

MANUAL DE USO

SOLUCIONADOR DE LABERINTOS

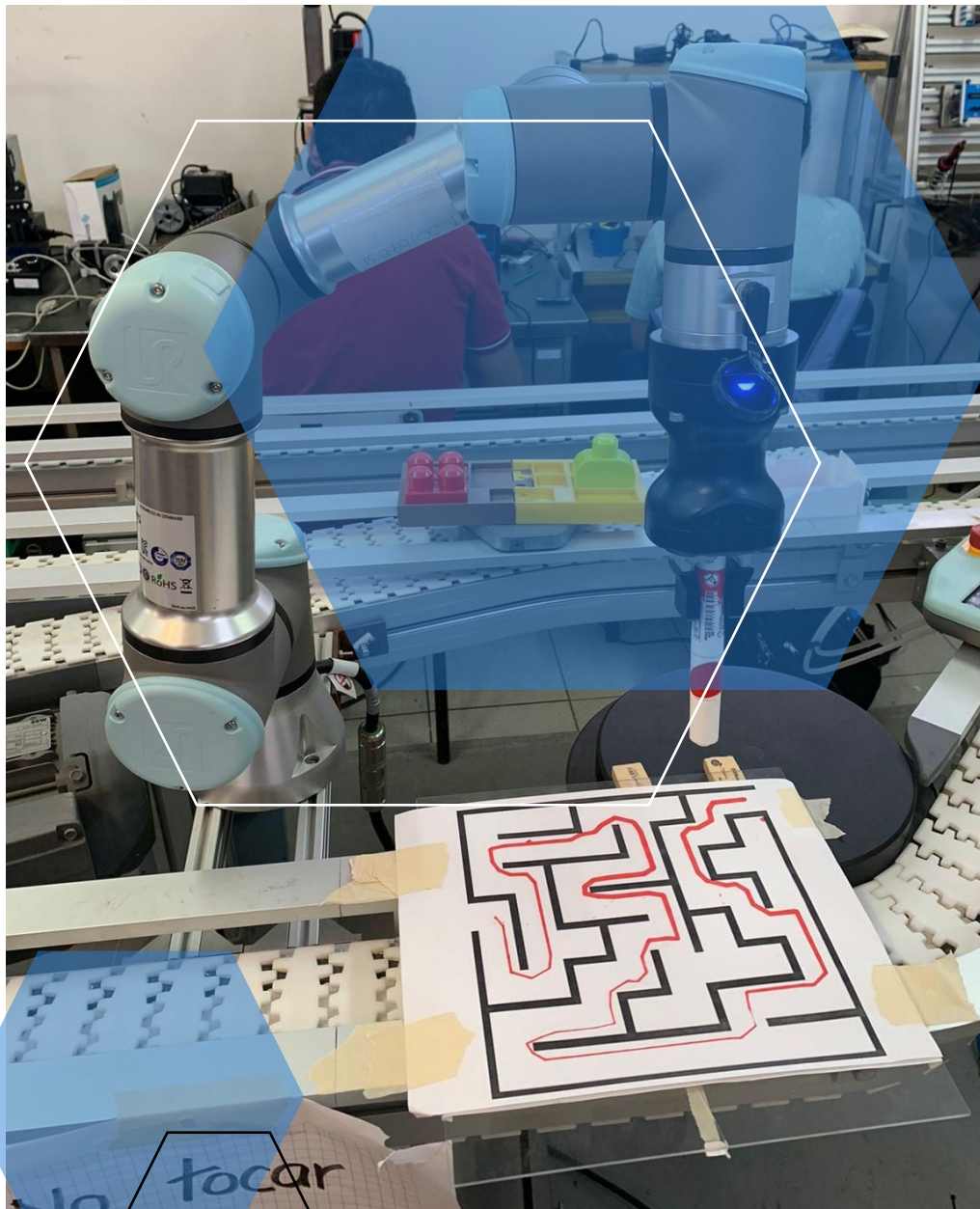
INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL

Catedrático

Dr. Luis Felipe
Marín Urias.



Universidad Veracruzana



RESOLUCIÓN DE LABERINTOS POR MEDIO DE AGENTES INTELIGENTES Y UN ROBOT MANIPULADOR

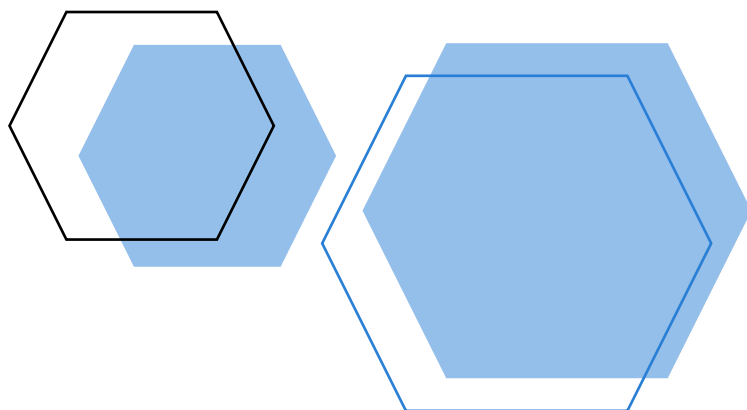
INTEGRANTES:

- * Cerecero Amador María Cristina.
- * Salgado Gómez Kevin.
- * Sosa Guzmán Mariana.
- * Ruíz Ríos Eduardo.

