

PRUEBAS AUTOMÁTICAS EN UNA PLATAFORMA EDUCATIVA WEB DE ALTA DISPONIBILIDAD

...

Autor: Felicidad Abad Quintanilla
Tutores: Jose María Cañas Plaza
David Valladares



1

INTRODUCCIÓN

2

OBJETIVOS Y PLANIFICACIÓN

3

HERRAMIENTAS UTILIZADAS

4

AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM

5

EJERCICIO INTRODUCTORIO AL PROCESAMIENTO DE IMAGEN.

6

CONCLUSIONES



INTRODUCCIÓN

TECNOLOGÍAS WEB

Herramientas y estándares para crear el contenido de Internet.

Se dividen en:

Tecnologías web del lado cliente.

Se ejecutan en el navegador

- HTML5
- CSS
- JavaScript

Tecnologías web del lado servidor.

Para desarrollar la funcionalidad de una aplicación web

- Lenguajes de programación lado servidor (Python, Java, ...)
- Entornos del servidor
- Bases de datos

INTRODUCCIÓN

CI/CD

Automatizar las tareas repetitivas para agilizar el desarrollo de software

SISTEMAS CI/CD

Procesos para implementar CI/CD en un proyecto de desarrollo web

INTRODUCCIÓN

Doméstico



Figura 1.6: Robot roomba.

Médico



Figura 1.7: Robot Aeo.

Ocio



Figura 1.8: Robot Dog-e.



Figura 1.9: Robot Spot de Boston Dynamic.

La plataforma web
sobre que se realiza
este Trabajo de Fin de
Grado: **UNIBOTICS**



OBJETIVOS Y PLANIFICACIÓN

OBJETIVOS:

- Implementar pruebas unitarias de frontend
- Implementar pruebas integrales
- Implementar un nuevo ejercicio procesamiento de imagen

METODOLOGÍA:

- Se divide el trabajo en distintas fases
- Reuniones semanales para comprobar avances y resolver dudas



HERRAMIENTAS UTILIZADAS

01. SELENIUM

02. DJANGO

03. DOCKER

04. GITHUB ACTIONS

05. PYTHON

06. HTML

07. YAML

08. OPENCV

09. UNIBOTICS

AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM

Se implementan pruebas automatizadas en el despliegue a producción.

1. **Pruebas unitarias de frontend.**
2. **Pruebas integrales**

Django proporciona un módulo para ayudar con la automatización: ***django.test***

Se usa la clase *LiveServerTestCase*.



AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM

Para configurar el entorno de prueba:

01

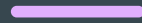
**Instalar
controladores**

Se instalan los
controladores de
Selenium.



