

---

# TRES EN RAYA CON RASPBERRY PI

---

DAVID ÁLVAREZ

GUILLERMO CREUS



*Escuela Técnica Superior de Ingeniería Industrial de Barcelona*  
*Universidad Politécnica de Cataluña*

BARCELONA, 12 DE MAYO DE 2019

# Índice

<b>1. Introducción</b>	<b>1</b>
1.1. Replanteamiento del proyecto . . . . .	1
1.2. Objetivos . . . . .	1
1.3. Alcance . . . . .	2
<b>2. Herramientas e implementación</b>	<b>2</b>
2.1. Tres en raya . . . . .	2
2.2. Interfaz gráfica . . . . .	3
2.3. Animación . . . . .	3
<b>3. Funcionamiento</b>	<b>5</b>
3.1. Tres en Raya . . . . .	5
3.2. Movimiento . . . . .	7
<b>4. Planificación y costes</b>	<b>7</b>
<b>5. Resultados y conclusiones</b>	<b>9</b>
<b>A. Árbol de directorios</b>	<b>11</b>
<b>B. Código más representativo</b>	<b>12</b>
B.1. Estrategia . . . . .	12
B.2. Movimiento . . . . .	21
B.3. Programa principal . . . . .	28
B.4. Animación . . . . .	34
B.5. Servidor . . . . .	42

## 1. Introducción

El objetivo del proyecto consiste en la construcción de un dispositivo mediante la programación de un microcontrolador (Raspberry Pi) y los periféricos oportunos. Se comenzó el proyecto con la idea de controlar un brazo robótico casero (creado en ETSEIB) mediante una Raspberry Pi.

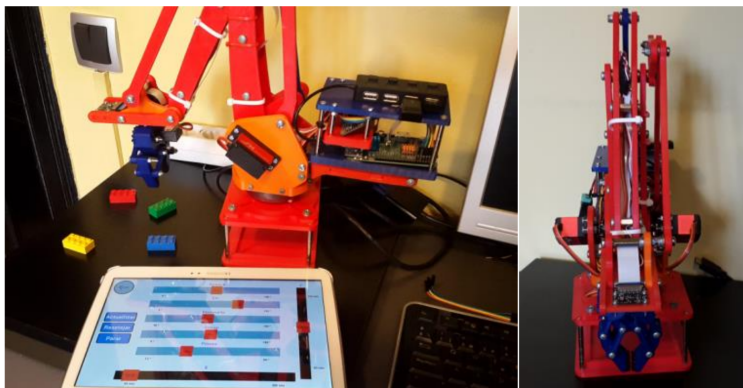


Figura 1: Brazo robótico casero.

La idea inicial se basaba en que la Raspberry fuese el cerebro de una partida de tres en raya con un humano. Es decir, que fuese la encargada de recoger el input de un jugador, decidir donde mover su pieza (de forma inteligente), mover el brazo para recoger las piezas correspondientes. Tras consultar la documentación del brazo robótico se dedicaron 6 semanas para el desarrollo del código de movimiento.

Una vez terminado y calibrado el robot se decidió a probarlo, advirtiéndolo un problema que podría ser crucial. El servo encargado de levantar piezas deslizaba, con lo que era imposible llevar al robot a una posición concreta. Con la idea de solucionar este problema se desmontó parte del robot y cambió la pieza problemática. No obstante, otro servo diferente empezó a fallar tras una semana de uso para comprobaciones. Sin piezas de recambio y con la imposibilidad de comprar servos nuevos debido a su coste elevado se decidió trasladar el proyecto hacia una simulación.

### 1.1. Replanteamiento del proyecto

Finalmente, el proyecto se definió como el desarrollo de un programa capaz de jugar de manera inteligente al juego del Tres en Raya y realizar una simulación de un robot moviendo las piezas.

El nuevo rol de la Raspberry Pi es ser el host del servidor donde correrá el código, simulación, etc. Se podría decir que en vez de controlar un periférico “real”, controla el estado de la simulación que no deja de ser una interpretación de la realidad.

### 1.2. Objetivos

El producto final pretende ser una interfaz web donde el jugador pueda realizar una partida contra la máquina (o máquina vs. jugador) y ver el progreso de la partida en la simulación.

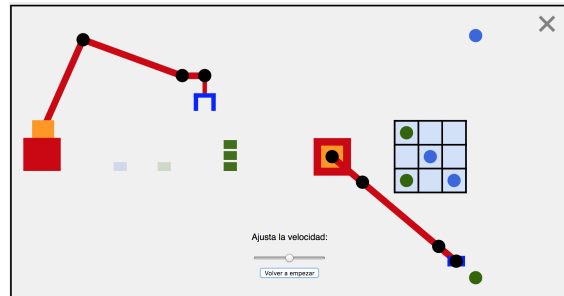
Sabiendo que el juego del tres en raya es un juego de suma cero (si se suman las pérdidas y las ganancias dan 0) se puede evitar la derrota siempre que no se realicen movimientos incorrectos. Por

lo tanto, no solo se plantea la creación de un programa “inteligente” que incluya una simulación, sino que se creará un programa capaz de evitar perder siempre.

En cuanto a la simulación, debido a la complejidad de gráficos 3D se decidió dividirla en dos planos; alzado y planta. De esta forma, no se compromete la imagen espacial sin adentrarse en el mundo de simulaciones 3D. Uno de los objetivos de la simulación es ser capaz de sincronizar los dos planos e integrar la rotación en el alzado, es decir, que los objetos en el alzado desaparezcan gradualmente a medida que rota la planta.



(a) Juego tres en raya interactivo.



(b) Simulación brazo robótico.

Figura 2: Interfaz Web.

### 1.3. Alcance

Una vez definido el objetivo del proyecto se procede al alcance del mismo:

- Puesta en marcha de un servidor capaz de generar una interfaz web que recoja el input (jugada humano), genere un output adecuado (jugada AI) y lo plasme en una animación.

## 2. Herramientas e implementación

A continuación recogemos las herramientas informáticas usadas para desarrollar este proyecto así como los diferentes lenguajes de programación que se han usado.

### 2.1. Tres en raya

Para desarrollar el tres en raya se han explorado manualmente todas las posibilidades (exceptuando simetrías) y expresado en forma de árbol. A partir de ahí, se ha trabajado en Python respetando la estructura definida anteriormente. Es decir, Python sabe lo que hacer gracias a un árbol implementado y lo único que debe hacer es bajar por las ramas correctas. Este árbol está internamente en Python siguiendo como si fuera un grafo. Esto es, consiste en una lista de nodos numerados y otras lista de conexiones entre nodos.

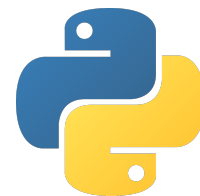


Figura 3: Lenguaje utilizado en el tres en raya.

## 2.2. Interfaz gráfica

Quisimos también ofrecer una interfaz gráfica en la que el usuario pudiese interactuar con el proyecto. Donde se pudiera jugar al tres en raya de manera sencilla y teniendo una respuesta mucho más visual del programa. Entre otras opciones que se consideraron, se optó finalmente por una interfaz web debido a que es un formato muy versátil y teníamos cierta experiencia anterior.



Para poder hacer funcionar la web es necesario un servidor. En nuestro caso nos decidimos por el servidor web Apache. Se hizo esta elección teniendo en cuenta que es software libre y que es el servidor más usado en el mundo. Se configuró correctamente para poder funcionar con PHP  y que éste pudiera ejecutar código en Python .



Figura 4: Servidor Web Apache.

Se está usando el lenguaje PHP como un enlace entre la web y el código de Python. Es decir, PHP es el que se encarga de ejecutar el código de Python en el servidor web con los parámetros adecuados dependiendo de la petición del usuario. Cuando el usuario interactúa con la web realizando algún movimiento en el tablero, es el código en PHP el que se encarga de transmitir este nuevo movimiento (junto con los anteriores) al código de Python para que este de un movimiento respuesta y mueva los servomotores a los ángulos adecuados. Por último es también PHP el que se encarga de construir la web (con la nueva información devuelta por Python) y así mostrársela al usuario.

Este funcionamiento global se encuentra recogido de manera esquemática en la figura 5 en la página siguiente.

## 2.3. Animación

Para la simulación se han usado principalmente tres herramientas:



Figura 6: Herramientas utilizadas en la simulación.

A grandes tiros, *Hypertext Markup Language* (HTML) permite crear webs. En este caso, HTML proporciona el espacio donde se realizará la simulación. *Cascading Style Sheets* (CSS) es el lenguaje utilizado para modificar la presentación de un documento HTML. CSS es la herramienta que ha permitido crear las figuras y determinar la disposición en el HTML. Por último, *JavaScript* es el lenguaje de programación que ha permitido modificar el CSS. En otras palabras, al modificar la presentación del documento HTML se crea la simulación.

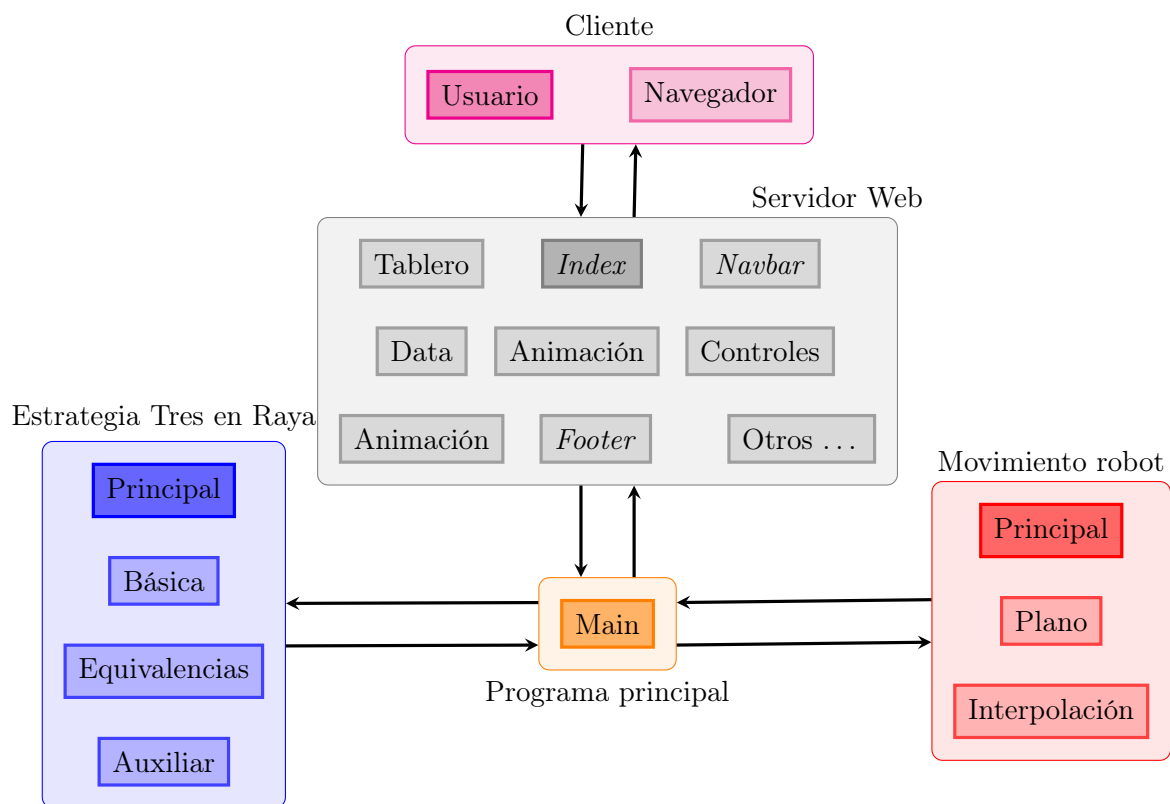


Figura 5: Arquitectura del programa.

### 3. Funcionamiento

A continuación se explicará el funcionamiento y la lógica detrás de este proyecto, que se encarga de decidir el siguiente movimiento (estrategia del tres en raya) y de animar el robot virtual desarrollado.

#### 3.1. Tres en Raya

Como se ha comentado anteriormente no solo se pretendía crear un jugador al 3 en raya, sino que fuese invencible. Esto es posible debido a la condición de ser un juego de suma cero, ya que nos permite desarrollar las ramas posibles antes y evitar aquellas que beneficien al oponente (y obviamente guiar al jugador a la derrota).

Realizar lo mencionado no es tan sencillo como parece. Aunque sea un juego con pocas posibilidades respecto a otros como el ajedrez, GO, etc. cuenta con  $9! = 362,880$  posibilidades (9 casillas al principio, 8 en la siguiente tirada ...). Para cualquier dispositivo sería una carga extra que se debería optimizar. La propuesta para este trabajo es crear un programa en Python que conozca todas los casos base (exceptuando simetrías) y extrapole a los otros respuestas mediante simetrías. Por ejemplo, si se desea saber cual es la jugada óptima si la primera jugada toca en un extremo se explorará la rama y se escogerá una rama en la que el jugador no pueda ganar. En este caso, como se ve en la figura 7, la jugada óptima tocaría en el medio.