CONTENIDO

INTRODUCCIÓN	3
Objetivo	3
Requerimientos.....	3

¿Cómo empezar?	4
Laberinto.....	4
Programa	5
Robot manipulador	8
Trabajo futuro.....	9



INTRODUCCIÓN

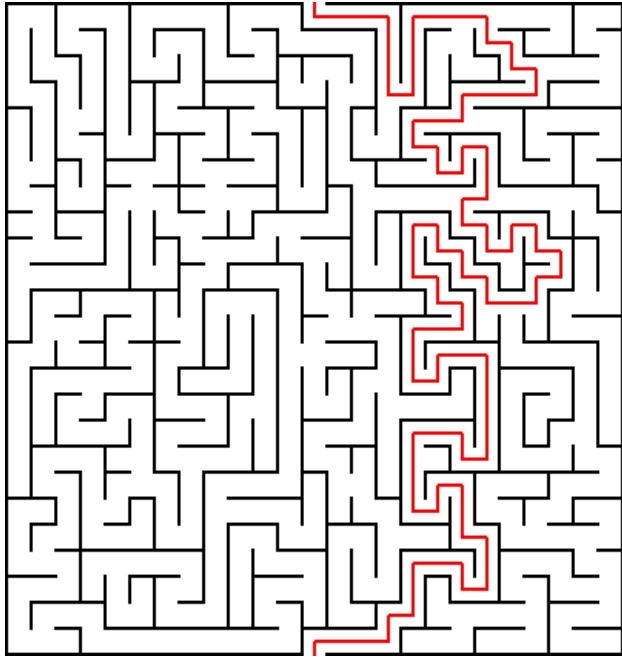
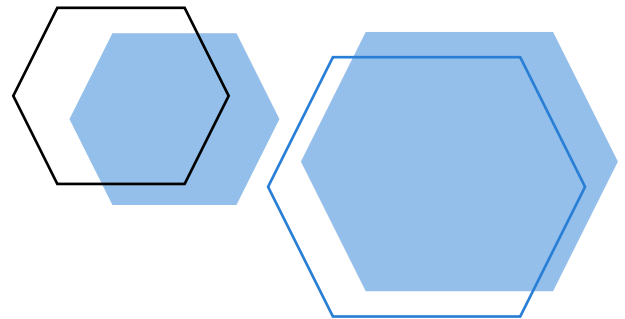


ILUSTRACIÓN 1. IMAGEN DE REFERENCIA.

OBJETIVO

El agente es capaz de encontrar la salida, es decir el camino que conduce a la salida en un laberinto. Dicho agente estará conformado de dos componentes: un brazo robótico que le permitirá interactuar con el mundo exterior (su entorno) de manera que pueda trazar el camino en un laberinto físico, y un software que permita representar el entorno para encontrar la solución.



REQUERIMIENTOS

PARA SU USO ES NECESARIO CONTAR CON:

- Una computadora con Python versión ≥ 3.8 instalada y con acceso a internet.
- Un robot manipulador en este caso se utilizó un UR3e (Con grados de libertad suficientes para adoptar las configuraciones necesarias para resolver el laberinto seleccionado, asegurándose que el laberinto este dentro del espacio alcanzable del robot).
- Un laberinto impreso.
- Un plumón (o cualquier utensilio para que el robot dibuje sobre la hoja).



ILUSTRACIÓN 2. IMAGEN DE REFERENCIA.

¿CÓMO EMPEZAR?

LABERINTO

SELECCIONA UN LABERINTO

Al hacer la elección del laberinto se toma en cuenta que no hay restricciones con su forma, su contorno o marco, se debe definir cuál será el punto de inicio y el punto de fin al que se desea llegar. Es necesario contar con el laberinto de forma digital y de forma impresa, se debe imprimir el laberinto con dimensiones de 20.7 x 20.7 cm en una hoja tamaño carta.

Algunas otras consideraciones importantes es que el laberinto propuesto no tenga marcas de agua, que este en escala de grises (blanco y negro) y que la imagen en el tamaño antes especificado no tenga incluido bordes blancos, se muestra un ejemplo a continuación.