02

**Crear archivo
auxiliar
operations.py**

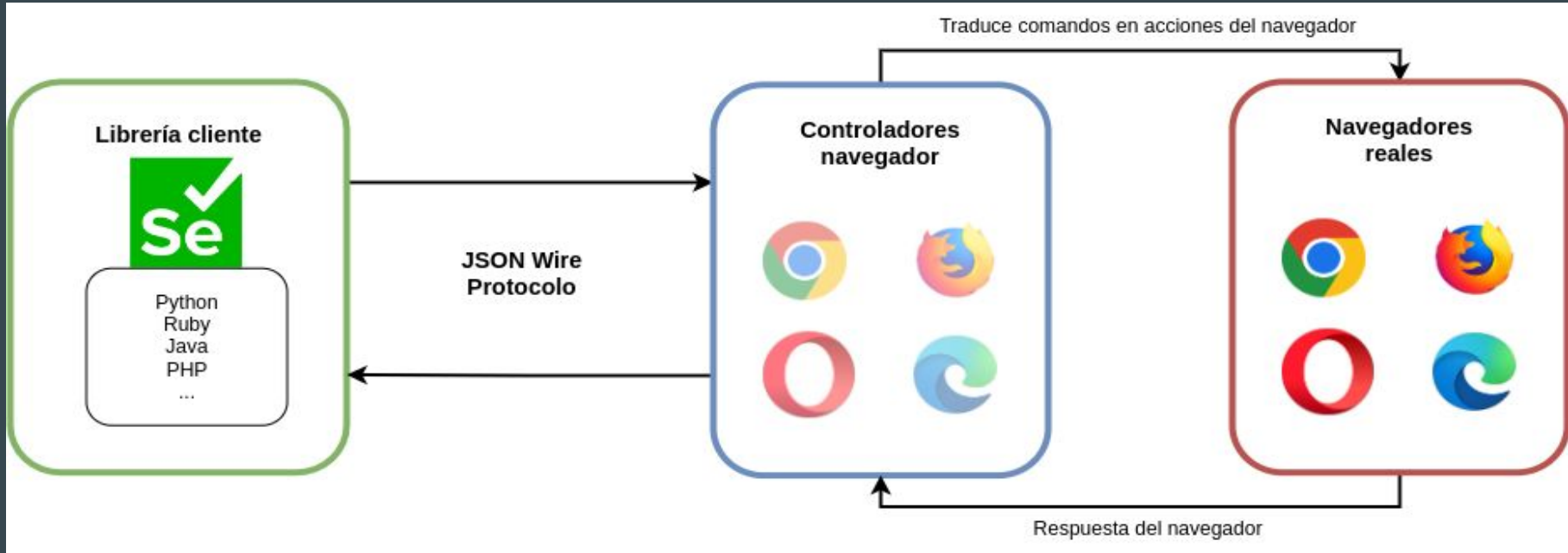


03

**Configurar los
métodos setUp() y
tearDown()**



AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM

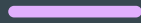


AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM

Para configurar el entorno de prueba:

01

**Instalar
controladores**



02

**Crear archivo
auxiliar
operations.py**



03

**Configurar los
métodos setUp() y
tearDown()**

Se crea el archivo
operations.py para definir en
él operaciones de creación de
usuario y ejercicio



AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM

Para configurar el entorno de prueba:

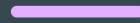
01

**Instalar
controladores**



02

**Crear archivo
auxiliar
operations.py**



03

**Configurar los
métodos setUp() y
tearDown()**

Funciones setUp() para
preparar entorno y
tearDown() para limpieza



AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM (FRONTEND)

01. Entrar en la página e iniciar sesión

The screenshot displays the Unibotics website interface. The top navigation bar includes the Unibotics logo, a search bar, and a user profile dropdown for 'FELICIDADAMQ'. The main content area features a grid of exercise cards, each with a title, a goal description, and a corresponding image. The cards are:

- Follow line**: The goal of this exercise is to perform a PID reactive control capable of following the line.
- Obstacle avoidan...**: The objective of this practice is to implement the logic of the VFF navigation algorithm.
- Basic Vacuum cle...**: The objective of this exercise is to implement the logic of a navigation algorithm for an
- Localized Vacuu...**: The objective of this exercise is to implement the logic of a navigation algorithm for an

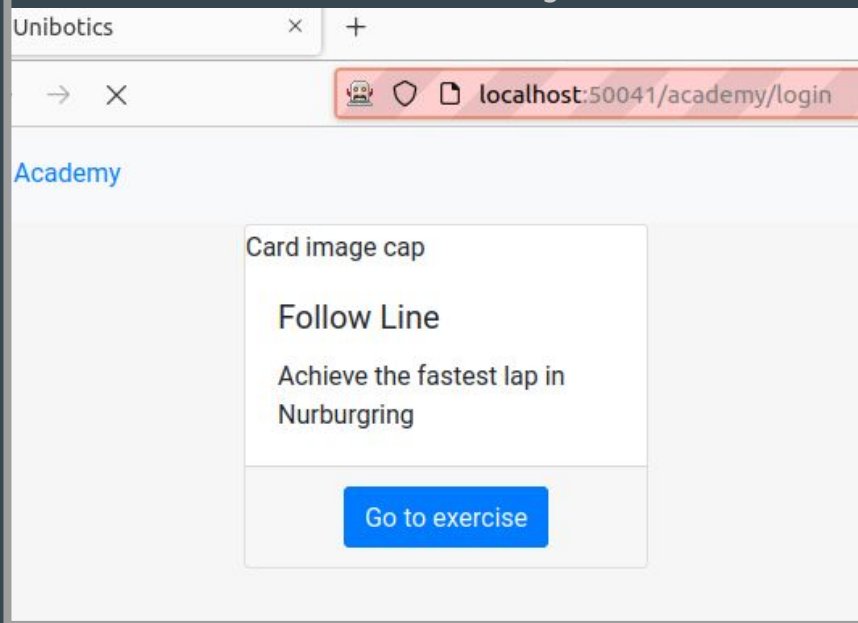
Below the grid, there are buttons for 'FORUM', 'SIGN IN', and 'SIGN UP'. A tooltip shows a value of '91.33 x 50'.