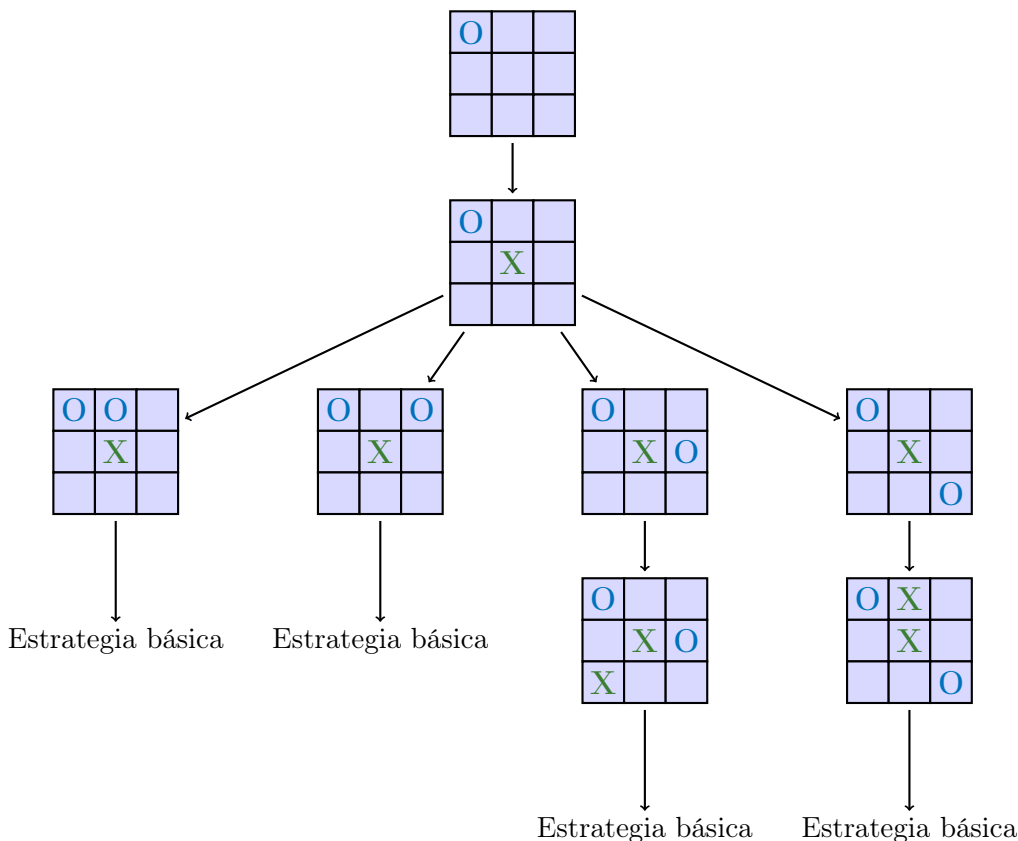


Figura 7: Una rama como ejemplo.

A partir de esta jugada se diferencian cuatro posibilidades del oponente. Para cada posibilidad se

explorará el árbol y se decidirá que jugada interesa. Como el tablero se va llenando, llegará un punto en el que la partida sea tablas o se ha forzado la victoria mediante un jaque. En este momento, se despliega la estrategia básica que consiste en lo siguiente:

1. Comprueba si podemos ganar:
  - a) PUEDO: juega y gana
  - b) NO PUEDO: siguiente
2. Comprueba si podemos perder:
  - a) PUEDO: tapo la posibilidad y sigo jugando
  - b) NO PUEDO: jugada aleatoria

Una vez definido el algoritmo para jugar uno se pregunta, ¿Se debe repetir el proceso de encontrar la rama si el jugador en vez de tirar al extremo superior izquierdo lo hace en el derecho? La respuesta de un humano sería claramente no, ya que sabe que existe una simetría respecto el eje  $y$ , con lo que extrapolaría los resultados obtenidos anteriormente.

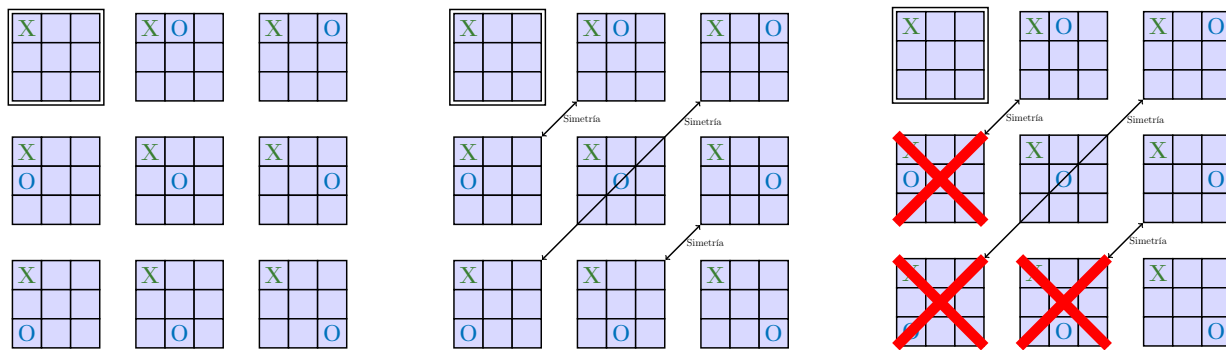


Figura 8: Tableros simétricos equivalentes.

La esencia del algoritmo se basa en las ramas desarrolladas y en las simetrías. El ordenador trabaja con un tablero interno y en cada jugada identifica la simetría que induce ese movimiento. Nótese que solo existen 5 simetrías:

1. Identidad.
2. Eje  $x$ .
3. Eje  $y$ .
4. Diagonal 1.
5. Diagonal 2.

Esta simetría se guarda en el vector de simetrías y cada vez que quiere traducir algún resultado lo aplica el vector de simetrías (Composición de simetrías  $\equiv$  simetría).



### 3.2. Movimiento

Para el movimiento se han definido dos planos de actuación: planta y alzado. Esta simplificación respecto a la simulación 3D implica que la planta y el alzado deben ir sincronizados.

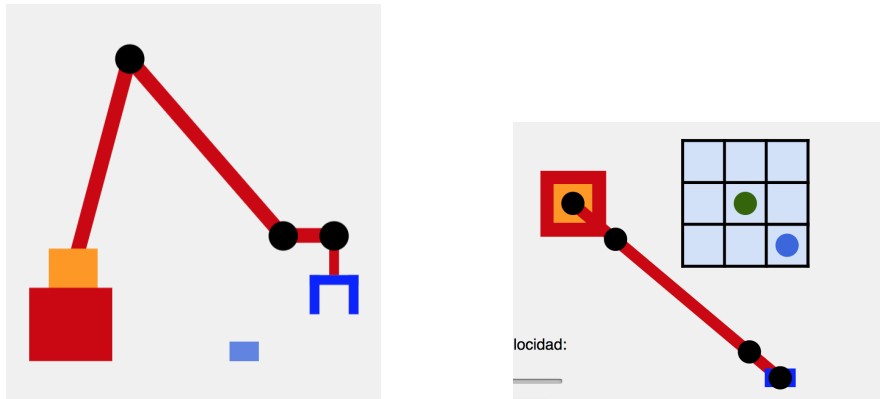


Figura 9: Brazo robótico a escala

Se ha utilizado un modelo a escala del brazo de la figura 1 (en la página 1) mediante CSS. Se ha definido cada elemento por separado y se han ido uniendo mediante “containers” (para poder mover varios elementos a la vez).

El hecho que sea a escala implica que la simulación sea más realista y los ángulos de movimientos se pueden reciclar del programa creado para el movimiento del brazo real. El funcionamiento de la simulación del alzado es relativamente sencilla (una vez creados los subprogramas):

1. Se definen los puntos del espacio donde se encuentran los almacenes de piezas y el tablero.
2. Mediante resolución analítica de las ecuaciones de enlace calcula los ángulos necesarios para orientar las barras y poder coger la pieza.
3. Llama a la función de JS encargada de mover las barras a los ángulos calculados.
4. Calcula los ángulos para dejar la pieza en el tablero.
5. Mismo que en (2).

Para añadir la simulación de la planta lo único que hay que añadir es una función que acorte la longitud de las barras cuando roten las barras del alzado. Esto se consigue calculando proyecciones respecto al eje x.

## 4. Planificación y costes

La planificación del proyecto se ha dividido de la siguiente manera:

- 12 Febrero - 26 Febrero:
  - Definición del proyecto.






- Investigación librerías servos.
- 26 Febrero - 23 Abril
  - Robot.
    - Código movimiento.
    - Pruebas movimiento.
    - Reparación.
  - 3 en raya.
    - Desarrollo árbol de posibilidades.
    - Implementación en Python.
  - Diseño Web.
    - Creación HTML/CSS Tablero.

A partir de este momento se decide realizar la simulación:

- 23 Abril -21 Mayo
  - Simulación
  - Servidor

El desarrollo de este proyecto ha requerido diferentes lenguajes de programación, en la tabla 1 se resumen el número de líneas de código por lenguaje de programación y el número de archivos.

Tabla 1: Líneas de código y número de archivos.

Lenguaje	# Archivos	# Líneas total
Python 	9	1025
JavaScript 	2	366
PHP  + HTML 	18	1315
CSS 	3	696
TOTAL	32	3402

Este número de líneas ha sido calculado de manera sencilla usando tuberías en la terminal. Por ejemplo, para contar el número de líneas de código que hay entre HTML y PHP en el servidor web se puede hacer:

```
$ cat Servidor\ Web/**/*.{html,php} | wc -l
```

Donde se ha usado el programa `cat` que imprime por terminal el contenido de un archivo y se le pasa este output mediante una tubería al programa `wc` que se encarga de contar el número de líneas. De manera similar, para contar el número de archivos escritos, por ejemplo, en Python, se puede hacer:

```
$ ls -l Servidor\ Web/**/*.*py | wc -l
```

- Nota: Estos comandos podrían fallar si no está activa la funcionalidad de **\*\*** en la terminal, esto se puede hacer (en Bash al menos) con el comando `shopt -s globstar`.

A continuación se presentan en la tabla 2 los costes del proyecto suponiendo que una empresa interesada en el ámbito encargase un proyecto similar al realizado:

Tabla 2: Costes del proyecto.

Tarea	Duración (h)	Precio/hora (€/h)	Precio (€)
Desarrollo de código en clase	$2 \cdot (13 \cdot 2)$	20	1040
3 en raya (árbol)	4	20	80
Código 3 en raya (implementación)	$2 \cdot 6$	20	240
Investigación librerías Servos	2	20	40
Código movimiento (robot real)	$2 \cdot 14$	20	560
Reparación robot	2	20	40
Simulación	$2 \cdot 14$	30	840
Creación Diseño Web	5	30	150
Puesta en marcha servidor	10	30	300
TOTAL	144		3290

## 5. Resultados y conclusiones

El proyecto, como se ha ido exponiendo a lo largo del trabajo ha migrado de la realidad a la web. Por esa razón, se anima al lector a visitar la web donde encontrará todo lo relacionado con este proyecto. En la pestaña "Documentación" se encuentra una copia de este informe en pdf y el código fuente (ya que ha sido desarrollado en  $\text{\LaTeX}$ ).

Para empezar a jugar se debe pinchar en "Empieza tú" o "Empieza el brazo" dependiendo de la modalidad de juego. Una vez dentro en la parte superior aparecen dos botones: "Controles" y "Simulación". Para ver el tablero se pinchará en la primera opción. En esta pantalla se puede configurar la dificultad del juego: "Fácil" (movimientos aleatorios), "Medio" (movimientos evitando la derrota e intentando ganar pero puede perder) e "Imposible" (nunca pierde).

Una vez introducida la jugada deseada, se podrá ver cómo mueve el robot pinchando en el botón "Simulación". Se puede ver un ejemplo de simulación en la figura 2 (en la página 2). Como cabe esperar, la simulación muestra el movimiento del jugador seguido del de la máquina, es decir, un turno completo.