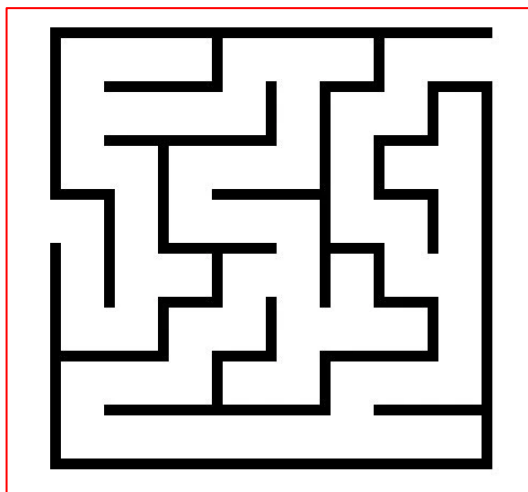


ILUSTRACIÓN 3. LABERINTO DE FORMA INCORRECTA

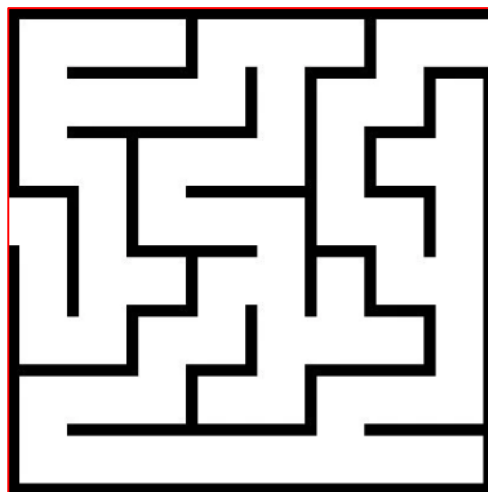


ILUSTRACIÓN 4. LABERINTO DE FORMA CORRECTA

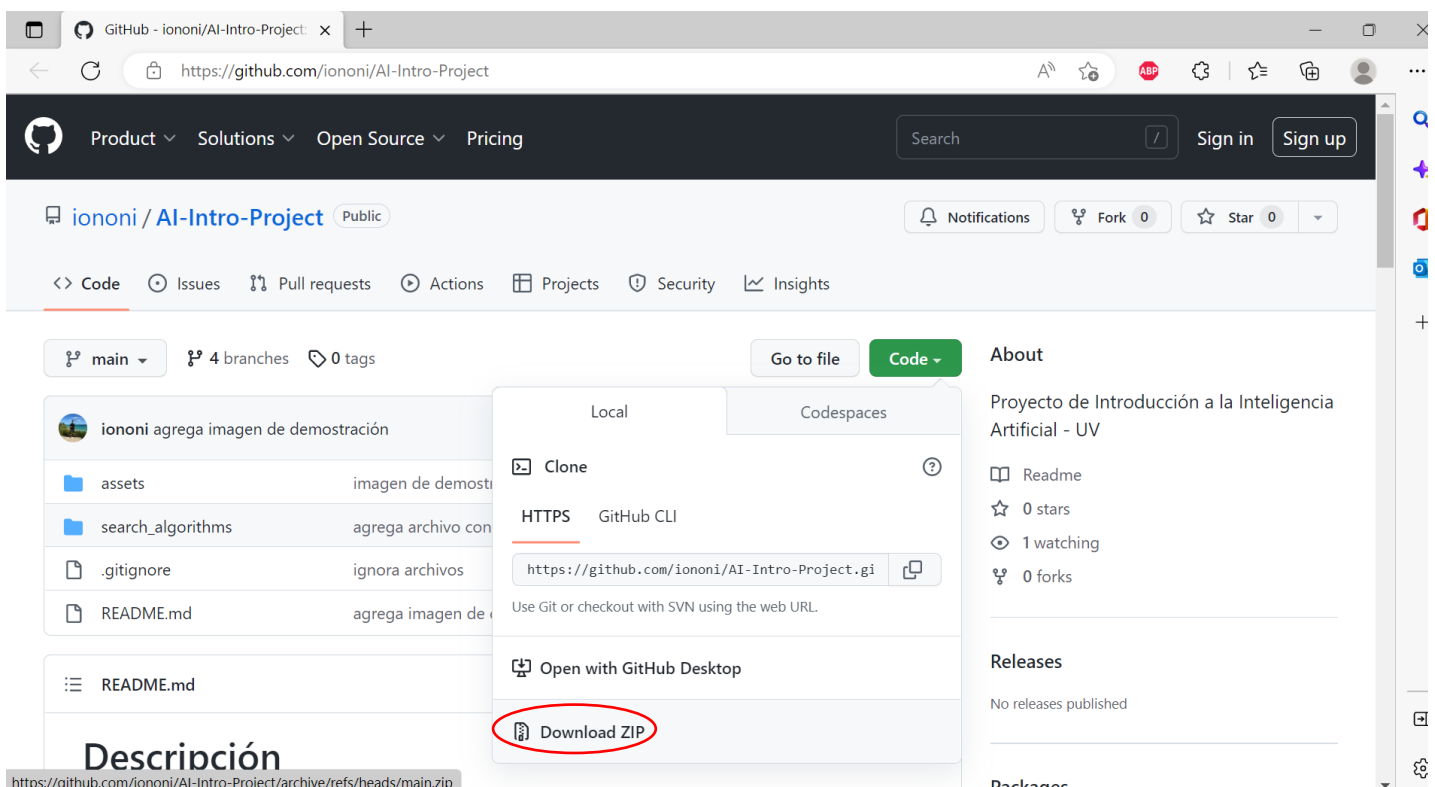
PROGRAMA

El primer paso es instalar las librerías necesarias en la ventana de comandos, usando el comando “pip install <nombre de la libreria>”. Las librerías necesarias son:

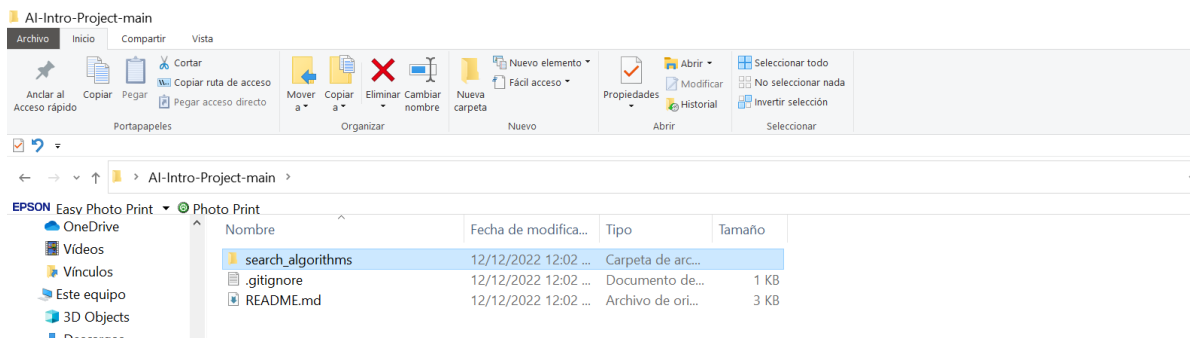
- numpy
- cv2 (open-cv)
- pillow (PIL)

