Robot camarero

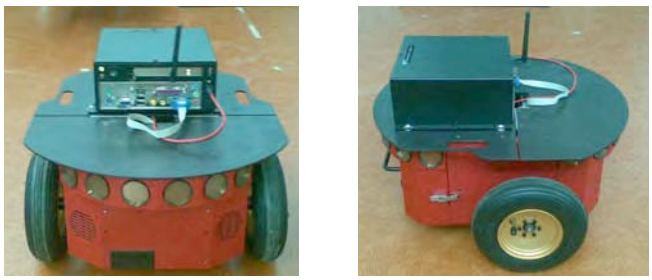
Mª del Mar Sastre Escudero, Ernesto Sigg Rodríguez, Mauro Zanlongo

Asignatura: Robótica

Abstract— **Este documento presenta todos los aspectos utilizados para la implementación de un robot camarero sobre el robot Pioneer de la asignatura Robótica perteneciente a los estudios de ingeniería en informática. Se explicará la arquitectura de control utilizada a través de los niveles estratégico, táctico y ejecutivo, los comportamientos y cálculos realizados, además de una serie de pruebas con el simulador.**

VI. Introducción

La práctica consiste en programar un robot Pioneer con las librerías Aria para implementar la problemática de un negocio de la hostelería (bar, restaurante…). En este caso, el robot corresponde al camarero y las mesas del entorno son los puntos objetivo. La idea de la problemática consiste en que todas las mesas deben estar servidas en el menor tiempo posible y en un orden establecido. Además, en el entorno, podemos encontrar obstáculos como pueden ser columnas o las mismas mesas, las cuales el robot deberá esquivar cuando no sean puntos objetivo, utilizando una técnica de evitación de obstáculos. La realización la práctica se ha ejecutado sobre un robot Pionner 3DX, como muestra las figuras 1 y 2, usando las librerías Aria.



(a) (b)

Fig. 1 (a). Vista frontal del robot Pioneer 3DX. (b). Vista lateral del robot Pioneer 3DX

VII. El Robot Camarero

Como se ha explicado brevemente en la introducción, la implementación simula la gestión de las mesas de un local de hostelería (bar, restaurante…). Para ello, se ha creado un entorno que consta de mesas (objetivos), obstáculos (pueden ser columnas del local o las mismas mesas cuando no son objetivos) y un punto inicial (simula la barra del local). Cada mesa se puede encontrar en uno de los siguientes estados:

1) Vacía: No hay ninguna persona en la mesa.

2) Sin nota: La mesa está ocupada pero todavía no se ha realizado el pedido.

3) Con nota: La mesa está ocupada y ya se ha realizado el pedido.

4) Servida: El robot ha servido la mesa con el pedido solicitado.

Al inicio, el robot se encarga de calcular una prioridad para cada mesa en función de la distancia respecto a la barra (esta prioridad será fija) para, posteriormente, calcular los caminos que deberá seguir para servir y realizar los pedidos de las mesas (por prioridad). De esta manera, el robot realizará dos recorridos: un recorrido de ida para servir a las mesas que ya hayan pedido, y un recorrido de vuelta para tomar nota a las que todavía no lo han hecho. Una vez hecho el camino de vuelta, el robot regresará a la barra. Además, cada vez que el robot llega a la barra, puede percibir cambios en el estado de las mesas, como por ejemplo, que las mesas que ya se hayan servido se pueden vaciar, las mesas vacías pueden llenarse… Estos cambios se producen mediante variables aleatorias. Se ha optado por hacer esta implementación ya que dotar al robot de la capacidad de percibir cambios en el entorno en cualquier momento, significaría trabajar con procesos concurrentes, lo cual no es el objetivo de esta práctica.

En resumen, podemos decir que el algoritmo que sigue el robot al más nivel es el siguiente:

```
Calcular_prioridad_mesas;
Mientras (1) {
        Recorrer_camino_ida (servir mesas);
        Recorrer_camino_vuelta (tomar nota);
        Regresar_a_barra;
        Generar_cambios_entorno (cambios de estado);
}
```

Para finalizar, se mostrará el aspecto que debería tener la práctica simulando un entorno real de un local de hostelería, como podría ser un bar. En la figura 2 se muestra la barra (donde inicialmente empezará el robot a moverse) y 6 mesas a su alrededor que deberá servir a medida que se le envíen peticiones. Además se encuentran 4 columnas en el local que deberá evitar el robot a medida que vaya atendiendo a las mesas.