En cuanto a conclusiones se debe recalcar dos lecciones importantes que se han aprendido a lo largo del proyecto. La primera es la dificultad de realizar proyectos con elementos reales. Esto se aplica al proyecto ya que por culpa de una pieza que a gran escala parece insignificante no se ha podido realizar el proyecto que se tenía en mente al principio de curso. Por falta de tiempo y presupuesto las consecuencias han sido devastadoras para los objetivos iniciales. En un ambiente laboral se podría

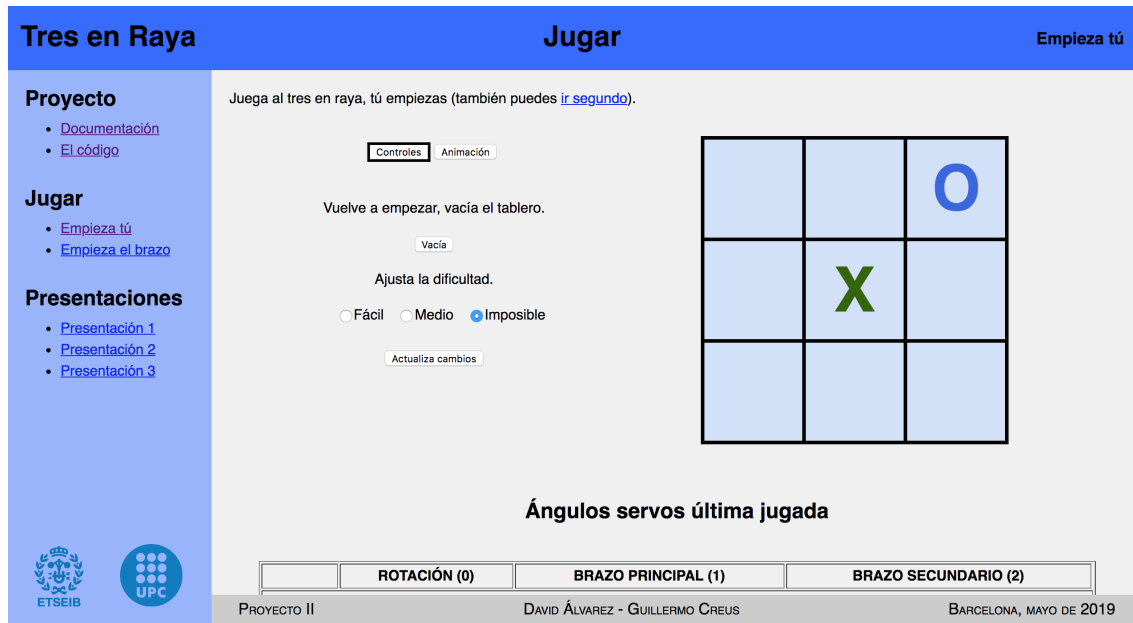


Figura 10: Web - Resultado final

encontrar una solución que no comprometiese tanto los objetivos iniciales y tampoco generase unas pérdidas importantes (como sería comprar nuevos servos). Esta alternativa se podría haber gestado en forma de creación de una pieza metálica que sustituya la defectuosa. Cabe destacar que esta opción fue estudiada personalmente pero fue desechada por falta de contactos en la industria.

No obstante, queda claro que los proyectos con elementos reales (y no reales también pero la resolución depende más de los individuos) generan unos problemas importantes que deben ser solucionados para llevarlo a cabo. La capacidad de resolución de problemas de la vida real es el pan de cada día del ingeniero y es su deber saber sortear, en la medida de lo posible, los problemas que amenacen las ideas/objetivos iniciales.

La segunda lección es la importancia de las simulaciones en la vida ingenieril. Como se ha comentado anteriormente los problemas de la vida real generan complicaciones grandes, desgaste del material, gasto energético, etc. Proporcionar una simulación a escala, como se ha desarrollado en este proyecto, es muy interesante de cara a realizar un proyecto ya que se reducen mucho las horas de experimentación. Se puede comprobar todo en el programa de ordenador y una vez funcione experimentar.

Por último, se debe mencionar que este proyecto se ha realizado con filosofía código libre por lo que en la pestaña “El código” se encuentran todos los scripts necesarios para correr la web, realizar simulaciones y emular lo conseguido en este proyecto.

## A. Árbol de directorios

A continuación se muestra en la figura 11 el árbol de directorios del Servidor Web, con todos los diferentes archivos que se han creado.

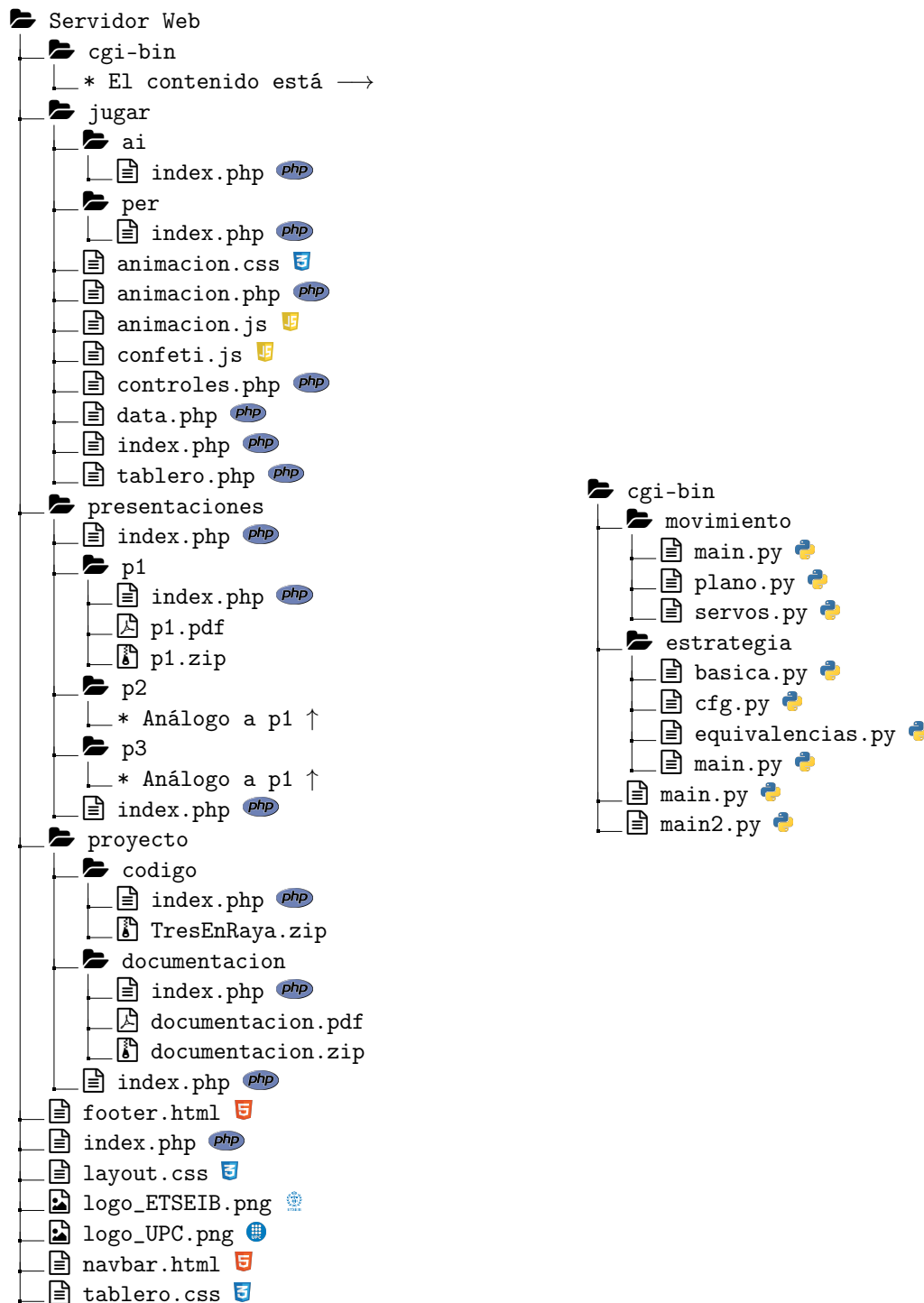


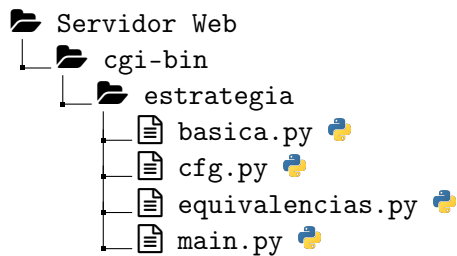
Figura 11: Árbol de directorios.

## B. Código más representativo

A continuación se recoge el código desarrollado considerado más representativo en este proyecto. Se puede consultar el código completo en la [web](#) del proyecto.

### B.1. Estrategia

Aquí se recoge todo el código desarrollado en Python 🐍 relacionado con la estrategia de juego del Tres en Raya. Este código en el servidor se encuentra (de acuerdo con la figura 11 de la página 11) en el siguiente directorio:



#### Estrategia Básica (*basica.py*)

```
basica.py
1  """
2  Estrategia de juego básica y movimiento aleatorio (usado para los niveles de
3  dificultad).
4  """
5
6  import sys
7  import random
8
9
10 def game_end(M):
11     x0=M[0][0]+M[1][0]+M[2][0]
12     x1=M[0][1]+M[1][1]+M[2][1]
13     x2=M[0][2]+M[1][2]+M[2][2]
14     y0=M[0][0]+M[0][1]+M[0][2]
15     y1=M[1][0]+M[1][1]+M[1][2]
16     y2=M[2][0]+M[2][1]+M[2][2]
17     d1=M[0][0]+M[1][1]+M[2][2]
18     d2=M[0][2]+M[1][1]+M[2][0]
19
20     if x0 == 0 or x1 == 0 or x2 == 0 or y0 == 0 or y1 == 0 or y2 == 0 \
21        or d1 == 0 or d2 == 0:
22         return "User wins"
23     if x0 == 3 or x1 == 3 or x2 == 3 or y0 == 3 or y1 == 3 or y2 == 3 \
24        or d1 == 3 or d2 == 3:
25         return "AI wins"
26
27     tie = True
28     for i in range(3):
29         for j in range(3):
```

```

30         if M[i][j] == -3:
31             tie = False
32             break
33
34     if tie:
35         return "Tie"
36
37     return "Not ended"
38
39
40 def check_win(M,i,j):
41     #Comprobamos posibles jugadas ganadoras
42     if (i==0 and j==0):
43         if (M[1][0]+M[2][0]==2 or M[0][1]+M[0][2]==2 or M[1][1]+M[2][2]==2): return
44         ↪ 1
45
46     elif (i==0 and j==1):
47         if (M[0][0]+M[0][2]==2 or M[1][1]+M[2][1]==2): return 1
48
49     elif (i==0 and j==2):
50         if (M[0][0]+M[0][1]==2 or M[1][2]+M[2][2]==2 or M[2][0]+M[1][1]==2): return
51         ↪ 1
52
53     elif (i==1 and j==0):
54         if (M[0][0]+M[2][0]==2 or M[1][1]+M[1][2]==2): return 1
55
56     elif (i==1 and j==1):
57         if (M[0][0]+M[2][2]==2 or M[2][0]+M[0][2]==2 or M[0][1]+M[2][1]==2 or
58         ↪ M[1][0]+M[1][2]==2): return 1
59
60     elif (i==1 and j==2):
61         if (M[1][0]+M[1][1]==2 or M[0][2]+M[2][2]==2): return 1
62
63     elif (i==2 and j==0):
64         if (M[2][1]+M[2][2]==2 or M[0][0]+M[1][0]==2 or M[1][1]+M[0][2]==2): return
65         ↪ 1
66
67     elif (i==2 and j==1):
68         if (M[2][0]+M[2][2]==2 or M[0][1]+M[1][1]==2): return 1
69
70     elif (i==2 and j==2):
71         if (M[2][0]+M[2][1]==2 or M[0][2]+M[1][2]==2 or M[0][0]+M[1][1]==2): return
72         ↪ 1
73
74     return 0
75
76 def check(M,i,j):
77     #Comprobamos posibles jaques
78     if (i==0 and j==0):
79         if (M[1][0]+M[2][0]==0 or M[0][1]+M[0][2]==0 or M[1][1]+M[2][2]==0): return
80         ↪ 1
81
82     elif (i==0 and j==1):

```

```

77         if (M[0][0]+M[0][2]==0 or M[1][1]+M[2][1]==0): return 1
78
79     elif (i==0 and j==2):
80         if (M[0][0]+M[0][1]==0 or M[1][2]+M[2][2]==0 or M[2][0]+M[1][1]==0): return
81         ↪ 1
82
83     elif (i==1 and j==0):
84         if (M[0][0]+M[2][0]==0 or M[1][1]+M[1][2]==0): return 1
85
86     elif (i==1 and j==1):
87         if (M[0][0]+M[2][2]==0 or M[2][0]+M[0][2]==0 or M[0][1]+M[2][1]==0 or
88         ↪ M[1][0]+M[1][2]==0): return 1
89
90     elif (i==1 and j==2):
91         if (M[1][0]+M[1][1]==0 or M[0][2]+M[2][2]==0): return 1
92
93     elif (i==2 and j==0):
94         if (M[2][1]+M[2][2]==0 or M[0][0]+M[1][0]==0 or M[1][1]+M[0][2]==0): return
95         ↪ 1
96
97     elif (i==2 and j==1):
98         if (M[2][0]+M[2][2]==0 or M[0][1]+M[1][1]==0): return 1
99
100    elif (i==2 and j==2):
101        if (M[2][0]+M[2][1]==0 or M[0][2]+M[1][2]==0 or M[0][0]+M[1][1]==0): return
102        ↪ 1
103
104    return 0
105
106def moveRandom(M):
107    """
108    Mover aleatoriamente a una casilla vacía.
109    """
110    movs = []
111    for ip in range(3):
112        for jp in range(3):
113            if M[ip][jp] == -3:
114                movs.append([ip, jp])
115    if len(movs) > 0:
116        [ip, jp] = movs[random.randint(0, len(movs) - 1)]
117        M[ip][jp] = 1
118        return ip, jp
119    return -1, -1
120
121def move(M):
122    #Entrada de matriz M 3x3 con 0s (humano), 1s (máquina) y previamente inicializada
123    ↪ en "-3"s (importantente que sea así para que funcione -- por tema sumas de
124    ↪ check_win) --> decide una jugada para la máquina (estrategia basic)
125
126    #Compruebo jugadas ganadoras para la máquina
127    for i in range(3):

