
TRES EN RAYA CON RASPBERRY PI

DAVID ÁLVAREZ

GUILLERMO CREUS



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

BARCELONA, 1 DE MAYO DE 2019

Índice

1. Introducción	1
1.1. Objetivos	1
1.2. Alcance	1
1.3. Antecedentes	1
2. Herramientas e implementación	1
3. Funcionamiento	1
3.1. Tres en Raya	1
3.2. Movimiento	1
4. Planificación y costes	1
5. Resultados y conclusiones	1
6. Bibliografía	1
A. Código completo	4
A.1. Estrategia	4
A.2. Movimiento	14
A.3. Programa principal	21
A.4. Animación	26
A.5. Servidor	34

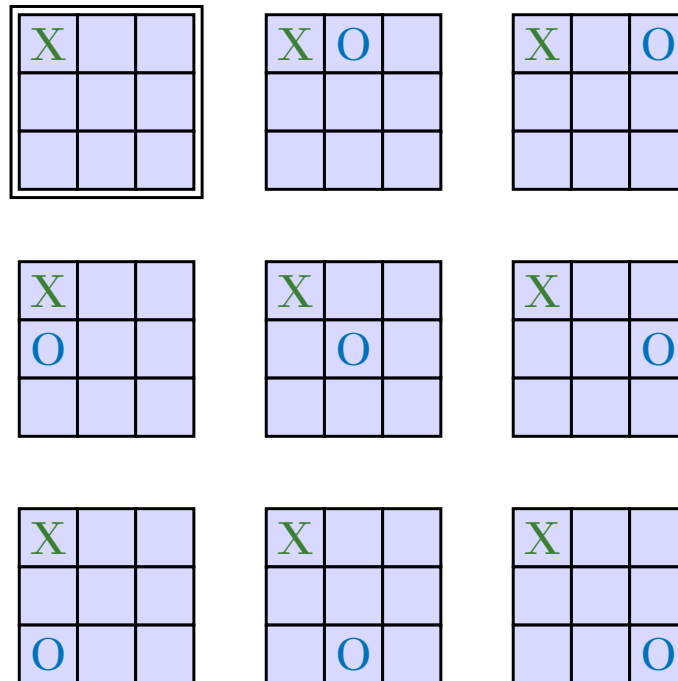


Figura 1: Tableros simétricos equivalentes.

1. Introducción

Este proyecto ha consistido en el desarrollo de un programa capaz de jugar de manera inteligente al juego del Tres en Raya.

1.1. Objetivos

1.2. Alcance

1.3. Antecedentes

2. Herramientas e implementación

3. Funcionamiento

3.1. Tres en Raya

3.2. Movimiento

4. Planificación y costes

5. Resultados y conclusiones

6. Bibliografía

- Tres en Raya

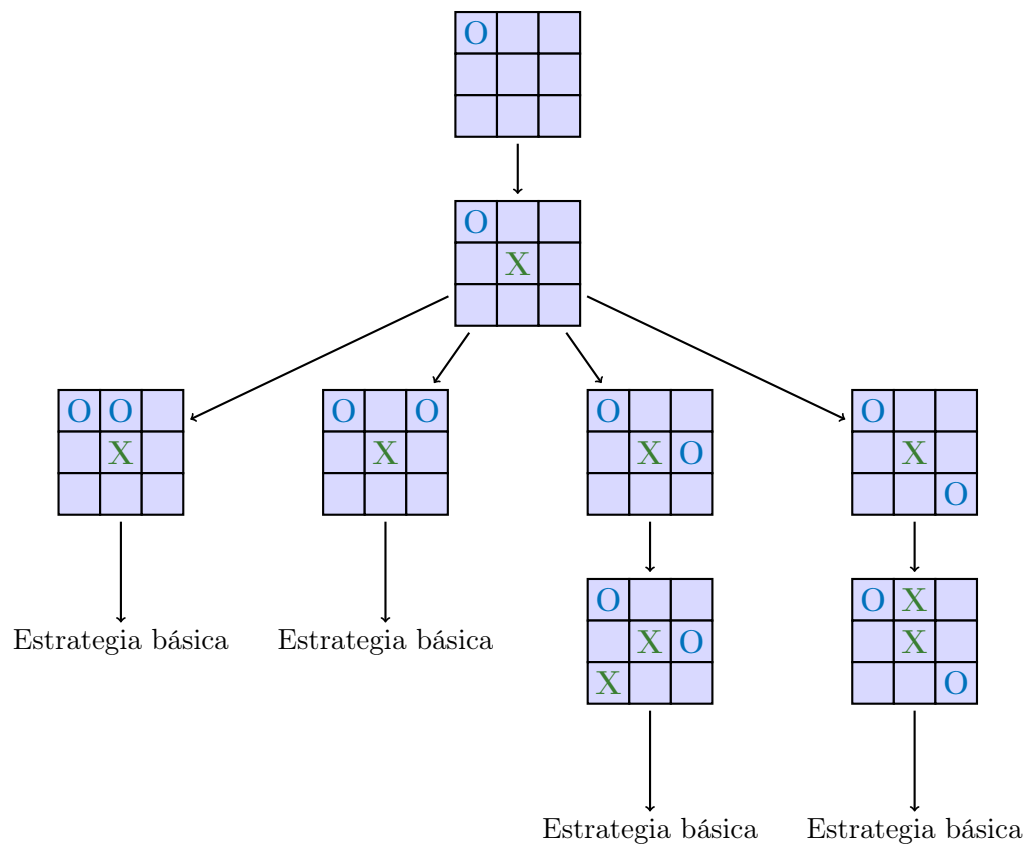


Figura 2: Una rama como ejemplo.

