Arquitectura de Control DPA

EDUARDO JAVIER PÉREZ RODRÍGUEZ

Departamento de Tecnología Electrónica
Teléfono : 95 213 71 75
Mail : edu@dte.uma.es

GUÍA RÁPIDA DE LA "ARQUITECTURA DE CONTROL DPA"

- ***Servidor DPA:*** (IP: 192.168.187.67)

  Activación servidor DPA:

  ```
  Login: dpaserver
  Password: dpa
  ```

  Activación administrador remoto DPA:

  ```
  Login: dpaadmin
  Password: dpa
  ```

- ***Servidor P2AT:*** (IP: 10.0.0.1)

  Activación servidor P2AT:

  ```
  Login: p2atserver
  Password: p2at
  ```

  Desactivación servidor P2AT:

  ```
  <CTRL>+<C>
  [p2atserver@p2at] su              {password: mazinger}
  [root@p2at] halt
  ```

  apagar el robot cuando Linux haya finalizado de apagar (mensaje: "System halted")

- ***Portátil (Router):*** (IP: 192.168.187.117; 10.0.0.2)

  Activación Router:

  Encender el portátil y conectarlo al robot P2AT con el cable cruzado.

- ***Módulos Arquitectura:*** (IP: 192.168.187.66)

  Activación módulos: *IFRobot, Navigation, MapMetric, PathPlanning, P2AT*

  ```
  Login: robot
  Password: p2at
  ```

- ***Módulos Arquitectura:*** (IP: 192.168.187.65)

  Activación módulos: *IFUser*

  ```
  Login: robot
  Password: p2at
  ```

GUÍA DE USUARIO DE LA "ARQUITECTURA DE CONTROL DPA"

La arquitectura de control DPA usada en el Departamento de Tecnología Electrónica es el sistema utilizado para controlar actualmente al conjunto de robots que posee el mismo. Para ser capaz de poder manejar dichos robots (enviarles comandos y recibir información del entorno a través de los sensores) es necesario saber usar dicha arquitectura de control.

La arquitectura de control posee 3 elementos básicos que hay que administrar:

- ***SERVIDOR DPA:***

  Es el elemento principal y más importante de la arquitectura de control, pues permite el intercambio de datos entre toda una red distribuida de ordenadores para procesar la información que reciben los sensores del robot y enviarle los comandos adecuados al mismo. Es un servidor software que debe estar funcionando en una máquina Linux conocida. Actualmente se encuentra en la máquina 192.168.187.67, y tiene asignado el puerto de acceso 7000.

  Antes de usar la arquitectura de control para manejar cualquier robot es necesario verificar que el servidor DPA está funcionando, y si se detecta cualquier anomalía hay que reiniciarlo, para lo cual quizás sea preciso en primer lugar detener su ejecución. La forma de realizar estas operaciones se describe a continuación:

  COMPROBACION FUNCIONAMIENTO SERVIDOR DPA:

  Hay que verificar que se está ejecutando el proceso "DPAServer" en la máquina 192.167.187.67. Este proceso, que es el servidor en sí, se ejecuta en modo comando, por lo que suele estar ejecutándose en una consola de texto. Se recuerda que en el sistema Linux existen 7 consolas distintas para acceder al sistema operativo, 6 de texto y 1 gráfica. Para cambiar de una consola a otra hay que pulsar la secuencia de teclas (<CTRL> + <ALT> + <Fn>), donde "Fn" es la tecla de función "n" que se corresponde con la consola que se quiere activar. Las consolas de texto son las 6 primeras (F1 – F6), mientras que la consola gráfica es la consola número 7 (F7). El sistema operativo Linux arranca por defecto en la consola de texto número 1, o en la consola gráfica si se configura para ello.

  NOTA: La consola gráfica no siempre está activa: depende de la configuración de arranque del sistema operativo Linux. Por lo tanto, es posible que en algunos casos si se activa la consola gráfica no se vea nada y haya que volver a una consola cualquiera de texto

  Al funcionar el servidor en modo texto se ejecuta normalmente desde una consola de texto, normalmente la 1, que es donde suele arrancar el sistema, aunque se puede ejecutar desde cualquier otra consola de texto. Una vez ejecutándose muestra alguna información por pantalla (dirección IP de la máquina host donde está ubicado, puerto de acceso, …) e indica que se está ejecutando mediante el mensaje "DPA SERVER INITIALITED". Si por

cualquier causa su ejecución ha sido detenida muestra esta circunstancia con el mensaje "DPA SERVER SHUTDOWNED".

Para comprobar si el servidor DPA está funcionando existen 2 procedimientos:

- procedimiento **visual**: consiste en ir cambiando entre las sucesivas consolas de texto (F1 – F6) para verificar si en alguna de ellas se muestra por pantalla el mensaje "DPA SERVER INITIALITED", lo cual indica que el servidor DPA se está ejecutando en dicha consola. Hay que comprobar también que en dicha consola no se muestra el mensaje "DPA SERVER SHUTDOWNED", pues eso indicaría que el servidor DPA ha dejado de ejecutarse.

- procedimiento por **comandos**: consiste en acceder al ordenador que alberga al servidor DPA mediante un login y password adecuados (si no se posee una cuenta se puede acceder en modo **root**, con contraseña **mazinger**):

  ```
  login: root
  password: mazinger
  ```

  ver la lista de procesos que se están ejecutando mediante el comando "ps":

  ```
  [root@dpa] ps –ax
  ```

  y buscar entre la lista de procesos alguno (pueden existir más de uno) que se llame "DPAServer". Se suele aplicar un filtro "grep" al comando "ps" para realizar esta búsqueda de forma más rápida:

  ```
  [root@dpa] ps –ax | grep DPA
  ```

  Si tras realizar esta búsqueda se encuentra como mínimo un proceso con el nombre "DPAServer" significa que el servidor se está ejecutando correctamente, y si no se encuentra ninguno significa que el servidor no se está ejecutando.