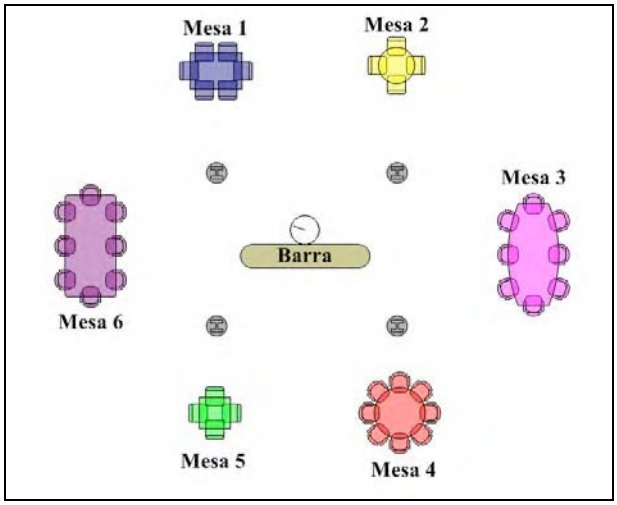


Fig. 2 Entorno real del robot camarero

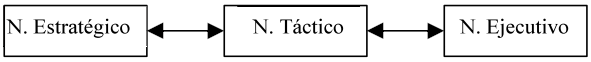


Fig. 2 Arquitectura de control

VIII. Arquitectura del Robot

La arquitectura de control consiste en 3 niveles, que permiten separar la implementación de los programas de alto nivel de la estructura básica del robot. Estos tres niveles son: estratégico, táctico y ejecutivo.

A. *Nivel Estratégico*

Es el nivel donde se ejecuta la misión a alto nivel. Indica al nivel táctico cuáles son los sub-objetivos por donde tiene que pasar el robot para ejecutar la tarea. Este nivel se ejecuta cada cierto tiempo y, a través del nivel táctico, se comprueba si el robot ha llegado al objetivo o no.

B. *Nivel Táctico*

En este nivel se ejecutan los comportamientos que lleva a término el robot, como ir a objetivo, evitar obstáculos… Se encarga de que el robot llegue a los sub-objetivos indicados por el nivel estratégico de manera que no colisione con los obstáculos encontrados en su camino. Este nivel se llama periódicamente por el nivel estratégico.

C. *Nivel Ejecutivo*

Es el encargado de interactuar con la parte física del robot (ultrasonidos y motores). La mayor parte del proceso ejecutivo viene implementado por las librerías del Aria y por el µ-microcontrolador del propio robot. En él se definen los métodos para mover y parar el robot, obtener su posición y los valores de los sensores.

IX. Comportamientos

Para implementar el nivel táctico se utilizan comportamientos reactivos. Un comportamiento es un módulo que a partir de una serie de estímulos exteriores da una determinada respuesta. Esta respuesta es de tipo reactivo, es decir, no se realiza ningún tipo de planificación y sólo depende de los datos que se tengan en ese momento. En este caso, se han generado dos comportamientos: ir a objetivo y evitar obstáculos.

A. *Ir a objetivo*

El comportamiento de ir a objetivo tiene como salida la dirección a la que tiene que ir el robot para llegar a ese punto sin tener en cuenta si existen obstáculos en su camino.

A. *Evitar obstáculos*

El comportamiento para evitar obstáculos tiene como salida la dirección a la que tiene que ir el robot para evitarlos.

La técnica utilizada para implementar dichos comportamientos es la denominada “campos de potencial”. El campo de potencial es un método que consiste en modelizar al robot como si fuera una partícula sometida a una serie de fuerzas. Por ejemplo, en el caso de evitar obstáculos, éstos son los que generan fuerzas de repulsión sobre el robot, mientras que el punto objetivo genera una fuerza de atracción. Cada comportamiento genera una fuerza y la dirección final del robot se obtiene a partir de la suma ponderada del resultado de cada uno de estos comportamientos.

X. Cálculos

Los cálculos empleados para implementar los dos comportamientos comentados en apartados anteriores son los siguientes:

B. *Ir a objetivo*

El funcionamiento de este comportamiento es el siguiente: dados dos puntos, el punto actual (PA) y el punto objetivo (PO) se tiene que definir el vector entre dichos puntos a través de las siguientes fórmulas (1), (2) y (3) a partir de la figura 4.