```



```

124         for j in range(3):
125             if(M[i][j] == -3 and check_win(M,i,j)):
126                 return i,j
127
128         for i in range(3):
129             for j in range(3):
130                 if(M[i][j]==-3 and check(M,i,j)):
131                     return i,j
132         return -1,-1
133
134
135 def moveBasic(M):
136     i, j = move(M)
137     # Mover al azar si no hay ningún movimiento.
138     if i == -1 and j == -1:
139         return moveRandom(M)
140
141     M[i][j] = 1
142     return i, j

```

basica.py

### Auxiliar (cfg.py)

```

1  """
2  Usado únicamente para poder acceder a las variables de manera global entre
3  módulos.
4  """
5
6  # Árbol de decisiones: 3 ramas con los nodos en orden y 3 vectores de conexiones
7  # entre nodos.
8  rama1 = [
9      [0, 0],
10     [1, 1],
11     [0, 1], [0, 2], [1, 2], [2, 2],
12     "EB", "EB", [2, 1], [0, 1],
13     "EB", "EB"
14 ]
15 conex1 = [
16     [1], [2, 3, 4, 5], [6], [7], [8], [9], [], [], [10], [11], [], []
17 ]
18 rama2 = [
19     [1, 1],
20     [0, 0],
21     [0, 1], [0, 2], [1, 2], [2, 2],
22     "EB", "EB", "EB", [0, 2],
23     "EB"
24 ]
25 conex2 = [
26     [1], [2, 3, 4, 5], [6], [7], [8], [9], [], [], [], [10], []

```

```

27 ]
28 rama3 = [
29     [0,1],
30     [0,0],
31     [0,2], [1,0], [1,1], [1,2], [2,0], [2,1], [2,2],
32     [2,0], [1,1], "EB", [2,0], [1,1], "EB", [1,1],
33     [1,0], "EB", "EB", "EB", [0,2], [1,0], [1,2], [2,0], [2,1],
34     [2,2], [1,0], "EB",
35     "EB", "EB"
36 ]
37 conex3 = [
38     [1], [2, 3, 4, 5, 6, 7, 8], [9], [10], [11], [12], [13], [14], [15], [16], [17],
39     ↪ [18], [19], [20, 21, 22, 23, 24], [25], [26], [27],
40     ↪ [27], [27], [27], [28], [29], [30], [31], [32]
41 ]
42 # Esta es la (única) rama cuando comienza la IA.
43 rama4 = [
44     [0, 0],
45     [0, 1], [0, 2], [1, 1], [1, 2], [2, 2],
46     [2, 0], [2, 0], [2, 2], [0, 2], [2, 0],
47     [1, 0], [1, 0], "EB", [0, 1], [1, 0],
48     [2, 2], [2, 2], [2, 0], [0, 2],
49     "EB", "EB", "EB", "EB"
50 ]
51 conex4 = [
52     [1, 2, 3, 4, 5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16],
53     ↪ [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32]
54 ]
55 # Conjunto de ramas y de conexiones.
56 ramas = [rama1, rama2, rama3, rama4]
57 conex = [conex1, conex2, conex3, conex4]
58
59 # Tableros inicializados como vacíos.
60 board = [
61     [-3, -3, -3],
62     [-3, -3, -3],
63     [-3, -3, -3]
64 ]
65 boardInt = [
66     [-3, -3, -3],
67     [-3, -3, -3],
68     [-3, -3, -3]
69 ]
70
71 # La lista sims es de simetrías y eb es estrategia básica.
72 rama = -1
73 nodo = -1
74 sims = []

```

```
76 eb = False
```

```
_____ cfg.py _____
```

### Equivalencias (*equivalencias.py*)

```
_____ equivalencias.py _____
1  """
2  Diferentes funciones para aplicar simetrías y para comprobar si dos tableros
3  son equivalentes.
4  """
5
6  import random
7
8
9  def simetria(boardP, sim):
10     """
11     Realiza la sim-ésima simetría al tablero.
12     Las simetrías están numeradas en sentido horario comenzando por las 12:00.
13     Casos especiales:
14         * -1: si coinciden.
15         * -2: si la simetría no existe.
16     """
17     boardC = []
18     for i in range(3):
19         boardC.append(list(boardP[i]))
20
21     if sim == 0:
22         for i in range(3):
23             boardC[i][0], boardC[i][2] = boardC[i][2], boardC[i][0]
24         return boardC
25
26     if sim == 1:
27         boardC[0][0], boardC[2][2] = boardC[2][2], boardC[0][0]
28         boardC[0][1], boardC[1][2] = boardC[1][2], boardC[0][1]
29         boardC[1][0], boardC[2][1] = boardC[2][1], boardC[1][0]
30         return boardC
31
32     if sim == 2:
33         for j in range(3):
34             boardC[0][j], boardC[2][j] = boardC[2][j], boardC[0][j]
35         return boardC
36
37     if sim == 3:
38         boardC[0][1], boardC[1][0] = boardC[1][0], boardC[0][1]
39         boardC[0][2], boardC[2][0] = boardC[2][0], boardC[0][2]
40         boardC[1][2], boardC[2][1] = boardC[2][1], boardC[1][2]
41         return boardC
42
43     if sim == -1:
44         return boardC
```

```

45
46     return -2
47
48
49 def simetriaMultiple(boardP, sims):
50     """
51     Realiza múltiples simetrías.
52     """
53     boardC = []
54     for i in range(3):
55         boardC.append(list(boardP[i]))
56
57     for sim in sims:
58         boardC = simetria(boardC, sim)
59
60     return boardC
61
62
63 def simetriaMultipleInversa(boardP, sims):
64     """
65     Realiza la inversa de una simetría múltiple.
66     """
67     boardC = []
68     for i in range(3):
69         boardC.append(list(boardP[i]))
70
71     for i in range(len(sims)):
72         boardC = simetria(boardC, sims[len(sims) - i - 1])
73
74     return boardC
75
76
77 def equivalente(boardA, boardB):
78     """
79     Comprueba si los dos tableros son equivalente y devuelve el número de la
80     simetría que convierte A en B.
81     Las simetrías están numeradas en sentido horario comenzando por las 12:00.
82     """
83     for sim in range(-1, 4):
84         boardSim = simetria(boardA, sim)
85         if boardSim == boardB:
86             return sim
87
88     return -2
89
90
91 def aleatorizar(board):
92     """
93     Añade una simetría extra (que no modifique el tablero) para hacer aleatorios
94     los movimientos.
95     """
96     posSims = [-1]
97     for sim in range(4):

```

```

98         if equivalente(board, simetria(board, sim)) == -1:
99             posSims.append(sim)
100
101     return posSims[random.randint(0, len(posSims) - 1)]

```

equivalencias.py

### Programa principal (*main.py*)

```

main.py
1  """
2  Estrategia general, importa estrategia básica.
3  """
4
5  import random, sys
6  import estrategia.cfg as cfg
7  from estrategia.equivalencias import *
8  from estrategia.basica import moveBasic, moveRandom
9
10
11 def actualiza(i, j):
12     """
13     Actualiza el tablero que ve el jugador y el resto de variables internas
14     (como el tablero que ve la máquina).
15     """
16     cfg.board[i][j] = 0
17     if cfg.eb == True:
18         return
19
20     # Si es el primer movimiento se detecta la rama inicial.
21     elif cfg.nodo == -1:
22         for i in range(3):
23             posSig = cfg.ramas[i][0]
24
25             cfg.boardInt[posSig[0]][posSig[1]] = 0
26             sim = equivalente(cfg.boardInt, cfg.board)
27             if sim != -2:
28                 cfg.sims.append(sim)
29                 cfg.rama = i
30                 cfg.nodo = 0
31                 return
32             cfg.boardInt[posSig[0]][posSig[1]] = -3
33
34     # A partir del segundo movimiento detectamos el nodo.
35     for pos in range(len(cfg.conex[cfg.rama][cfg.nodo])):
36         nodoSig = cfg.conex[cfg.rama][cfg.nodo][pos]
37         posSig = cfg.ramas[cfg.rama][nodoSig]
38
39         if (posSig == "EB"):
40             cfg.eb = True
41             return

```

```

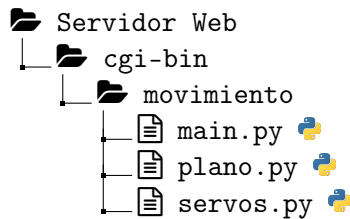
42
43     cfg.boardInt[posSig[0]][posSig[1]] = 0
44     sim = equivalente(simetriaMultiple(cfg.board, cfg.sims), cfg.boardInt)
45
46     if sim != -2:
47         cfg.sims.append(sim)
48         cfg.nodo = nodoSig
49         return
50     cfg.boardInt[posSig[0]][posSig[1]] = -3
51
52
53 def move():
54     """
55     Decide el siguiente movimiento a realizar. Actualiza el tablero y devuelve
56     cuál es el movimiento.
57     """
58     if sys.argv[5] == "Easy":
59         if random.randint(0, 100) > 50:
60             return moveRandom(cfg.board)
61
62     if cfg.eb == True:
63         i, j = moveBasic(cfg.board)
64         return i, j
65
66     if len(cfg.conex[cfg.rama][cfg.nodo]) > 1:
67         print("Error de longitud")
68
69     cfg.nodo = cfg.conex[cfg.rama][cfg.nodo][0]
70     move = cfg.ramas[cfg.rama][cfg.nodo]
71
72     cfg.sims.append(aleatorizar(cfg.boardInt))
73
74     if move == "EB":
75         cfg.eb = True
76         i, j = moveBasic(cfg.board)
77         return i, j
78
79     # Copia del tablero para poder detectar el movimiento.
80     boardC = []
81     for i in range(3):
82         boardC.append(list(cfg.board[i]))
83
84     cfg.boardInt[move[0]][move[1]] = 1
85     cfg.board = simetriaMultipleInversa(cfg.boardInt, cfg.sims)
86
87     for i in range(3):
88         for j in range(3):
89             if boardC[i][j] != cfg.board[i][j]:
90                 return i, j
91
92     return -1, -1

```

main.py

## B.2. Movimiento

Aquí se recoge todo el código desarrollado en Python 🐍 relacionado con el movimiento del brazo robótico. Este código en el servidor se encuentra (de acuerdo con la figura 11 de la página 11) en el siguiente directorio:



### Programa principal (*main.py*)

```

1  """
2  Se encarga de coordinar el movimiento del brazo robótico.
3  - Define la posición espacial de las casillas del tablero y de los
4  almacenes.
5  - Contiene funciones que permiten mover piezas de una posición
6  (espacial) a otra.
7  """
8
9  from math import *
10 from movimiento.plano import verticalMove
11 from movimiento.servos import *
12
13
14 # DEFINICION DE VARIABLES
15 # Almacén más separado, si no parece que no es capaz de llegar.
16 # TODO: Revisar esto.
17 R = 283.582
18 ang = (40*pi)/180
19
20 # V1 es un vector con la posición de 4 "X"'s
21 V1 = [[R*cos(ang), R*sin(ang)], [R*cos(ang), R*sin(ang)], [R*cos(ang), R*sin(ang)],
22 ↪ [R*cos(ang), R*sin(ang)]]
23 # U1 indica el número de pieza a coger en almacén de "X"'s
24 U1 = 0
25
26 # V2 es un vector con la posición de 4 "O"'s
27 V2 = [[R*cos(-ang), R*sin(-ang)], [R*cos(-ang), R*sin(-ang)], [R*cos(-ang),
28 ↪ R*sin(-ang)], [R*cos(-ang), R*sin(-ang)]]
29 # U1 indica el número de pieza a coger en almacén de "O"'s
30 U2 = 0
31
32 # Alturas del almacén.
33 H = [50, 33.333, 16.667, 0]
34
35 # Unión de variables del almacén.
36 V = [V1, V2]