ACTIVACION SERVIDOR DPA

Para activar el servidor DPA basta con seleccionar una consola de texto cualquiera y entrar en una cuenta automática creada a tal efecto que lanza automáticamente el servidor DPA:

```
login: dpaserver
password: dpa
```

El servidor DPA se lanza automáticamente, monstrando el mensaje "DPA SERVER INITIALITED" por pantalla y bloqueando la consola de texto desde la que se ha lanzado, por lo que esta consola ya no se puede usar y no responde ni siquiera a la secuencia (<CTRL> + <C>). En algunas ocasiones puede que el servidor indique un error por pantalla y no se inicialice. Esto suele ocurrir tras haber desactivado previamente el servidor, pues el sistema operativo necesita un

tiempo para cerrar los sockets y demás recursos que usa el servidor. En este caso basta con esperar un tiempo (un par de minutos suele ser más que suficiente) y volver a lanzar el servidor (para lo cual habrá que salir de la cuenta con el comando "exit" y volver a entrar en la misma).

DESACTIVACION SERVIDOR DPA

Para desactivar el servidor DPA existen 2 procedimientos:

- procedimiento por **comandos**: hay que buscar la consola activa del servidor DPA y leer la información que muestra por pantalla. Uno de los datos que muestra es el número de proceso que hay que matar para desactivar el servidor mediante el mensaje "TO SHUTDOWN THIS DPA SERVER: KILL p", donde "p" es el número de proceso que tiene asignado el servidor DPA. Por lo tanto, basta acceder al ordenador que alberga al servidor DPA mediante la cuenta de **root**, con contraseña **mazinger** (cualquier otra cuenta no dejará desactivar el servidor DPA, pues no se es propietario del mismo):

  ```
  login: root
  password: mazinger
  ```

  y ejecutar el comando "kill" con este parámetro:

  ```
  [root@dpa] kill p
  ```

  donde hay que sustituir "p" por el número de proceso adecuado.

  NOTA: Es MUY IMPORTANTE NO ejecutar el comando "kill" con la clásica opción "-9":

  `[root@dpa] kill -9 p`

  pues esto desactivaría inadecuadamente el servidor DPA y dejaría todos los recursos asignados al mismo sin liberar: sockets, zonas de memoria compartida, ...

  El número de proceso "p" a usar con el comando "kill" también se puede obtener de otra forma en modo comando, mediante el comando "ps". Si ejecutamos:

  ```
  [root@dpa] ps –ax | grep DPA
  ```

  el primer proceso existente que se llame "DPAServer" (puede y suele haber más de uno) se corresponde con el servidor DPA, por lo que el número de proceso asociado que muestre el comando "ps" es el número "p" buscado.

- procedimiento por **administración remota**: para gestionar adecuadamente el servidor DPA se ha implementado una herramienta de administración remota que monitoriza su funcionamiento y permite realizar algunas operaciones de configuración sobre el mismo, como desactivarlo o reiniciarlo. Para activar el dicha herramienta de administración remota con seleccionar una consola de texto cualquiera y entrar en una cuenta automática creada a tal efecto que lanza automáticamente el administrador remoto:

```
login: dpaadmin
password: dpa
```

Tras iniciar la herramienta se presenta por pantalla un menú en modo texto, y seleccionando la opción 8 ("RESTART") se resetea el servidor DPA y se vuelve a su estado inicial o mediante la opción 9 ("SHUTDOWN") se desactiva el servidor DPA.

- ***ROBOT P2AT:***

Existen diferentes robots en el Departamento de Tecnología Electrónica, cada uno de los cuales usa sus propios comandos de movimiento y exploración del entorno, por lo que cualquiera que necesite usar un robot necesita conocer sus comandos específicos. Esta solución no parece muy adecuada en un entorno de pruebas como es el que existe en el departamento, pues implica que cada vez que se quiera cambiar de robot habría que reescribir toda la parte del código de aquella aplicación que accede al mismo.

Para evitar esta situación y garantizar una total transparencia de cara al usuario final se ha implementado un servidor específico para cada robot, similar al servidor DPA, que siempre presenta los mismos comandos de movimiento y de acceso a los sensores de cara al usuario, proporcionando un interface estandar de acceso a cualquier robot. Obviamente se debe implementar un servidor para cada robot, de forma que aunque el interface de cara al usuario es siempre el mismo (los mismos comandos), la parte de este servidor que transforma estos comandos estandar en los comandos adecuados para cada robot es específica de cada uno de ellos. De esta forma, gracias a este servidor, cambiar de robot NO implica el cambio de ninguna línea de código en las aplicaciones que acceden a los mismos, pues se mantienen los mismos comandos y estructura de datos que llevan la información de los sensores.

Actualmente existen 2 robots en el departamento plenamente operativos, un Nomad 200 y un Pioneer P2AT, si bien es el P2AT el más usado. Este servidor sólo se ha implementado de momento para el dicho robot P2AT, por lo que, de momento, la arquitectura no es aplicable al Nomad 200. La dirección IP del robot P2AT es 10.0.0.1.

Para usar el robot P2AT en el contexto de la arquitectura de control del Departamento de Tecnología Electrónica es IMPRESCINDIBLE, pues, activar el servidor específico del robot P2AT. Tras finalizar su operación, es necesario desactivar dicho servidor antes de apagar el robot P2AT.

