

Unibotics: plataforma web de programación de robots en ingeniería

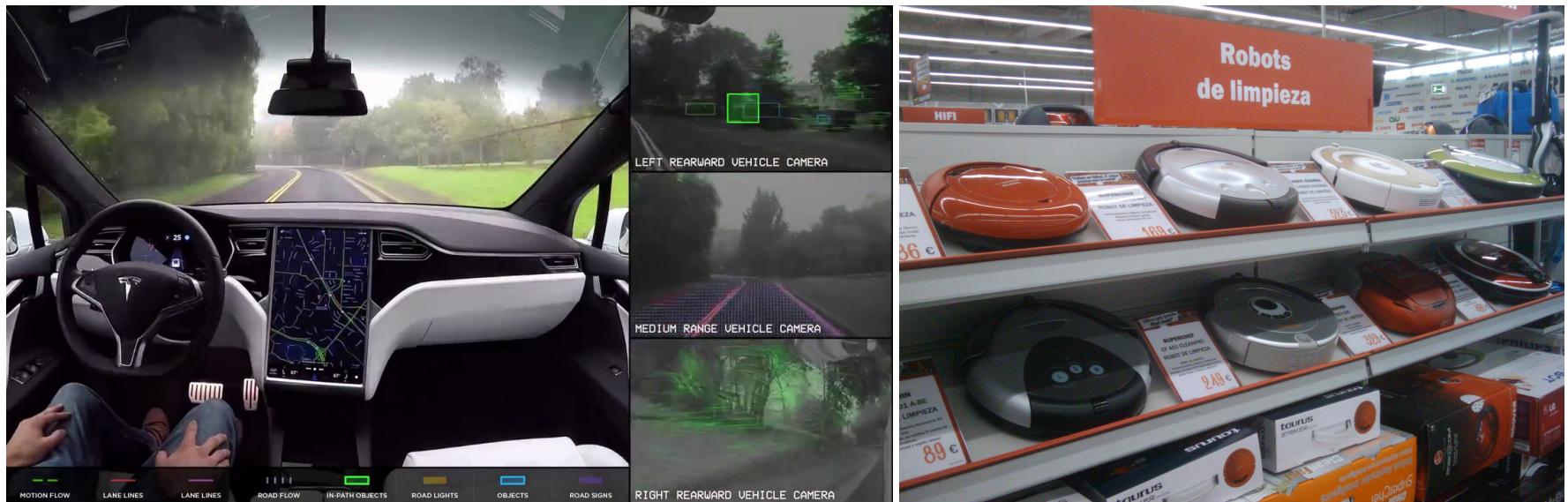


Asociación de Robótica e Inteligencia Artificial JdeRobot
CIF G88145909

*josemaria.plaza@gmail.com, david.rolan@urjc.es,
pedro.arias@upm.es, d.valladaresvigara@gmail.com*

- Robótica una tecnología en auge
- Plataforma Unibotics
- Contenidos
- ¿Cómo está hecha?
- Conclusiones

Robótica, una tecnología en auge



- Cada vez más aplicaciones robóticas útiles en la sociedad

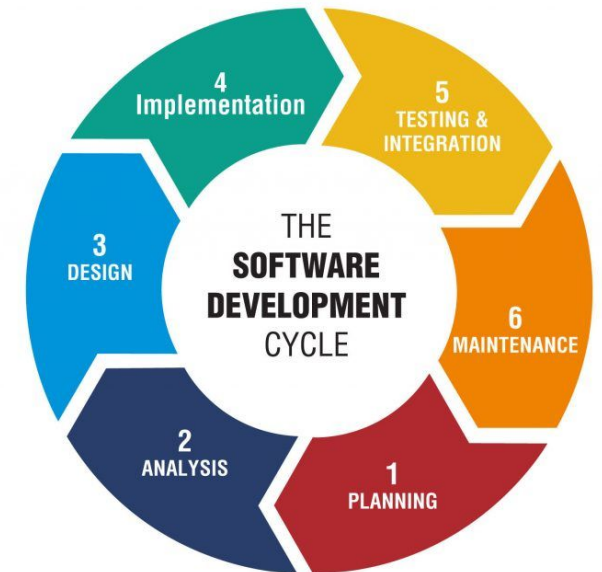
Demanda creciente de formación en robótica

- Nociones básicas útiles para todos en XXI, educación preuniversitaria
- Introducción divertida a las tecnologías STEAM
- Demanda laboral creciente
- Ingenierías, profesionales
- Grados, másteres, doctorados
- TheConstruct, Riders.ai
- MOOCs, videos OK pero lado práctico flojo

Programación de robots

- *Robot = hardware + software*
- *Hardware = sensores + actuadores + computadoras*
- **En el software reside la inteligencia**
- Lenguajes: C++, Python, Matlab...
- Middleware ROS: ROS1, ROS2
 - funcionalidad ya resuelta: bibliotecas, drivers, nodos...
 - herramientas
- **Algoritmos**: percepción, planificación, control
- Programación basada en datos: redes neuronales

- Interfaces gráficas: HRI y útiles para depuración
- Simuladores: útiles para depuración
Gazebo, Webots, Carla, CoppeliaSim...
- Ciclo normal de desarrollo software:
 - diseño
 - editar el código fuente en un fichero de texto
 - generar el ejecutable
 - depuración
- ROS: aplicación robótica es una *orquesta de nodos* que se comunican entre sí



Programar robots es complejo

- El software de un robot combina muchas piezas: drivers, herramientas, bibliotecas, nodos...
- Entorno + aplicación en un sistema operativo
- Una aplicación robótica tiene muchas dependencias: ROS, openCV, PX4, TensorFlow, PyTorch...
- Instalación complicada
- Heterogeneidad
- Algoritmos sofisticados