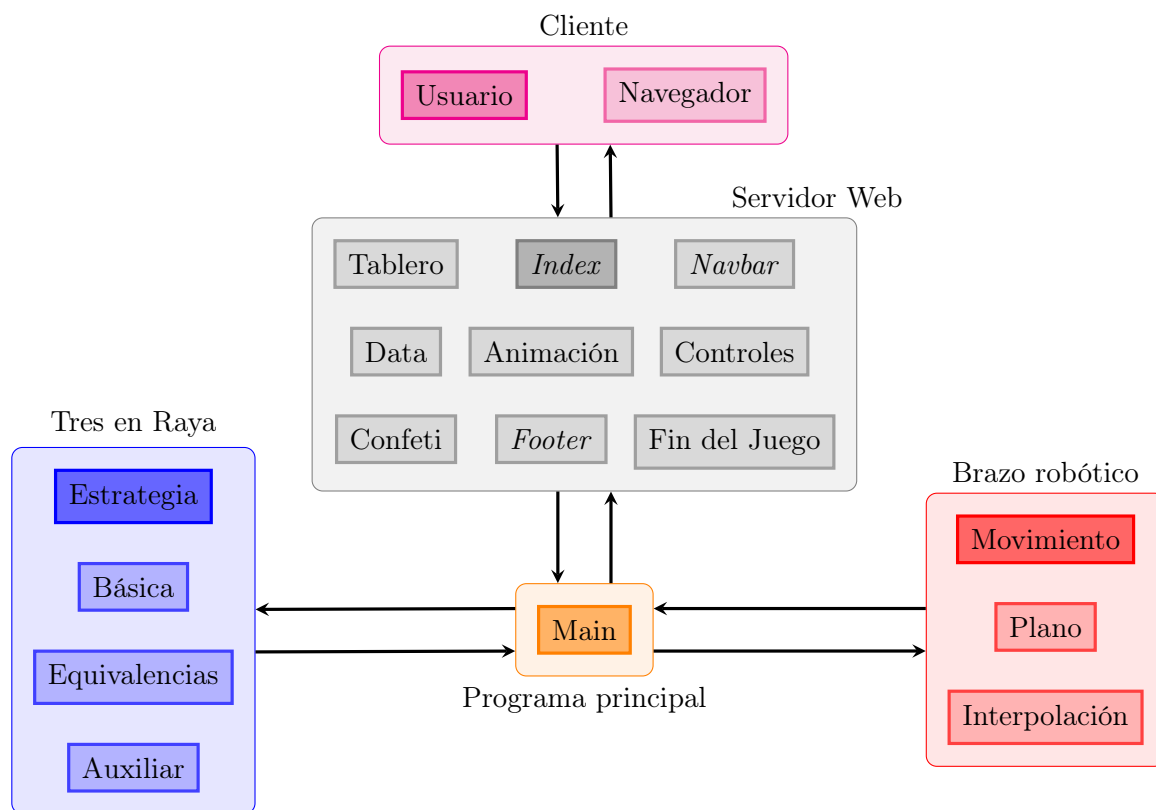


Figura 3: Arquitectura del programa.

- Estrategia: estrategia.py

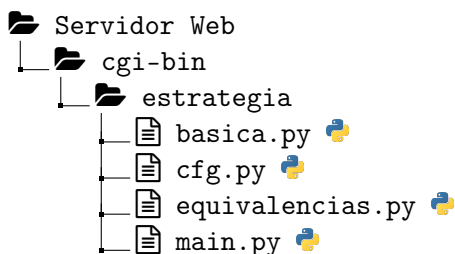
A continuación se muestra el árbol de Se recogen mostrando primero los directorios y después en orden alfabético.

A. Código completo

A continuación se recoge todo el código desarrollado en este proyecto.

A.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 4 de la página 5) 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:
```

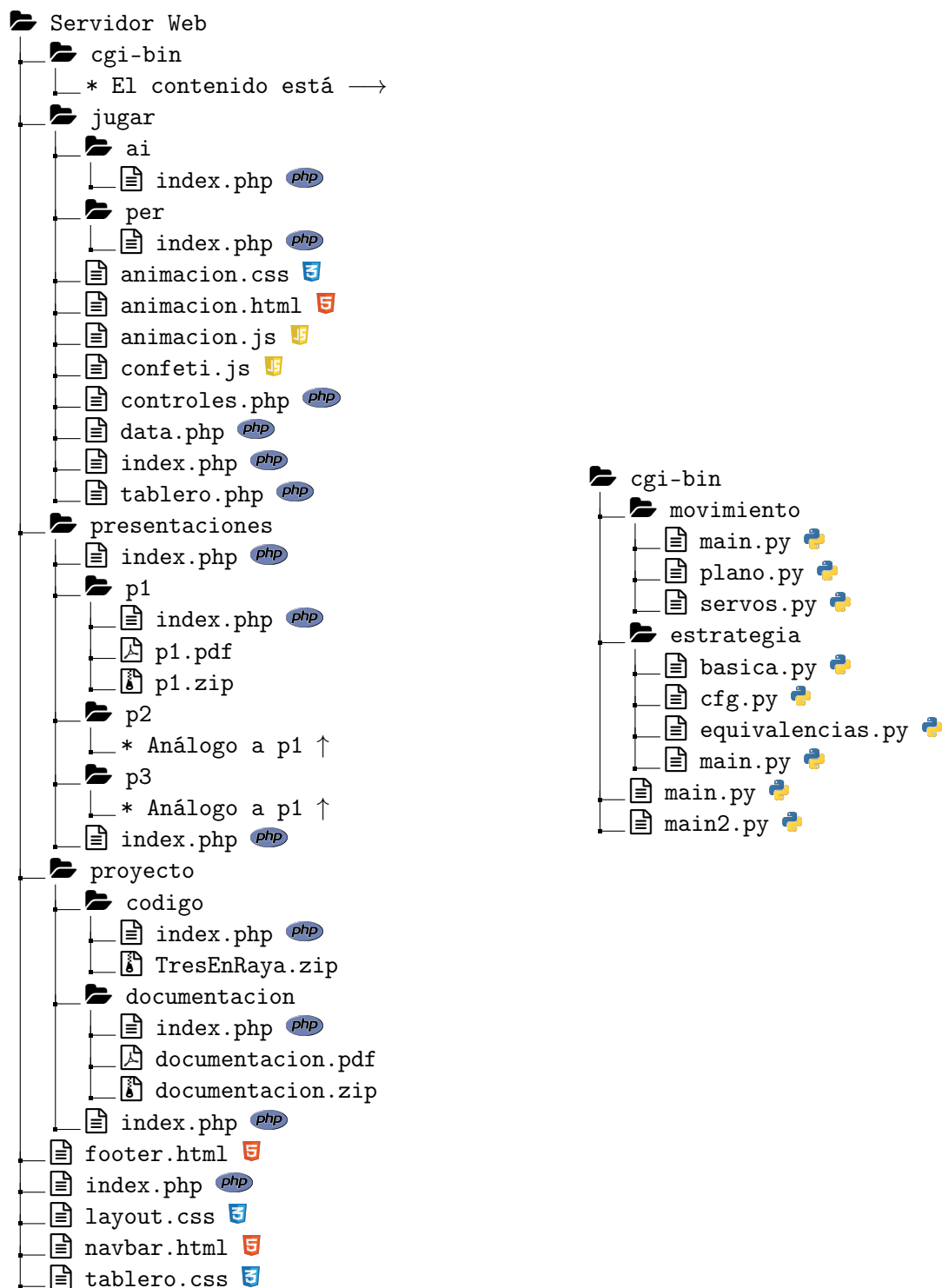


Figura 4: Árbol de directorios.