ACTIVACION SERVIDOR P2AT

Para activar este servidor sólo hay que encender el robot P2AT, el cual opera bajo el sistema operativo Linux, y entrar en una cuenta automática creada a tal efecto que lanza automáticamente el servidor del robot:

```
login: p2atserver
```

```
password: p2at
```

con lo que el robot queda ya totalmente operativo para ser usado en la arquitectura de control.

NOTA: Por problemas de configurarión al principio el robot suele mostrar un mensaje de error por pantalla. Basta pulsar <ENTER> para acceder al "login" normal e ignorar el mensaje de error

DESACTIVACION SERVIDOR P2AT

Para desactivar el servidor P2AT basta con detener su ejecución mediante la secuencia <CTRL>+<C>:

```
[p2atserver@p2at] <CTRL>+<C>
```

y proceder a apagar el robot mediante el comando "halt" (esto SOLO se puede hacer desde la cuenta **root**, por lo que es necesario entrar en dicha cuenta mediante el comando "su", recordando que la contraseña es **mazinger**):

```
[p2atserver@p2at] su          {password: mazinger}
[root@ p2at] halt
```

con lo que el sistema comienza a cerrarse y pasado un tiempo (tras el mensaje "System Halted") se puede ya apagar.

PORTATIL

Es importante observar que el robot P2AT no posee ningún enlace radio para comunicarse con la red del departamento, por lo que en teoría se necesitaría un enlace ethernet mediante el cable adecuado para que cualquier aplicación tuviese acceso al mismo. Esto es inviable en un sistema móvil, por lo que se ha usado otra estrategia para garantizar la movilidad del sistema. Sobre el robot P2AT se ha ubicado un portátil, que posee un enlace ethernet y un enlace radio. El enlace ethernet se conecta directamente al robot P2AT, y mediante el enlace radio se conecta el portátil a la red del departamento. De esta forma, gracias al portátil (el cual actúa como un ROUTER) se puede acceder al robot P2AT vía radio. Por lo tanto, es necesario encender el portátil y colocarlo sobre el robot P2AT, conectándolo al mismo medio un CABLE DE RED CRUZADO (si el cable de red con con el que se conecta no es cruzado no funcionará). El portátil se configura automáticamente como Router tras arrancar, por lo que no es necesario entrar en ninguna cuenta. La IP del portátil es 192.168.187.117.

- ***MÓDULOS:***

Una vez lanzado el servidor DPA y el servidor del robot se pueden lanzar ya las aplicaciones específicas que controlan el funcionamiento del robot. En el Departamento de Tecnología Electrónica se han implementado diferentes módulos, todo operando sobre Linux, cada uno de los cuales se encarga de controlar una parte de la funcionalidad necesaria para que el robot navege libremente por su entorno.

Los módulos IMPRESCINDIBLES para que el robot pueda navegar son (existen más con otras funcionalidades añadidas, como exploración inteligente del entorno con algoritmos genéticos y demás, pero no son estrictamente necesarios para operar con el robot):

- IFRobot: módulo que envía comandos de movimiento al robot y recibe información de los sensores.

- Navigation: módulo que permite la navegación reactiva del robot mediante los campos potenciales para evitar colisiones con obstáculos de su entorno, tanto fijos como móviles.

- MapMetric: módulo que genera un mapa métrico del entorno donde se encuentra el robot a partir de la información de los sensores.

- PathPlanning: módulo que calcula un camino para dirigir al robot desde la posición actual hacia una posición determinada.

- IFUser: módulo interface de usuario que permite enviar órdenes al robot (dirigirse a un lugar, explorar el entorno, ...).

- P2AT: módulo encargado de configurar adecuadamente TODOS los parámetros necesarios en el funcionamiento del robot (tamaño de celda del mapa métrico, distancia de seguridad para evitar colisiones, ...).

Para poder operar con el robot, pues, es necesario lanzar como mínimo estos 6 módulos. Todos son ejecutados en modo texto, excepto el interface de usuario "IFUser" que está implementado en modo gráfico. Al ser la arquitectura de control distribuida gracias a la existencia del servidor DPA se pueden ejecutar estos módulos en cualquier ordenador, si bien se ha comprobado que el funcionamiento del sistema es correcto (es decir, que la carga del sistema no es excesiva y no se ralentiza la operación del mismo) al ejecutar los 5 módulos en modo texto ("IFRobot", "Navigation", "MapMetric", "PathPlanning" y "P2AT") en un único ordenador y el módulo en modo gráfico ("IFUser"), que es el que mayor número de recursos consume por ser gráfico, en otro ordenador, por lo que esta va a ser la distribución descrita en lo sucesivo. En concreto los 5 módulos en modo texto se van a ejecutar sobre el ordenador 192.168.187.66 y el módulo en modo gráfico en el ordenador 192.168.187.65.