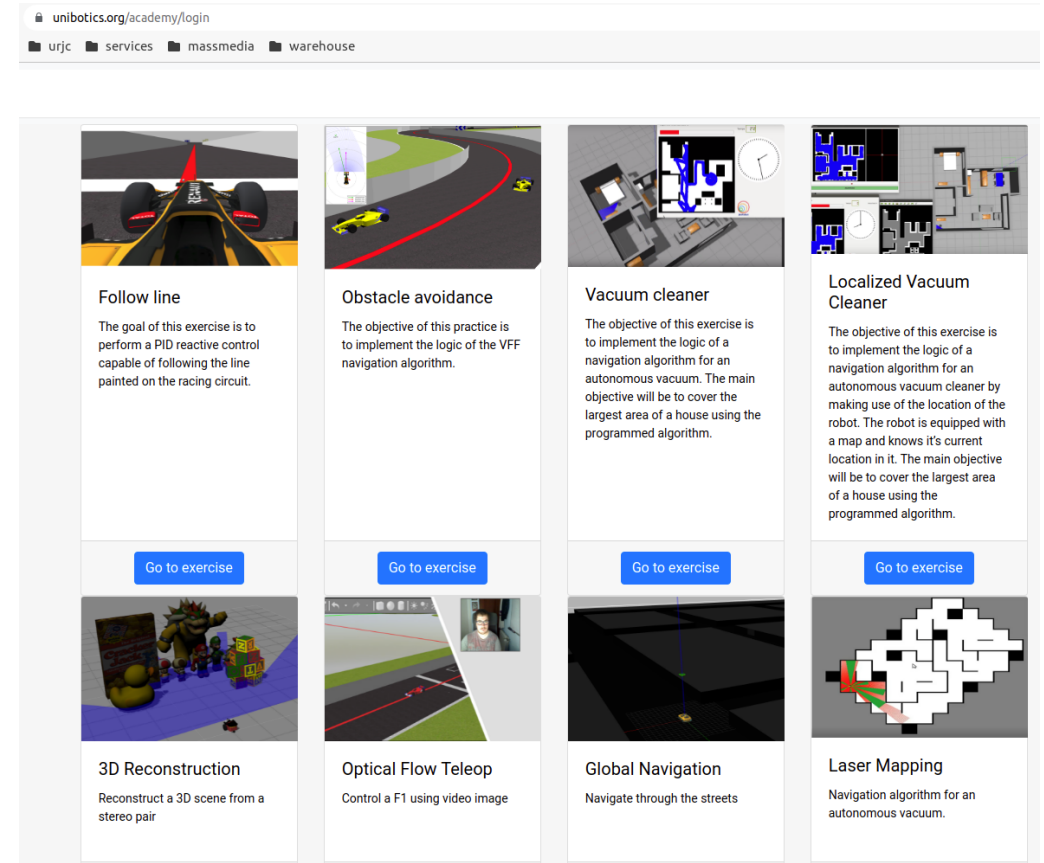


Unibotics

- Herramienta para aprender robótica
- Programar robots desde el minuto 1, simplificar curva de entrada
- Plataforma **web** de **programación de robots**
 - <https://unibotics.org>
 - browser como editor y como GUI
- Dependencias pre-instaladas en un contenedor docker
- Multiplataforma: Linux, Windows, MacOS, tabletas, móvil
- Abierta y gratuita
- Internacional, todo en inglés

Características

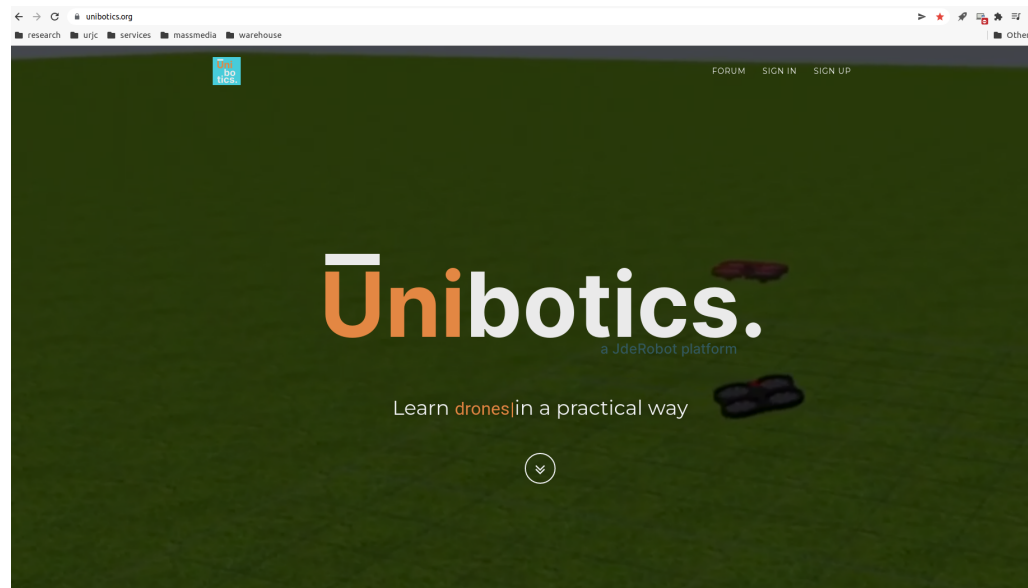
- Grado, master, doctorado
- Énfasis en programación, lenguaje **Python**
- Contenidos: **colección de retos**
Learn by doing
- Robots físicos y simulados
- Interacción social, foro
- Gamificación
- Estadísticas internas
- Evaluación automática



The screenshot shows the Unibotics Academy login page. The browser address bar is `unibotics.org/academy/login`. There are navigation links for `urjc`, `services`, `massmedia`, and `warehouse`. The main content area displays a grid of exercise cards, each with a title, description, and a 'Go to exercise' button.