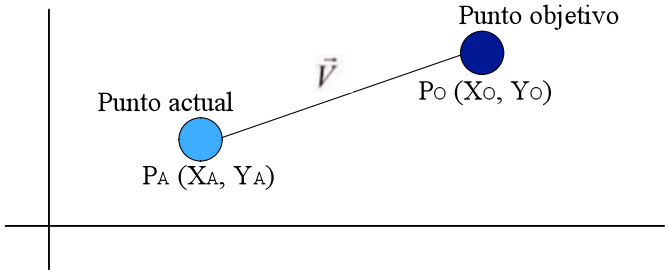


Fig. 4 Ir a objetivo

$$\vec{V} = (X_O - X_A, Y_O - Y_A) \quad (1)$$

$$\vec{V} = \sqrt{X_A^2 + X_O^2} \quad (2)$$

$$\alpha = \arctan\left(\frac{y_v}{x_v}\right) \quad (3)$$

Evitar obstáculos

El funcionamiento de este comportamiento se basa en la técnica “campos de potencial” explicada en el apartado anterior. El robot posee 16 sensores, 8 en la parte delantera y 8 en la parte trasera. Cada sensor genera un vector de repulsión a partir de los datos de cada sensor, es decir, genera 16 vectores de repulsión. A cada vector generado, se le aplica un peso según la posición de cada sensor. En nuestro caso, los sensores traseros tienen 100 veces menos peso que los sensores delanteros. Una vez ponderados se suman para obtener el vector de repulsión total, utilizando la fórmula (4). Cuando se detecte algún obstáculo muy cerca del robot, a una distancia menor a un valor determinado denominado *distancia mínima* (ver punto V.C), se supondrá que el peligro de colisión es inminente. En este caso, se deshabilitan todos los demás comportamientos y sólo estará activo el de evitar obstáculos.

En el caso del robot camarero se han considerado dos posturas en lo que se refiere a la velocidad que debe alcanzar el robot para moverse por el entorno. Por una parte, una versión de la práctica se ha realizado utilizando la velocidad máxima del robot, definida en el fichero de configuración. La segunda versión, la velocidad alcanzada está en función de la cantidad de obstáculos que detecta. Cuando el módulo del vector del comportamiento de evitar obstáculos sea muy alto, la velocidad se reduce, en cambio, cuando el robot se mueve por un entorno donde no encuentra obstáculos cerca de él, la velocidad aumenta.

$$\vec{V} = \sum_{i=1}^{16} p_i \cdot \vec{V}_i \quad (4)$$

XI. PARÁMETROS DEL ROBOT

Para que el robot pueda moverse de un punto objetivo a otro evitando obstáculos, tiene que leer de un fichero de configuración los parámetros de la arquitectura. Este fichero sigue una estructura en el que cada fila corresponde a un parámetro en concreto conocido por nosotros. Los parámetros definidos para el correcto funcionamiento del robot son los siguientes:

- *maxDist:* Es la distancia máxima a partir de la cual los objetos detectados no se consideran obstáculos, definida en mm.
- *velMax:* Es la velocidad máxima a la que puede moverse el robot hacia los puntos objetivo, definida en mm./s.
- *distMin:* Es la distancia mínima de colisión, definida en mm.
- *distObj:* Es la distancia mínima a partir de la cual consideramos que hemos llegado al objetivo, definida en mm.
- *DRot:* Es la diferencia máxima entre la rotación actual del ángulo y la que tendrá que tener, definida en grados.
- *T:* Es el período de tiempo entre llamadas, definido en ms.
- *pesoObjetivo:* Es el peso del comportamiento de ir hacia el objetivo.
- *pesoObstaculo:* Es el peso del comportamiento de evitar obstáculos.
- *numMesas:* Es el número de mesas que se encuentran en el entorno.
- *Parámetros de las mesas:* A partir de aquí se encuentran los parámetros asociados a cada mesa, a través de un conjunto de 2 filas para cada mesa. La primera fila corresponde al estado inicial de cada mesa, relacionado con los 4 estados mencionados en el apartado II. La segunda fila corresponde a la posición (x, y) de cada una de las mesas. Esta posición no se modifica a lo largo de la ejecución de la práctica.

XII. PRUEBAS EXPERIMENTALES

Una vez explicado todo el funcionamiento del robot camarero, se mostrará a través de la figura 5 la siguiente prueba: ir desde un punto inicial (simulando la barra) a un punto objetivo (simulando una mesa) con un obstáculo situado en medio del camino a realizar. En dicha figura se observan dos caminos: la flecha roja simula el camino que debería hacer el robot si no hubiera ningún obstáculo en su camino para alcanzar el objetivo. Por otro lado, la flecha verde simula el camino que debe realizar el robot en caso de encontrarse un obstáculo en su camino. Como se puede observar, sigue el método utilizado que se ha explicado anteriormente de evitar obstáculo.