También se hace uso de la librería collections, pero esta se instala automáticamente al momento de instalar Python.

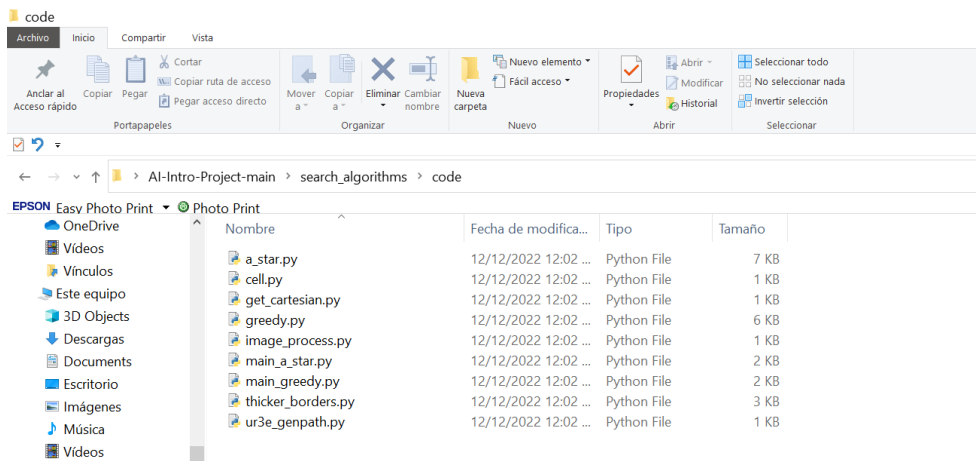
Después procedemos a descargar el archivo zip o clonar el repositorio que se encuentra a continuación: <https://github.com/iononi/AI-Intro-Project.git>



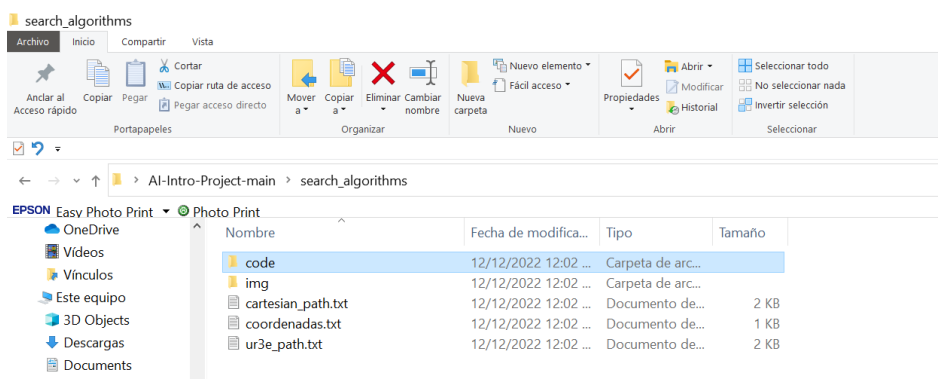
Al completarse la descarga, puede descomprimir la carpeta de archivos y abrir la carpeta “search_algorithms”.