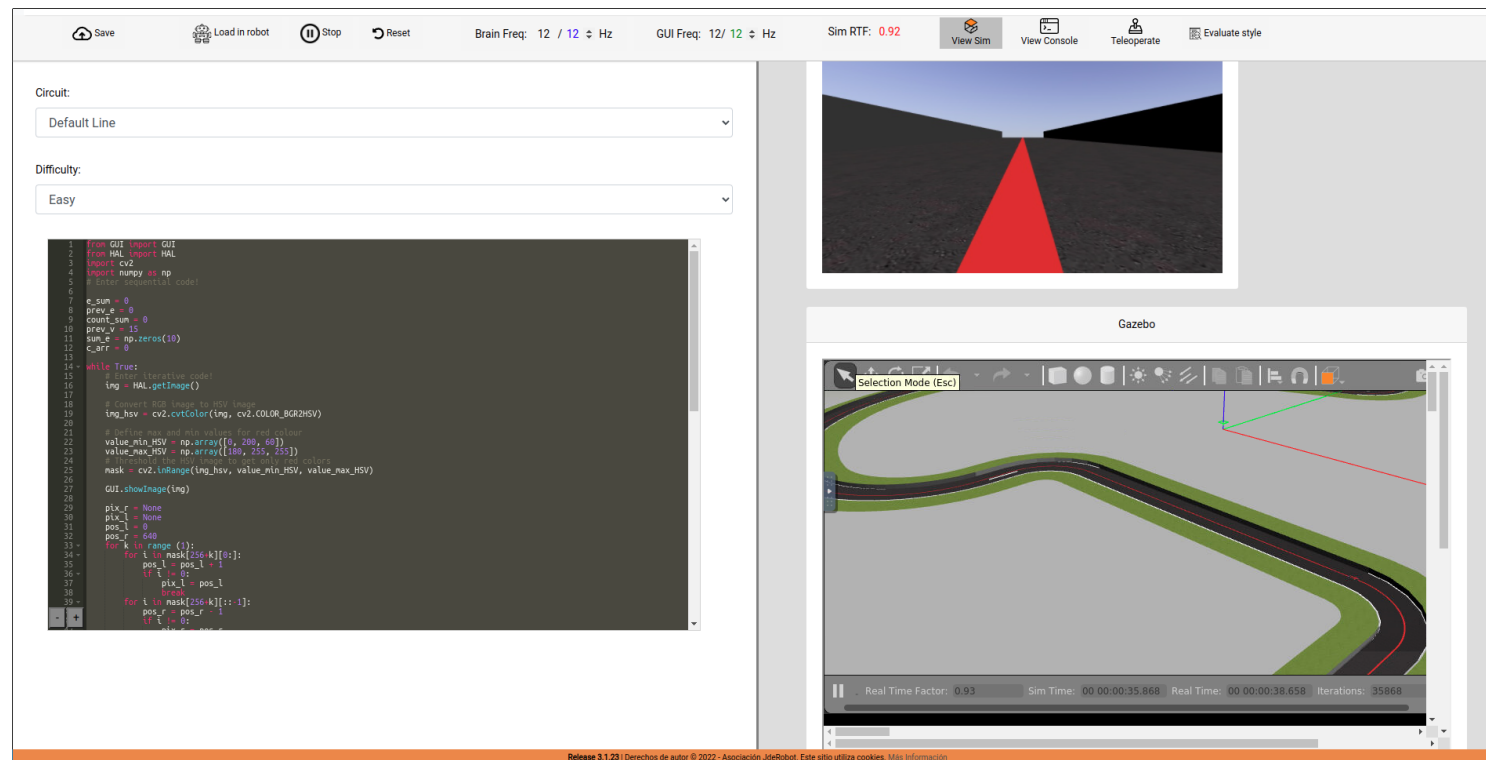
Exercise Title	Description
Follow line	The goal of this exercise is to perform a PID reactive control capable of following the line painted on the racing circuit.
Obstacle avoidance	The objective of this practice is to implement the logic of the VFF navigation algorithm.
Vacuum cleaner	The objective of this exercise is to implement the logic of a navigation algorithm for an autonomous vacuum. The main objective will be to cover the largest area of a house using the programmed algorithm.
Localized Vacuum Cleaner	The objective of this exercise is to implement the logic of a navigation algorithm for an autonomous vacuum cleaner by making use of the location of the robot. The robot is equipped with a map and knows its current location in it. The main objective will be to cover the largest area of a house using the programmed algorithm.
3D Reconstruction	Reconstruct a 3D scene from a stereo pair
Optical Flow Teleop	Control a F1 using video image
Global Navigation	Navigate through the streets
Laser Mapping	Navigation algorithm for an autonomous vacuum.

- **En línea**, basta con un navegador web
- Accesibilidad: desde cualquier sitio (clase, casa), a cualquier hora
- No hay que instalar (casi) nada
- Alumnos URJC literalmente *nada*, granja de ordenadores
- Basada en RoboticsAcademy (open source y offline)
- Basada en ROS y en simulador Gazebo



Editor e Interfaz gráfica en página web

- Escenario + **cuerpo** + **cerebro**
- Se programa el cerebro, que se conecta al cuerpo (sensores, actuadores)
- APIs: robot y GUI (visualización para depurar)



The screenshot displays the Unibotics web interface. On the left, there is a code editor with the following Python code:

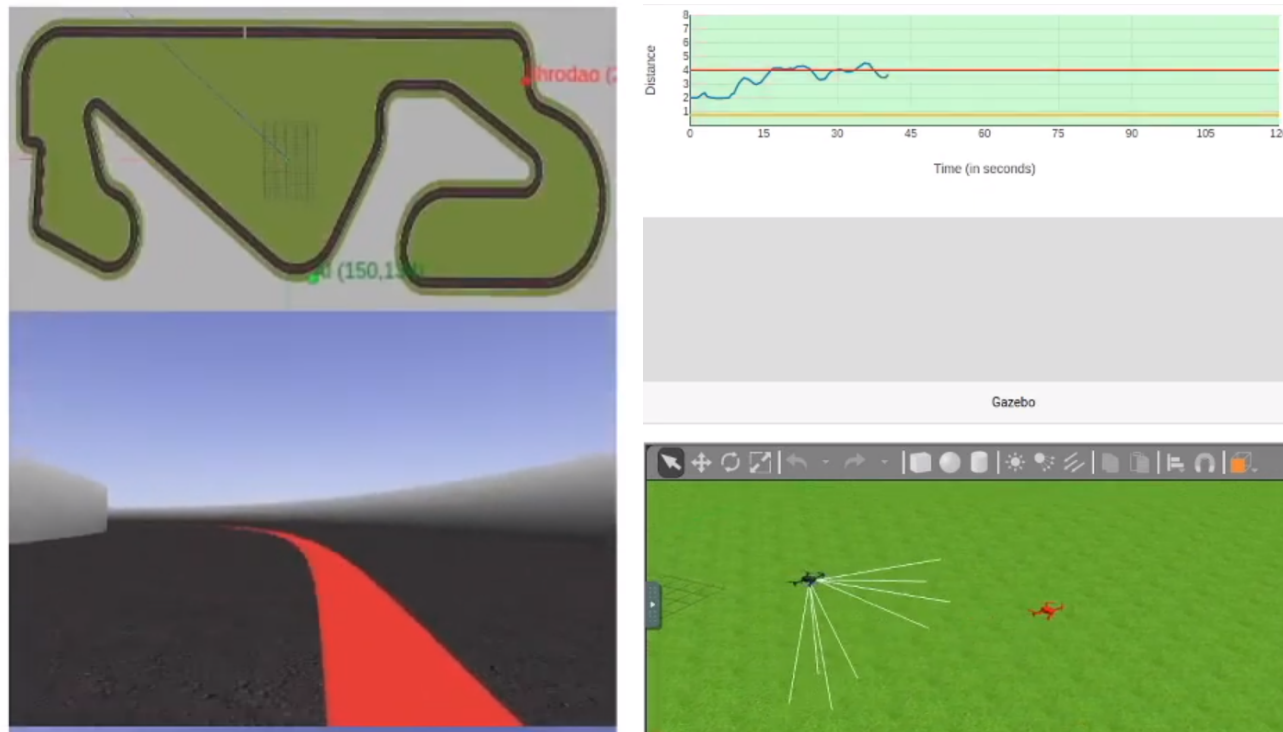
```

1 from GUI import GUI
2 from HAL import HAL
3 import cv2
4 import numpy as np
5 # Import the robot model
6
7 e_sum = 0
8 prev_e = 0
9 count_sum = 0
10 prev_v = 15
11 sum_e = np.zeros(10)
12 c_dT = 0
13
14 while True:
15     # Enable iterative code!
16     img = HAL.getImage()
17
18     # Convert image to HSV
19     img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
20
21     # Define the range of the color
22     value_min_HSV = np.array([0, 200, 60])
23     value_max_HSV = np.array([180, 255, 255])
24     # Create a mask for the color
25     mask = cv2.inRange(img_hsv, value_min_HSV, value_max_HSV)
26
27     GUI.showImage(img)
28
29     pix_r = None
30     pix_l = None
31     pos_l = 0
32     pos_r = 440
33
34     for k in range(1):
35         for i in mask[256:k][0:]:
36             pos_l = pos_l + 1
37             if i > 0:
38                 pix_l = pos_l
39
40         for i in mask[256:k][::-1]:
41             pos_r = pos_r - 1
42             if i > 0:
43                 pix_r = pos_r
44
45     # Calculate the error
46     e = (pos_r - pos_l) / 2
47     e_sum = e_sum + e
48     count_sum = count_sum + 1
49     prev_e = e
50     prev_v = prev_v - e
51     sum_e[count_sum - 1] = e
52     c_dT = c_dT + e
53     # Calculate the control signal
54     v = prev_v + c_dT
55     # Send the control signal to the robot
56     HAL.setVelocity(v)
57
58     # Wait for the next iteration
59     time.sleep(0.01)
    
```

On the right, there is a Gazebo simulation window showing a robot on a track. The interface includes a top toolbar with buttons for Save, Load in robot, Stop, Reset, Brain Freq, GUI Freq, Sim RTF, View Sim, View Console, Teleoperate, and Evaluate style. The simulation window shows a top-down view of a track with a red cone in the center. The Gazebo window also includes a toolbar with Selection Mode (Esc) and various simulation controls. At the bottom of the Gazebo window, the Real Time Factor is 0.93, Sim Time is 00:00:00:35.868, Real Time is 00:00:00:38.658, and Iterations are 35868.

Gamificación, juegos robóticos

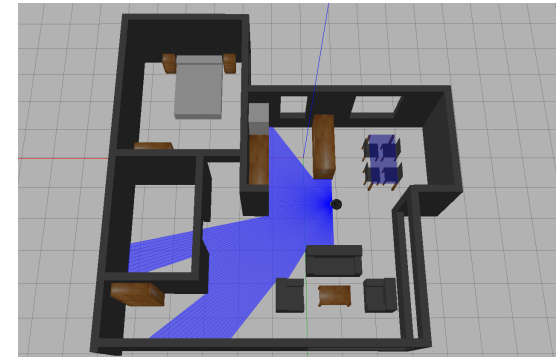
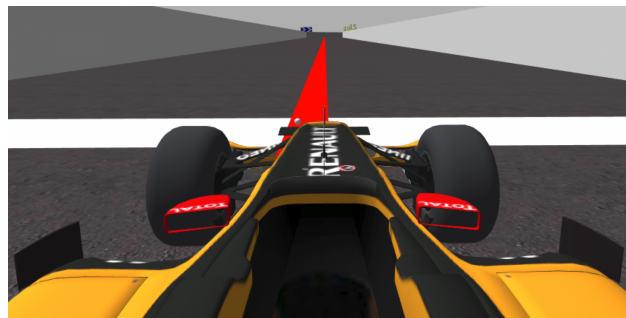
- Puntuación automática, *mayor motivación*
- *Oponentes* automáticos de varios niveles, bots. Torneos
- SigueLíneas competitivo, gato-ratón con drones