```

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):
44             ↪ return 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):
51             ↪ return 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):
65             ↪ return 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):
72             ↪ return 1
73
74     return 0
75
76 def check(M,i,j):
77     #Comprobamos posibles jaques

```



```

73     if (i==0 and j==0):
74         if (M[1][0]+M[2][0]==0 or M[0][1]+M[0][2]==0 or M[1][1]+M[2][2]==0):
75             ↪ return 1
76
77     elif (i==0 and j==1):
78         if (M[0][0]+M[0][2]==0 or M[1][1]+M[2][1]==0): return 1
79
80     elif (i==0 and j==2):
81         if (M[0][0]+M[0][1]==0 or M[1][2]+M[2][2]==0 or M[2][0]+M[1][1]==0):
82             ↪ return 1
83
84     elif (i==1 and j==0):
85         if (M[0][0]+M[2][0]==0 or M[1][1]+M[1][2]==0): return 1
86
87     elif (i==1 and j==1):
88         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
89             ↪ M[1][0]+M[1][2]==0): return 1
90
91     elif (i==1 and j==2):
92         if (M[1][0]+M[1][1]==0 or M[0][2]+M[2][2]==0): return 1
93
94     elif (i==2 and j==0):
95         if (M[2][1]+M[2][2]==0 or M[0][0]+M[1][0]==0 or M[1][1]+M[0][2]==0):
96             ↪ return 1
97
98     elif (i==2 and j==1):
99         if (M[2][0]+M[2][2]==0 or M[0][1]+M[1][1]==0): return 1
100
101     elif (i==2 and j==2):
102         if (M[2][0]+M[2][1]==0 or M[0][2]+M[1][2]==0 or M[0][0]+M[1][1]==0):
103             ↪ return 1
104
105     return 0
106
107 def moveRandom(M):
108     """
109     Mover aleatoriamente a una casilla vacía.
110     """
111     movs = []
112     for ip in range(3):
113         for jp in range(3):
114             if M[ip][jp] == -3:
115                 movs.append([ip, jp])
116     if len(movs) > 0:
117         [ip, jp] = movs[random.randint(0, len(movs) - 1)]
118         M[ip][jp] = 1
119         return ip, jp
120     return -1, -1
121
122 def move(M):
123     #Entrada de matriz M 3x3 con 0s (humano), 1s (máquina) y previamente
124     ↪ inicializada en "-3"s (importantente que sea así para que funcione -- por
125     ↪ tema sumas de check_win) --> decide una jugada para la máquina (estrategia
126     ↪ basic)

```

```

121
122     #Compruebo jugadas ganadoras para la máquina
123     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"

```

cfg.py

```

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],
39     → [17], [], [18], [19], [], [20, 21, 22, 23, 24], [25], [], [], [], [26],
40     → [27], [27], [27], [27], [28], [29], [], [], []
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],
53     → [16], [17], [], [18], [19], [20], [21], [22], [23], [], [], [], []
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

```

```

74 nodo = -1
75 sims = []
76 eb = False

```

cfg.py

Equivalencias (*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*)

```

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

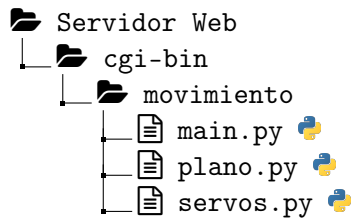
```

```
92     return -1, -1
```

```
main.py
```

A.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 4 de la página 5) 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 R1 = 200
18 R2 = 230
19 ang1 = 0.9*(pi/2)
20 ang2 = 0.7*(pi/2)
21
22 # V1 es un vector con la posición de 4 "X"'s
23 V1 = [[R1*cos(ang1), R1*sin(ang1)], [R1*cos(ang2), R1*sin(ang2)], [R2*cos(ang1),
24 ↪ R2*sin(ang1)], [R2*cos(ang2), R2*sin(ang2)]]
25 # U1 indica el número de pieza a coger en almacén de "X"'s
26 U1 = 0
27
28 # V2 es un vector con la posición de 4 "O"'s
29 V2 = [[R1*cos(-ang1), R1*sin(-ang1)], [R1*cos(-ang2), R1*sin(-ang2)],
30 ↪ [R2*cos(-ang1), R2*sin(-ang1)], [R2*cos(-ang2), R2*sin(-ang2)]]

```



```

29 # U1 indica el número de pieza a coger en almacén de "0"'s
30 U2 = 0
31
32 # Unión de variables del almacén.
33 V = [V1, V2]
34 U = [U1, U2]
35
36 # Posicion del tablero de casillas ancho_tablero*ancho_tablero (mm2)
37 ancho_tablero = 50
38 # Tablero también más separado.
39 # TODO: Revisar esto.
40 x_inicial_t = 80
41
42 fila_1 = [[x_inicial_t + 2*ancho_tablero, -ancho_tablero], [x_inicial_t +
↪ 2*ancho_tablero, 0], [x_inicial_t + 2*ancho_tablero, ancho_tablero]]
43 fila_2 = [[x_inicial_t + ancho_tablero, -ancho_tablero], [x_inicial_t +
↪ ancho_tablero, 0], [x_inicial_t + ancho_tablero, ancho_tablero]]
44 fila_3 = [[x_inicial_t, -ancho_tablero], [x_inicial_t, 0], [x_inicial_t,
↪ ancho_tablero]]
45
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):
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, h0)
77     S[1] = phi1
78     S[2] = phi2