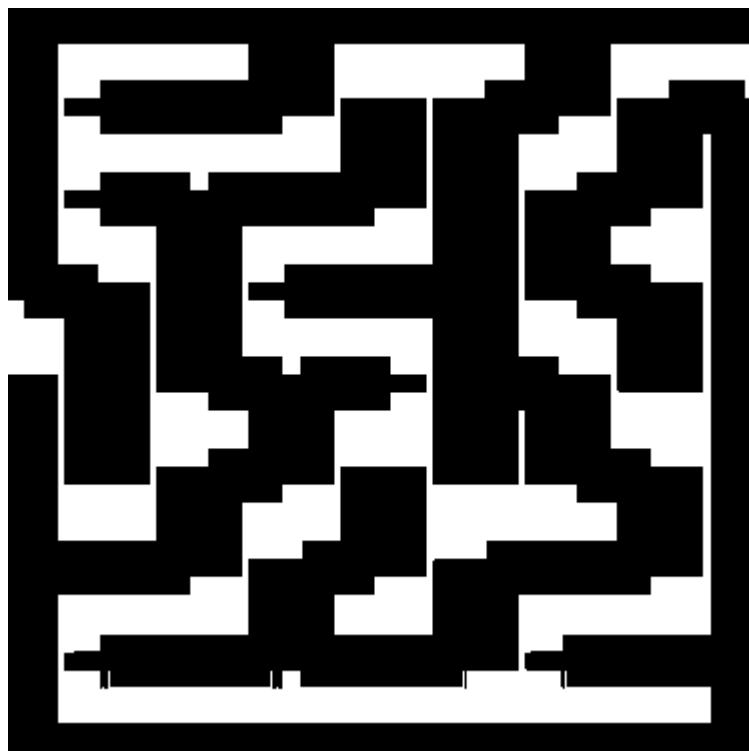
Posteriormente, en esa carpeta abrimos la carpeta code.



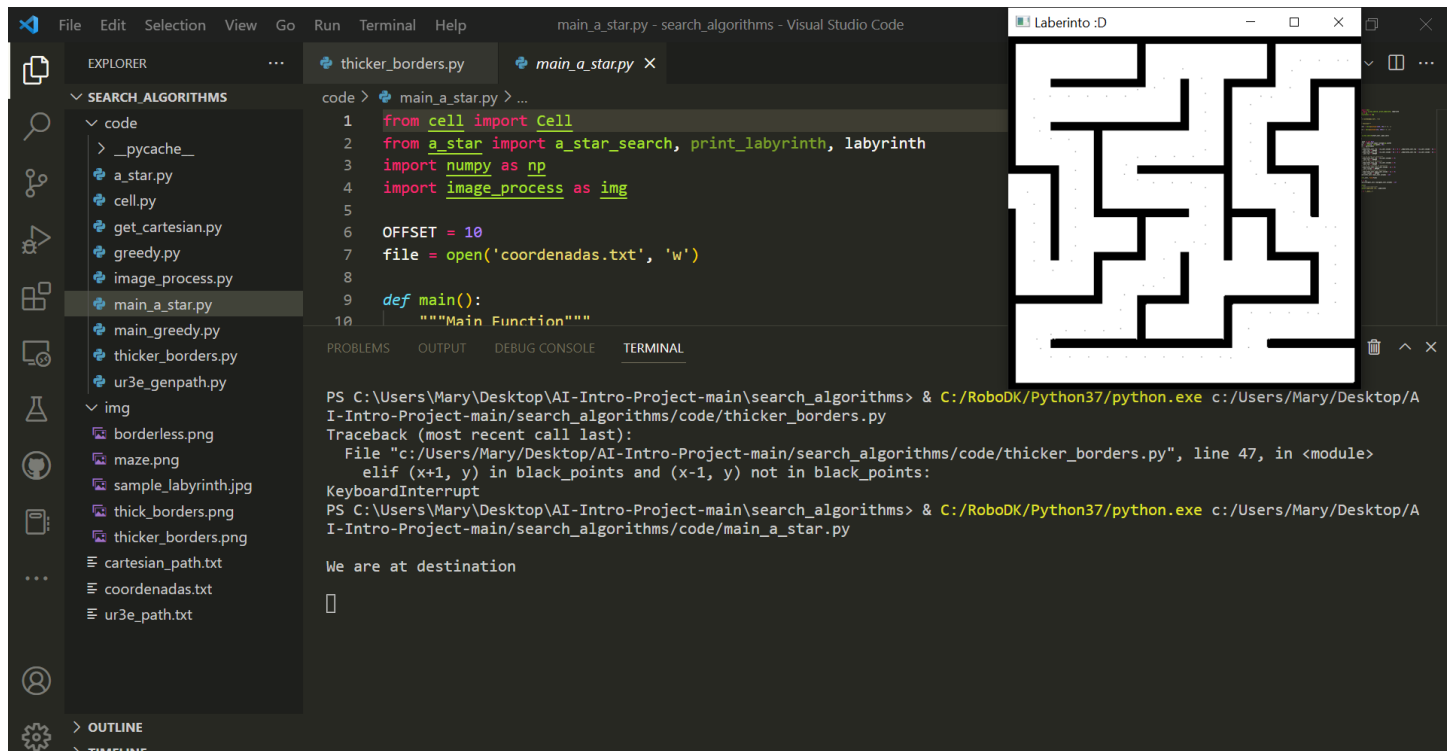
Ahora podemos visualizar todos los códigos que se utilizarán.