Para ejecutar los módulos basta entrar en los correspondientes ordenadores (192.168.187.66 y 192.168.187.65) con una cuenta apropiada creada a tal efecto:

```
login: robot
password: p2at
```

y abrir varias consolas o terminales para ejecutar los distintos módulos (el sistema operativo de estos ordenadores está configurado para arrancar en modo gráfico y no en modo consola). Para arrancar estos módulos es necesario indicarles mediante 2 parámetros la dirección IP del host que alberga al servidor DPA y el puerto de acceso al mismo (esta información la muestra por pantalla el servidor DPA en la

consola de texto donde se activa). Así pues, en el ordenador 192.168.187.66 habrá que ejecutar en 5 consolas o terminales distintos:

```
[robot@pc7066te] IFRobot 192.168.187.67 7000
[robot@pc7066te] Navigation 192.168.187.67 7000
[robot@pc7066te] MapMetric 192.168.187.67 7000
[robot@pc7066te] PathPlanning 192.168.187.67 7000
[robot@pc7066te] P2AT 192.168.187.67 7000
```

y en el ordenador 192.168.187.65 habrá que ejecutar en 1 consola o terminal:

```
[robot@pc7065te] IFUser 192.168.187.67 7000
```

NOTA: Se han creado unos accesos directos a estos módulos en el escritorio de estos dos ordenadores, por lo que no es necesario ejecutarlos en modo texto mediante los comandos anteriores si no se desea: basta con hacer "click" en estos iconos para lanzarlos con lo parámetros adecuados.

Es necesario saber que, como es lógico, hasta que TODOS los parámetros de funcionamiento del sistema no estén correctamente configurados el mismo no puede comenzar a operar. Como el módulo responsable de configurar dichos parámetros es el "P2AT", esto significa que hasta que este módulo no se ejecute el resto de módulos no comenzarán a operar, por lo que estarán en un estado de espera hasta que el módulo "P2AT" se ejecute. Sin embargo, el orden con el que se inicien los módulos es TOTALMENTE indiferente. La sincronización correcta entre ellos se realiza de forma automática y cualquiera de ellos se puede ejecutar el primero o el último.

Por último, hay que considerar un último aspecto sobre el módulo "IFRobot". Este es el único módulo que accede al robot del sistema (enviando comandos de movimiento y recibiendo información de los sensores), por lo que este módulo debe saber dónde se encuentra el robot y cómo se accede a él. Al usar un portátil como Router, el acceso al robot NO es directo, es decir, HAY que indicarle al sistema que para acceder al robot (IP 10.0.0.1) es necesario dirigirse al portátil que actúa como Router (IP 192.168.187.117). Esto se realiza mediante el comando "route" (tanto en Linux como en Windows). Así pues, en aquel ordenador desde el que deseemos tener acceso al robot es IMPRESCINDIBLE actualizar la tabla de encaminamiento e indicarle al sistema cómo se accede al robot (a través del portátil). Esto hay que hacerlo indistintamente si estamos ejecutando una aplicación en Linux o en Windows. Por lo tanto, en aquel ordenador donde se ejecute el módulo "IFRobot" (que es el único que accede al robot) hay que ejecutar el siguiente comando en Linux (pues el módulo "IFRobot" opera en Linux) para actualizar la tabla de encaminamiento del sistema (esto SOLO se puede hacer desde la cuenta **root**, por lo que es necesario entrar en dicha cuenta mediante el comando "su", recordando que la contraseña es **mazinger**):

```
[robot@pc7066te] su                {password: mazinger}
[root@pc7066te] route add 10.0.0.1 gw 192.168.187.117
[root@pc7066te] exit
```

Es recomendable salir de la cuenta **root** mediante el comando "exit" antes de seguir operando con el sistema.

Si se desea acceder al robot desde cualquier aplicación bajo Windows (de momento no es el caso en el sistema implementado) se debe modificar la tabla de encaminamiento ejecutando el siguiente comando desde una ventana de comandos MS-DOS:

```
C:\> route add 10.0.0.1 192.168.187.117
```

Antes de finalizar, indicar que el ordenador 192.168.187.66 está configurado para actualizar automáticamente su tabla de encaminamiento al arrancar en Linux, por lo que NO es necesario ejecutar el comando anterior. Por eso es recomendable ejecutar el módulo "IFRobot" sobre dicho ordenador.

MANEJO DEL ROBOT

El manejo del robot es bastante sencillo mediante el interface de usuario mostrado por el módulo "IFUser". Se muestra el mapa métrico que está creando el robot, así como el robot (un círculo con una línea apuntando en la dirección de avance del robot). Para moverlo hacia una posición del mapa métrico basta situar el ratón sobre dicha posición y pulsar uno de los 2 botones del ratón, con la siguiente diferencia:

- si se pulsa el botón IZQUIERDO del ratón el robot se dirige hacia la posición de destino indicada usando el "PathPlanning", es decir, calculando un camino libre de obstáculos hacia dicha posición y siguiéndolo.

- si se pulsa el botón DERECHO del ratón el robot se dirige hacia la posición de destino indicada sin usar el "PathPlanning", es decir, en línea recta y usando únicamente los campos potenciales para evitar obstáculos.

Así mismo, existe un botón de "STOP" que detiene completamente al robot.

Por último, indicar algunas recomendaciones de manejo del robot. Si bien el módulo de navegación reactiva y evitación de obstáculos "Navigation" opera correctamente en general, existen algunas situaciones donde las lecturas de los sensores pueden ignorar un obstáculo que se encuentra extremadamente cerca del robot (por ejemplo cuando el ángulo de incidencia del sonar es muy elevado y se aleja de la perpendicular), por lo que en ciertas circunstancias se pueden producir colisiones con obstáculos del entorno, tanto fijos como móviles. Así pues, es recomendable realizar un seguimiento visual del robot para detenerlo en caso de peligro de colisión. Además, hay que tener en cuenta que la ejecución de órdenes no es inmediata, tardando aproximadamente 1 segundo en ejecutarse (fruto del inevitable retardo de operación de los módulos). Por lo tanto, desde que el botón "STOP" se pulsa hasta que el robot efectivamente se para transcurre aproximadamente un segundo, tiempo que hay que tener en cuenta para evitar colisiones.

