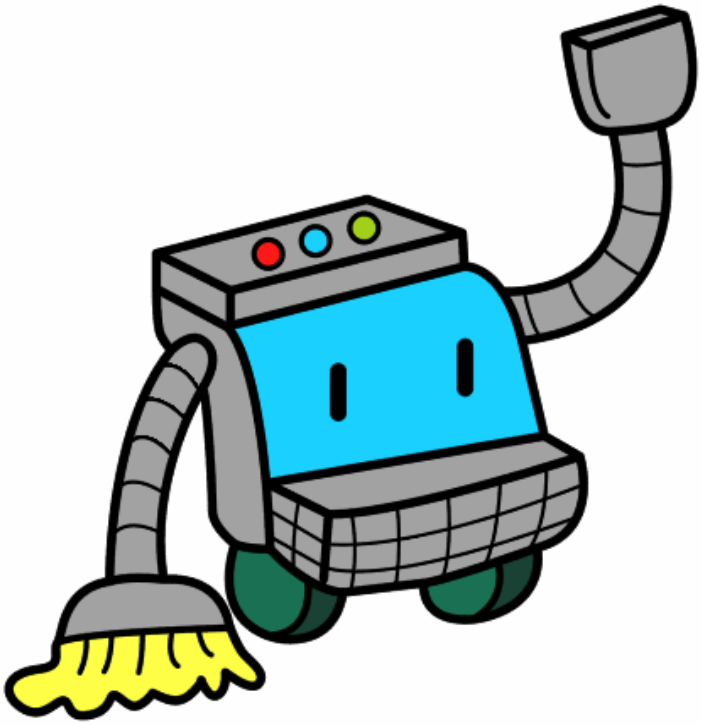

TRABAJO OBLIGATORIO 1

Robot de limpieza

Enunciado

Implementación de una arquitectura basada en el comportamiento e híbrida

GR08
Diego Domínguez Castillejo
Miguel Reino Mateos
Andrés Ripoll Cabrera

Índice

Pág.

1. Introducción

La idea principal de nuestro trabajo deriva de un robot Roomba, en el que nuestro robot va recolectando basura (zonas grises) para llevarla a la papelera (zona negra). Además, incluimos en la sala unos charcos (luces azules), por los que el robot realiza una vuelta sobre ellos para fregarlos. También está atento por si alguien se deja la luz del baño encendida (luz roja) e ir inmediatamente a apagarla.

Para la elaboración del trabajo hemos empleado parte del controlador de motorSchemas en la arquitectura basada en el comportamiento, agregándole nuevas funcionalidades. En la implementación de la arquitectura híbrida hemos añadido funcionalidades propias de una arquitectura de este tipo (calcular rutas, planificar mapas…)

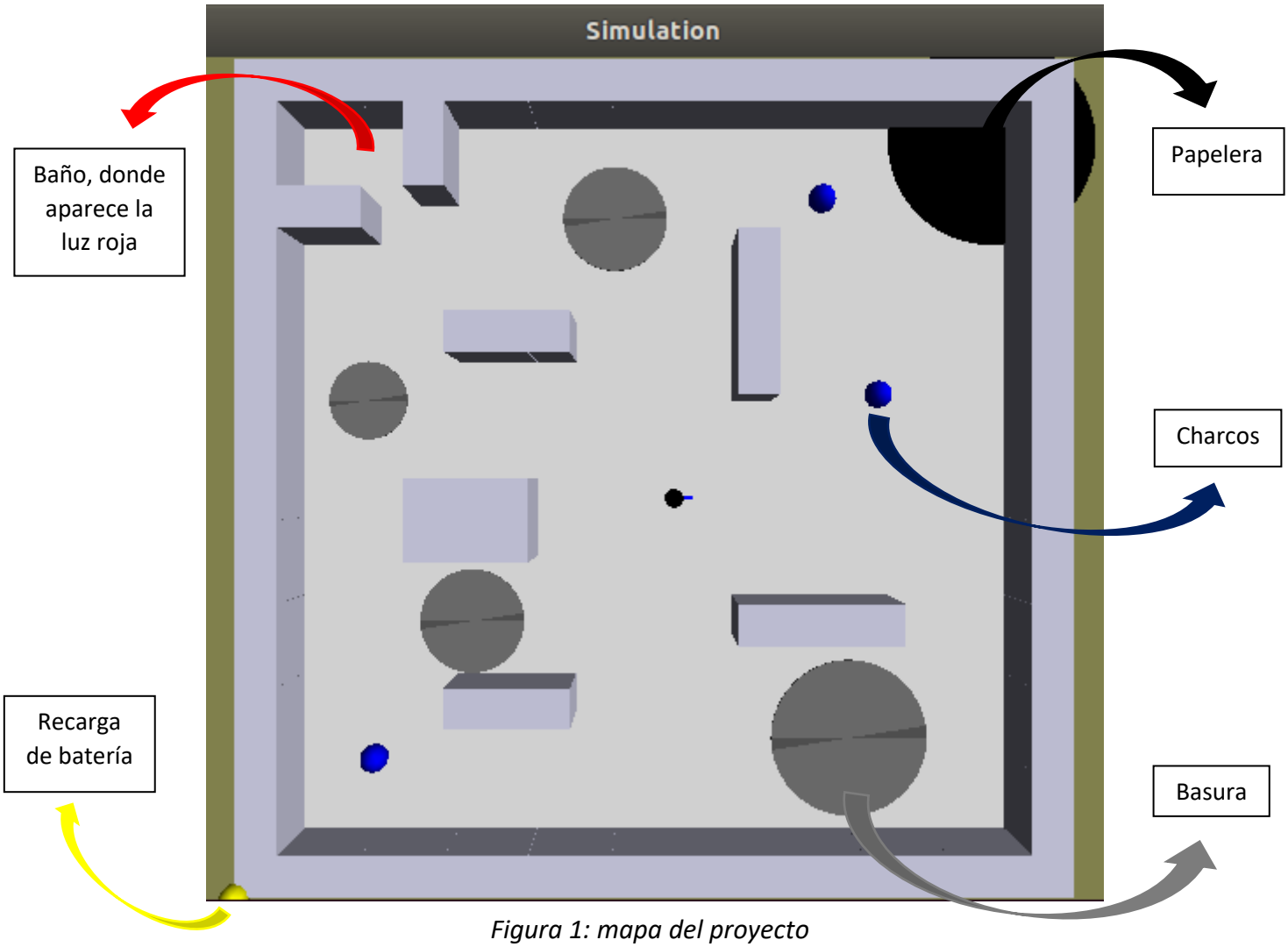

Figura 1: mapa del proyecto

2. Arquitectura basada en el comportamiento

2.1. Esquema

Esta arquitectura está dividida en 6 comportamientos, 4 reciclados y 2 nuevos: “Navigate”, “Forage”, “Load Battery”, “Avoid”, “Dry Floor” y “Switch Bathroom”. En el siguiente esquema vemos cómo se inhiben entre sí y sus prioridades.