Primero ejecutamos “thicker_borders.py” este algoritmo es el encargado de engrosar los bordes del laberinto para evitar que el algoritmo busque una solución pegada a los bordes originales; la imagen retornada funciona como máscara.



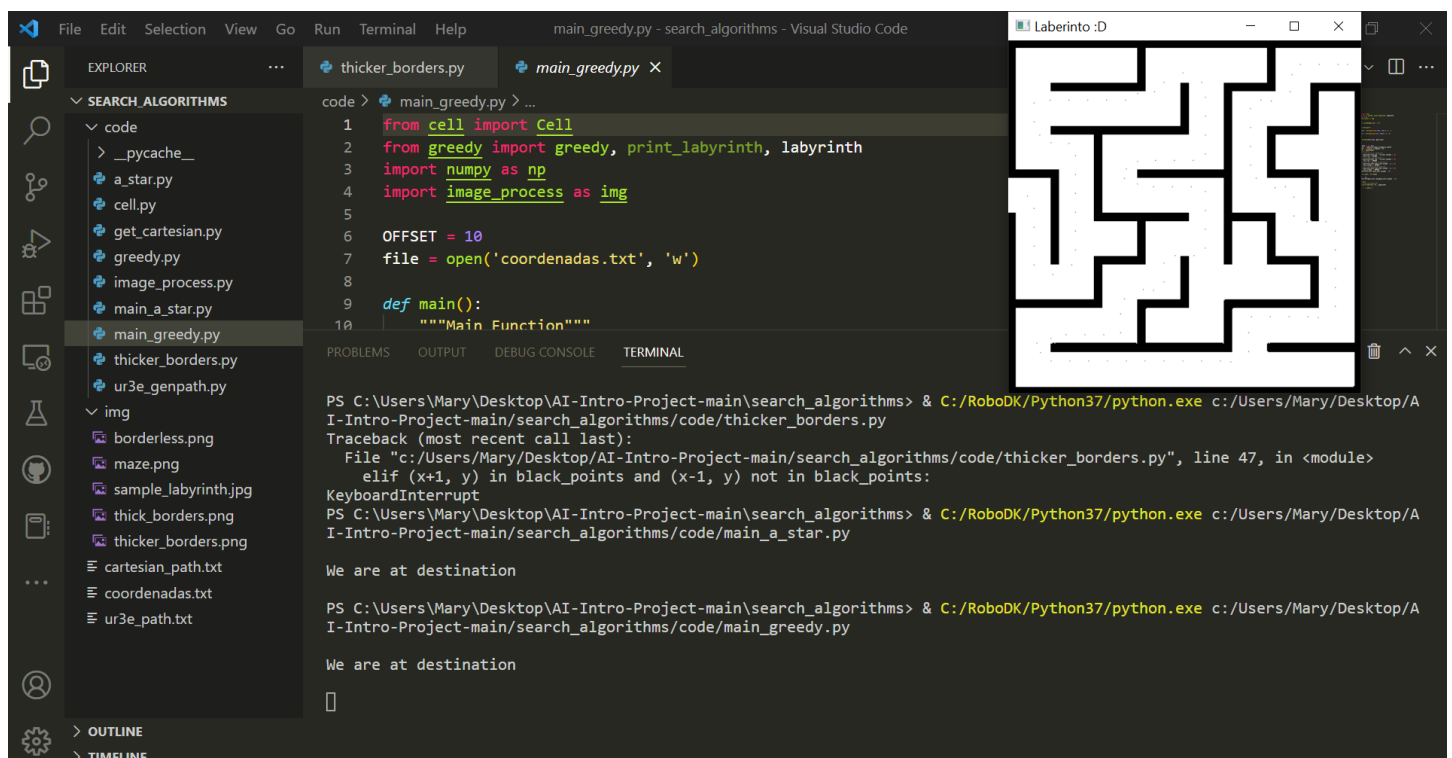
Para continuar ejecutamos el programa principal “main”, en la carpeta search_algorithms se encuentran los algoritmos “main_a_star” y “main_greedy”, tenemos dos métodos de búsqueda por **a estrella (a*)** y **greedy**, se ejecutará el código principal de acuerdo con el algoritmo de búsqueda que se desee utilizar.



```
1 from cell import Cell
2 from a_star import a_star_search, print_labyrinth, labyrinth
3 import numpy as np
4 import image_process as img
5
6 OFFSET = 10
7 file = open('coordenadas.txt', 'w')
8
9 def main():
10     """Main Function"""
```

```
PS C:\Users\Mary\Desktop\AI-Intro-Project-main\search_algorithms> & C:/RoboDK/Python37/python.exe c:/Users/Mary/Desktop/AI-Intro-Project-main/search_algorithms/code/thicker_borders.py
Traceback (most recent call last):
  File "c:/Users/Mary/Desktop/AI-Intro-Project-main/search_algorithms/code/thicker_borders.py", line 47, in <module>
    elif (x+1, y) in black_points and (x-1, y) not in black_points:
KeyboardInterrupt
PS C:\Users\Mary\Desktop\AI-Intro-Project-main\search_algorithms> & C:/RoboDK/Python37/python.exe c:/Users/Mary/Desktop/AI-Intro-Project-main/search_algorithms/code/main_a_star.py

We are at destination
```



```
1 from cell import Cell
2 from greedy import greedy, print_labyrinth, labyrinth
3 import numpy as np
4 import image_process as img
5
6 OFFSET = 10