```

```

35 U = [U1, U2]
36
37 # Posicion del tablero de casillas ancho_tablero*ancho_tablero (mm2)
38 ancho_tablero = 33.33333
39 # Tablero también más separado.
40 # TODO: Revisar esto.
41 x_inicial_t = 116.988
42
43 fila_1 = [[x_inicial_t, ancho_tablero], [x_inicial_t + ancho_tablero,
↪ ancho_tablero], [x_inicial_t + 2*ancho_tablero, ancho_tablero]]
44 fila_2 = [[x_inicial_t, 0], [x_inicial_t + ancho_tablero, 0], [x_inicial_t +
↪ 2*ancho_tablero, 0]]
45 fila_3 = [[x_inicial_t, -ancho_tablero], [x_inicial_t + ancho_tablero,
↪ -ancho_tablero], [x_inicial_t + 2*ancho_tablero, -ancho_tablero]]
46 tablero = [fila_1, fila_2, fila_3]
47
48 # Vector con los ángulos de los servos
49 S = [0]*6
50
51
52 def reset_servos():
53     global S
54     S = [0]*6
55     moveServos(S)
56
57
58 def movePieceFromTo(p0, pf, h):
59     """
60     Mueve una pieza sobre el plano (horizontal) de una posición p0 = [x0, y0] a
61     una pf = [xf, yf].
62
63     La pieza utilizada es una goma Marca: Milan, Modelo: 430
64     Medidas: 2.8 x 2.8 x 1.3 cm.
65     """
66     printServosAngles(S)
67
68     # Posicionar pinza abierta por encima de la pieza (en posición de
69     # inicio).
70     r = sqrt(pow(p0[0], 2) + pow(p0[1], 2))
71     S[0] = atan(p0[1]/p0[0])
72     S[4] = -S[0] # MODIFICAR POR TEMA ANGULOS NEGATIVOS
73     # ancho = 34 # Le dejo margen. Hay q vigilar q no toque a otras piezas
74     # S[5] = acos((ancho+18)/52)
75     # h0 = 26*sin(S[5])+68
76     phi1, phi2, phi3, phi4 = verticalMove(r, h)
77     S[1] = phi1
78     S[2] = phi4
79
80     moveServos(S)
81     printServosAngles(S)
82
83     # # Cerrar pinza para coger pieza.
84     # ancho = 25 # A 25 mm. (< 28) la pinza hara fuerza - MODIFICAR

```



```

85     # S[5] = acos((ancho+18)/52)
86     # # h0 = 26*sin(S[5])+68
87     # phi1, phi2, phi3, phi4 = verticalMove(r, h0)
88     # S[1] = phi1
89     # S[2] = phi4
90
91     # moveServos(S)
92     # printServosAngles(S)
93
94     # # Subir la pinza para que no se choque
95     # phi1, phi2, phi3, phi4 = verticalMove(r, h0+10) # TODO: MODIFICAR
96     # S[1] = phi1
97     # S[2] = phi4
98
99     # moveServos(S)
100    # printServosAngles(S)
101
102    # Mover la pieza hasta la posición final
103    r = sqrt(pow(pf[0], 2) + pow(pf[1], 2))
104    S[0] = atan(pf[1]/pf[0])
105    S[4] = -S[0] # MODIFICAR POR TEMA ANGULOS NEGATIVOS
106    phi1, phi2, phi3, phi4 = verticalMove(r, 0)
107    S[1] = phi1
108    S[2] = phi4
109
110    moveServos(S)
111    printServosAngles(S)
112
113    # # Bajar pinza sobre posición final.
114    # phi1, phi2, phi3, phi4 = verticalMove(r, h0)
115    # S[1] = phi1
116    # S[2] = phi4
117
118    # moveServos(S)
119    # printServosAngles(S)
120
121    # # Soltar pieza en la posición final.
122    # ancho = 34 # > 28
123    # S[5] = acos((ancho+18)/52)
124    # # h0 = 26*sin(S[5])+68
125    # phi1, phi2, phi3, phi4 = verticalMove(r, h0)
126    # S[1] = phi1
127    # S[2] = phi4
128
129    # moveServos(S)
130    # printServosAngles(S)
131
132    # # Subir la pinza para que no se choque
133    # phi1, phi2, phi3, phi4 = verticalMove(r, h0+50) # MODIFICAR
134    # S[1] = phi1
135    # S[2] = phi4
136
137    # moveServos(S)

```

```

138     # printServosAngles(S)
139
140     # Dejar el brazo en posición por defecto para permitir ver el tablero.
141     # Esta posición se podría mejorar
142     reset_servos()
143     printServosAngles(S)
144
145
146 def movePiece(i, j, tipo):
147     """
148     Posiciona una pieza (de un tipo) en una posición concreta del tablero.
149     """
150     if tipo == "X":
151         tipo = 0
152     else:
153         tipo = 1
154
155     print("%.1f" % V[tipo][U[tipo]][0], end = ",")
156     print("%.1f" % V[tipo][U[tipo]][1], end = ",")
157     print("%.1f" % tablero[i][j][0], end = ",")
158     print("%.1f" % tablero[i][j][1], end = ",")
159
160     movePieceFromTo(V[tipo][U[tipo]], tablero[i][j], H[U[tipo]])
161     U[tipo] += 1
162
163     print(i, end = ",")
164     print(j, end = ",")

```

main.py

### Plano (*plano.py*)

```

1     """
2     Resolución (analítica) de las ecuaciones de enlace en planos verticales.
3     - Permite pasar de posiciones en el plano a ángulos de los servomotores
4     - Se definen los parámetros del brazo.
5     """
6
7     from math import sin, cos, acos, asin, pi, sqrt
8
9
10    def resolverSistemaGeneral(p1, p2, d1, d2):
11        """
12        Resuelve el sistema:
13             $p1 = d1 \cdot \cos(b1) + d2 \cdot \cos(b2)$ 
14             $p2 = d1 \cdot \sin(b1) + d2 \cdot \sin(b2)$ 
15            Donde  $d1$ ,  $d2$ ,  $p1$  y  $p2$  son parámetros y  $b1$ ,  $b2$  son los ángulos a obtener.
16        """
17         $c1 = p1 \cdot p1 + p2 \cdot p2 - d1 \cdot d1 + d2 \cdot d2$ 
18         $c2 = 2 \cdot d2 \cdot p1$ 

```

plano.py

```

19     c3 = 2*d2*p2
20     c4 = c2*c2 + c3*c3
21     c5 = 2*c1*c2
22     c6 = c1*c1 - c3*c3
23
24     if (c5*c5 - 4*c4*c6 < 0):
25         print("RAIZ COMPLEJA")
26         return []
27
28     raiz = sqrt(c5*c5 - 4*c4*c6)
29
30     aes = [(c5 + raiz)/(2*c4), (c5 - raiz)/(2*c4)]
31
32     sols = []
33     for a in aes:
34         if abs(a) <= 1:
35             b2s = [acos(a), -acos(a)]
36             for b2 in b2s:
37                 sinb1 = (p2 - d2*sin(b2))/d1
38                 if abs(sinb1) <= 1:
39                     b1s = [asin(sinb1), pi - asin(sinb1)]
40                     for b1 in b1s:
41                         sols.append([b1, b2])
42
43     return sols
44
45
46 def extraerSolucion2(phiss, phi1, phi4):
47     """
48     Devuelve una única solución que cumpla las ecuaciones del sistema 2 y con
49     los ángulos de los servos dentro del rango de funcionamiento.
50     """
51     for phis in phiss:
52         phi2 = phis[0]
53         phi3 = phis[1]
54
55         if abs(l3*cos(phi2) + l1*cos(phi3) - l2*cos(phi1) + l3*cos(phi4)) < eps and
56             ↪ \
57             abs(l3*sin(phi2) + l1*sin(phi3) - l2*sin(phi1) + l3*sin(phi4)) < eps and
58             ↪ \
59             phi2 >= 0 and phi2 <= pi and phi3 >= 0 and phi3 <= pi:
60             return phi2, phi3
61
62 def resolverSistema2(phi1, phi4):
63     """
64     Resuelve el sistema:
65         px = l2*cos(phi1) + l1*cos(phi4) + l4*cos(35°)
66         py = l2*sin(phi1) + l1*sin(phi4) + h
67     Donde l1, l2, l4 son parámetros del brazo; px, py y h son los parámetros de
68     la función y phi1, phi4 son los ángulos a obtener.
69     Solo devuelve una solución.
70     """

```

```

70     phiss = resolverSistemaGeneral(l2*cos(phi1) - l3*cos(phi4),
71                                   l2*sin(phi1) - l3*sin(phi4),
72                                   l3, l1)
73     return extraerSolucion2(phiss, phi1, phi4)
74
75
76 def extraerSolucion1(phiss, px, py):
77     """
78     Devuelve una única solución que cumpla las ecuaciones del sistema 1 y con
79     los ángulos de los servos dentro del rango de funcionamiento.
80     """
81     for phis in phiss:
82         phi1 = phis[0]
83         phi4 = phis[1]
84
85         if abs(l2*cos(phi1) + l1*cos(phi4) + l4*cos((35*pi)/180) - px) < eps and \
86            abs(l2*sin(phi1) + l1*sin(phi4) + h - py) < eps and \
87            phi1 >= 0 and phi1 <= pi and phi4 <= pi/2 and phi4 >= -pi/2:
88             return phi1, phi4
89
90
91 def resolverSistema1(px, py):
92     """
93     Resuelve el sistema:
94         px = l2*cos(phi1) + l1*cos(phi4) + l4*cos(35°)
95         py = l2*sin(phi1) + l1*sin(phi4) + h
96     Donde l1, l2, l4 son parámetros del brazo; px, py y h son los parámetros de
97     la función y phi1, phi4 son los ángulos a obtener.
98     Solo devuelve una solución.
99     """
100    phiss = resolverSistemaGeneral(px - l4*cos((35*pi)/180), py - h, l2, l1)
101    return extraerSolucion1(phiss, px, py)
102
103
104 def verticalMove(px, py):
105     """
106     Dada una posición (px, py) en un plano vertical, devuelve los ángulos de los
107     servos que corresponden a esa posición.
108     """
109     # Sistema 1.
110     phi1, phi4 = resolverSistema1(px, py)
111     # Sistema 2.
112     phi2, phi3 = resolverSistema2(phi1, phi4)
113     return phi1, phi2, phi3, phi4
114
115
116 # Tolerancia.
117 eps = 1e-6
118
119 # Definir parámetros del brazo robótico.
120 l1 = 160
121 l2 = 148
122 l3 = 54

```

```

123 14 = 42
124 15 = 68.81
125 dx = 34.4
126 dy = 24.22
127 # Consideraremos la altura como un parámetro más.
128 h = 9.404993

```

plano.py

### Servomotores (*servos.py*)

```

1  """
2  Define y mueve simultáneamente y de manera progresiva los servos.
3  -- Todo lo de mover los servos está por ahora comentado --
4
5  Los servos están numerados de la siguiente manera:
6  0: ROTACION
7  1: BRAZO PRINCIPAL
8  2: BRAZO SECUNDARIO
9  3: NO MUEVE NADA
10 4: PINZA ROTACIÓN
11 5: PINZA APERTURA
12 """
13
14 # from adafruit_servokit import ServoKit
15 from time import sleep
16 from math import *
17 from threading import Thread
18
19
20 # kit = ServoKit(channels = 16)
21
22 # Variable real de ángulos en servos.
23 Sp = [0]*6
24
25
26 def rad2Deg(phi):
27     """
28     Convierte de radianes a grados.
29     """
30     return (phi* 180)/pi
31
32
33 def printServosAngles(S):
34     """
35     Devuelve información de los ángulos de los servos.
36     """
37     # for i in range(6):
38     #     # El 3 no es un servo.
39     #     if i != 3:

```

```

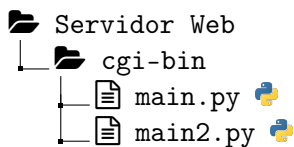
40     #         print(round(rad2Deg(S[i])), end = ",")
41     print(round(rad2Deg(S[0])), end = ",")
42     print(round(rad2Deg(S[1])), end = ",")
43     print(round(rad2Deg(S[2])), end = ",")
44
45
46     def moveServo(servo, angle):
47         """
48         Mueve el servo a un determinado ángulo (en radianes) de manera progresiva.
49         """
50         global Sp
51         angle = rad2Deg(angle)
52         steps = 50
53         time = 0 # Cambiar este valor al hacer la conexión real con el brazo.
54         timeStep = time/steps
55         angleIni = Sp[servo]
56         h = (angle - angleIni)/steps
57         for i in range(steps):
58             angleIni = angleIni + h
59             # kit.servo[servo].angle = round(angleIni)
60             sleep(timeStep)
61
62         Sp[servo] = round(angleIni)
63
64
65     def moveServos(angles):
66         """
67         Mueve los servos simultáneamente a los ángulos dados (en radianes).
68         """
69         Thread(target = moveServo, args = [0, angles[0]]).start()
70         Thread(target = moveServo, args = [1, angles[1]]).start()
71         Thread(target = moveServo, args = [2, angles[2]]).start()
72         Thread(target = moveServo, args = [4, angles[4]]).start()
73         Thread(target = moveServo, args = [5, angles[5]]).start()