DESCRIPCIÓN DE LA "ARQUITECTURA DE CONTROL DPA"

1.1. DESCRIPCIÓN DEL TRABAJO DE INVESTIGACIÓN REALIZADO

1.1.1. INTRODUCCIÓN

En los últimos años, el campo de la robótica móvil ha experimentado un importante resurgir en el interés de la comunidad científica e investigadora. Un gran número de investigadores han centrado sus esfuerzos en el desarrollo de agentes autónomos móviles, más conocidos como robots, capaces de desenvolverse en entornos total o parcialmente desconocidos sin ningún tipo de intervención humana. El principal fruto de este esfuerzo es la aparición de nuevas aplicaciones y algoritmos cada vez más sofisticados, los cuales han permitido la aparición de un gran número de robots cada vez más potentes.

No obstante, a pesar de dicho esfuerzo y aumento en el bagaje científico disponible para la implementación de dichos sistemas autónomos móviles, la comunidad científica en general todavía no ha dado solución a un gran número de cuestiones en su carrera por el desarrollo de robots de propósito general. El principal motivo que origina esta incertidumbre se deriva de la dificultad en el estudio de dicho tipo de sistemas.

Se puede definir un robot como "un sistema capaz de extraer información del entorno que le rodea y usar el conocimiento que tiene de él para moverse sin peligro de una forma útil e intencionada". Es decir, un robot es un sistema "inteligente" que es capaz de adaptarse a su entorno y llevar a cabo diferentes tareas de forma no supervisada, modificando dicho entorno si es necesario para la consecución de determinados fines.

Desde el punto de vista de la ingeniería, el problema de diseño de robots se aborda bajo una perspectiva de diseño modular, en la que el comportamiento global del robot se divide en una serie de tareas básicas menores que deberán ser integradas adecuadamente para formar el sistema completo. Obviamente, esta subdivisión en tareas dependerá de la aplicación final para la que se deba diseñar el robot.

Existen diversas aproximaciones y algoritmos para resolver las diferentes tareas básicas que debe realizar un robot, cada una con sus especificaciones, sus necesidades y sus prestaciones. La parte más compleja en el diseño de robots consiste en la integración de las distintas tareas básicas en las que se divide el comportamiento global del robot, mediante una estructura que permita la cooperación de todas ellas simultáneamente y en paralelo para poder obtener así la funcionalidad final buscada. Esta estructura básica que permite la cooperación paralela de las diferentes tareas básicas es lo que se conoce como arquitectura de control.

Una arquitectura de control, pues, es un mecanismo que permite integrar diferentes tareas básicas, definiendo la interacción y controlando el flujo de información entre ellas, lo cual posibilita la cooperación paralela de las mismas a fin de implementar el comportamiento global deseado, el cual determinará la interacción entre el robot y el entorno que lo rodea.

Resumiendo, el comportamiento global de cualquier robot puede ser dividido en una serie de tareas básicas menores, dependiendo esta descomposición de la aplicación final para la cual esté diseñado el robot. Es la correcta cooperación paralela de todas estas

tareas la que permite la implementación del comportamiento global final, y es la arquitectura de control la que debe facilitar los mecanismos adecuados para la correcta interacción entre las diferentes tareas básicas que constituyen el sistema. Una implementación modular y flexible de la arquitectura de control es con mucho la parte más importante y compleja en la implementación de cualquier robot, pues es directamente responsable de las prestaciones finales del mismo.

1.1.2. ARQUITECTURA DE CONTROL PROPUESTA

Con el objetivo de implementar un agente autónomo móvil dotado con comportamientos inteligentes se pretende desarrollar una arquitectura de control potente, modular, flexible y robusta. Independientemente del tipo de arquitectura de control implementada, esta debe integrar las diferentes tareas básicas de las que consta el robot, garantizando la interacción y controlando el flujo de información entre ellas. Esta es la finalidad última de cualquier arquitectura de control, ya sea deliberativa, reactiva o híbrida.

La primera fase en la implementación de la arquitectura de control consiste en el desarrollo software de la misma, es decir, en la implementación de los mecanismos adecuados capaces de permitir la creación de los diferentes módulos y el intercambio de información entre ellos, a fin de permitir la correcta cooperación de los mismos.

El objetivo final es el de diseñar una arquitectura de control que, mediante el uso de las librerías adecuadas, permita que todo el esfuerzo de diseño del agente autónomo móvil se centre en la división de los comportamientos globales deseados en tareas básicas menores, y estas a su vez sean implementadas mediante bs módulos adecuados. Serán los mecanismos software implementados mediante estas librerías los que permitirán que la arquitectura de control gestione el intercambio de información entre los distintos módulos de una forma eficaz, todo ello de una forma totalmente transparente al diseñador.

Para garantizar que la cooperación entre módulos se pueda llevar a cabo, la arquitectura de control implementada tiene que poseer una serie de características mínimas:

- Paralelismo: debe permitir que diferentes módulos se ejecuten concurrentemente a fin de que las diferentes tareas básicas se puedan llevar a cabo de forma concurrente.
- Interconexión: debe permitir el flujo de información entre módulos distintos para garantizar la cooperación entre ellos.
- Distribuida: debe permitir que diferentes módulos sean ejecutados en diferentes procesadores a fin de distribuir la carga total del sistema.

Debido a que Linux fue concebido originalmente con una filosofía multiusuario y está especialmente diseñado para procesamiento paralelo y distribuido es la plataforma de desarrollo elegida para la implementación software de la arquitectura de control. Además, para garantizar un sistema distribuido es indispensable el uso de sockets para permitir el intercambio de información entre diferentes máquinas, por lo que los sockets deberán estar presentes en la implementación de la arquitectura de control propuesta.