```

```

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

```

```

132     phi1, phi2, phi3, phi4 = verticalMove(r, h0+50) # MODIFICAR
133     S[1] = phi1
134     S[2] = phi2
135     moveServos(S)
136     printServosAngles(S)
137
138     # Dejar el brazo en posición por defecto para permitir ver el tablero.
139     # Esta posición se podría mejorar
140     reset_servos()
141     printServosAngles(S)
142
143
144 def movePiece(i, j, tipo):
145     """
146     Posiciona una pieza (de un tipo) en una posición concreta del tablero.
147     """
148     if tipo == "X":
149         tipo = 0
150     else:
151         tipo = 1
152
153     print("%.1f" % V[tipo][U[tipo]][0], end = ",")
154     print("%.1f" % V[tipo][U[tipo]][1], end = ",")
155     print("%.1f" % tablero[i][j][0], end = ",")
156     print("%.1f" % tablero[i][j][1], end = ",")
157
158     movePieceFromTo(V[tipo][U[tipo]], tablero[i][j])
159     U[tipo] += 1
160
161     print(i, end = ",")
162     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*p1 + p2*p2 - d1*d1 + d2*d2
18     c2 = 2*d2*p1
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
56             ↪ and \
57                 abs(l3*sin(phi2) + l1*sin(phi3) - l2*sin(phi1) + l3*sin(phi4)) < eps
58             ↪ and \
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     
$$px = l2*cos(phi1) + l1*cos(phi4) + l4*cos(35^\circ)$$

66     
$$py = l2*sin(phi1) + l1*sin(phi4) + h$$


```

```

66     Donde l1, l2, l4 son parámetros del brazo; px, py y h son los parámetros de
67     la función y phi1, phi4 son los ángulos a obtener.
68     Solo devuelve una solución.
69     """
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             ↪ \
87             abs(l2*sin(phi1) + l1*sin(phi4) + h - py) < eps and \
88             phi1 >= 0 and phi1 <= pi and phi4 <= pi/2 and phi4 >= -pi/2:
89                 return phi1, phi4
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 l4 = 42
124 l5 = 68.81
125 dx = 34.4
126 dy = 24.22
127 # Consideraremos la altura como un parámetro más.
128 h = 0

```

plano.py

Servomotores (*servos.py*)

```

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     """

```

```

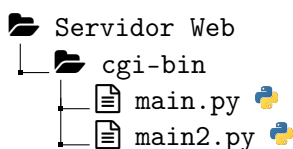
35     Devuelve información de los ángulos de los servos.
36     """
37     for i in range(6):
38         # El 3 no es un servo.
39         if i != 3:
40             print(floor(rad2Deg(S[i])), end = ",")
41
42
43 def moveServo(servo, angle):
44     """
45     Mueve el servo a un determinado ángulo (en radianes) de manera progresiva.
46     """
47     global Sp
48     angle = rad2Deg(angle)
49     steps = 50
50     time = 0 # Cambiar este valor al hacer la conexión real con el brazo.
51     timeStep = time/steps
52     angleIni = Sp[servo]
53     h = (angle - angleIni)/steps
54     for i in range(steps):
55         angleIni = angleIni + h
56         # PARA QUE NO HAYA PROBLEMAS TRUNCO!!!!!!!!
57         # kit.servo[servo].angle = floor(angleIni)
58         sleep(timeStep)
59
60     Sp[servo] = floor(angleIni)
61
62
63 def moveServos(angles):
64     """
65     Mueve los servos simultáneamente a los ángulos dados (en radianes).
66     """
67     Thread(target=moveServo, args=[0, angles[0]]).start()
68     Thread(target=moveServo, args=[1, angles[1]]).start()
69     Thread(target=moveServo, args=[2, angles[2]]).start()
70     Thread(target=moveServo, args=[4, angles[4]]).start()
71     Thread(target=moveServo, args=[5, angles[5]]).start()

```

servos.py

A.3. Programa principal

Aquí se recoge todo el código desarrollado en Python 🐍 encargado de fusionar/coordinar el código de las secciones anteriores (A.1 y A.2). Este código en el servidor se encuentra (de acuerdo con la figura 4 de la página 5) 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         movs = []
48         for i in range(int(len(movsStr)/2)):
49             movs.append([int(movsStr[2*i]), int(movsStr[2*i + 1])])
50

```



```

51     for i in range(len(movs) - 1):
52         if i%2 == 0:
53             cfg.board[movs[i][0]][movs[i][1]] = 0
54         else:
55             cfg.board[movs[i][0]][movs[i][1]] = 1
56
57     cfg.rama = int(sys.argv[2])
58     cfg.nodo = int(sys.argv[3])
59     simsStr = sys.argv[4]
60     for i in range(len(simsStr)):
61         if simsStr[i] != "-":
62             if i != 0 and simsStr[i - 1] != "-":
63                 cfg.sims.append(int(simsStr[i]))
64             elif i != 0 and simsStr[i - 1] == "-":
65                 cfg.sims.append(int(simsStr[i-1:i+1]))
66             else:
67                 cfg.sims.append(int(simsStr[i]))
68
69     cfg.boardInt = simetriaMultiple(cfg.board, cfg.sims)
70
71     if sys.argv[5] == "False":
72         cfg.eb = False
73     else:
74         cfg.eb = True
75
76     actualiza(movs[len(movs) - 1][0], movs[len(movs) - 1][1])
77
78     return movs[len(movs) - 1][0], movs[len(movs) - 1][1]
79
80
81 def nextUrl(i, j):
82     """
83     Crea e imprime la siguiente dirección web. También el tablero.
84     """
85     url = "&rama=" + str(cfg.rama)
86     url += "&nodo=" + str(cfg.nodo)
87     url += "&sims="
88     for sim in cfg.sims:
89         url += str(sim)
90     url += "&eb=" + str(cfg.eb)
91     url += "&movs=" + sys.argv[1] + str(i) + str(j)
92
93     return url
94
95
96 def printData(i, j):
97     """
98     Imprime por pantalla diferentes datos.
99     """
100     data = board2Str(cfg.board) + ","
101     data += nextUrl(i, j)
102     print(data, end = ",")
103

```

```

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

```

main.py

Programa principal 2 (*main2.py*)