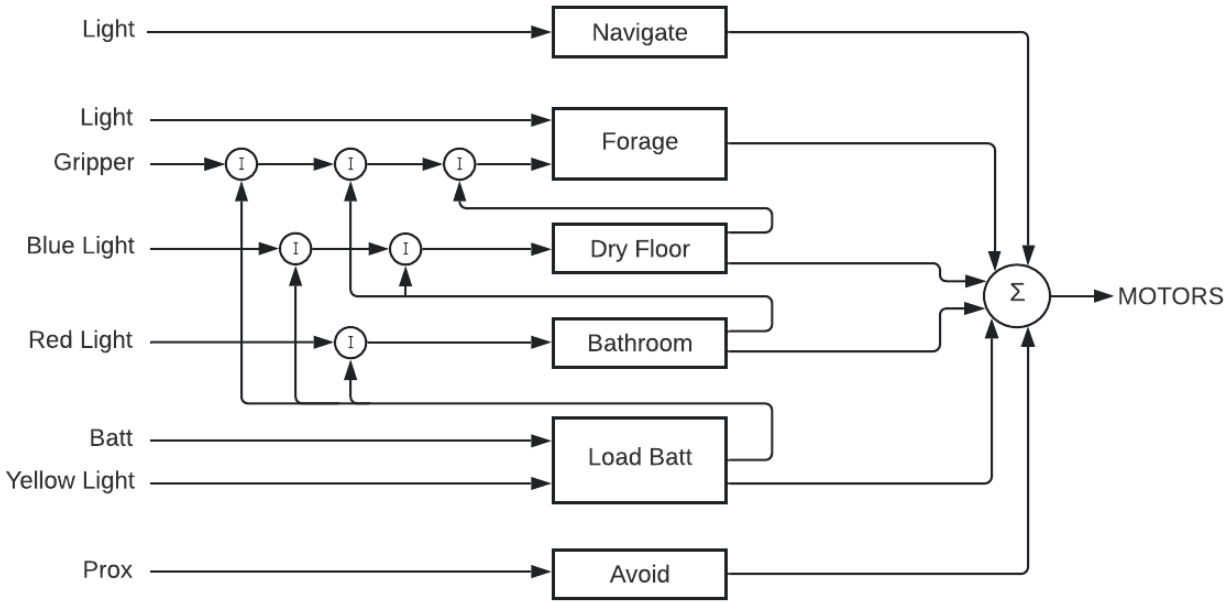

Figura 2: esquema de comportamientos

Explicación de los comportamientos:

- Navigate: esta función es trivial, lo único que hace es moverse recto hacia delante. Tiene la prioridad mínima.

- Forage: función de recolección de basura. Al llegar a una zona gris recoge basura, siempre que no haya recogido ya y no la haya dejado todavía. Cambia de color al azul cuando lleva un objeto encima, y se mueve en dirección contraria a la luz amarilla, es decir, hacia la zona negra o papelera.

- Dry Floor: función de secado de charcos. Cuando el robot detecta que tiene un charco cerca, gira alrededor de él para secarlo. Es inhibido por las funciones que describiremos a continuación, pero no se detiene aunque lleve encima basura, puesto que le dimos más importancia al secado de suelo, antes que llegar a la papelera.

- Switch Bathroom: alguien se ha dejado la luz del baño encendida, así que nuestro robot se dirige allí para apagarlo. Consideramos que es una función bastante prioritaria por lo que solo la inhibe la batería. En cuanto detecta que el baño está encendido, el robot cambia al amarillo.

- Load Battery: si el robot se está quedando sin batería, cambia al color rojo, disminuye su velocidad y deja lo que este haciendo para irse a la zona amarilla donde se recarga.

- Avoid: gracias a esta función el robot evita chocarse con paredes y obstáculos, que representan las paredes de la habitación y muebles o columnas que esta pueda tener. Tiene máxima prioridad, independientemente de cualquier otra función que esté realizando. Cuando está en marcha, el robot cambia al color verde.

2.2. Implementación del código

A continuación detallaremos qué funciones hemos implementado y qué archivos hemos editado, los cuales hemos adjuntado.

- iri1controller.cpp

En este fichero se definen las distintas funciones del robot, y sus prioridades e inhibiciones. Las funcionalidades que hemos aprendido en la asignatura se pueden entender gracias al manual de esta, en el capítulo 4:
http://robolabo.etsit.upm.es/asignaturas/irin/apuntes/manual.pdf

- Navigate: movimiento en línea recta del robot.

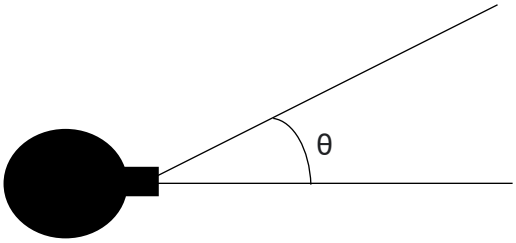

Tabla de activación para Navigate		
Ángulo θ	Velocidad	Flag de prioridad
0	0.5 (50% del máximo)	Siempre 1

- Forage: esta función depende del sensor pinza o gripper. Cuando entra en una zona gris, esta se activa y el robot se orienta en contra de la luz, si no lo estaba ya.

Tabla de activación para Forage		
Ángulo θ (dirección)	Velocidad	Flag de prioridad
$\tan^{-1}(\frac{posicion\ y_{destino}}{posicion\ x_{destino}})$	1 - S_{max}	1

- DryFloor: para implementar esta función nos hemos basado en el comportamiento de un Vehículo II de Braitenberg, cobarde, que al detectar con los sensores la luz azul, gira en torno a ella. En concreto hemos adaptado el código que presentó unos de los integrantes (Diego Domínguez) como Voluntario 2. Con este método, recoge la luz de los 3 sensores de la izquierda y derecha tal que:

$$S_{izq} = S_0 + S_1 + S_2 \qquad \text{y} \qquad S_{der} = S_7 + S_6 + S_5$$

Y según el nivel de intensidad que reciben estos sensores, aumenta la velocidad de la rueda derecha o izquierda. En cuanto su tabla de activación, varía ligeramente porque

en vez de introducir los valores de dirección y velocidad, metemos rueda derecha y rueda izquierda. Además el flag de prioridad no está siempre activo.