Existen diversas aproximaciones a la implementación software de una arquitectura de control. Últimamente se han desarrollado varios sistemas, teniendo en común muchos de ellos el empleo de servidores capaces de gestionar el intercambio de información distribuida en el sistema, basándose fundamentalmente en la existencia de un agente

específico que controla todo el proceso de intercomunicación entre los módulos del sistema.

La arquitectura implementada se ha denominado *DPA* ("Distributed and Parallel Architecture"). Su filosofía se basa en la existencia de un servidor de datos que mantiene actualizada toda la información que debe ser compartida en el sistema.

El servidor central de la arquitectura *DPA* implementada se ha denominado *DPAServer*. A este servidor de datos se conectan todos los módulos del sistema, realizando operaciones de lectura y escritura sobre la información contenida en el servidor. Así pues, cualquier módulo puede actualizar los datos que almacena el servidor, estando así disponibles para cualquier otro módulo que los necesite.

La solución propuesta sigue el esquema mostrado en la Figura 1.

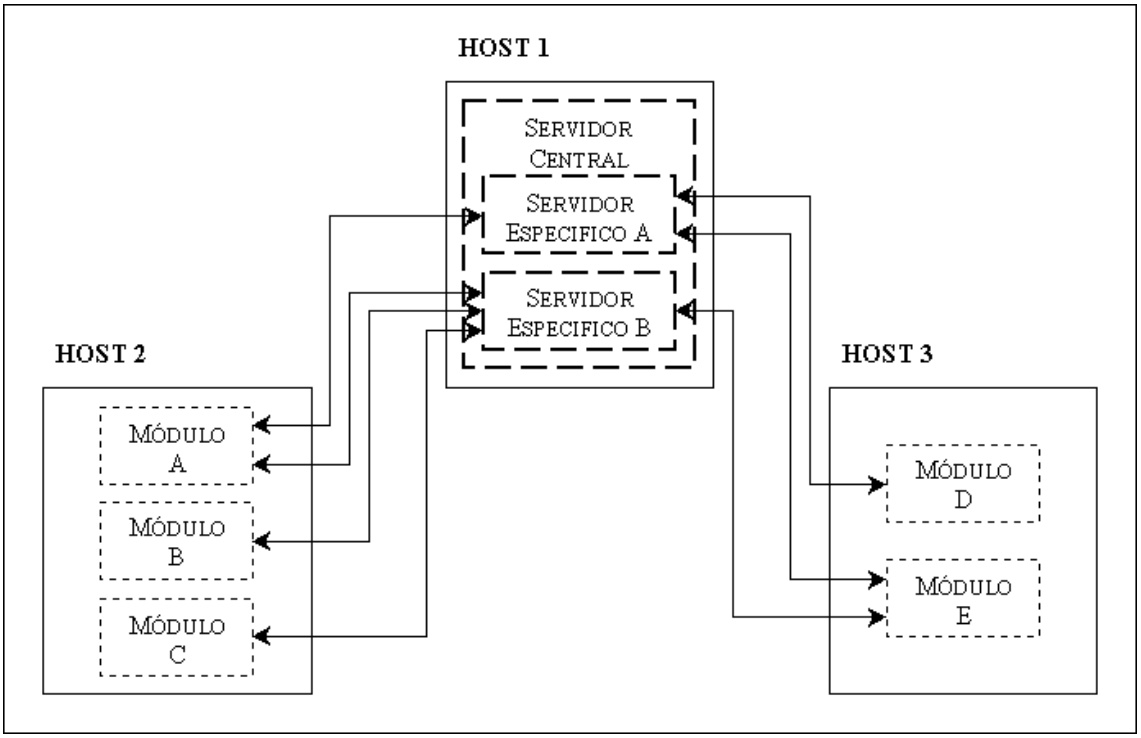

Figura 1: Arquitectura de control propuesta

Una filosofía de este tipo tiene importantes y numerosas ventajas:

- Facilidad de uso del sistema
- Eficiencia del sistema
- Fácil depuración del sistema
- Monitorización del sistema
- Fácil ampliabilidad del sistema
- Flexibilidad del sistema
- Independencia de la plataforma

La idea final de la arquitectura, pues, es que el servidor central esté siempre activo en una máquina conocida a través de un puerto conocido, como lo están servicios más conocidos como FTP, Telnet, WWW, ... De esta forma estará siempre disponible para

cualquier sistema que se quiera implementar sobre dicha arquitectura. Además, al ser un servicio genérico de intercambio de información entre módulos distribuidos, ni siquiera es necesario que el sistema final implementado sea un sistema específicamente diseñado para desarrollar un robot: cualquier aplicación distribuida que necesite un proceso de intercambio de datos puede utilizar esta arquitectura.

Desde el punto de vista de la configuración y del control del sistema, sería interesante realizar una monitorización interactiva del funcionamiento del servidor, así como una reconfiguración on-line de su funcionamiento. Esta funcionalidad es extremadamente útil para la depuración del sistema a desarrollar, pues permite controlar y visualizar directamente el flujo de información de los módulos del sistema. La mejor alternativa para implementar esta característica es la construcción de un administrador remoto que pueda ser ejecutado en cualquier máquina y que se conecte al servidor central mediante un protocolo específico que permita la reconfiguración del mismo. El administrador remoto de la arquitectura *DPA* implementada se ha denominado *DPAAdmin*.

La facilidad de uso de la arquitectura de control, de cara al usuario final, depende fundamentalmente de la facilidad de uso de las librerías implementadas para crear y acceder a las conexiones. En este sentido, las librerías implementadas han sido desarrolladas con la meta final de la transparencia y la facilidad de uso. El lenguaje escogido para implementar las librerías ha sido C, por su universalidad y por ser uno de los lenguajes más extendidos hoy en día.