```

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):
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     movs = []
48     for i in range(int(len(movsStr)/2)):
49         movs.append([int(movsStr[2*i]), int(movsStr[2*i + 1])])
50
51     for i in range(len(movs) - 1):
52         if i%2 == 0:
53             cfg.board[movs[i][0]][movs[i][1]] = 1
54         else:
55             cfg.board[movs[i][0]][movs[i][1]] = 0
56
57     cfg.rama = int(sys.argv[2])
58     cfg.nodo = int(sys.argv[3])
59     simsStr = sys.argv[4]
60     for i in range(len(simsStr)):
61         if simsStr[i] != "-":
62             if i != 0 and simsStr[i - 1] != "-":
63                 cfg.sims.append(int(simsStr[i]))
64             elif i != 0 and simsStr[i - 1] == "-":
65                 cfg.sims.append(int(simsStr[i-1:i+1]))
66             else:
67                 cfg.sims.append(int(simsStr[i]))
68
69     cfg.boardInt = simetriaMultiple(cfg.board, cfg.sims)
70
71     if sys.argv[5] == "False":
72         cfg.eb = False
73     else:
74         cfg.eb = True
75
76     if len(movs) == 1:
77         cfg.rama = -1
78
79     actualiza(movs[len(movs) - 1][0], movs[len(movs) - 1][1])
80
81     return movs[len(movs) - 1][0], movs[len(movs) - 1][1]
82

```




```

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

```

main2.py

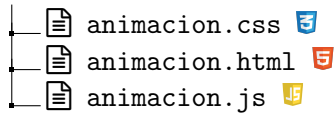
A.4. Animación

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 4 de la página 5) en el siguiente directorio:

```

└─ Servidor Web
   └─ jugar

```



Estilos animación (*animacion.css*)

```
animacion.css
1  #alzado {
2      position: absolute;
3      left: 25px;
4      top: 250px;
5      transform-origin: 0 0;
6      transform: scale(.65, .65);
7  }
8
9  #base {
10     width: 50px;
11     height: 50px;
12     position: absolute;
13     bottom: 55px;
14     left: 15px;
15     background-color: #FF9900;
16 }
17
18 #tierra {
19     width: 80px;
20     height: 65px;
21     position: absolute;
22     bottom: 0;
23     background-color: #CC0000;
24 }
25
26
27 /* Barra 1 */
28 #barra1Cont {
29     position: absolute;
30     bottom: 77.5px;
31     left: 40px;
32     transform-origin: 0 7.5px;
33 }
34
35 #barra1 {
36     width: 250px;
37     height: 15px;
38     background-color: #CC0000;
39 }
40
41
42 /* Barra 2 */
43 #barra2 {
44     width: 250px;
45     height: 15px;
46     position: absolute;
```

```
47     bottom: 77.5px;
48     left: 290px;
49     background-color: #CC0000;
50     transform-origin: 0 7.5px;
51 }
52
53
54 /* Pinza */
55 #pinzaCont {
56     position: absolute;
57     bottom: 45px;
58     left: 515px;
59 }
60
61 #pinza {
62     width: 10px;
63     height: 40px;
64     position: absolute;
65     top: -40px;
66     left: 20px;
67     background-color: #CC0000;
68 }
69
70 #piezaPinza {
71     width: 30px;
72     height: 20px;
73     position: absolute;
74     left: 10px;
75     top: 12px;
76     background-color: #2B6587;
77     display: none;
78 }
79
80 #tenaza {
81     width: 50px;
82     height: 10px;
83     position: absolute;
84     top: 390px;
85     left: 530px;
86     background-color: #0000FF;
87 }
88
89 #tenazas_h {
90     width: 50px;
91     height: 10px;
92     position: absolute;
93     top: 0;
94     left: 0;
95     background-color: #0000FF;
96 }
97
98 #tenazas_v1 {
99     width: 10px;
```

```
100     height: 40px;
101     position: absolute;
102     top: 0;
103     left: 0;
104     background-color: #0000FF;
105     transform-origin: 0 0;
106 }
107
108 #tenazas_v2 {
109     width: 10px;
110     height: 40px;
111     position: absolute;
112     top: 0;
113     left: 40px;
114     background-color: #0000FF;
115 }
116
117
118 /* Articulaciones */
119 #articulacion {
120     width: 30px;
121     height: 30px;
122     border-radius: 50%;
123     background-color: Black;
124 }
125
126 #articulacionPos1 {
127     position: absolute;
128     left: 235px;
129     bottom: -7.5px;
130 }
131
132 #articulacionPos2 {
133     position: absolute;
134     left: 10px;
135     bottom: 25px;
136 }
137
138
139 /* Almacén de piezas. */
140 #almacen * {
141     width: 30px;
142     height: 20px;
143     position: absolute;
144     left: 406px;
145     background-color: #2b6587;
146 }
147
148 #almacen {
149     opacity: 0.1;
150     transition: opacity 2s;
151     transition-timing-function: ease-in;
152 }
```

```
153
154 #almacenPieza0 {
155     bottom: 72px;
156 }
157
158 #almacenPieza1 {
159     bottom: 48px;
160 }
161
162 #almacenPieza2 {
163     bottom: 24px;
164 }
165
166 #almacenPieza3 {
167     bottom: 0;
168 }
169
170
171 /* Piezas tablero. */
172 #tablero {
173     opacity: 0.1;
174     transition: opacity 2s;
175     transition-timing-function: easy-in;
176 }
177
178 #tablero * {
179     width: 30px;
180     height: 20px;
181     position: absolute;
182     bottom: 0;
183     background-color: #2B6587;
184     display: none;
185 }
186
187 #tableroPieza0 {
188     left: 170px;
189 }
190
191 #tableroPieza1 {
192     left: 215px;
193 }
194
195 #tableroPieza2 {
196     left: 260px;
197 }
198
199
200 /* Control de velocidad. */
201 #velocidad {
202     position: absolute;
203     top: 425px;
204     left: 375px;
```


205 }

animacion.css

Animación HTML (*animacion.html*)

animacion.html

```
1 <div id="alzado">
2   <div id="barra2"></div>
3
4   <div id="barra1Cont">
5     <div id="barra1"></div>
6     <div id="articulacionPos1">
7       <div id="articulacion"></div>
8     </div>
9   </div>
10
11   <div id="base"></div>
12   <div id="tierra"></div>
13
14   <div id="pinzaCont">
15     <div id="pinza"></div>
16     <div id="tenazas_h"></div>
17     <div id="tenazas_v1"></div>
18     <div id="tenazas_v2"></div>
19     <div id="articulacionPos2">
20       <div id="articulacion"></div>
21     </div>
22     <div id="piezaPinza"></div>
23   </div>
24
25   <div id="almacen">
26     <div id="almacenPieza0"></div>
27     <div id="almacenPieza1"></div>
28     <div id="almacenPieza2"></div>
29     <div id="almacenPieza3"></div>
30   </div>
31
32   <div id="tablero">
33     <div id="tableroPieza0"></div>
34     <div id="tableroPieza1"></div>
35     <div id="tableroPieza2"></div>
36   </div>
37
38 </div>
```

animacion.html

Animación JS (*animacion.js*)