The DevTools console on the right shows the HTML structure of the page, highlighting the 'Sign In' button. The relevant HTML snippet is:

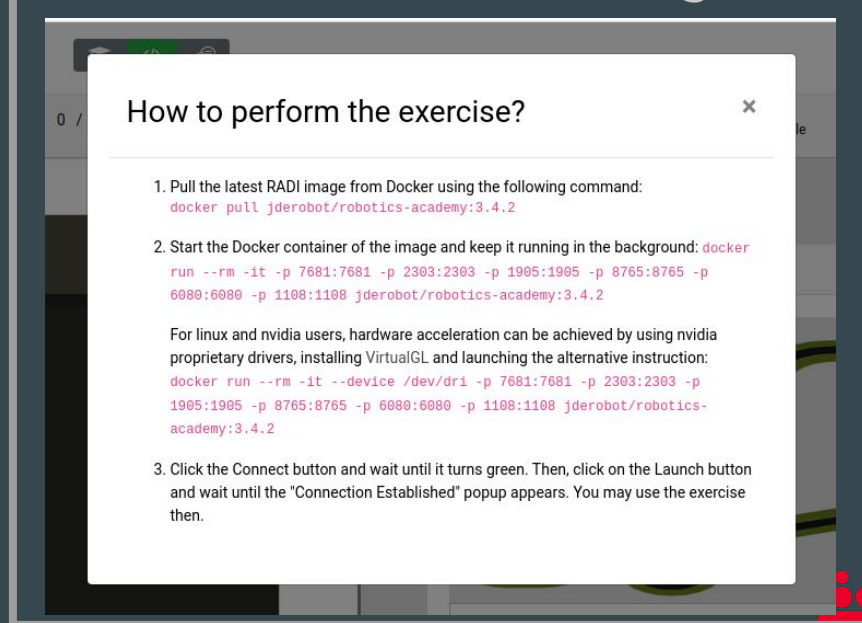
```
<li class="page-scroll">
  <a href="/academy/login"> == $0
    "Sign In"
  </a>
</li>
```

AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM (FRONTEND)

02. Seleccionar ejercicio



03. Cerrar ventana emergente



AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM (FRONTEND)

04. Se realizan las pruebas sobre todos los botones del frontend

05. Resultados

05.1 Errores:

```
-----  
Ran 10 tests in 565.623s  
  
FAILED (errors=7)  
Destroying test database for alias 'default'...
```

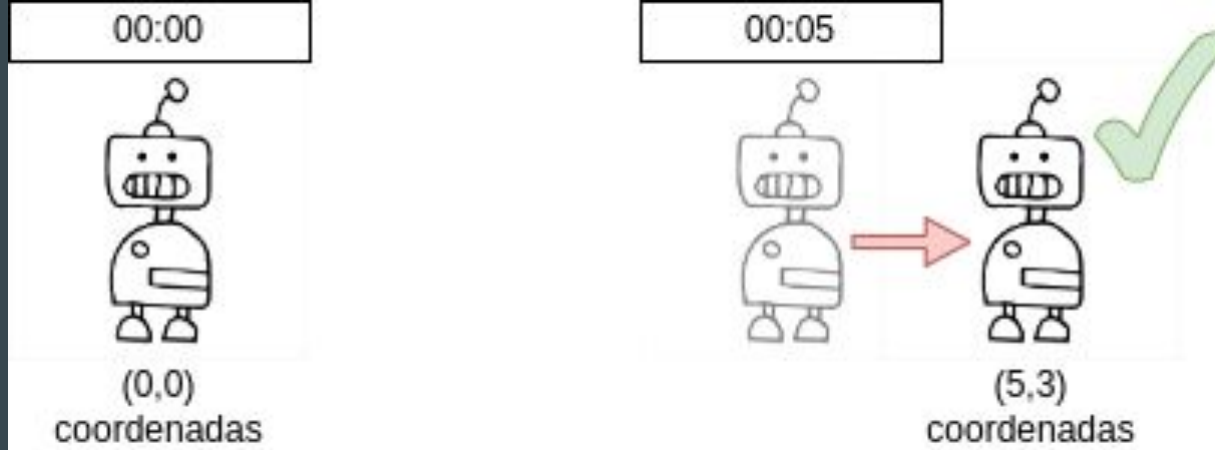
05.2 Resultado correcto:

```
-----  
Ran 10 tests in 642.393s  
  
OK  
Destroying test database for alias 'default'...
```



AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM (INTEGRALES)

Se implementan pruebas integrales para comprobar que el sistema al completo funciona