En lo que respecta al secado completo (apagar la luz), decidimos probar a ver para que valor daba una vuelta completa y comprobamos que cuando $S_5 = 0.9$ era óptimo (utilizamos S_5 y no S_2 porque gira en sentido horario).

- SwitchBathroom: cuando los sensores de luz roja detectan que esta se ha encendido, el robot se orienta hacia ella. La implementación del código de movimiento es muy similar al de GoLoad.

Estuvimos probando cuando se considera que estábamos lo suficientemente cerca de la luz para apagarlas, y fue para un valor de:

$$S_{total} = S_0 + S_7 = 1.36$$

- Load Battery (GoLoad): cuando la batería llega a su 30% (umbral de batería), el robot necesita ir a cargarse, gracias a los sensores de luz amarilla. Sin embargo, cuando se está cargando, hemos establecido que hasta que no esté la carga al 0.9 no salga de la zona de recarga.

- Avoid (ObstacleAvoidance): función que, gracias a los sensores de proximidad impide la colisión con obstáculos. Debido a que utilizamos un umbral de proximidad pequeño, de 0.3 concretamente, nuestro robot se aproxima menos a los obstáculos. Si utilizáramos un umbral mayor pasaría más cerca de ellos.

Tabla de activación para Avoid		
Ángulo θ (dirección)	Velocidad	Flag de prioridad
$\tan^{-1}\left(\frac{\sum_{i=1}^{8} Prox_i * \cos\theta_i}{\sum_{i=1}^{8} Prox_i * \sin\theta_i}\right)$	$Prox_{max}$	1

- iri1controller.h

Aquí solo incluimos las cabeceras de los nuevos métodos introducidos, los sensores y las variables globales.

- iri1exp.cpp

En este documento, diseñamos el mapa de nuestro proyecto. En nuestro caso no hemos querido llenar la sala de obstáculos y hemos querido dar un toque de realidad con la zona del baño.

- motorSchemas1Param.txt

Fichero de parámetros donde hay que actualizar el número de zonas grises, luz roja y luces azules y sus posiciones. También es donde se incluyen las características de los sensores.

- redlightobject.cpp

Aquí hemos variado las propiedades de la luz roja. Lo hemos programado de tal manera que al cabo de 500 y 4000 pasos se encienda.

