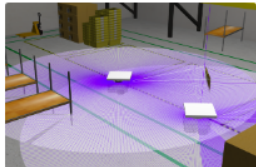

Plataforma RoboticsAcademy

- <https://jderobot.github.io/RoboticsAcademy/>
- **Contenidos educativos**
- *Learn by doing*
- ¡Divertidos! *Gamificación*
- Lenguaje Python
- Simulador Gazebo
- Middleware ROS
- Foro
- Open source, internacional



Contenidos educativos: conjunto de ejercicios

- Conducción autónoma
- Robótica móvil
- Drones
- Robots industriales
- Visión Artificial

		
<p>Follow Line</p> <p>Complete a lap following the line painted on the racing circuit.</p> <p>Go!</p>	<p>Obstacle avoidance</p> <p>Local navigation. Virtual Force Field navigation algorithm using a F1.</p> <p>Go!</p>	<p>Global Navigation</p> <p>Global Navigation. Navigating a Tele Taxi autonomously.</p> <p>Go!</p>
		
<p>Car Junction</p> <p>Car Junction. Automated vehicle must stop and pass once the road is clear.</p> <p>Go!</p>	<p>Vacuum Cleaner</p> <p>Navigation algorithm for an autonomous vacuum.</p> <p>Go!</p>	<p>Vacuum Cleaner Loc</p> <p>Navigation algorithm for an autonomous vacuum with localization.</p> <p>Go!</p>
		
<p>Single Robot Amazon Warehouse (ROS2)</p>	<p>Multi Robot Amazon Warehouse (ROS2)</p>	<p>Drone Gymkhana</p> <p>Learn about different drone controls</p>

Visual Follow Line

Versions to run the exercise

Currently, there are 2 versions for running this exercise:

- ROSNode Templates
- Web Templates(Current Release)

The instructions for both of them are provided as follows.

Goal

The goal of this exercise is to perform a PID reactive control capable of following the line painted on the racing circuit.



Gallery

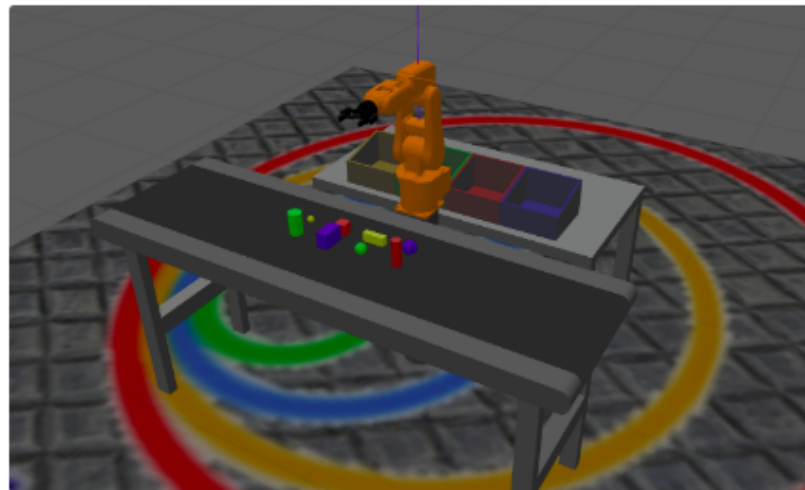
The students program a Formula1 car in a race circuit to follow the red line in the middle of the road.

Instructions for Web Templates

⚙️ TOC Visual Follow Line
Versions to run the exercise
Goal
Instructions for Web Templates
Installation
How to perform the exercise?
Instructions for ROSNode Templates
Installation
How to perform the exercise?
How to run your solution?
Theory
Control System
Types of Control System
Open Loop Control System
Closed Loop Control System
PID Control
Tuning Methods
Real Life Example
Hints
Detecting the Line to Follow
Coding the Controller
Illustrations
Demonstrative Video
Contributors
References

Pick and Place

The goal of this exercise is to learn the underlying infrastructure of Industrial Robot exercises (ROS + MoveIt + our own industrial robotics API) and get familiar with the key components needed for more complex exercises by completing the task of pick and place multiple objects and classify them by color or shape.



Gallery.

Installation

1. Install the [General Infrastructure](#) of the JdeRobot Robotics Academy.
2. Install MoveIt

```
sudo apt install ros-melodic-moveit
sudo apt-get install ros-melodic-ros-control ros-melodic-ros-cont
```

TOC Pick and Place
Installation
How to run the exercise
How should I solve the exercise
API
Environment Information
Convert Pose Message
Basic Robot Movement
Setup Grasp message
Pick and Place
Theory
Robot Workspace
Position and Orientation
Forward Kinematics and Inverse Kinematics
Hints
Relationship among ROS, MoveIt, Rviz, Gazebo, JdeRobot provided API
Object and Target lists:
Why does the robot sometimes cannot move to some desired pose?
Ignorable ERROR and WARNING
Demonstration video of the solution