Contenidos

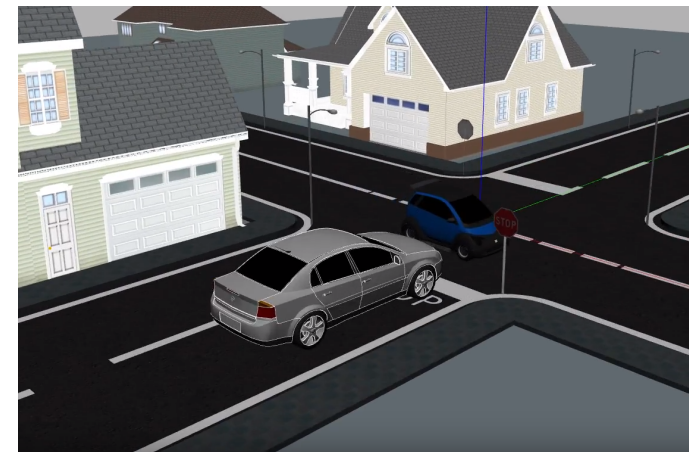
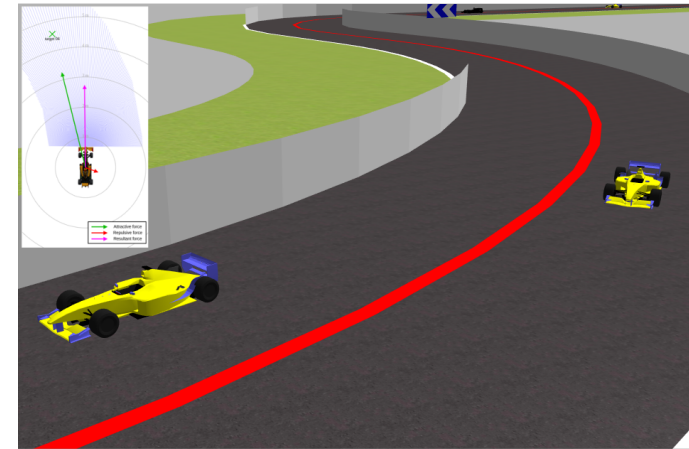
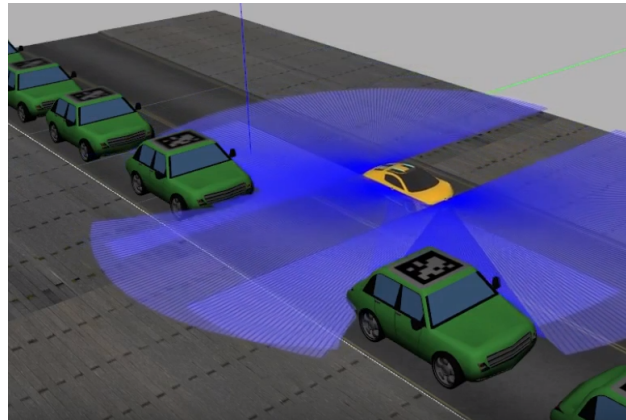
Robots móviles

- Construcción de mapas
- Autolocalización de robots
- Aspiradora gama baja, gama alta
- Formula1 sigue línea, control visual



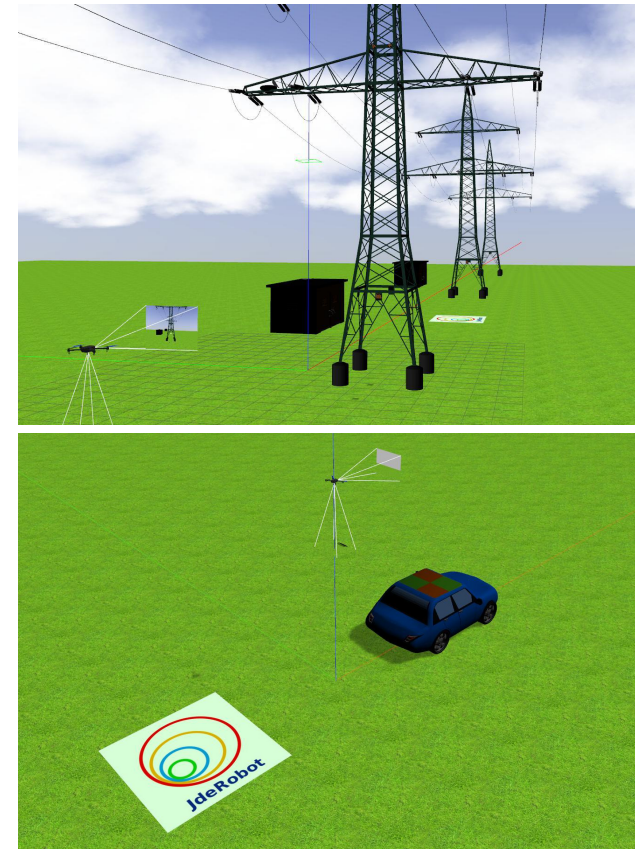
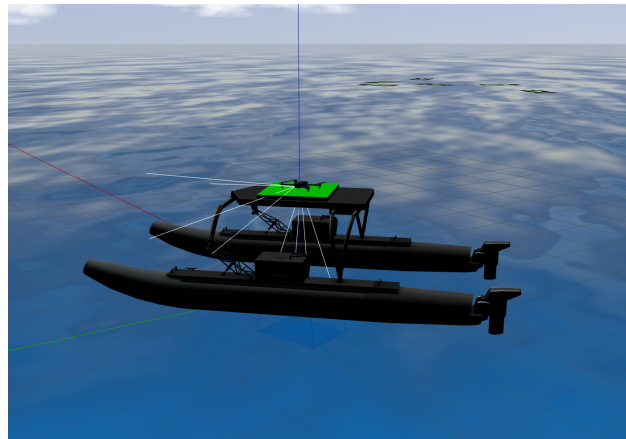
Conducción autónoma

- Navegación local y global
- Esquivar obstáculos
- Negociar un cruce
- Autoaparcamiento
- Teletaxi