2.3. Simulación

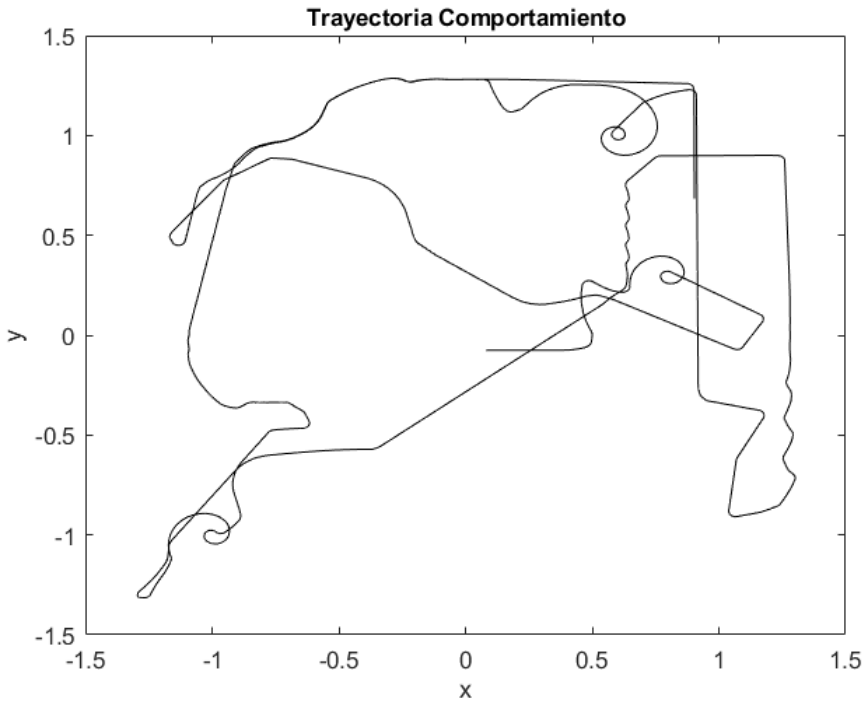

Figura 3: Simulación trayectoria descrita por el comportamiento

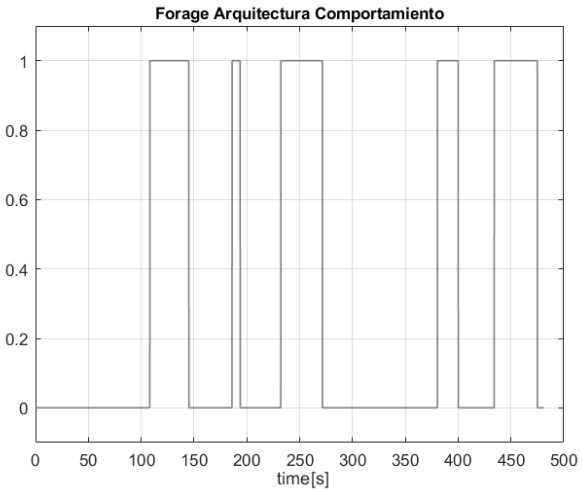

Figura 4: Sensor Forage del Comportamiento

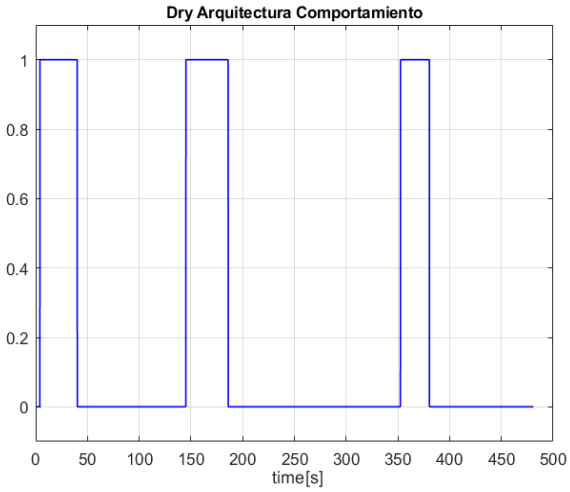

Figura 5: Sensor Dry del Comportamiento

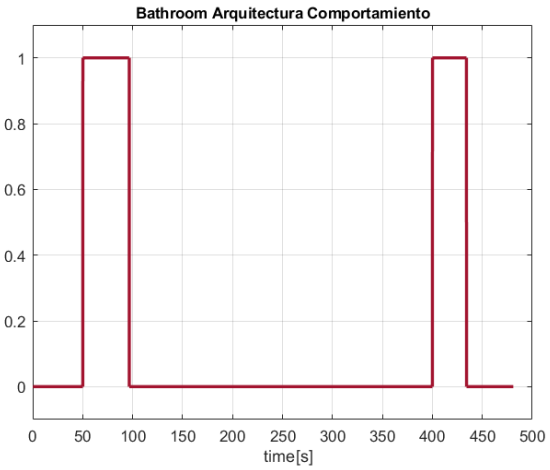

Figura 6: Sensor Bathroom del Comportamiento

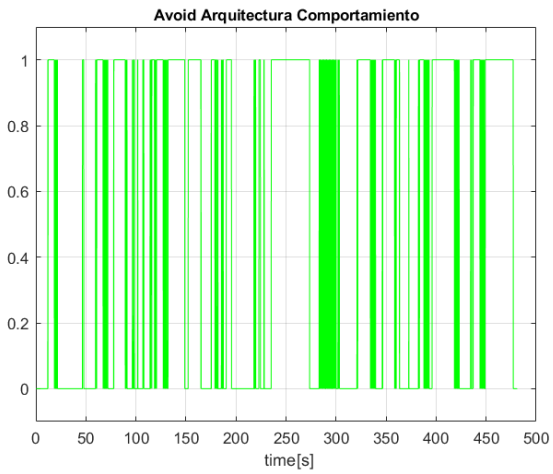

Figura 7: Sensor Avoid del Comportamiento

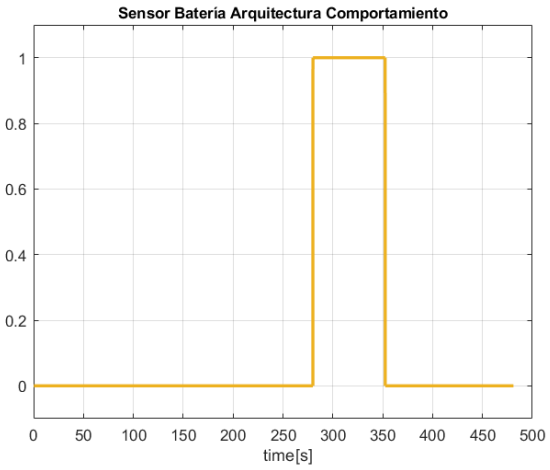

Figura 8: Sensor Batería del Comportamiento

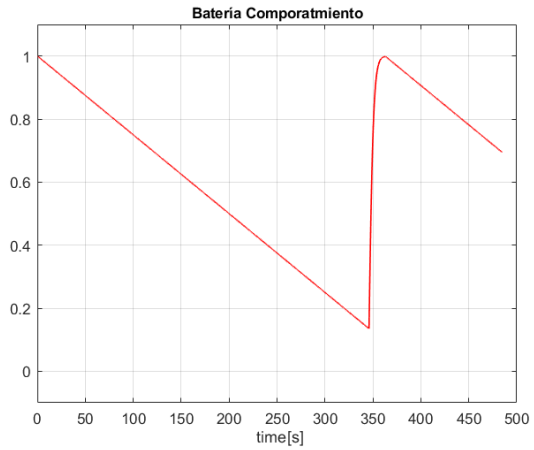

Figura 9: Almacenamiento de Batería del Comportamiento

Como se puede apreciar en la figura 3, vemos los giros que realiza el robot alrededor de los charcos para secarlos, que coinciden con los representados en el mapa de la introducción, y aunque en esta figura no se distinguen las zonas de recolección de basura, en la figura 4 sí podemos ver la presencia de este comportamiento. La figura 5 nos permite ver las tres veces que el robot detecta los charcos para secarlos. Además, debido a que dentro del recorrido la luz del baño se activa dos veces, en la figura 6 se aprecia las dos activaciones de este sensor. Como se puede observar de la figura 7, el comportamiento más frecuente realizado por el robot es el de Avoid ya que el mapa cuenta con varios obstáculos, además de estar cerrado.

Para finalizar, en la figura 10 podemos visualizar el comportamiento de la batería y de su respectivo sensor. Es importante remarcar que cuando la batería se descarga al 0.3 (umbral de batería) se activa el sensor de ir a cargar y cuando la batería se recarga al 0.9, comienza a abandonar la zona de carga (luz amarilla), resultando en una carga completa al salir de la misma debido al uso de un coeficiente de carga elevado.