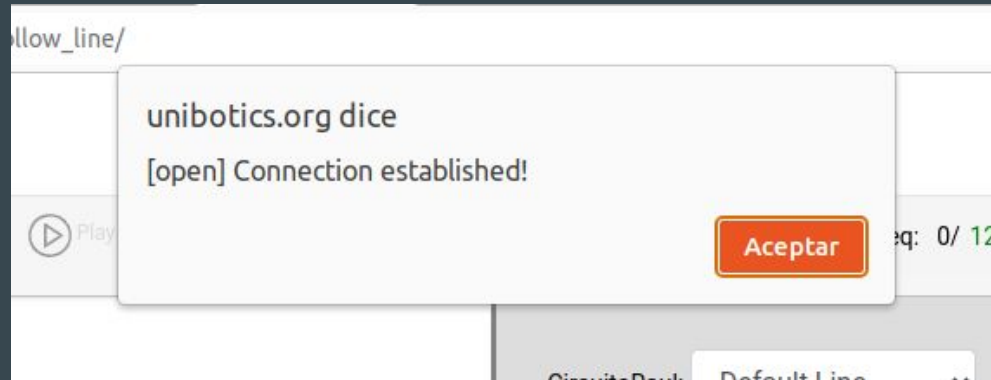


AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM (INTEGRALES)

01. Automatización levantamiento contenedor RADI

02. Seleccionar un ejercicio y levantarlo

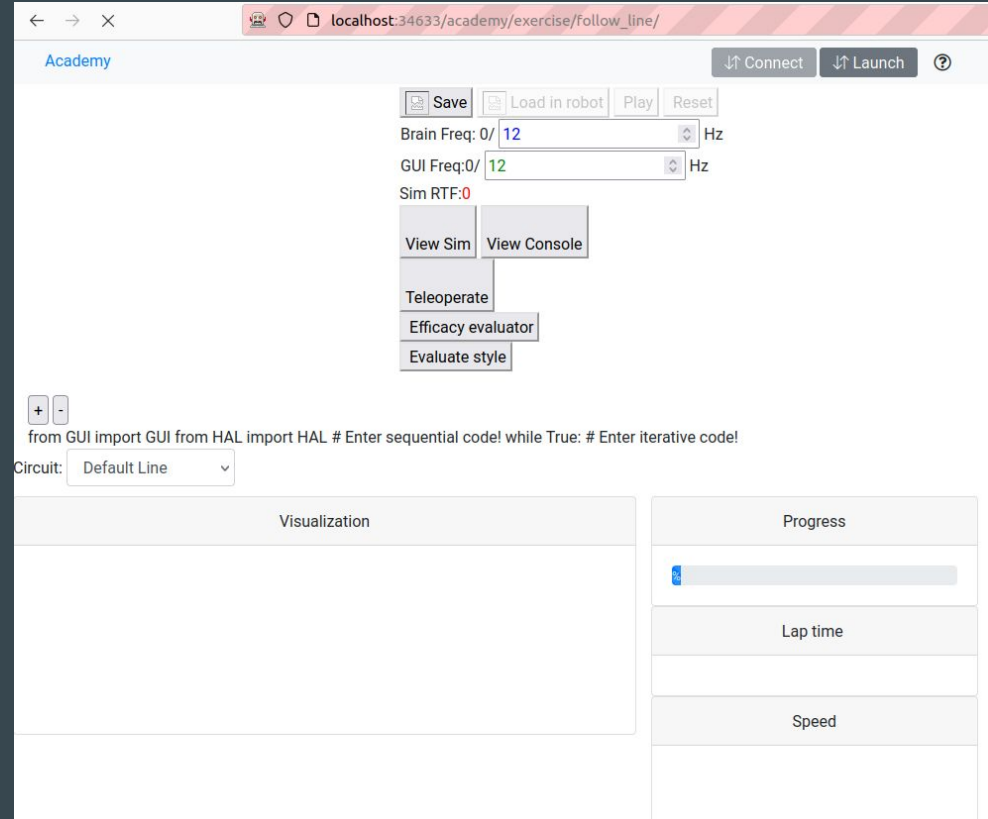
02.2 Para preparar un ejercicio se pulsa el botón *“Launch”* y el manager.py realiza distintos procesos.



AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM (INTEGRALES)