```

servos.py

### B.3. Programa principal

Aquí se recoge todo el código desarrollado en Python 🐍 encargado de fusionar/coordinar el código de las secciones anteriores (B.1 y B.2). Este código en el servidor se encuentra (de acuerdo con la figura 11 de la página 11) en el siguiente directorio:



Programa principal 1 (*main1.py*)

```

1      """
2      Es el programa principal para cuando comienza a jugar el usuario, que coordina
3      la estrategia y el movimiento del brazo robótico.
4
5      Se le pasan por parámetros al ejecutarlo el estado de una partida ya comenzada,
6      entonces:
7          1. Posiciona servomotores en posición de inicio.
8          2. Mueve (físicamente) la ficha del usuario a la posición seleccionada.
9          3. Da una respuesta al tablero (de acuerdo a la dificultad).
10         4. Mueve (físicamente) la ficha de respuesta.
11         5. Devuelve diferentes datos (entre ellos las próximas url's) para que la
12         página web se pueda actualizar. Estos datos se devuelven en una línea y
13         separados por comas.
14     """
15
16     import sys
17     import estrategia.cfg as cfg
18     from estrategia.main import actualiza, move
19     from estrategia.basica import game_end
20     from estrategia.equivalencias import simetriaMultiple
21     import movimiento.main
22     from movimiento.main import movePiece, reset_servos
23
24
25     def board2Str(M):
26         """
27         Convierte el tablero a string.
28         """
29         boardStr = ""
30         for i in range(3):
31             for j in range(3):
32                 if (M[i][j] == 0):
33                     boardStr += "0"
34                 elif (M[i][j] == 1):
35                     boardStr += "X"
36                 else:
37                     boardStr += "."
38
39         return boardStr
40
41
42     def readVariables():
43         """
44         Leer las variables y guardarlas. Devuelve la última jugada realizada.
45         """
46         movsStr = sys.argv[1]
47
48         # Devolver posición en almacén.
49         print(int(len(movsStr)/4), end = ",")
50

```

```

51     movs = []
52     for i in range(int(len(movsStr)/2)):
53         movs.append([int(movsStr[2*i]), int(movsStr[2*i + 1])])
54
55     for i in range(len(movs) - 1):
56         if i%2 == 0:
57             cfg.board[movs[i][0]][movs[i][1]] = 0
58         else:
59             cfg.board[movs[i][0]][movs[i][1]] = 1
60
61     cfg.rama = int(sys.argv[2])
62     cfg.nodo = int(sys.argv[3])
63     simsStr = sys.argv[4]
64     for i in range(len(simsStr)):
65         if simsStr[i] != "-":
66             if i != 0 and simsStr[i - 1] != "-":
67                 cfg.sims.append(int(simsStr[i]))
68             elif i != 0 and simsStr[i - 1] == "-":
69                 cfg.sims.append(int(simsStr[i-1:i+1]))
70             else:
71                 cfg.sims.append(int(simsStr[i]))
72
73     cfg.boardInt = simetriaMultiple(cfg.board, cfg.sims)
74
75     if sys.argv[5] == "False":
76         cfg.eb = False
77     else:
78         cfg.eb = True
79
80     actualiza(movs[len(movs) - 1][0], movs[len(movs) - 1][1])
81
82     return movs[len(movs) - 1][0], movs[len(movs) - 1][1]
83
84
85 def nextUrl(i, j):
86     """
87     Crea e imprime la siguiente dirección web. También el tablero.
88     """
89     url = "&rama=" + str(cfg.rama)
90     url += "&nodo=" + str(cfg.nodo)
91     url += "&sims="
92     for sim in cfg.sims:
93         url += str(sim)
94     url += "&eb=" + str(cfg.eb)
95     url += "&movs=" + sys.argv[1] + str(i) + str(j)
96
97     return url
98
99
100 def printData(i, j):
101     """
102     Imprime por pantalla diferentes datos.
103     """

```



```

104     data = board2Str(cfg.board) + ","
105     data += nextUrl(i, j)
106     print(data, end = ",")
107
108
109     # Iniciar los servos.
110     reset_servos()
111     # Leer movimiento humano y mover la pieza correspondiente.
112     i, j = readVariables()
113     movePiece(i, j, "O")
114     if game_end(cfg.board) != "User wins":
115         # Decidir movimiento respuesta y mover la pieza correspondiente.
116         i, j = move()
117         # Si se puede hacer movimiento, mover la pieza.
118         if i != -1 and j != -1:
119             movePiece(i, j, "X")
120     # Devolver datos necesarios.
121     printData(i, j)
122     # Mirar si la partida ha terminado.
123     print(game_end(cfg.board))

```

main.py

## Programa principal 2 (*main2.py*)

```

1  """
2  Es el programa principal para cuando el usuario juega segundo, que coordina la
3  estrategia y el movimiento del brazo robótico.
4  """
5  Se le pasan por parámetros al ejecutarlo el estado de una partida ya comenzada,
6  entonces:
7  1. Posiciona servomotores en posición de inicio.
8  2. Mueve (físicamente) la ficha del usuario a la posición seleccionada.
9  3. Da una respuesta al tablero (de acuerdo a la dificultad).
10 4. Mueve (físicamente) la ficha de respuesta.
11 5. Devuelve diferentes datos (entre ellos las próximas url's) para que la
12 página web se pueda actualizar. Estos datos se devuelven en una línea y
13 separados por comas.
14 """
15
16 import sys
17 import estrategia.cfg as cfg
18 from estrategia.main import actualiza, move
19 from estrategia.basica import game_end
20 from estrategia.equivalencias import simetriaMultiple
21 import movimiento.main
22 from movimiento.main import movePiece, reset_servos
23
24
25 def board2Str(M):

```

main2.py

```

26     """
27     Convierte el tablero a string.
28     """
29     boardStr = ""
30     for i in range(3):
31         for j in range(3):
32             if (M[i][j] == 0):
33                 boardStr += "0"
34             elif (M[i][j] == 1):
35                 boardStr += "X"
36             else:
37                 boardStr += "."
38
39     return boardStr
40
41
42 def readVariables():
43     """
44     Leer las variables y guardarlas. Devuelve la última jugada realizada.
45     """
46     movsStr = sys.argv[1]
47
48     # Devolver posición en almacén.
49     print(int(len(movsStr - 2)/4), end = ",")
50
51     movs = []
52     for i in range(int(len(movsStr)/2)):
53         movs.append([int(movsStr[2*i]), int(movsStr[2*i + 1])])
54
55     for i in range(len(movs) - 1):
56         if i%2 == 0:
57             cfg.board[movs[i][0]][movs[i][1]] = 1
58         else:
59             cfg.board[movs[i][0]][movs[i][1]] = 0
60
61     cfg.rama = int(sys.argv[2])
62     cfg.nodo = int(sys.argv[3])
63     simsStr = sys.argv[4]
64     for i in range(len(simsStr)):
65         if simsStr[i] != "-":
66             if i != 0 and simsStr[i - 1] != "-":
67                 cfg.sims.append(int(simsStr[i]))
68             elif i != 0 and simsStr[i - 1] == "-":
69                 cfg.sims.append(int(simsStr[i-1:i+1]))
70             else:
71                 cfg.sims.append(int(simsStr[i]))
72
73     cfg.boardInt = simetriaMultiple(cfg.board, cfg.sims)
74
75     if sys.argv[5] == "False":
76         cfg.eb = False
77     else:
78         cfg.eb = True

```




```

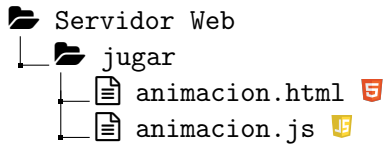
79
80     if len(movs) == 1:
81         cfg.rama = -1
82
83     actualiza(movs[len(movs) - 1][0], movs[len(movs) - 1][1])
84
85     return movs[len(movs) - 1][0], movs[len(movs) - 1][1]
86
87
88 def nextUrl(i, j):
89     """
90     Crea e imprime la siguiente dirección web. También el tablero.
91     """
92     url = "&rama=" + str(cfg.rama)
93     url += "&nodo=" + str(cfg.nodo)
94     url += "&sims="
95     for sim in cfg.sims:
96         url += str(sim)
97     url += "&eb=" + str(cfg.eb)
98     url += "&movs=" + sys.argv[1] + str(i) + str(j)
99
100    return url
101
102
103 def printData(i, j):
104     """
105     Imprime por pantalla diferentes datos.
106     """
107     data = board2Str(cfg.board) + ","
108     data += nextUrl(i, j)
109     print(data, end = ",")
110
111
112 # Iniciar los servos.
113 reset_servos()
114 # Leer movimiento humano y mover la pieza correspondiente.
115 i, j = readVariables()
116 movePiece(i, j, "X")
117 if game_end(cfg.board) != "User wins":
118     # Decidir movimiento respuesta y mover la pieza correspondiente.
119     i, j = move()
120     # Si se puede hacer movimiento, mover la pieza.
121     if i != -1 and j != -1:
122         movePiece(i, j, "O")
123 # Devolver datos necesarios.
124 printData(i, j)
125 # Mirar si la partida ha terminado.
126 print(game_end(cfg.board))

```

main2.py

## B.4. Animación

Aquí se recoge el código desarrollado en JavaScript  que hace posibles las animaciones del brazo robótico virtual que se muestra en la web. También se añade su correspondiente HTML  y CSS . Este código en el servidor se encuentra (de acuerdo con la figura 11 de la página 11) en el siguiente directorio:



### Animación HTML (*animacion.php*)

```

1  <div id ="alzado">
2    <div id ="barra2A"></div>
3
4    <div id="barra1ACont">
5      <div id="barra1A"></div>
6      <div id="articulacionPos1A">
7        <div id="articulacion"></div>
8      </div>
9    </div>
10
11  <div id ="baseA"></div>
12  <div id="tierraA"></div>
13
14  <div id="pinzaContA">
15    <div id="extensionA"></div>
16    <div id ="pinzaA"></div>
17    <div id="tenazas_hA"></div>
18    <div id="tenazas_v1A"></div>
19    <div id="tenazas_v2A"></div>
20    <div id="articulacionPos2A">
21      <div id="articulacion"></div>
22    </div>
23    <div id="articulacionPos3A">
24      <div id="articulacion"></div>
25    </div>
26    <div id="piezaPinzaA"></div>
27  </div>
28
29  <div id="almacenA">
30    <div id="almacenPieza0A"></div>
31    <div id="almacenPieza1A"></div>
32    <div id="almacenPieza2A"></div>
33    <div id="almacenPieza3A"></div>
34  </div>
35
36  <div id="tableroA">
37    <div id="tableroPieza0A"></div>
  
```

```

38     <div id="tableroPieza1A"></div>
39     <div id="tableroPieza2A"></div>
40 </div>
41 </div>
42
43
44 <div id="planta">
45     <div id="almacenXsP"></div>
46     <div id="almacenOsP"></div>
47
48     <div id="robotP">
49         <div id="barra2ContP">
50             <div id="barra2P"></div>
51             <div id="articulacionPos2P">
52                 <div id="articulacion"></div>
53             </div>
54         </div>
55
56         <div id="barra1ContP">
57             <div id="barra1P"></div>
58             <div id="articulacionPos1P">
59                 <div id="articulacion"></div>
60             </div>
61         </div>
62
63         <div id="articulacionPos3P">
64             <div id="articulacion"></div>
65         </div>
66
67         <div id="extensionP"></div>
68
69         <div id="pinzaContP">
70             <div id="articulacionPos4P">
71                 <div id="articulacion"></div>
72             </div>
73             <div id="pinzaP"></div>
74             <div id="piezaPinzaP"></div>
75         </div>
76     </div>
77
78 <div id="baseP"></div>
79 <div id="tierraP"></div>
80
81 <table id="boardP">
82     <?php
83     for ($i = 0; $i < 3; ++$i) {
84         echo '<tr>';
85         for ($j = 0; $j < 3; ++$j) {
86             if ($board[3*$i + $j] == "X")
87                 echo '<td><div id="tableroPiezaXP"></div></td>';
88             else if ($board[3*$i + $j] == "O")
89                 echo '<td><div id="tableroPiezaOP"></div></td>';
90             else

```

```

91         echo '<td><a href="'. $url. $i. $j. '"><button
           ↳ id="ticP"><span></span></button></a></td>';
92     }
93     echo '</tr>';
94 }
95 ?>
96 </table>
97 </div>
98
99
100 <script type="text/javascript">
101     async function comenzarAnimacion() {
102         await sleep(500);
103
104         move_piece(3, 1);
105         await sleep(550*delay);
106
107         move_piece(-2, 3);
108         await sleep(550*delay);
109
110         reset();
111     }
112 </script>