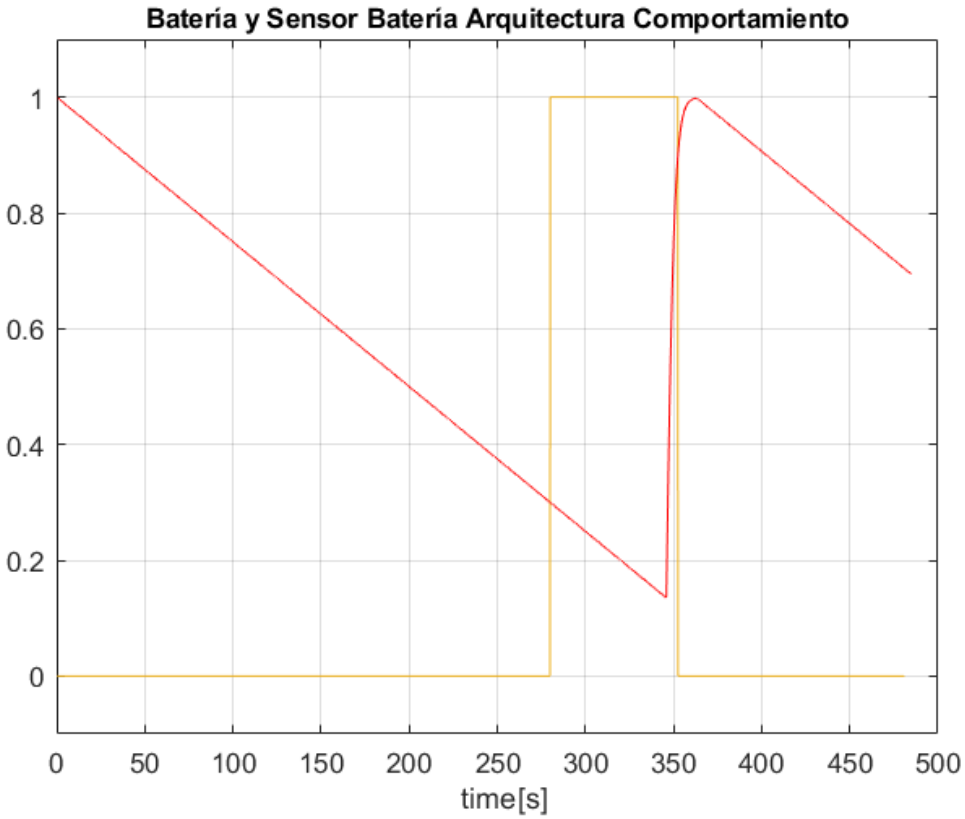

Figura 10: Batería y Sensor Batería del Comportamiento

3. Arquitectura híbrida

3.1 Esquema

La diferencia con respecto a la arquitectura basada en comportamiento es que cuando llega a una zona gris (Prey) o negra (Nest) imprime el mapa que ha ido descubriendo.

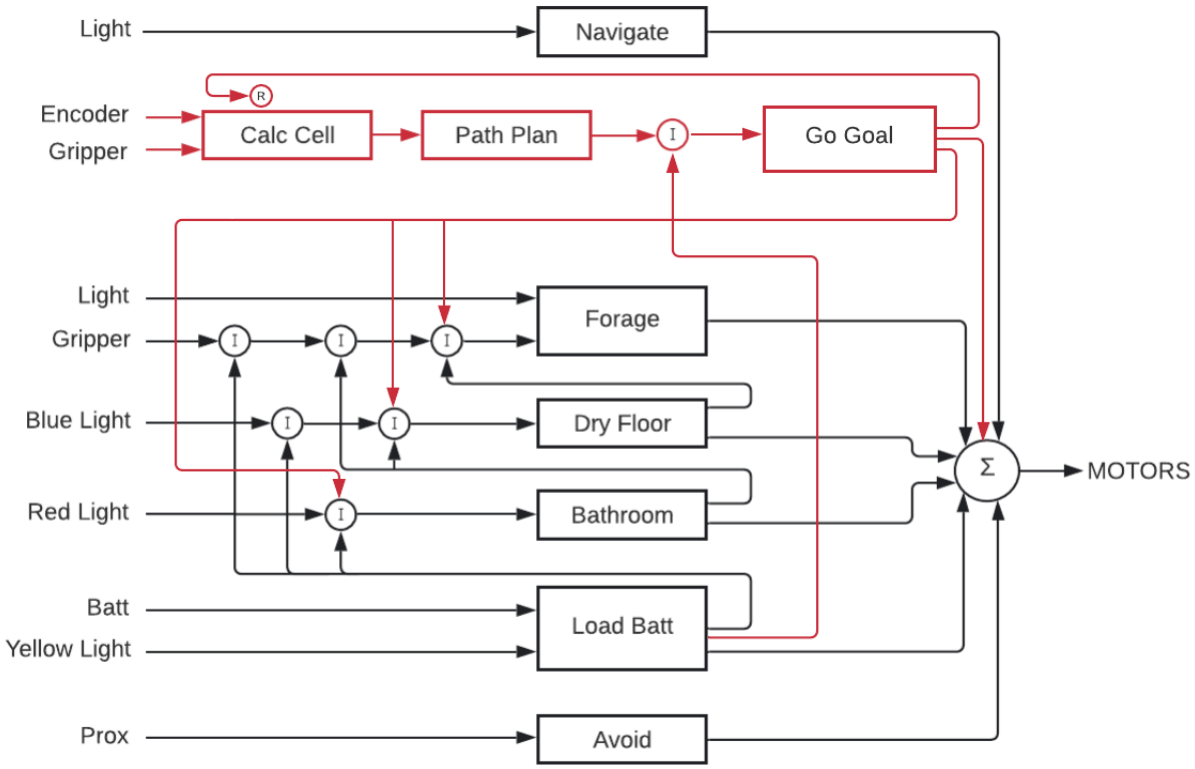

Figura 11: Esquema comportamiento Híbrido

Para no ser redundantes, en esta parte del trabajo solo comentaremos los cambios y novedades en relación con la arquitectura anterior.

- Calc Cell: función que crea un mapa igual que el original, en el que va marcando las celdas por las que va pasando y por tanto están libres de obstáculos, los Preys y el Nest.

- Path Planning: planifica la ruta utilizando algoritmo en estrella A* que realiza el robot siempre que haya localizado un Prey o Nest.

- Go Goal: es una nueva función de la arquitectura híbrida que en la anterior arquitectura formaba parte de Forage, puesto que se encarga de ir a dejar la basura. En este experimento hemos decidido que esta función inhiba al resto para que vaya directo a la papelera en cuanto ha planificado la ruta. Mediante GoGoal se utiliza la ruta previamente creada por Path Planning ya sea para llegar a una zona gris o una negra. Cabe destacar que, debido al umbral de proximidad comentado anteriormente

y el uso de las ocho direcciones del robot, puede causar diversos errores acumulativos de odometría al recorrer la ruta, provocando que el robot se pierda.

3.2. Implementación del código

- iri2controller.cpp

Como código base utilizaremos el ejemplo de arquitectura híbrida subido a robolabo. Solo hay que tener en cuenta el inhibidor de GoGoal, que se activa para Forage, DryFloor y SwitchBathroom.

- motorSchemas2Param.txt

Para esta arquitectura hemos actualizado la arena, incluyendo un charco más y aumentando el tamaño de la zona negra, para de esta manera ver mejor todas las funciones incluidas y minimizar los posibles errores, como por ejemplo la planificación de una ruta repetitiva, que impedía la realización de las demás tareas.