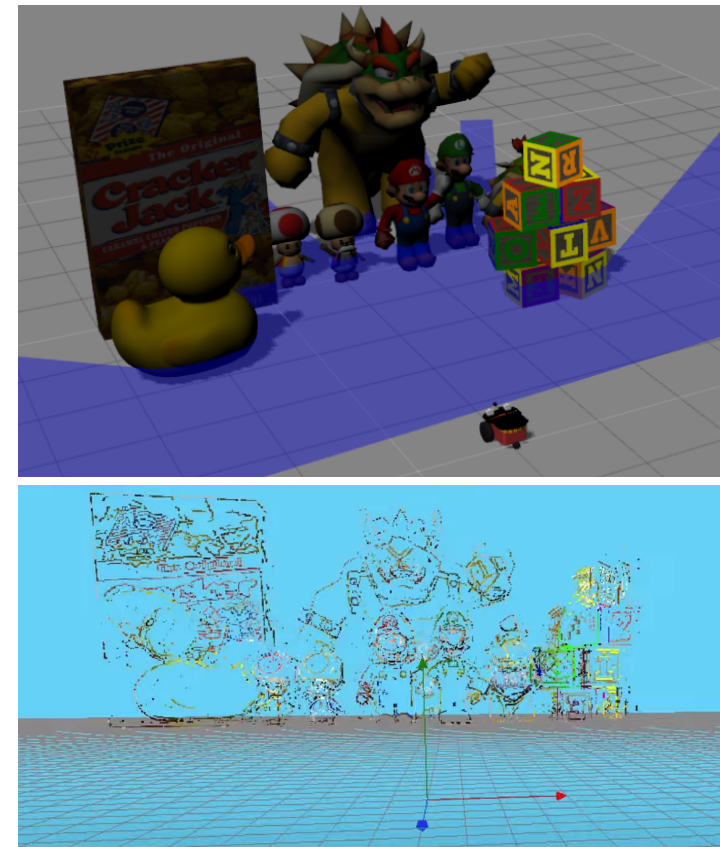
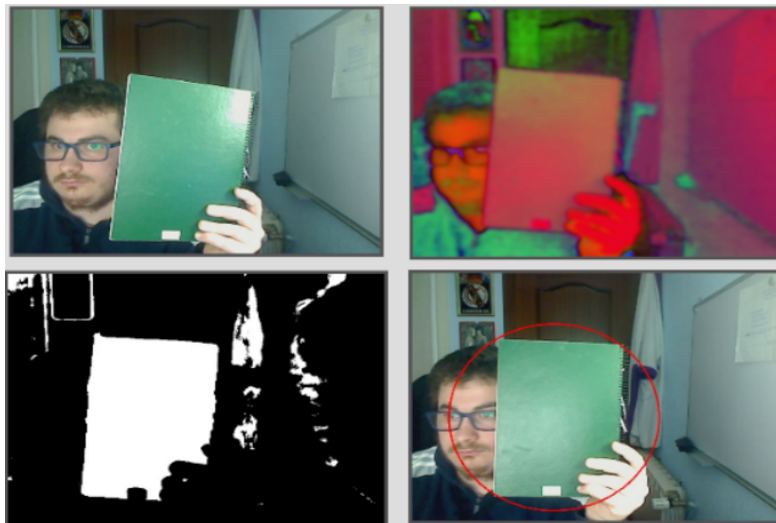
Drones

- Rescate de personas
- Entrega de paquetes
- Inspección de torre eléctrica



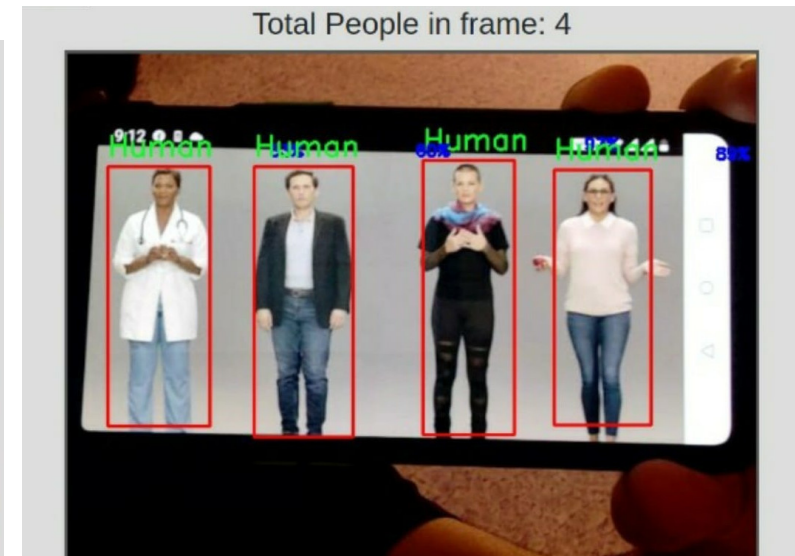
Visión artificial

- Filtro de color
- Flujo óptico
- Reconstrucción 3D



Aprendizaje automático

- DeepLearning sobre imágenes
- Clasificación de números
- Detección de personas en imagen



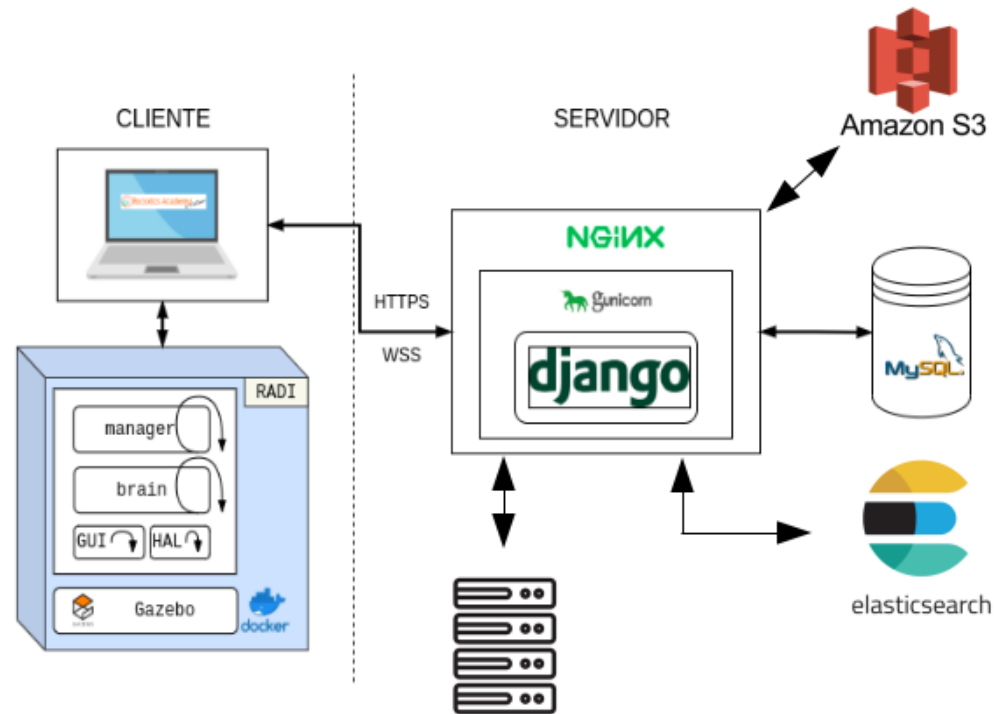
Desarrolladores: ¿Cómo está hecha?