03. Automatización de envío de código de usuario.

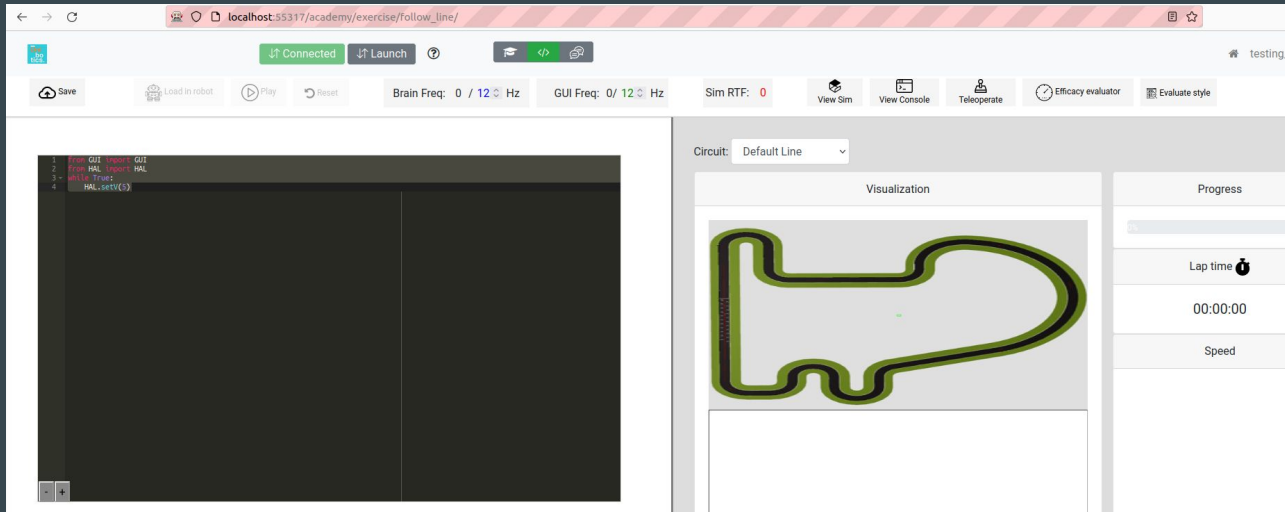
03.1 La clase *LiveServerTestCase* no envía archivos estáticos:



AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM (INTEGRALES)

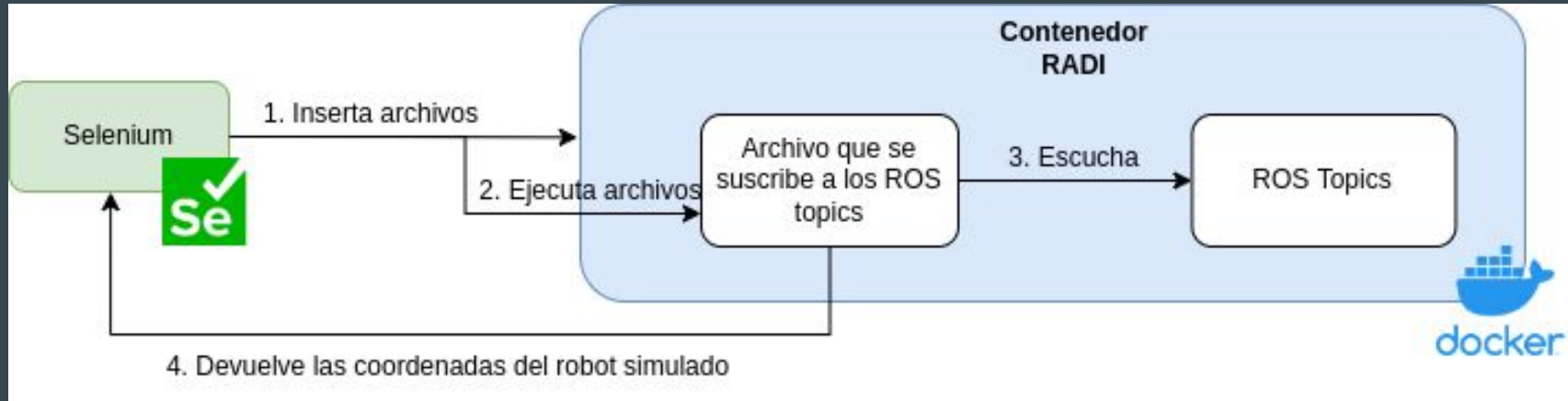
03. 03.2 Se sustituye por *StaticLiveServerTestCase*

03.3 `execute_script()` para Selenium y `setValue()` para el editor ACE.