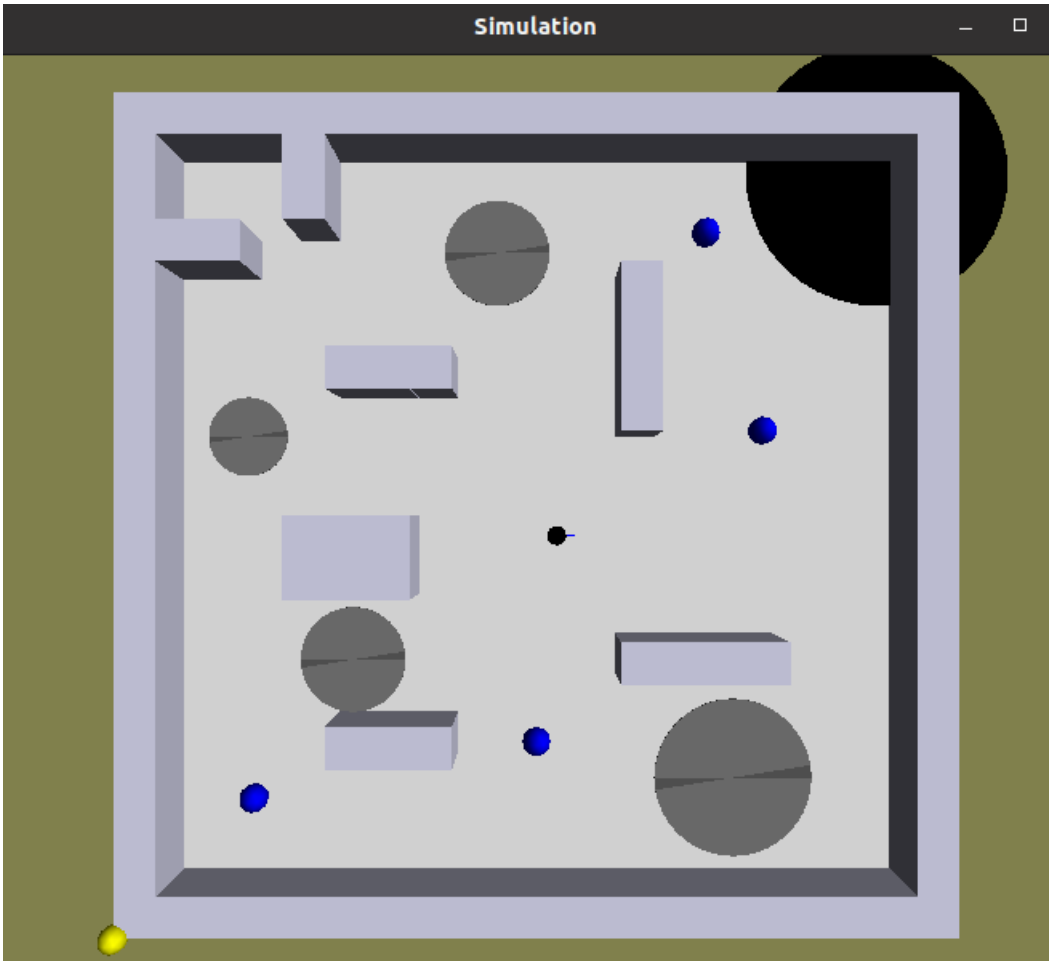

Figura 12: Arena Arquitectura Híbrida

- El resto de ficheros están actualizados a este experimento pero no incluyen ninguna novedad significativa.