```

1  function sleep(delay) {
2      return new Promise(resolve => setTimeout(resolve, delay));
3  }
4
5
6  async function move_barra1(phi0, phif) {
7      var phi = phi0;
8      var inc = 1;
9      if (phi0 > phif)
10         inc = -1;
11
12     while (phi != phif) {
13         await sleep(delay);
14         phi += inc;
15
16         barra1Cont.style.transform = "rotate(-" + phi.toString() + "deg)";
17
18         var xBarra2 = xBarra2Ini + length1*(Math.cos(phi*Math.PI/180) - 1);
19         var yBarra2 = yBarra2Ini + length1*Math.sin(phi*Math.PI/180);
20         barra2.style.left = xBarra2.toString() + "px";
21         barra2.style.bottom = yBarra2.toString() + "px";
22
23         var xPinza = posPinza[0] + length1*(Math.cos(phi*Math.PI/180) -
24             ↪ Math.cos(phi0*Math.PI/180));
25         var yPinza = posPinza[1] + length1*(Math.sin(phi*Math.PI/180) -
26             ↪ Math.sin(phi0*Math.PI/180));
27         pinzaCont.style.left = xPinza.toString() + "px";
28         pinzaCont.style.bottom = yPinza.toString() + "px";
29     }
30
31     posPinza = [xPinza, yPinza];
32 }
33
34 async function move_barra2(phi0, phif) {
35     var phi = phi0;
36     var inc = 1;
37     if (phi0 > phif)
38         inc = -1;
39
40     var transPinzaIni = document.getElementById("pinzaCont").style.transform;
41     while (phi != phif) {
42         await sleep(delay);
43         phi += inc;
44
45         barra2.style.transform = "rotate(" + phi.toString() + "deg)";
46
47         var xPinza = posPinza[0] + length2*(Math.cos(phi*Math.PI/180) -
48             ↪ Math.cos(phi0*Math.PI/180));
49         var yPinza = posPinza[1] + length2*(-Math.sin(phi*Math.PI/180) +
50             ↪ Math.sin(phi0*Math.PI/180));

```

```

48     pinzaCont.style.left = xPinza.toString() + "px";
49     pinzaCont.style.bottom = yPinza.toString() + "px";
50 }
51
52     posPinza = [xPinza, yPinza];
53 }
54
55
56 async function move_barras(phi1, phi2) {
57     var sleepTime1 = 1.35*delay*Math.abs(phi1 - angulosBarras[0]);
58     var sleepTime2 = 1.35*delay*Math.abs(phi2 - angulosBarras[1]);
59     sleepTime = 1.1*(sleepTime1 + sleepTime2);
60
61     move_barra1(angulosBarras[0], phi1);
62     await sleep(sleepTime1);
63     move_barra2(angulosBarras[1], phi2);
64     await sleep(sleepTime2);
65     angulosBarras = [phi1, phi2];
66 }
67
68
69 async function move_piece(noAlmacen, noTablero) {
70     // Ir al almacén.
71     tablero.style.opacity = "0.1";
72     move_barras(angulosBarras[0] - 1, 20);
73     await sleep(sleepTime);
74
75     almacen.style.opacity = "1";
76     move_barras(almacenAngs[noAlmacen][0], almacenAngs[noAlmacen][1]);
77     await sleep(sleepTime);
78     document.getElementById("piezaPinza").style.display = "block";
79     document.getElementById("almacenPieza" + noAlmacen).style.display = "none";
80
81     // Ir al tablero.
82     almacen.style.opacity = "0.1";
83     tablero.style.opacity = "1";
84     move_barras(tableroAngs[noTablero][0], tableroAngs[noTablero][1]);
85     await sleep(sleepTime);
86     piezaPinza.style.display = "none";
87     document.getElementById("tableroPieza" + noTablero).style.display = "block";
88 }
89
90
91 async function reset() {
92     move_barras(90, 73);
93     await sleep(sleepTime);
94 }
95
96
97 // Función para probar.
98 async function move() {
99     await sleep(500);
100    reset();

```





```

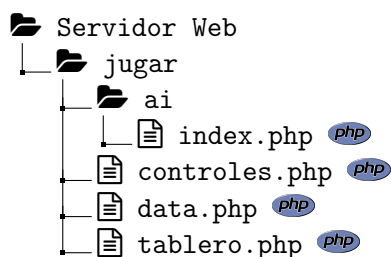
101     await sleep(280*delay + 500);
102
103     move_piece(0, 0);
104     await sleep(250*delay);
105
106     move_piece(1, 2);
107     await sleep(250*delay);
108
109     move_piece(2, 1);
110     await sleep(250*delay);
111
112     reset();
113 }
114
115
116 var length1 = document.getElementById("barra1").offsetWidth;
117 var length2 = document.getElementById("barra2").offsetWidth;
118
119 var xBarra2Ini =
120   ↪ getComputedStyle(document.getElementById("barra2")).getPropertyValue("left");
121 var yBarra2Ini =
122   ↪ getComputedStyle(document.getElementById("barra2")).getPropertyValue("bottom");
123 // Conversión.
124 xBarra2Ini = Number(xBarra2Ini.slice(0, xBarra2Ini.length - 2))
125 yBarra2Ini = Number(yBarra2Ini.slice(0, yBarra2Ini.length - 2))
126
127 var delay = 25 - Number(document.getElementById("velSlider").value);
128 var sleepTime = 0;
129
130 var posPinza = [515, 45];
131 var angulosBarras = [0, 0];
132
133 var almacenAngs = [[48, 31], [45, 35], [42, 39], [39, 42]];
134 var tableroAngs = [[71, 75], [67, 68], [61, 63]];

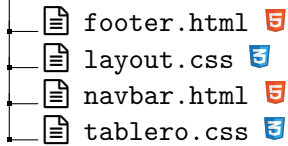
```

animacion.js

A.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 4 de la página 5).





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*)