```

animacion.php

### Animación JS (*animacion.js*)

```

1 // Funciones para web.
2 function mostrarAnimacion() {
3     botonControl.style.border = "";
4     animacion.style.visibility = "visible";
5     planta.style.visibility = "hidden";
6     alzado.style.top = "250px";
7     alzado.style.transform = "scale(.7, .7)";
8     animacion.style.height = "300px";
9     animacion.style.width = "500px";
10    animacion.style.top = "120px";
11    animacion.style.left = "50px";
12    animacion.style.background = "#F0F0F0";
13    animacion.style.border = "";
14    velocidad.style.display = "none";
15    botonAnimacion.style.border = "3px inset Black";
16    cerrarPant.style.display = "none";
17    pantCompl.style.display = "block";
18    comenzarAnimacion();
19 }
20
21
22 function mostrarAnimacionCompleta() {

```

animacion.js

```
23     planta.style.visibility = "visible";
24     alzado.style.top = "290px";
25     alzado.style.transform = "scale(.8, .8)";
26     animacion.style.height = "530px";
27     animacion.style.width = "96.5%";
28     animacion.style.top = "10px";
29     animacion.style.left = "17px";
30     animacion.style.border = "3px solid Black";
31     animacion.style.background = "##E6E6E6";
32     velocidad.style.display = "block";
33     cerrarPant.style.display = "block";
34     pantCompl.style.display = "none";
35 }
36
37
38 function cerrarAnimacion() {
39     control.style.visibility = "visible";
40     animacion.style.visibility = "hidden";
41     planta.style.visibility = "hidden";
42     mostrarControles();
43 }
44
45
46 function mostrarControles() {
47     animacion.style.visibility = "hidden";
48     control.style.visibility = "visible";
49     botonControl.style.border = "3px inset Black";
50     botonAnimacion.style.border = "";
51 }
52
53
54 function almacenColorA(color) {
55     almacenPieza0A.style.background = color;
56     almacenPieza1A.style.background = color;
57     almacenPieza2A.style.background = color;
58     almacenPieza3A.style.background = color;
59 }
60
61
62 function piezaPinzaColorA(color) {
63     piezaPinzaA.style.background = color;
64 }
65
66
67 function actualizaAlmacenA(noAlmacen) {
68     for (var i = 0; i < 4; ++i) {
69         if (i < noAlmacen)
70             document.getElementById("almacenPieza" + i + "A").style.display =
71                 ↪ "none";
72         else
73             document.getElementById("almacenPieza" + i + "A").style.display =
74                 ↪ "block";
75     }
76 }
```

```

74 }
75
76
77 // Funciones para movimiento del brazo robótico virtual.
78 function sleep(delay) {
79     return new Promise(resolve => setTimeout(resolve, delay));
80 }
81
82
83 async function rotacionP(phi0, phif) {
84     sleepTime = 1.35*delay*Math.abs(phi0 - phif);
85
86     var phi = phi0;
87     var inc = 1;
88
89     if (phi0 > phif)
90         inc = -1;
91
92     while (phi != phif) {
93         await sleep(delay);
94         phi += inc;
95         robotP.style.transform = "rotate(" + (-phi) + "deg";
96         pinzaContP.style.transform = "rotate(" + phi + "deg";
97     }
98 }
99
100
101 async function move_barra1(phi0, phif) {
102     var phi = phi0;
103     var inc = 1;
104     if (phi0 > phif)
105         inc = -1;
106
107     while (phi != phif) {
108         await sleep(delay);
109         phi += inc;
110
111         barra1ACont.style.transform = "rotate(-" + phi + "deg";
112
113         var xBarra2A = xBarra2AIniA + length1A*Math.cos(phi*Math.PI/180);
114         var yBarra2A = yBarra2AIniA + length1A*(Math.sin(phi*Math.PI/180) - 1);
115         barra2A.style.left = xBarra2A + "px";
116         barra2A.style.bottom = yBarra2A + "px";
117
118         var xPinza = posPinzaA[0] + length1A*(Math.cos(phi*Math.PI/180) -
119             ↪ Math.cos(phi0*Math.PI/180));
120         var yPinza = posPinzaA[1] + length1A*(Math.sin(phi*Math.PI/180) -
121             ↪ Math.sin(phi0*Math.PI/180));
122         pinzaContA.style.left = xPinza + "px";
123         pinzaContA.style.bottom = yPinza + "px";
124
125         var length1P = length1A*(Math.cos(phi*Math.PI/180));
126         barra1P.style.width = length1P + "px";

```



```

125     articulacionPos1P.style.left = length1P - 15 + "px";
126     barra2ContP.style.left = length1P + 40 + "px";
127     extensionP.style.left = 40 + length1P +
    ↪     document.getElementById("barra2P").offsetWidth + "px";
128     pinzaContP.style.left = 40 + length1P +
    ↪     document.getElementById("barra2P").offsetWidth + 51.6 + "px";
129 }
130
131 posPinzaA = [xPinza, yPinza];
132 }
133
134
135 async function move_barra2(phi0, phif) {
136     var phi = phi0;
137     var inc = 1;
138     if (phi0 > phif)
139         inc = -1;
140
141     var transPinzaIni = document.getElementById("pinzaContA").style.transform;
142     while (phi != phif) {
143         await sleep(delay);
144         phi += inc;
145
146         barra2A.style.transform = "rotate(" + phi + "deg)";
147
148         var xPinza = posPinzaA[0] + length2A*(Math.cos(phi*Math.PI/180) -
    ↪     Math.cos(phi0*Math.PI/180));
149         var yPinza = posPinzaA[1] + length2A*(-Math.sin(phi*Math.PI/180) +
    ↪     Math.sin(phi0*Math.PI/180));
150         pinzaContA.style.left = xPinza + "px";
151         pinzaContA.style.bottom = yPinza + "px";
152
153         var length2P = length2A*(Math.cos(phi*Math.PI/180));
154         barra2P.style.width = length2P + "px";
155         articulacionPos2P.style.left = length2P - 15 + "px";
156         extensionP.style.left = 40 + document.getElementById("barra1P").offsetWidth
    ↪     + length2P + "px";
157         pinzaContP.style.left = 40 + document.getElementById("barra1P").offsetWidth
    ↪     + length2P + 51.6 + "px";
158     }
159
160     posPinzaA = [xPinza, yPinza];
161 }
162
163
164 async function move_barras(phi1, phi2) {
165     var sleepTime1 = 2*delay*Math.abs(phi1 - angulosBarrasA[0]);
166     var sleepTime2 = 2*delay*Math.abs(phi2 - angulosBarrasA[1]);
167     sleepTime = 1.1*(sleepTime1 + sleepTime2);
168
169     move_barra1(angulosBarrasA[0], phi1);
170     await sleep(sleepTime1);
171     move_barra2(angulosBarrasA[1], phi2);

```

```
172     await sleep(sleepTime2);
173     angulosBarrasA = [phi1, phi2];
174 }
175
176
177 // async funcion move_piece() {
178
179 // }
180
181
182 async function move_piece(noAlmacen, noTablero) {
183     actualizaAlmacenA(Math.abs(noAlmacen) - 1);
184     if (noAlmacen >= 0) {
185         noAlmacen -= 1;
186         noTablero -= 1;
187         almacenAngP = Math.abs(almacenAngP);
188         almacenColorA("#3862E0");
189         piezaPinzaColorA("#3862E0");
190         if (noTablero >= 0)
191             document.getElementById("tableroPieza" + noTablero +
192                 ↪ "A").style.background = "#3862E0";
193         else {
194             noTablero *= -1;
195             document.getElementById("tableroPieza" + noTablero +
196                 ↪ "A").style.background = "#336600";
197         }
198     }
199     else {
200         noAlmacen *= -1;
201         noAlmacen -= 1;
202         almacenAngP = -Math.abs(almacenAngP);
203         almacenColorA("#336600");
204         piezaPinzaColorA("#336600");
205         if (noTablero >= 0) {
206             noTablero -= 1;
207             document.getElementById("tableroPieza" + noTablero +
208                 ↪ "A").style.background = "#336600";
209         }
210         else {
211             noTablero *= -1;
212             noTablero -= 1;
213             document.getElementById("tableroPieza" + noTablero +
214                 ↪ "A").style.background = "#3862E0";
215         }
216     }
217 }
218
219 // Ir al almacén.
220 rotacionP(0, almacenAngP);
221 await sleep(sleepTime);
222 tableroA.style.opacity = "0.1";
223 almacenA.style.opacity = "1";
224
225 move_barras(angulosBarrasA[0] - 1, 20);
```





```

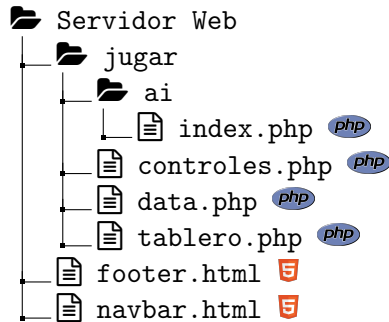
221     await sleep(sleepTime);
222
223     move_barras(almacenAngsA[noAlmacen][0], almacenAngsA[noAlmacen][1]);
224     await sleep(sleepTime);
225     document.getElementById("piezaPinzaA").style.display = "block";
226     document.getElementById("almacenPieza" + noAlmacen + "A").style.display =
        ↪ "none";
227
228     // Ir al tablero.
229     rotacionP(almacenAngP, 0);
230     await sleep(sleepTime);
231
232     almacenA.style.opacity = "0.1";
233     tableroA.style.opacity = "1";
234     move_barras(tableroAngsA[noTablero][0], tableroAngsA[noTablero][1]);
235     await sleep(sleepTime);
236     piezaPinzaA.style.display = "none";
237     document.getElementById("tableroPieza" + noTablero + "A").style.display =
        ↪ "block";
238 }
239
240
241 async function reset() {
242     move_barras(90, 77);
243     await sleep(sleepTime);
244 }
245
246 var length1A = document.getElementById("barra1A").offsetWidth;
247 var length2A = document.getElementById("barra2A").offsetWidth;
248
249 var xBarra2AIniA =
    ↪ getComputedStyle(document.getElementById("barra2A")).getPropertyValue("left");
250 var yBarra2AIniA =
    ↪ getComputedStyle(document.getElementById("barra2A")).getPropertyValue("bottom");
251 // Conversión.
252 xBarra2AIniA = Number(xBarra2AIniA.slice(0, xBarra2AIniA.length - 2))
253 yBarra2AIniA = Number(yBarra2AIniA.slice(0, yBarra2AIniA.length - 2))
254
255 var delay = 25 - Number(document.getElementById("velSlider").value);
256 var sleepTime = 0;
257
258 var posPinzaA = [120.588, 33.151];
259 var angulosBarrasA = [90, 77];
260
261 var almacenAngsA = [[46, 24], [43, 29], [39, 33], [36, 36]];
262 var tableroAngsA = [[76, 73], [69, 67], [61, 61]];
263
264 var almacenAngP = 40;

```

animacion.js

## B.5. Servidor

El resto de la parte de archivos del servidor han sido desarrollados en diversos lenguajes, como son PHP , HTML , CSS  y JavaScript . En este caso, debido a la extensión, solo se mostrarán los archivos considerados más representativos. Más concretamente se mostrarán los recogidos en el siguiente árbol (de acuerdo con la figura 11 de la página 11).



El contenido de los archivos a continuación se muestra siguiendo el mismo orden en el que aparecen en el árbol anterior.