Las funciones implementadas permiten que cualquier módulo pueda explotar por completo la funcionalidad de la arquitectura de control implementada, posibilitando el acceso selectivo a los datos compartidos en la arquitectura. Estas librerías han sido implementadas tanto en Linux como en Windows, lo cual posibilita que tanto los módulos desarrollados en Linux como los módulos desarrollados en Windows puedan usar la arquitectura de control, lo que pone de manifiesto la independencia de la arquitectura respecto de la plataforma.

1.1.3. IMPLEMENTACIÓN DE LAS LIBRERÍAS

La facilidad de uso de la arquitectura de control, de cara al usuario final, depende fundamentalmente de la facilidad de uso de las librerías implementadas para crear y acceder a las conexiones. En este sentido, las librerías implementadas han sido desarrolladas con la meta final de la transparencia y la facilidad de uso.

El lenguaje escogido para implementar las librerías ha sido C, por su universalidad y por ser uno de los lenguajes más extendidos hoy en día. Las funciones implementadas para usar la arquitectura de control son:

- Connection NewConnection(char *host, int port, char *name, int size): crea una conexión identificada por el nombre *name* y de un tamaño *size* en el servidor central ubicado en la máquina *host* y al que se accede por el puerto *port*, inicializando la conexión a cero. Devuelve la variable de tipo *Connection* que será la que identifique la conexión creada y mediante la cual se accederá a los datos contenidos en ella. Si la conexión ya existía sencillamente se conecta a ella para compartir los datos que contenga.
- int InputConnection(Connection conn, void *data, int pos, int size): accede a la porción de datos que comienza en la posición *pos* y tiene un tamaño *size* de la conexión identificada por *conn*, guardando los datos en la variable *data*. Si *size* es

cero, accede a los datos desde la posición *pos* hasta el final de los datos almacenados en la conexión. Devuelve la cantidad de datos leídos.

- int OutputConnection(Connection conn, void *data, int pos, int size): modifica la porción de datos que comienza en la posición *pos* y tiene un tamaño *size* de la conexión identificada por *conn*, guardando los datos de la variable *data*. Si *size* es cero, modifica los datos desde la posición *pos* hasta el final de los datos almacenados en la conexión. Devuelve la cantidad de datos escritos.
- int SizeConnection(Connection conn): devuelve el tamaño de la conexión identificada por *conn*. Esta función es útil por si algún módulo desea conocer explícitamente el tamaño de una conexión, pues este tamaño viene determinado por el primer módulo que creó la conexión.
- int CloseConnection(Connection conn): destruye la conexión identificada por *conn*. El uso de esta función es opcional, pues el sistema cuenta con un sistema de detección automático para destruir conexiones no usadas por ningún módulo, aunque en algunos casos puede ser recomendable destruir una conexión que no va a ser usada más tiempo. Devuelve un indicador de operación correcta (0) o de error (-1).

Las funciones implementadas permiten la creación de una conexión, su destrucción, y el acceso selectivo a los datos que contiene (no necesariamente a todos los datos que contiene la conexión). Mediante estas funciones cualquier módulo puede explotar por completo la funcionalidad de la arquitectura de control implementada. Con la función *NewConnection* se accede a la conexión deseada, identificada por su nombre. Asímismo, la función *InputConnection* tiene acceso a la información contenida en dicha conexión, mientras que la función *OutputConnection* modifica la información contenida en dicha conexión.

De esta forma, si varios módulos quieren intercambiar información tan sólo tienen que crear una conexión con el mismo nombre y leer o escribir en ella los datos oportunos. La lectura y escritura en las conexiones tienen control de acceso, es decir, mientras un módulo está leyendo o escribiendo datos de la conexión el resto de módulos no puede acceder a los datos de la misma. Este mecanismo mantiene la integridad de los datos existentes en la conexión. Un ejemplo de la utilización de estas librerías es el que se muestra en la Figura 2.