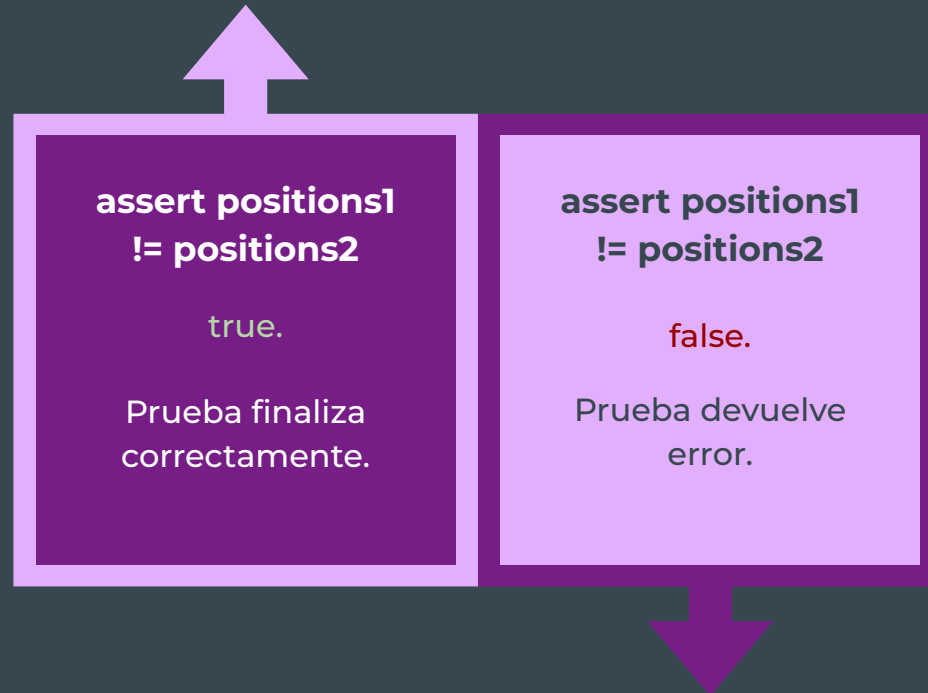
AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM (INTEGRALES)

04. Se inserta un archivo que se suscribe a un ROS Topic



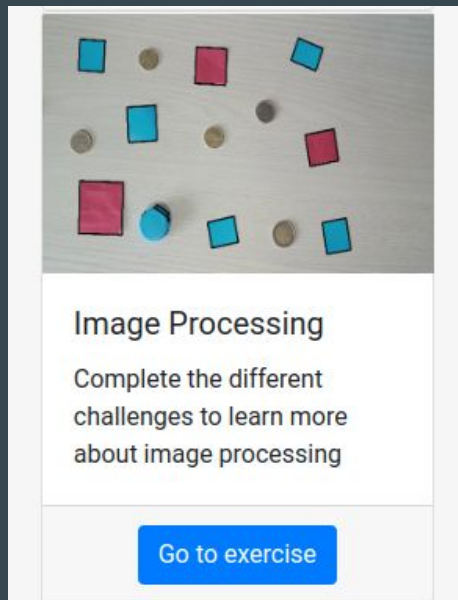
AUTOMATIZACIÓN DE PRUEBAS EN UNIBOTICS CON SELENIUM (INTEGRALES)

05. Se comprueban los resultados



EJERCICIO INTRODUCTORIO AL PROCESAMIENTO DE IMAGEN

Ejercicio que propone una serie de retos de procesamiento de imagen 2D con la librería OpenCV



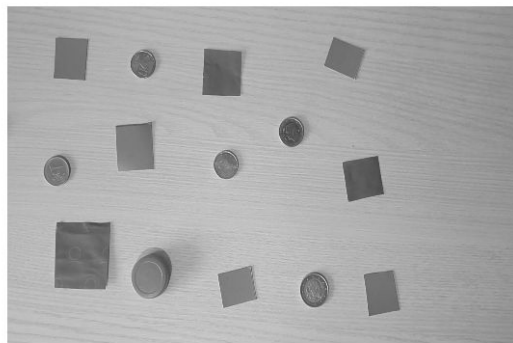
EJERCICIO INTRODUCTORIO AL PROCESAMIENTO DE IMAGEN (RETO 1)

01. Convertir los fotogramas a blanco y negro.

OpenCV por defecto usa el espacio de color BGR, el objetivo es transformar el valor de estos tres canales a uno sólo de nivel de gris

```
1 from GUI import GUI
2 from HAL import HAL
3 # Enter sequential code!
4 import cv2
5 import numpy as np
6
7 i = 0
8
9 while True:
10     # Enter iterative code!
11     image = HAL.getImage(i)
12     # Convertir la imagen a escala de grises
13     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
14
15     GUI.showImage(gray)
16
17     i = i + 1
```

Visualization



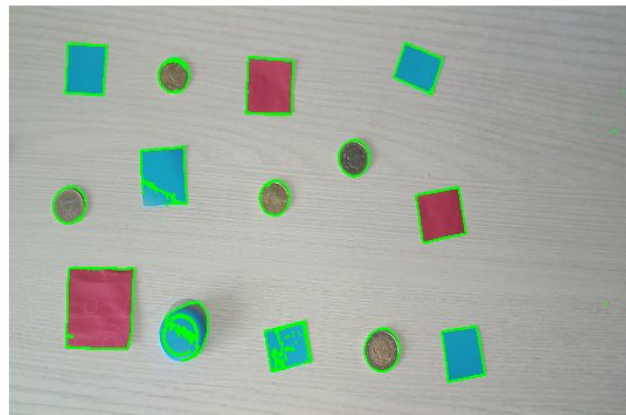
EJERCICIO INTRODUCTORIO AL PROCESAMIENTO DE IMAGEN (RETO 3)