```

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      <style type="text/css">
10     .data {
11       width: 90%;
12       min-width: 500px;
13       height: 70px;
14       text-align: center;
15     }
16   </style>
17 </head>
18
19
20 <body>
21   <header id="header">
22     <div style="float: left;"><a href="/proyecto/"><h1>Tres en
23   ↪ Raya</h1></a></div>
24     <div style="float: right;"><a href="/proyecto/jugar/ai/"><h3>Empieza el
25   ↪ brazo</h3></a></div>
26     <div style="margin: 0 auto; width: 100px;"><a
27   ↪ href="/proyecto/jugar/"><h1>Jugar</h1></a></div>
28   </header>
29
30   <?php
31   include("../../navbar.html")
32   ?>
33
34   <main id="main">
35     <div class="innertube">
36       <p>Juega al tres en raya, tú vas segundo (también puedes
37       <a href="/proyecto/jugar/per/" title="Empieza tú a jugar.">
38       empezar tú</a>).

```

```

39     </div>
40
41     <?php
42     $dif = 'alta';
43     if (isset($_GET['dif']))
44         $dif = $_GET['dif'];
45
46     if (isset($_GET['movs'])) {
47         $movs = $_GET['movs'];
48         $rama = $_GET['rama'];
49         $nodo = $_GET['nodo'];
50         $sims = $_GET['sims'];
51         $eb = $_GET['eb'];
52
53         if ($dif == 'baja')
54             $eb = 'Easy';
55         else if ($dif == 'media')
56             $eb = 'True';
57
58         $command = 'python3 ../../cgi-bin/main2.py '.$movs.' '.$rama.'
59             ↪ '.$nodo.' '.$sims.' '.$eb;
60         $output = exec($command);
61         $outputArray = split(" ", $output);
62         $n = sizeof($outputArray);
63
64         $board = $outputArray[$n - 3];
65         $url = '?dif='.$dif.$outputArray[$n - 2];
66     }
67     else {
68         $rand = rand(1, 4);
69         if ($rand == 1) {
70             $movs = "00";
71             $sims = -1;
72             $board = 'X.....';
73         }
74         else if ($rand == 2) {
75             $movs = "02";
76             $sims = 0;
77             $board = '..X.....';
78         }
79         else if ($rand == 3) {
80             $movs = "20";
81             $sims = 2;
82             $board = '.....X..';
83         }
84         else {
85             $movs = "22";
86             $sims = 1;
87             $board = '.....X';
88         }
89         $url =
90             ↪ '?dif='.$dif.'&rama=3&nodo=0&sims='.$sims.'&eb=False&movs='.$movs;
91         $outputArray = array('Not ended');

```

```

90         $n = sizeof($outputArray);
91     }
92     ?>
93
94     <?php
95     include("../tablero.php");
96     $page = 'ai';
97     include("../controles.php");
98     include("../data.php");
99     ?>
100 </main>
101
102
103 <?php
104 include("../../footer.html")
105 ?>
106 </body>
107 </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     <?php
14     if ($outputArray[$n - 1] == "AI wins")
15         echo '<p id="endMessage" style="color: red;">Lo sentimos, has
16         perdido.</p>';
17     else if ($outputArray[$n - 1] == "Tie")
18         echo '<p id="endMessage">Ha sido un empate.</p>';
19     else if ($outputArray[$n - 1] == "User wins")
20         echo '<p id="endMessage" style="color: green;">Enhorabuena, has
21         ganado.</p>';
22     else
23         echo '<br />';
24     ?>
25
26 <p>Vuelve a empezar, vacía el tablero.</p>
27 <?php

```

```

26     echo '<a href="/proyecto/jugar/'.$page.'/?dif=';
27     echo $dif.'">';
28     echo '<button>Vacía</button></a>';
29     ?>
30
31     <p>Ajusta la dificultad.</p>
32     <div align="center">
33         <?php
34         echo '<form action="/proyecto/jugar/'.$page.'/">';
35         $same = '<input type="radio" name="dif" value=';
36         $dif = 'alta';
37         if (isset($_GET['dif']))
38             $dif = $_GET['dif'];
39         $values = array('baja', 'media', 'alta');
40         $valuesDisp = array('Fácil', 'Medio', 'Imposible');
41         for ($i = 0; $i < 3; ++$i) {
42             if ($values[$i] == $dif)
43                 echo '<input checked type="radio" name="dif"
44                     ↪ value="'.$values[$i].'" />'.$valuesDisp[$i];
45             else
46                 echo '<input type="radio" name="dif" value="'.$values[$i].'"
47                     ↪ />'.$valuesDisp[$i];
48         }
49         echo '<br /><br />';
50         echo '<input type="submit" value="Actualiza cambios" />';
51         echo '</form>';
52         ?>
53     </div>
54
55     <?php
56     if ($outputArray[$n - 1] == "User wins")
57         echo '
58             <canvas id="canvas"></canvas>
59             <script type="text/javascript" src="../confeti.js"></script>
60             <style type="text/css">
61                 canvas {
62                     position: absolute;
63                     top: 0;
64                     left: 0;
65                     display: block;
66                     z-index: -1;
67                 }
68             </style>';
69     ?>

```

controles.php

Tabla datos diversos (*data.php*)

```

1  <div align="center">
2    <h2>Ángulos servos última jugada</h2>
3    <br />
4    <table border="1" class="data">
5      <tr>
6        <th></th>
7        <th>ROTACIÓN (0)</th>
8        <th>BRAZO PRINCIPAL (1)</th>
9        <th>BRAZO SECUNDARIO (2)</th>
10       <th>PINZA ROTACIÓN (4)</th>
11       <th>PINZA APERTURA (5)</th>
12     </tr>
13     <tr>
14       <td colspan="6">Movimiento humano.</td>
15     </tr>
16
17     <?php
18     for ($i = 0; $i < 9; ++$i) {
19       echo '<tr>';
20       echo '<td><b>Mov ' . $i . '</b></td>';
21       for ($j = 0; $j < 5; ++$j)
22         echo '<td>'. $outputArray[5*$i + $j + 4] . '</td>';
23       echo '</tr>';
24     }
25     ?>
26     <tr>
27       <td colspan="6">Movimiento brazo.</td>
28     </tr>
29     <?php
30     for ($ip = 9; $ip < 18; ++$ip) {
31       echo '<tr>';
32       $ipp = $ip - 9;
33       echo '<td><b>Mov ' . $ipp . '</b></td>';
34       for ($jp = 0; $jp < 5; ++$jp)
35         echo '<td>'. $outputArray[5*$ip + $jp + 10] . '</td>';
36       echo '</tr>';
37     }
38     ?>
39   </table>
40
41   <br />
42   <ul align="center">
43     <li><b>Mov 0:</b> Posición predeterminada.</li>
44     <li><b>Mov 1:</b> Pinza sobre la pieza a coger en el almacén (a la altura
45   ↪ justa para que al cerrar la pinza coja la pieza).</li>
46     <li><b>Mov 2:</b> Justo después de cerrar la pinza.</li>
47     <li><b>Mov 3:</b> Subir la pinza para que no se choque.</li>
48     <li><b>Mov 4:</b> Sobre la posición final.</li>
49     <li><b>Mov 5:</b> Pinza bajada sobre posición final.</li>
50     <li><b>Mov 6:</b> Pieza soltada en la posición final (con las pinzas
51   ↪ abiertas).</li>

```