7 file = open('coordenadas.txt', 'w')
8
9 def main():
10     """Main Function"""
```

```
PS C:\Users\Mary\Desktop\AI-Intro-Project-main\search_algorithms> & C:/RoboDK/Python37/python.exe c:/Users/Mary/Desktop/AI-Intro-Project-main/search_algorithms/code/thicker_borders.py
Traceback (most recent call last):
  File "c:/Users/Mary/Desktop/AI-Intro-Project-main/search_algorithms/code/thicker_borders.py", line 47, in <module>
    elif (x+1, y) in black_points and (x-1, y) not in black_points:
KeyboardInterrupt
PS C:\Users\Mary\Desktop\AI-Intro-Project-main\search_algorithms> & C:/RoboDK/Python37/python.exe c:/Users/Mary/Desktop/AI-Intro-Project-main/search_algorithms/code/main_a_star.py

We are at destination

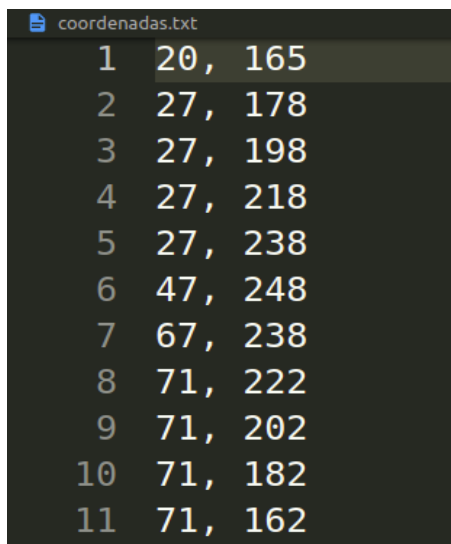
PS C:\Users\Mary\Desktop\AI-Intro-Project-main\search_algorithms> & C:/RoboDK/Python37/python.exe c:/Users/Mary/Desktop/AI-Intro-Project-main/search_algorithms/code/main_greedy.py

We are at destination
```

Al ejecutar el programa nos regresa la imagen del algoritmo resuelto punteada y nos genera un archivo tipo txt con los puntos de paso en coordenadas en pixeles de la solución del laberinto.

ROBOT MANIPULADOR

El programa “main” después de ser ejecutado nos devuelve un archivo de tipo txt, el cual contiene las coordenadas en pixeles de los puntos de paso que el robot deberá seguir, posteriormente dichos puntos se transforman a coordenadas cartesianas (x,y) en centímetro con respecto al origen del laberinto (el primer pixel que se encuentra en la esquina superior de lado izquierdo). para esto se debe ejecutar el programa “get_cartesian.py”.