- Tecnologías web
 - backend: Django, Elasticsearch, MySQL
 - frontend: JavaScript, HTML5, DASH
- Tecnologías robóticas
 - ROS (ROS1-Noetic, ROS2-Foxy)
 - simulador Gazebo (Gazebo-11)
- Tecnologías DevOps
 - contenedores docker
 - Amazon Web Services
 - GitHub actions
 - Selenium



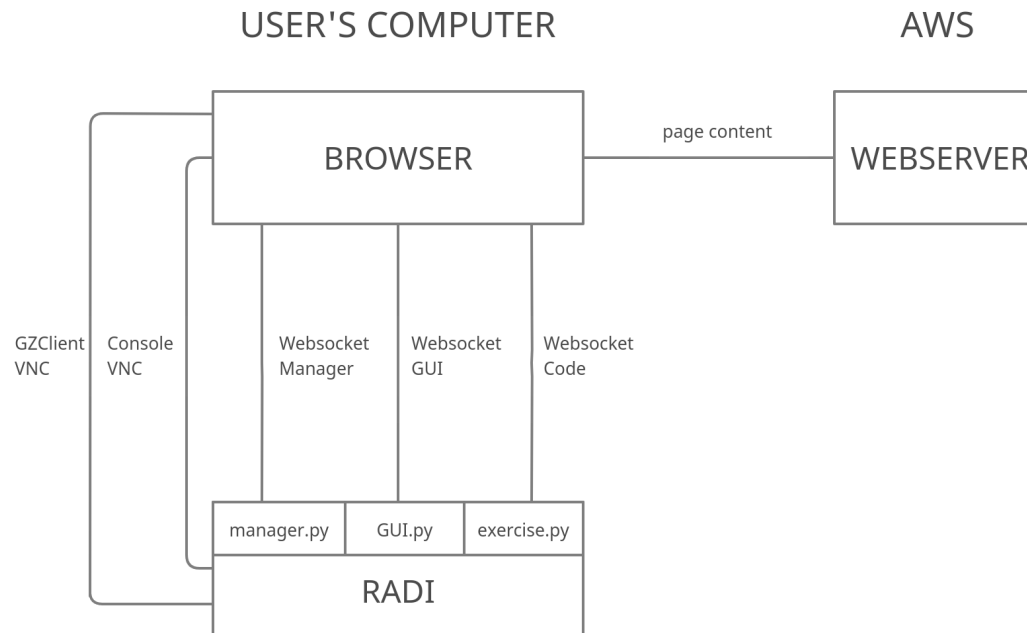
Diseño

- Lado usuario: frontend en browser, contenedor docker
- Lado servidor: Django, AWS, S3, granja, MySQL, Elasticsearch



Lado usuario: browser y contenedor

- Cada ejercicio: 1 plantilla Python y 1 página web que se comunican vía *websockets*
- Docker RADI: n `exercise.py`, 1 `manager.py`, [simulador](#)
- Cómputo distribuido, escalable a muchos usuarios simultáneos
- `manager.py`: arranca la simulación y la plantilla Python correspondientes al entrar en el ejercicio (y las cierra al salir)



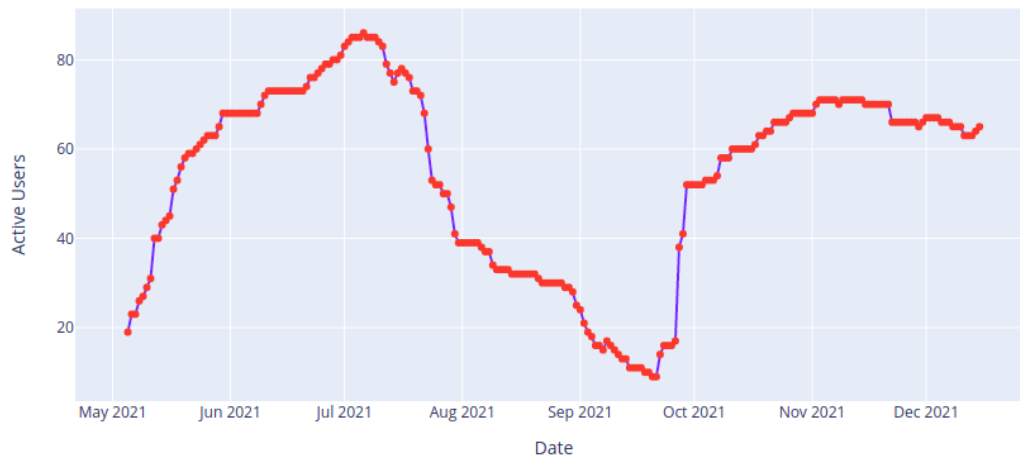
- `exercise.py`: brain, HAL y GUI
- HAL: conexión con sensores y actuadores del simulador
- Servidor VNC: visor del simulador y de consola

Estadísticas internas automáticas

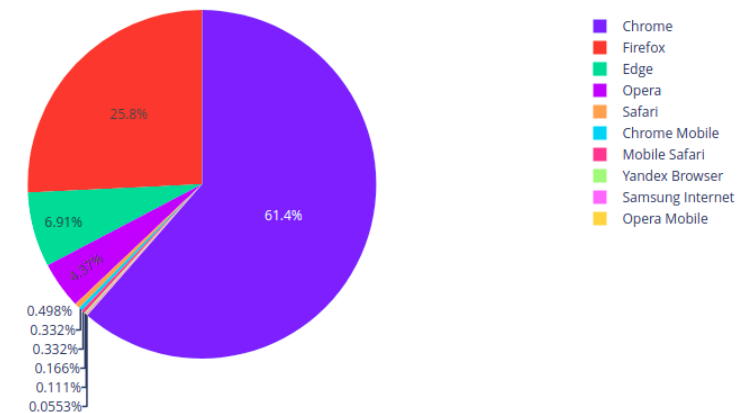
- Sondas graban eventos en Elasticsearch
- Se visualizan en una página web con DASH, gráficas dinámicas