```

50     <li><b>Mov 7:</b> Subir la pinza para que no se choque.</li>
51     <li><b>Mov 8:</b> Posición predeterminada.</li>
52 </ul>
53
54
55 <br /><br /><br /><br />
56 <h2>Movimiento piezas última jugada</h2>
57 <br />
58 <table border="1" class="data">
59     <tr>
60         <th>PIEZA COGIDA DE (ejes)</th>
61         <th>PIEZA COGIDA DE (número)*</th>
62         <th>PIEZA DEJADA EN (ejes)</th>
63         <th>PIEZA DEJADA EN (tablero)**</th>
64     </tr>
65     <tr>
66         <td colspan="4">Movimiento humano (mueve O's).</td>
67     </tr>
68     <tr>
69         <td>x = <?php echo $outputArray[0]; ?> , y = <?php echo $outputArray[1]
↪ <?> [mm]</td>
70         <td>Falta, pero va por orden.</td>
71         <td>x = <?php echo $outputArray[2]; ?> , y = <?php echo $outputArray[3]
↪ <?> [mm]</td>
72         <td>Fila = <?php echo $outputArray[49] + 1; ?> - Columna = <?php echo
↪ $outputArray[50] + 1; ?></td>
73     </tr>
74     <tr>
75         <td colspan="4">Movimiento brazo (mueve X's).</td>
76     </tr>
77     <tr>
78         <td>x = <?php echo $outputArray[51]; ?> , y = <?php echo
↪ $outputArray[52] ?> [mm]</td>
79         <td>Falta, pero va por orden.</td>
80         <td>x = <?php echo $outputArray[53]; ?> , y = <?php echo
↪ $outputArray[54] ?> [mm]</td>
81         <td>Fila = <?php echo $outputArray[100] + 1; ?> - Columna = <?php echo
↪ $outputArray[101] + 1; ?></td>
82     </tr>
83 </table>
84
85 <br />
86 <ul>
87     <li><b>*</b>Número de pieza en el almacén correspondiente (hay 4 huecos en
↪ cada almacén).</li>
88     <li><b>***</b>Posición relativa en el tablero (fila y columna).</li>
89     <li>Los ejes parten del brazo (que está puesto donde está el título de
90         la página, es decir, detrás del tablero). Eje de abscisas horizontal (en
91         pantalla de ordenador) y de ordenadas vertical (ídem).</li>
92 </ul>
93
94 <br /><br /><br />

```

```
95 </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">
3       <span style="float: left">Proyecto II</span>
4       David Álvarez, Guillermo Creus
5       <span style="float: right">
6           <a href="https://etseib.upc.edu/" target="_blank" title="Escuela Técnica
7               ↪ Superior de Ingeniería Industrial de Barcelona">
8               ETSEIB
9           </a>
10          -
11          <a href="https://upc.edu/" target="_blank" title="Universidad
12              ↪ Politécnica de Cataluña">
13              UPC
14          </a>
15      </span>
16  </p>
17 </footer>

```

```
footer.html
```

Estilos principal (*layout.css*)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
12          ↳ del 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      <li>
40        <a href="/proyecto/error/" title="Ve una partida.">
41          Ver partida
42        </a>
43      </li>
```

```
44     </ul>
45
46
47     <h1>
48         <a href="/proyecto/presentaciones/" title="Todas las presentaciones.">
49             Presentaciones
50         </a>
51     </h1>
52     <ul>
53         <li>
54             <a href="/proyecto/presentaciones/p1/" title="Presentación inicial.">
55                 Presentación 1
56             </a>
57         </li>
58         <li>
59             <a href="/proyecto/presentaciones/p2/" title="Segunda presentación.">
60                 Presentación 2
61             </a>
62         </li>
63         <li>
64             <a href="/proyecto/presentaciones/p3/" title="Presentación final.">
65                 Presentación 3
66             </a>
67         </li>
68     </ul>
69
70 </div>
71 </nav>
```

navbar.html

Estilos tablero (*tablero.css*)

```
1  #board {
2      border-collapse: collapse;
3      margin: -15px 150px 0 0;
4      float: right;
5      background-color: #D2E0F9;
6  }
7
8  #board td {
9      height: 110px;
10     width: 110px;
11     padding: 0;
12     border: 4px solid Black;
13     text-align: center;
14     font-size: 70px;
15     font-weight: bold;
16 }
17
```

tablero.css

```
18 #X {
19     color: #336600;
20 }
21
22 #O {
23     color: #3862E0;
24 }
25
26 #tic {
27     width: 100px;
28     height: 100px;
29     background-color: #D2E0F9;
30     border: 0px;
31     cursor: pointer;
32 }
33
34 #tic span {
35     text-align: center;
36     font-weight: bold;
37     font-size: 70px;
38 }
39
40 #tic span:after {
41     content: 'O';
42     opacity: 0;
43     transition: 1s;
44 }
45
46 #tic:hover span:after {
47     opacity: .75;
48 }
49
50
51 /* Actualizar con el tamaño. */
52 @media only all
53 and (max-width: 900px) {
54     #board {
55         margin: 0 auto;
56         float: none;
57     }
58 }
59
60 @media only all
61 and (max-width: 1050px)
62 and (min-width: 899px){
63     #board {
64         margin: -15px 50px 0 0;
65     }
66 }
67
68 @media only all
69 and (max-width: 1200px)
70 and (min-width: 1049px){
```

```
71     #board {  
72         margin: -15px 100px 0 0;  
73     }  
74 }
```

tablero.css