03. Encontrar contornos

Se propone binarizar la imagen, y utilizar las funciones *findContours()* y *drawContours()*

```
1 from GUI import GUI
2 from HAL import HAL
3 # Enter sequential code!
4 import cv2
5
6 i = 0
7
8 ~ while True:
9     # Enter iterative code!
10    image = HAL.getImage(i)
11
12    # Escala de grises
13    gris = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
14
15    # Imagen binaria
16    thresh, umbral = cv2.threshold(gris, 127, 255, cv2.THRESH_BINARY)
17    imagenInvertida = cv2.bitwise_not(umbral)
18
19    # Encontrar contornos
20    contour, _ = cv2.findContours(imagenInvertida, cv2.RETR_EXTERNAL, cv2.CHAIN_
21
22    # Dibujar contorno
23    cv2.drawContours(image, contour, -1, (0, 255, 0), 5)
24
25    GUI.showImage(image)
26    i=i+1
```

Visualization



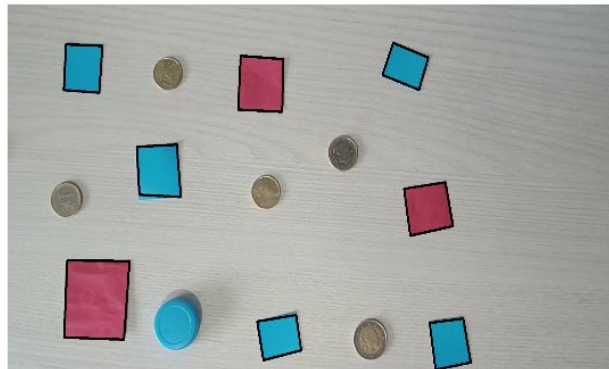
EJERCICIO INTRODUCTORIO AL PROCESAMIENTO DE IMAGEN (RETO 4)

04. Encontrar objetos rectangulares

El objetivo es usar operaciones morfológicas para segmentar la imagen y usar la función `cv2.approxPolyDP()`

```
23 # Crear máscaras binarias para cada rango de color
24 mask_red = cv2.inRange(img_hsv, lower_red, upper_red)
25 mask_blue = cv2.inRange(img_hsv, lower_blue, upper_blue)
26
27 # Unir las dos máscaras
28 mask = cv2.bitwise_or(mask_red, mask_blue)
29
30 # Aplicar la máscara a la imagen original para mostrar sólo los píxeles de color
31 result = cv2.bitwise_and(frame, frame, mask=mask)
32
33 # apply morphology
34 kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (9,9))
35 clean = cv2.morphologyEx(result, cv2.MORPH_OPEN, kernel)
36 kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (15,15))
37 clean = cv2.morphologyEx(result, cv2.MORPH_CLOSE, kernel)
38
39 # Convertir imagen HSV a BGR
40 bgr_img = cv2.cvtColor(clean, cv2.COLOR_HSV2BGR)
41
42 # Convertir imagen BGR a escala de grises
43 gray_img = cv2.cvtColor(bgr_img, cv2.COLOR_BGR2GRAY)
44
45 # Umbralizar la imagen
46 _, threshold = cv2.threshold(gray_img, 10, 255, cv2.THRESH_BINARY)
47
48 # Encontrar los contornos
49 contours, _ = cv2.findContours(threshold, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
50
51 rectangles = []
52 for contour in contours:
53     perimeter = cv2.arcLength(contour, True)
54     approx = cv2.approxPolyDP(contour, 0.04 * perimeter, True)
55     if len(approx) == 4:
```