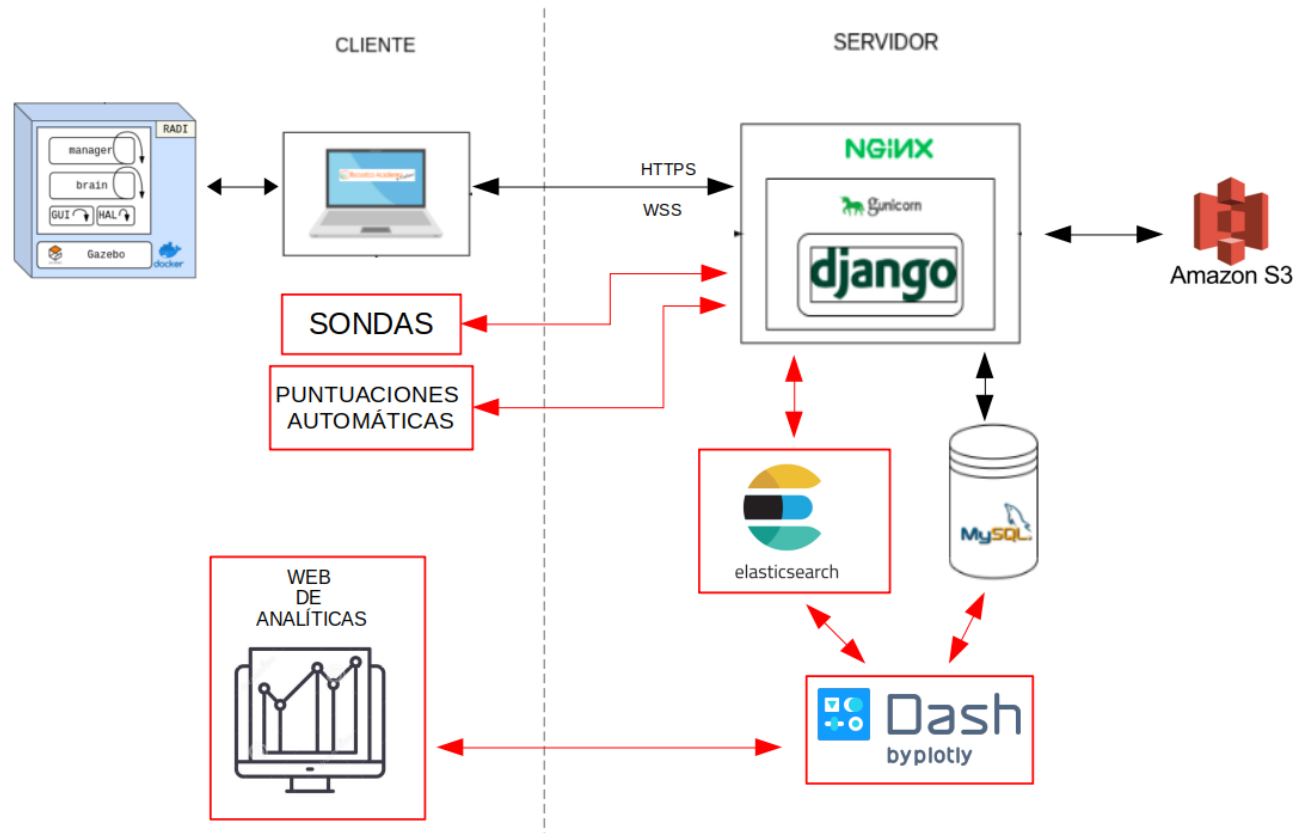
Active users

Start Date → End Date



Start Date → End Date



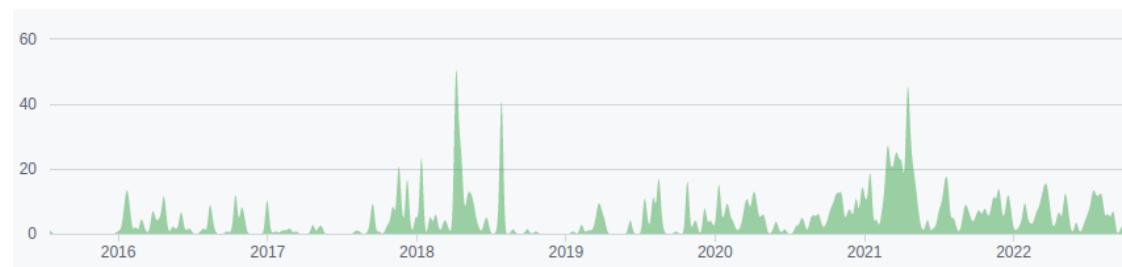


Despliegues, tecnologías DevOps

- D1 para desarrolladores, en local
- D2 en test: servidor de pruebas y minigranja
- D3 en producción: AWS y granja
- GitHub Actions
- Tests automáticos: Selenium

Conclusiones

- Tecnologías punteras: de robótica, de web y de DevOps
- Da servicio a estudiantes reales
 - URJC Master Visión Artificial: Visión en Robótica (50)(2020-)
 - URJC Grado Ing. Robótica Software: (2021-)
“Robótica móvil” (30), “Robótica de servicio” (20)
 - URJC Grado Ing. Telemática: “Robótica” (70)(2016-)
- Mejora continua, comunidad activa de desarrolladores socios (españoles e internacionales), TFGs, TFMs, GSoC, UPEs



Evolución

- RoboticsAcademy (2016-), instalable en local
- Unibotics (2018-), web
- Desplegado en la nube, 24x7
- En producción para alumnos URJC (2020-)
- Pilotos en UAH y UAlicante

¿En qué estamos ahora?

- Refactorizar frontend con REACT
- Torneos automáticos, retransmitidos por Twitch
- Juegos síncronos en línea, con WebRTC
- Aceptamos contribuidores, **oportunidad de aprender tecnologías**
 - posibilidad de Prácticas en Empresa
 - posibilidad de TFG
 - Google Summer of Code