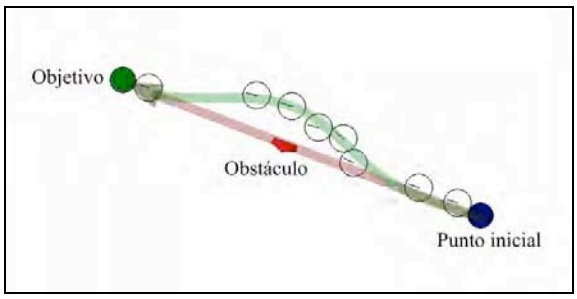


Fig. 5 Técnica evitar obstáculo

A continuación, en la figura 6, se muestra una captura de pantalla de los resultados que se han obtenido en una de las ejecuciones de la práctica a medida que el robot se va moviendo por el entorno realizando sus tareas. En un primer momento se visualizan el estado de cada una de las mesas ordenadas por la distancia que debe recorrer el robot para llegar a ellas. En este caso, la mesa 2 está vacía, la mesa 3 está ocupada pero todavía no ha sido atendida por el robot y las mesas y 0 ya han realizado el pedido al robot. Según se ha explicado anteriormente, el robot realiza dos caminos: uno de ida para servir a las mesas que ya hayan pedido y uno de vuelta para tomar nota a las que todavía no lo han hecho. De esta manera, la mesa 3 esperará a que el robot haya servido antes a las mesas 1 y 0. Una vez realizado los dos caminos regresa a la barra. A partir de ahí, se vuelven a generar cambios en el estado de las mesas, los cuales repercutirán en el camino que deberá recorrer el robot.

Se han realizado una serie de pruebas sobre el robot real (Pioneer 3DX) donde se ha podido observar la correcta ejecución de la práctica. Para ello se han tenido que realizar pequeños cambios en los parámetros del robot, como por ejemplo, *distMin* o *Drot* (ver punto VI). En la figura 7 se muestra un momento de la ejecución de la práctica donde se observa el comportamiento de evitar obstáculos.

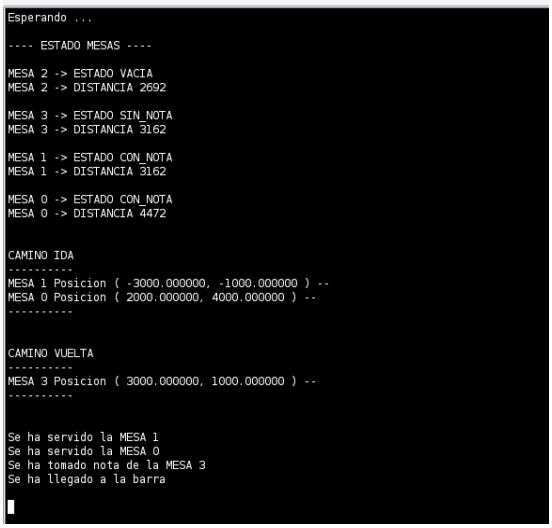


Fig. 6 Debug robot camarero

XIII. CONCLUSIONES Y MEJORAS

A modo de conclusión se han tenido en cuenta una serie de aspectos que pueden servir como punto de partida para posibles mejoras en el robot:

- La posibilidad de tener en cuenta el número de personas que ocupan cada mesa: con este dato se pueden calcular nuevas prioridades y nuevos caminos.
- Se pueden usar técnicas basadas en mapas para mejorar el recorrido del camino y la evitación de obstáculos.
- Existe la posibilidad de dotar al robot de capacidad para percibir cambios en el entorno en cualquier momento del ciclo de ejecución. Una posibilidad sería implementar un proceso concurrente que envíe "nuevos eventos" al robot.

Fig. 7 Prueba real – evitar obstáculos

Mª del Mar Sastre Escudero. Estudiante de 5º de ingeniería informática. Ingeniera Técnica en Informática de sistemas (UIB-2006). Trabajando actualmente en el Centro de Tecnologías de la Información (UIB) en desarrollo del proyecto de mecanización de la FUEIB.

Ernesto Sigg Rodríguez. ernesto.sigg@uib.es . Ingeniero Técnico en Informática de sistemas (UIB-2006). Trabajando actualmente para la Fundació Universitat-Empresa (UIB) en aplicaciones de vigilancia tecnológica.

Mauro Zanlongo. Estudiante de 5º de ingenieríainformática. Ingeniero Técnico en Informática de sistemas (UIB-2006). Trabajando actualmente para Axis Data S.L. en el desarrollo de aplicaciones B2B. Otros articulos: "The Intelligent Butler: a Virtual Agent Prototype for Disabled People Assistance".