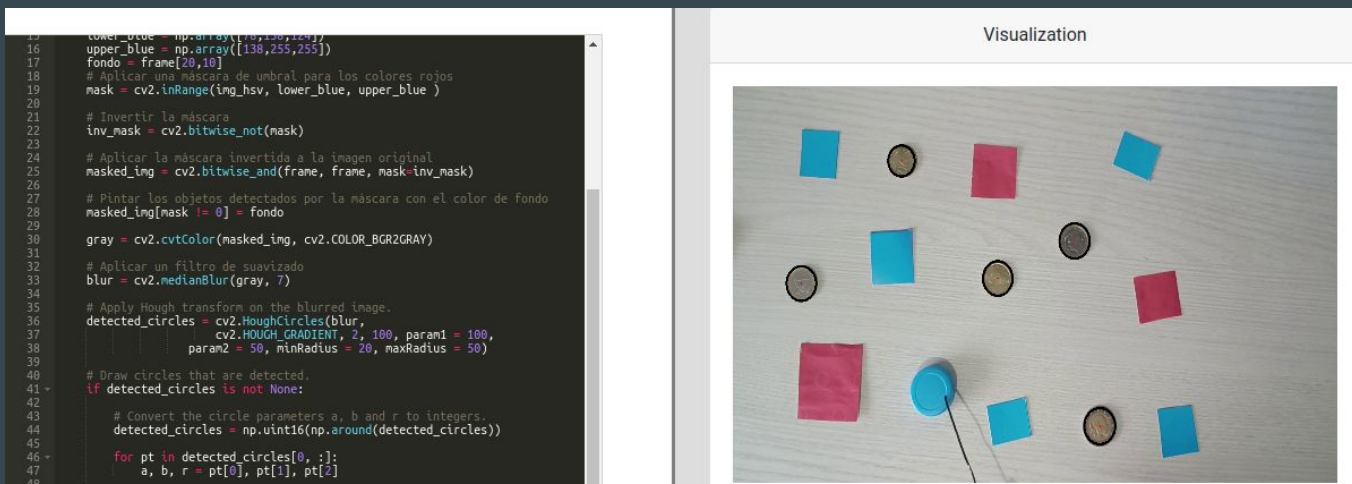
Visualization



EJERCICIO INTRODUCTORIO AL PROCESAMIENTO DE IMAGEN (RETO 5)

05. Encontrar monedas.

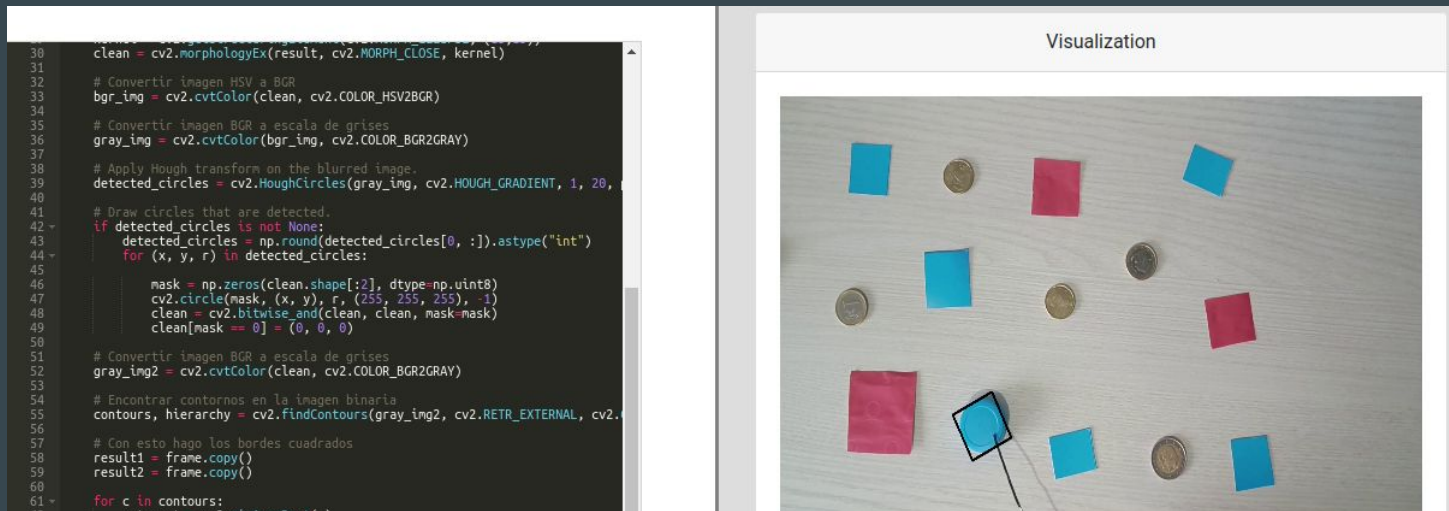
OpenCV ofrece la función `cv2.HoughCircles()` para encontrar cada círculo en el fotograma y `cv2.circle()` para dibujarlos




EJERCICIO INTRODUCTORIO AL PROCESAMIENTO DE IMAGEN (RETO 6)

06. Seguir círculo en movimiento.

Se propone solucionar este reto con operaciones morfológicas, `cv2.HoughCircles()` y `cv2.drawContours()`



CONCLUSIONES



**Recapitulación
de objetivos**

**Trabajos
futuros**