3.3. Simulación

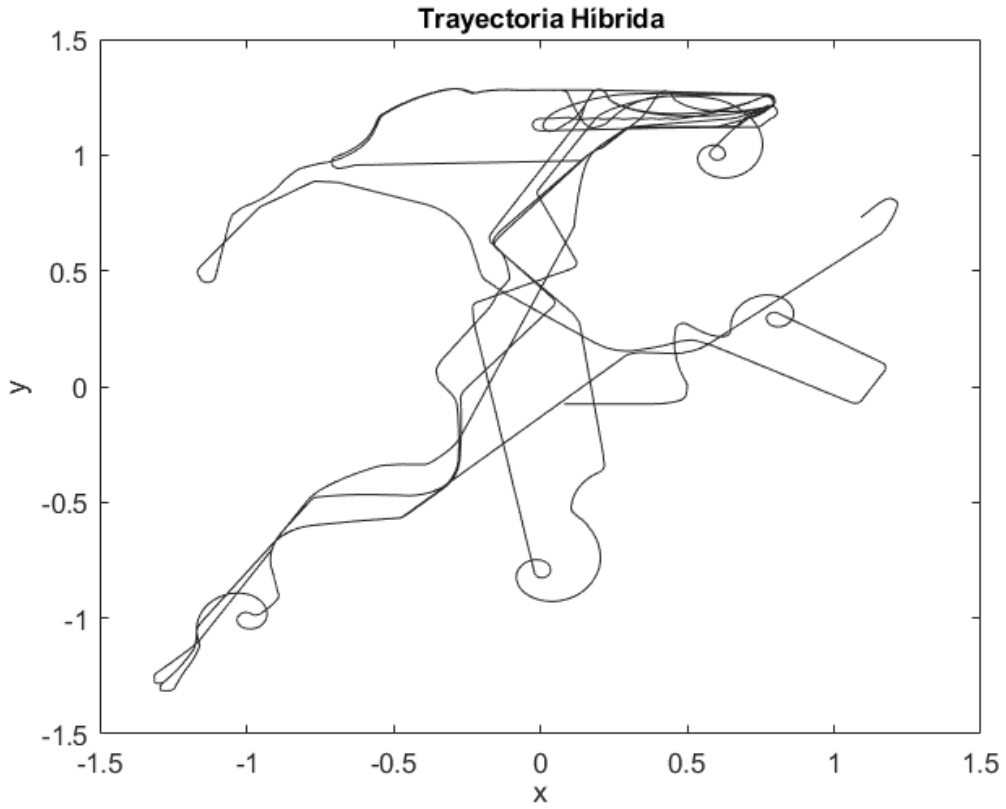

Figura 13: Simulación trayectoria descrita por híbrida

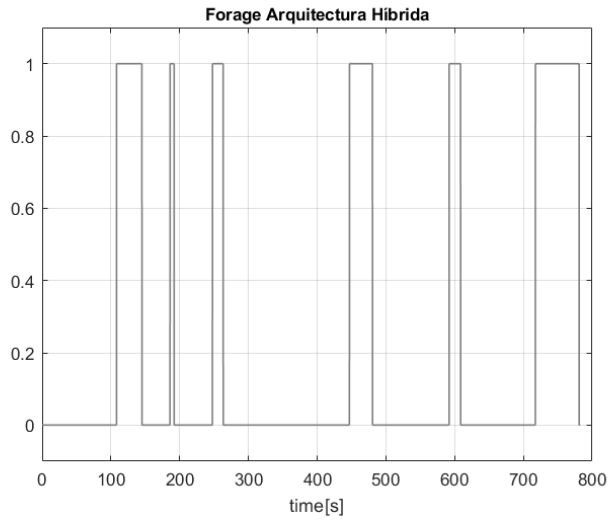

Figura 14: Sensor Forage de la Híbrida

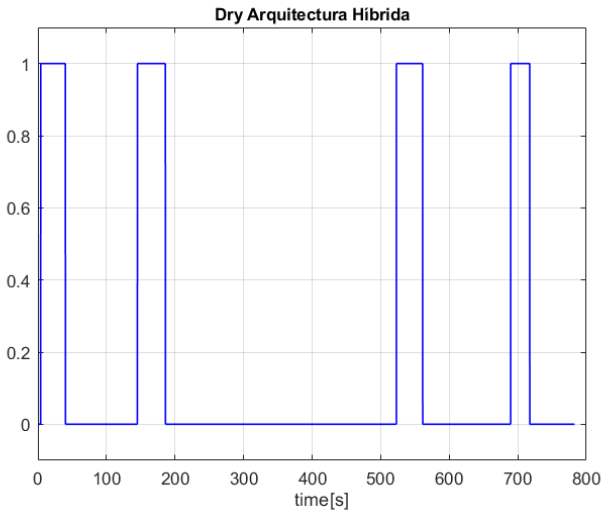

Figura 15: Sensor Dry de la Híbrida

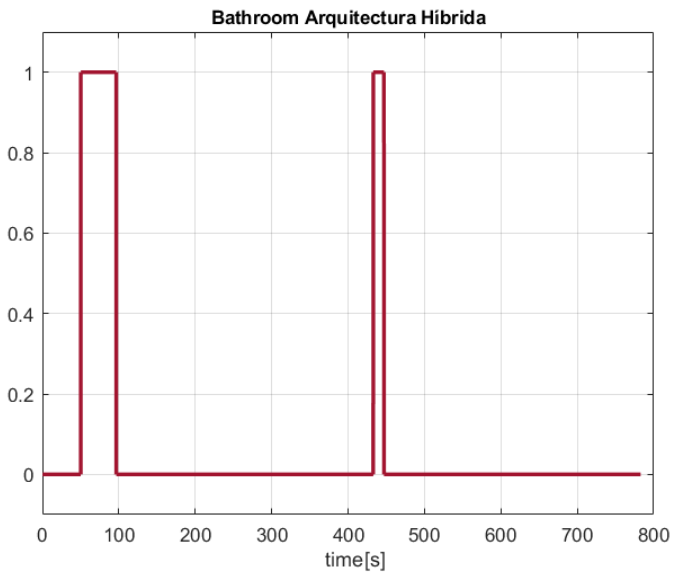

Figura 16: Sensor Bathroom de la Híbrida

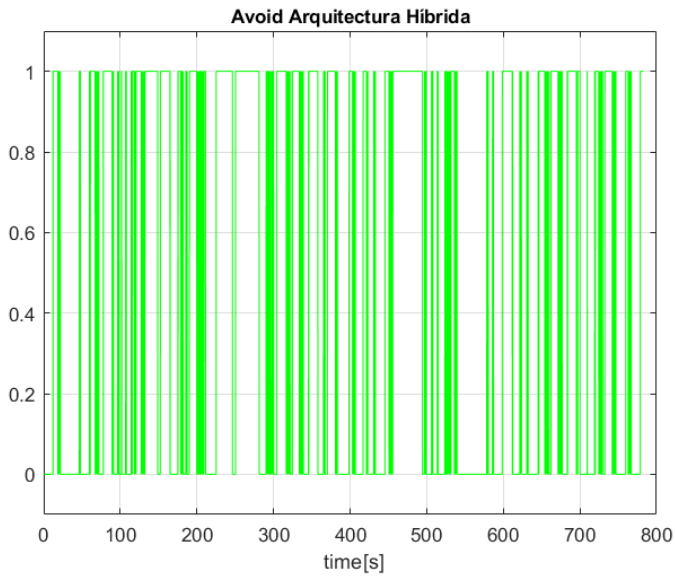

Figura 17: Sensor Avoid de la Híbrida

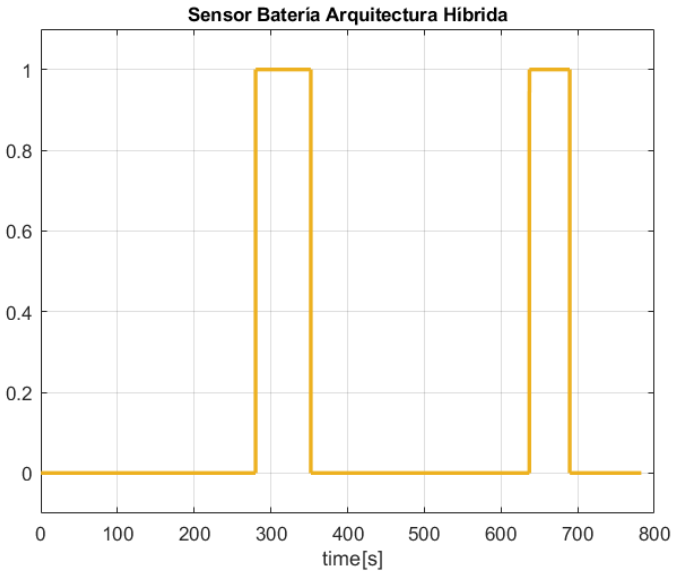

Figura 18: Sensor Batería de la Híbrida

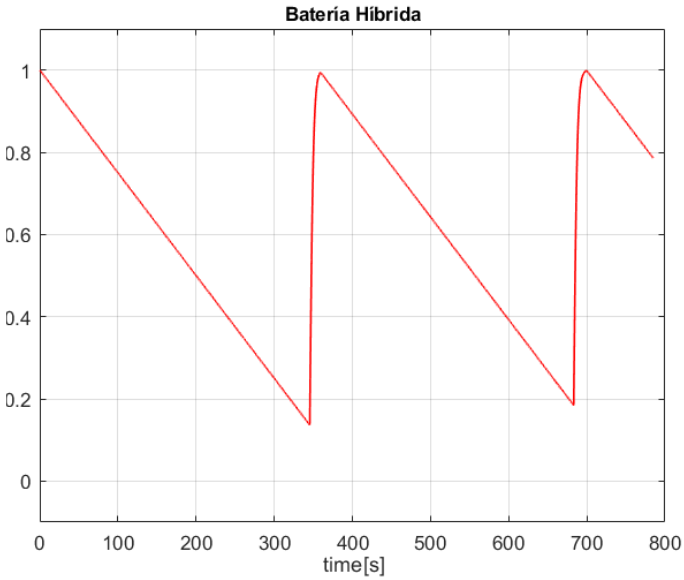

Figura 19: Almacenamiento de Batería de la Híbrida

Comparando con la arquitectura anterior, sigue un comportamiento similar. Si que es remarcable la diferencia de trayectoria que sigue con esta arquitectura puesto que ahora sigue una ruta definida, hasta que se acumule mucho error o el robot necesite cargarse. Por otra parte, observando la figura 15, vemos como, en contraposición a la figura 5, en esta arquitectura recorre un charco más, que previamente habíamos incluido.

La figura 20 muestra también el comportamiento de la batería y su sensor, pero en este caso el robot debe recargarse dos veces.

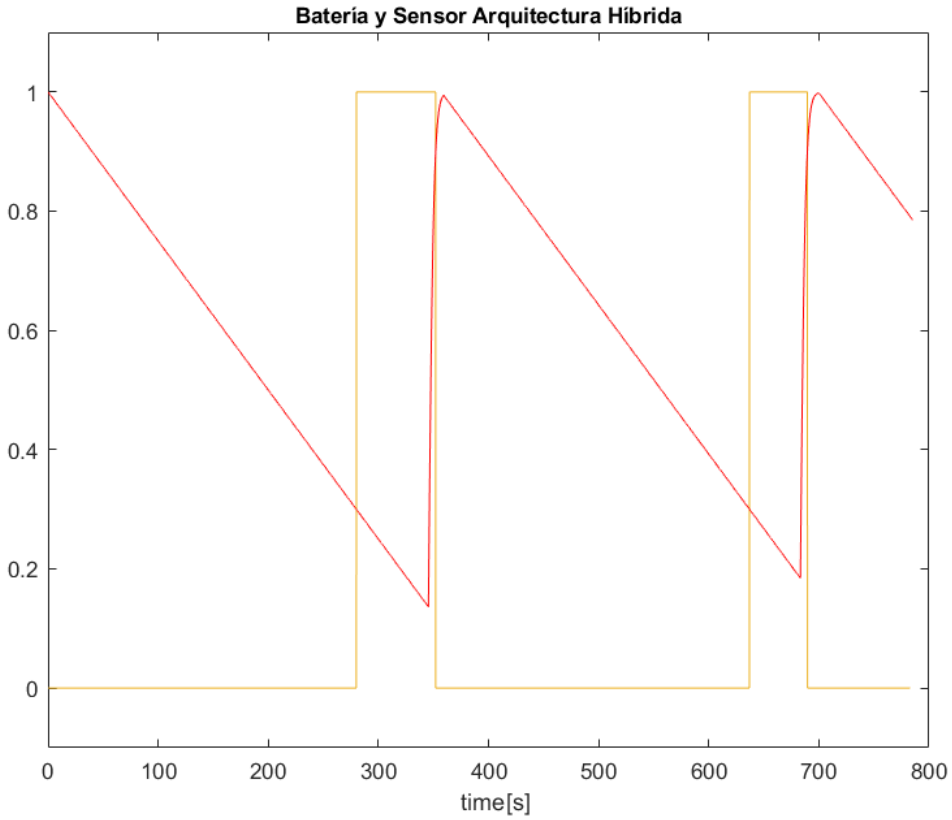

Figura 20: Batería y Sensor Batería de la Híbrida

4. Problemática y posibles futuras implementaciones

4.1. Problemática

Uno de los primeros problemas que hemos tenido ha sido a la hora de implementar un temporizador en la luz del baño. Pretendiamos utilizar un número aleatorio mediante la función rand, pero más tarde nos dimos cuenta de que esto era inviable, ya que la variable nos generaba un número por cada paso del robot, o bien nos generaba siempre el mismo número, por lo que decidimos descartarlo.

También tuvimos ciertos problemas a la hora de desarrollar el código, ya que casi siempre nos encontrábamos con errores de ejecución (error core), normalmente causados por no declarar una variable correctamente. Debido a que es la primera vez que programamos en C++, el código no es especialmente estético.