1	20,	165
2	27,	178
3	27,	198
4	27,	218
5	27,	238
6	47,	248
7	67,	238
8	71,	222
9	71,	202
10	71,	182
11	71,	162

**ILUSTRACIÓN 5. ROBOT UR3E
EJECUTANDO LA SOLUCIÓN DEL
LABERINTO**

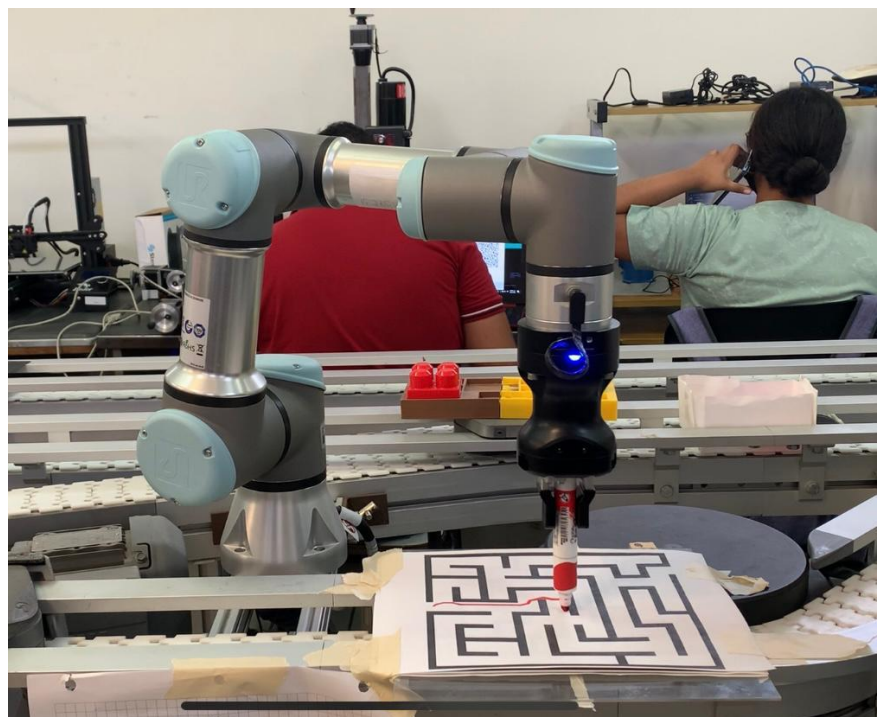
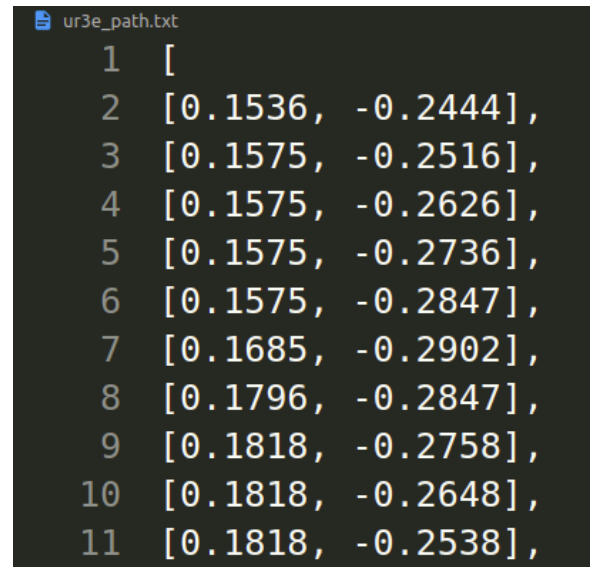


ILUSTRACIÓN 4. COORDENADAS OBTENIDAS DE "UR3E_GENPATH.PY"



1	[
2	[0.1536, -0.2444],
3	[0.1575, -0.2516],
4	[0.1575, -0.2626],
5	[0.1575, -0.2736],
6	[0.1575, -0.2847],
7	[0.1685, -0.2902],
8	[0.1796, -0.2847],
9	[0.1818, -0.2758],
10	[0.1818, -0.2648],
11	[0.1818, -0.2538],

**ILUSTRACIÓN 3. COORDENADAS EN
PIXELES OBTENIDAS DE "MAIN.PY"**

Ahora los puntos cartesianos obtenidos deben ser transformados con respecto al origen del robot, esto se logra ejecutando el programa “ur3e_genpath.py”, los puntos obtenidos son importados a un URScript. Se genera la trayectoria que hará el robot implementando un bucle que itera para cada punto, usando movimientos tipo “Move L”.

TRABAJO FUTURO

En caso de querer usar un laberinto distinto al pre-incluido, deberá anexar la imagen del laberinto en la carpeta “img” y realizar los cambios pertinentes en los códigos en los que se llama a la imagen del laberinto, puesto que este proyecto es capaz de resolver cualquier laberinto. Como trabajo a futuro, se implementaría una interfaz de usuario y mediante ella el usuario ingresaría la imagen del laberinto deseado, así como el punto de inicio, el punto final y el tamaño de la imagen en pixeles.