### Página principal jugar/ai/ (*index.php*)

```

1  <!DOCTYPE HTML>
2
3  <html>
4    <head>
5      <meta charset="UTF-8">
6      <title>Tres en Raya</title>
7      <link rel="stylesheet" type="text/css" href="../../layout.css" />
8      <link rel="stylesheet" type="text/css" href="../../tablero.css" />
9      <link rel="stylesheet" type="text/css" href="../../animacion.css" />
10     <style type="text/css">
11       .data {
12         width: 90%;
13         min-width: 500px;
14         height: 70px;
15         text-align: center;
16       }
17     </style>
18   </head>
19
20
21   <body>
22     <header id="header">
23       <div style="float: left;">
24         <a href="/proyecto/" title="Página principal.">
25           <h1>Tres en Raya</h1>
26         </a>
27       </div>
28       <div style="float: right;">

```

```

29         <a href="/proyecto/jugar/ai/" title="Tú juegas segundo.">
30             <h3>Empieza el brazo</h3>
31         </a>
32     </div>
33     <div style="margin: 0 auto; width: 100px;" title="Juega contra la
    ↪ máquina.">
34         <a href="/proyecto/jugar/">
35             <h1>Jugar</h1>
36         </a>
37     </div>
38 </header>
39
40
41 <?php
42 include("../../navbar.html")
43 ?>
44
45
46 <main id="main">
47     <div class="innertube">
48         <p>Juega al tres en raya, tú vas segundo (también puedes
49             <a href="/proyecto/jugar/per/" title="Empieza tú a jugar.">
50                 empezar tú</a>).
51         </p>
52     </div>
53
54     <?php
55     $dif = 'alta';
56     if (isset($_GET['dif']))
57         $dif = $_GET['dif'];
58
59     if (isset($_GET['movs'])) {
60         $movs = $_GET['movs'];
61         $rama = $_GET['rama'];
62         $nodo = $_GET['nodo'];
63         $sims = $_GET['sims'];
64         $eb = $_GET['eb'];
65
66         if ($dif == 'baja')
67             $eb = 'Easy';
68         else if ($dif == 'media')
69             $eb = 'True';
70
71         $command = 'python3 ../../cgi-bin/main2.py '.$movs.' '.$rama.'
    ↪ '.$nodo.' '.$sims.' '.$eb;
72         $output = exec($command);
73         $outputArray = split(",", $output);
74         $n = sizeof($outputArray);
75
76         $board = $outputArray[$n - 3];
77         $url = '?dif='.$dif.$outputArray[$n - 2];
78     }
79     else {

```

```

80     $rand = rand(1, 4);
81     if ($rand == 1) {
82         $movs = "00";
83         $sims = -1;
84         $board = 'X.....';
85     }
86     else if ($rand == 2) {
87         $movs = "02";
88         $sims = 0;
89         $board = '..X.....';
90     }
91     else if ($rand == 3) {
92         $movs = "20";
93         $sims = 2;
94         $board = '.....X..';
95     }
96     else {
97         $movs = "22";
98         $sims = 1;
99         $board = '.....X';
100    }
101    $url =
102    ↪ ')?dif='.$dif.'&rama=3&nodo=0&sims='.$sims.'&eb=False&movs='.$movs;
103    $outputArray = array('Not ended');
104    $n = sizeof($outputArray);
105 }
106
107 <?php
108 include("../tablero.php");
109 $page = 'ai';
110 include("../controles.php");
111 include("../data.php");
112 ?>
113
114 <div id="animacion">
115     <?php
116     include("../animacion.php");
117     ?>
118
119     <div id="velocidad" align="center">
120         <p>Ajusta la velocidad:</p>
121         <form action="/proyecto/jugar/ai/">
122             <?php
123             echo '<input type="range" id="velSlider" name="vel" min="5"
124             ↪ max="20" ' ;
125             if (isset($_GET['vel']))
126                 echo ' value="'.$_GET['vel'].'" ' ;
127             echo 'step="3" onchange="delay = 25 - this.value;" />';
128             ?>
129             <input type="submit" value="Vuelve a empezar" />
130         </form>
131     </div>

```

```

131
132     <button id="pantCompl" onclick="mostrarAnimacionCompleta();">
133         Pantalla completa
134     </button>
135     <a id="cerrarPant" href="javascript:void(0)"
136         ↪ onclick="cerrarAnimacion();">
137         &times;
138     </a>
139 </div>
140 </main>
141
142 <?php
143 include("../..//footer.html")
144 ?>
145
146 <script src="../animacion.js"></script>
147 <script type="text/javascript">
148     <?php
149     if (isset($_GET['vel'])) {
150         echo 'mostrarAnimacion()';
151         echo 'mostrarAnimacionCompleta()';
152     }
153     else
154         echo 'mostrarControles()';
155     ?>
156 </script>
157
158 </body>
159 </html>

```

index.php

### Controles (*controles.php*)

```

1 <style type="text/css">
2     #endMessage {
3         font-weight: bold;
4         font-size: 30px;
5     }
6     input[type = radio] {
7         margin-left: 20px;
8     }
9 </style>
10
11
12 <div class="control" align="center" id="control">
13     <button id="botonControl" onclick="mostrarControles();">
14         Controles
15     </button>

```

controles.php

```

16 <button id="botonAnimacion" onclick="mostrarAnimacion();">
17     Animación
18 </button>
19
20 <?php
21 if ($outputArray[$n - 1] == "AI wins")
22     echo '<p id="endMessage" style="color: red;">Lo sentimos, has perdido.</p>';
23 else if ($outputArray[$n - 1] == "Tie")
24     echo '<p id="endMessage">Ha sido un empate.</p>';
25 else if ($outputArray[$n - 1] == "User wins")
26     echo '<p id="endMessage" style="color: green;">Enhorabuena, has ganado.</p>';
27 else
28     echo '<br /><br />';
29 ?>
30
31 <p>Vuelve a empezar, vacía el tablero.</p>
32 <?php
33 echo '<a href="/proyecto/jugar/' . $page . '/?dif=';
34 echo $dif . '">';
35 echo '<button>Vacía</button></a>';
36 ?>
37
38 <p>Ajusta la dificultad.</p>
39 <div align="center">
40     <?php
41     echo '<form action="/proyecto/jugar/' . $page . '/">';
42     $same = '<input type="radio" name="dif" value=';
43     $dif = 'alta';
44     if (isset($_GET['dif']))
45         $dif = $_GET['dif'];
46     $values = array('baja', 'media', 'alta');
47     $valuesDisp = array('Fácil', 'Medio', 'Imposible');
48     for ($i = 0; $i < 3; ++$i) {
49         if ($values[$i] == $dif)
50             echo '<input checked type="radio" name="dif" value="' . $values[$i] . '"
51                 ↳ />' . $valuesDisp[$i];
52         else
53             echo '<input type="radio" name="dif" value="' . $values[$i] . '"
54                 ↳ />' . $valuesDisp[$i];
55     }
56     echo '<br /><br />';
57     echo '<input type="submit" value="Actualiza cambios" />';
58     echo '</form>';
59 ?>
60 </div>
61
62 <?php
63 if ($outputArray[$n - 1] == "User wins")
64     echo '
65     <canvas id="canvas"></canvas>
66     <script type="text/javascript" src="../../confeti.js"></script>

```



```

67     <style type="text/css">
68         canvas {
69             position: absolute;
70             top: 0;
71             left: 0;
72             display: block;
73             z-index: -1;
74         }
75     </style>';
76     ?>

```

controles.php

### Tabla datos diversos (*data.php*)

```

1  <div align="center">
2  <br />
3  <h2>Ángulos animación última jugada</h2>
4  <br />
5  <table border="1" class="data">
6      <tr>
7          <th></th>
8          <th>ROTACIÓN (0)</th>
9          <th>BRAZO PRINCIPAL (1)</th>
10         <th>BRAZO SECUNDARIO (2)</th>
11     </tr>
12     <tr>
13         <td colspan="6">Movimiento humano.</td>
14     </tr>
15
16     <?php
17     for ($i = 0; $i < 4; ++$i) {
18         echo '<tr>';
19         echo '<td><b>Mov ' . ($i + 1) . '</b></td>';
20         for ($j = 0; $j < 3; ++$j)
21             echo '<td>'. $outputArray[3*$i + $j + 5] . '</td>';
22         echo '</tr>';
23     }
24     ?>
25     <tr>
26         <td colspan="6">Movimiento brazo.</td>
27     </tr>
28     <?php
29     for ($ip = 4; $ip < 8; ++$ip) {
30         echo '<tr>';
31         echo '<td><b>Mov ' . ($ip - 3) . '</b></td>';
32         for ($jp = 0; $jp < 3; ++$jp)
33             echo '<td>'. $outputArray[3*$ip + $jp + 11] . '</td>';
34         echo '</tr>';
35     }

```

data.php

```

36     ?>
37 </table>
38
39 <br />
40 <ul align="center">
41     <li><b>Mov 1:</b> Posición predeterminada.</li>
42     <li><b>Mov 2:</b> Recogida de pieza del almacén.</li>
43     <li><b>Mov 3:</b> Pieza dejada sobre el tablero.</li>
44     <li><b>Mov 4:</b> Posición predeterminada.</li>
45 </ul>
46
47
48 <br /><br />
49 <h2>Movimiento piezas última jugada</h2>
50 <br />
51 <table border="1" class="data">
52     <tr>
53         <th>PIEZA COGIDA DE (ejes)</th>
54         <th>PIEZA COGIDA DE (número)*</th>
55         <th>PIEZA DEJADA EN (ejes)</th>
56         <th>PIEZA DEJADA EN (tablero)**</th>
57     </tr>
58     <tr>
59         <td colspan="4">Movimiento humano (mueve O's).</td>
60     </tr>
61     <tr>
62         <td>x = <?php echo $outputArray[1]; ?> , y = <?php echo $outputArray[2] ?>
        ↳ [mm]</td>
63         <td> <?php echo 4 - $outputArray[0]; ?> </td>
64         <td>x = <?php echo $outputArray[3]; ?> , y = <?php echo $outputArray[4] ?>
        ↳ [mm]</td>
65         <td>Fila = <?php echo $outputArray[17] + 1; ?> - Columna = <?php echo
        ↳ $outputArray[18] + 1; ?></td>
66     </tr>
67     <tr>
68         <td colspan="4">Movimiento brazo (mueve X's).</td>
69     </tr>
70     <tr>
71         <td>x = <?php echo $outputArray[19]; ?> , y = <?php echo $outputArray[20]
        ↳ ?> [mm]</td>
72         <td> <?php echo 4 - $outputArray[0]; ?> </td>
73         <td>x = <?php echo $outputArray[21]; ?> , y = <?php echo $outputArray[22]
        ↳ ?> [mm]</td>
74         <td>Fila = <?php echo $outputArray[35] + 1; ?> - Columna = <?php echo
        ↳ $outputArray[36] + 1; ?></td>
75     </tr>
76 </table>
77
78 <br />
79 <ul>
80     <li><b>*</b>Número de pieza en el almacén correspondiente (hay 4 huecos en
        ↳ cada almacén).</li>
81     <li><b>*</b>Posición relativa en el tablero (fila y columna).</li>

```

```

82     <li>Los ejes parten del brazo (que está puesto donde está el título de
83         la página, es decir, detrás del tablero). Eje de abscisas horizontal (en
84         pantalla de ordenador) y de ordenadas vertical (ídem).</li>
85     </ul>
86
87     <br /><br /><br />
88 </div>

```

data.php

### Tablero del Tres en Raya (*tablero.php*)

```

1  <table id="board">
2      <?php
3      for ($i = 0; $i < 3; ++$i) {
4          echo '<tr>';
5          for ($j = 0; $j < 3; ++$j) {
6              if ($board[3*$i + $j] != ".")
7                  echo '<td id="' . $board[3*$i + $j] . '>' . $board[3*$i + $j] . '</td>';
8              else if ($outputArray[$n - 1] != "Not ended")
9                  echo '<td></td>';
10             else
11                 echo '<td><a href="' . $url . $i . $j . '><button
12                     ↪ id="tic"><span></span></button></a></td>';
13             }
14             echo '</tr>';
15         }
16     </table>

```

tablero.php

### Pie de página (*footer.html*)

```

1  <footer id="footer">
2      <p align="center" style="font-variant: small-caps; font-size: 1.1em;">
3          <span style="float: left;">Proyecto II</span>
4          David Álvarez - Guillermo Creus
5          <span style="float: right;">
6              Barcelona, mayo de 2019
7          </span>
8      </p>
9  </footer>

```

footer.html

Barra de navegación (*navbar.html*)

```
1 <nav id="nav">
2   <div class="innertube">
3
4     <h1>
5       <a href="/proyecto/proyecto/" title="El proyecto.">
6         Proyecto
7       </a>
8     </h1>
9     <ul>
10      <li>
11        <a href="/proyecto/proyecto/documentacion/" title="La documentación del
12          ↪ proyecto.">
13          Documentación
14        </a>
15      </li>
16      <li>
17        <a href="/proyecto/proyecto/codigo/" title="Todo el código.">
18          El código
19        </a>
20      </li>
21    </ul>
22
23    <h1>
24      <a href="/proyecto/jugar/" title="Juega contra la máquina.">
25        Jugar
26      </a>
27    </h1>
28    <ul>
29      <li>
30        <a href="/proyecto/jugar/per/" title="Empieza tú a jugar.">
31          Empieza tú
32        </a>
33      </li>
34      <li>
35        <a href="/proyecto/jugar/ai/" title="Tú juegas segundo.">
36          Empieza el brazo
37        </a>
38      </li>
39    </ul>
40
41
42    <h1>
43      <a href="/proyecto/presentaciones/" title="Todas las presentaciones.">
44        Presentaciones
45      </a>
46    </h1>
47    <ul>
48      <li>
49        <a href="/proyecto/presentaciones/p1/" title="Presentación inicial.">
```

```
50         Presentación 1
51     </a>
52 </li>
53 <li>
54     <a href="/proyecto/presentaciones/p2/" title="Segunda presentación.">
55         Presentación 2
56     </a>
57 </li>
58 <li>
59     <a href="/proyecto/presentaciones/p3/" title="Presentación final.">
60         Presentación 3
61     </a>
62 </li>
63 </ul>
64 </div>
65
66 <div style="position: absolute; bottom: 0px; left: 8px;">
67     <a href="https://etseib.upc.edu/" target="_blank" title="Escuela Técnica
68     → Superior de Ingeniería Industrial de Barcelona" style="text-decoration:
69     → none;">
70         
71     </a>
72     <a href="https://upc.edu/" target="_blank" title="Universidad Politécnica de
73     → Cataluña" style="text-decoration: none;">
74         
75     </a>
76 </div>
77 </nav>
```

navbar.html