Los problemas que experimentamos al inicio del proyecto no se repitieron en el momento en el que decidimos implementar la arquitectura híbrida, gracias a la experiencia adquirida anteriormente.

Por último, para el correcto desempeño de las funciones del robot, tuvimos que compilar y ejecutar numerosas veces el código, cambiando cada vez los distintos parámetros y valores, ya sea el rango de los sensores, el coeficiente de carga o descarga, el umbral de proximidad, etc. Además, con el objetivo de que el robot no se quedase atascado en el baño tuvimos que adaptarlo a nuestras necesidades.

4.2. Posibles futuras implementaciones

Se nos ocurren diversas implementaciones, de las cuales vamos a mencionar algunas:

- Como una primera posible implementación se nos ocurre que cuando el robot se acerque al baño a apagar la luz, revise que no haya una persona dentro. Para ello, deberíamos implementar un nuevo sensor, o en su defecto una casilla diferente a las ya implementadas (gris o negra) que nos permitiera realizar esta labor.
- Con el objetivo de no caer en una ruta repetitiva en la arquitectura híbrida, habíamos pensado que el Path Planning nos recoja de manera aleatoria las zonas de color gris y que el robot no se dirija directamente a estas, haciendo de este modo que tenga que recorrer la arena en busca de otras distintas.
- Y por último, para darle más realismo al proyecto nos gustaría que cuando vaya recogiendo basura, las zonas grises vayan disminuyendo su tamaño, ofreciendo la sensación de que el robot está realizando una labor de limpieza correcta.

5. Conclusiones

Gracias al desarrollo de este trabajo, hemos conseguido adquirir los conocimientos necesarios para entender las arquitecturas basadas en el comportamiento y en el conocimiento, para poder introducir nuestras propias funcionalidades. Por otra parte, aunque el manejo de inhibidores pueda parecer sencillo, en realidad puede llegar a ser confuso si hacemos un uso excesivo de ellos.