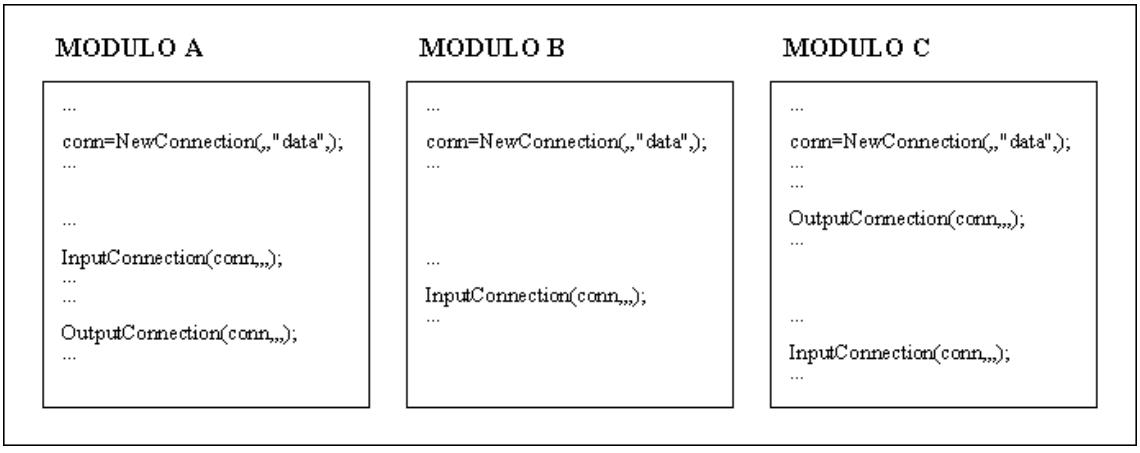

Figura 2: Ejemplo de uso de las librerías

Estas librerías han sido implementadas tanto en Linux como en Windows, lo cual posibilita que tanto los módulos desarrollados en Linux como los módulos desarrollados en Windows puedan usar la arquitectura de control, lo que pone de manifiesto la independencia de la arquitectura respecto de la plataforma. En estos casos, sin embargo, hay que tener especial cuidado con la compatibilidad entre los tipos de datos en ambas plataformas. Este aspecto debe ser contemplado por los propios módulos que efectúan el intercambio de datos, pues los servidores de datos específicos no gestionan el contenido de la información que almacenan, como no puede ser de otra forma si se quiere garantizar la generalidad del sistema.

1.1.4. PRUEBAS Y RESULTADOS

Una vez implementada la arquitectura de control se plantea la necesidad de analizar en profundidad su comportamiento mediante la realización de las pruebas necesarias. El aspecto básico a verificar es el correcto funcionamiento en la capacidad de intercambiar información entre módulos en un sistema distribuido.

La estrategia más adecuada parece la realización de una única prueba global del sistema completo, donde el objetivo básico de la prueba será verificar que diferentes módulos distribuidos a lo largo del sistema son capaces de intercambiar información entre sí.

Los módulos diseñados para intercambiar información pueden tener cualquier funcionalidad, aunque con el fin de demostrar la utilidad de la arquitectura de control en la implementación de sistema orientados a la robótica móvil, los módulos diseñados van a corresponder a una implementación real. Como el objetivo de esta prueba no es el de implementar un sistema complejo, sino más bien el de validar el correcto funcionamiento de la arquitectura de control, el sistema tan sólo va a construir un mapa probabilístico del entorno en el que se encuentra.

El robot utilizado para ello es un Nomad 200, el cual posee 16 sensores de ultrasonidos, 16 sensores de infrarrojos, 1 brújula digital y un sistema de odometría que da la posición del robot en cada instante. Para facilitar el desarrollo de la prueba, y sin pérdida de generalidad, se va a utilizar un simulador en lugar del robot real.

El sistema básico constará de 3 módulos:

- *Sensores*: encargado de recoger las medidas del entorno, mediante los sensores del robot, la brújula y el sistema de odometría.
- *Mapa*: encargado de construir el mapa probabilístico a partir de las medidas del entorno recogidas por el robot.
- *Visor*: encargado de representar el mapa probabilístico que está siendo construido por el sistema.

Un diagrama funcional del sistema es el que se muestra en la Figura 3.

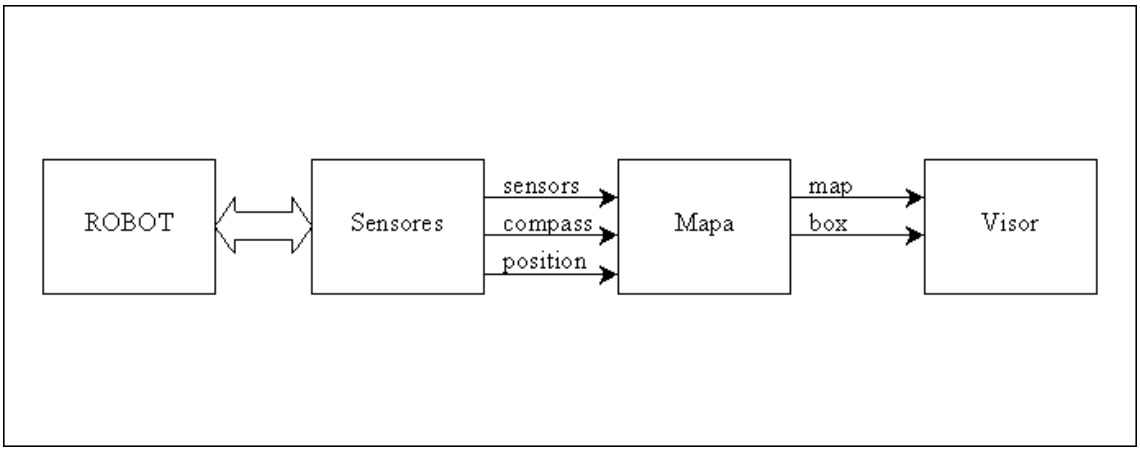

Figura 3: Diagrama funcional del sistema de prueba

Para poner de manifiesto la independencia de la arquitectura de control implementada respecto a la plataforma de desarrollo de los diferentes módulos, se van a implementar los módulos *Sensores* y *Mapa* sobre Linux y el módulo *Visor* sobre Windows.

Por último, para verificar el correcto funcionamiento de la arquitectura de control, se va a utilizar el administrador remoto, lo cual también permitirá verificar el funcionamiento del mismo.

El sistema de pruebas, pues, constará de 6 máquinas pertenecientes al Departamento de Tecnología Electrónica de la Universidad de Málaga, usando 2 plataformas (Linux y Windows) y 4 sistemas operativos (Red Hat Linux 4.2, Linux Mandrake 7.0, Linux Mandrake 7.2 y Windows 98), cada una de las cuales tendrá las siguientes características y funcionalidad:

- pc51te: máquina con dirección IP *150.214.59.51* y con Linux Mandrake 7.0, que ubicará el servidor central.
- pc38te: máquina con dirección IP *150.214.59.38* y con Linux Mandrake 7.0, que ubicará el administrador remoto.
- pc55te: máquina con dirección IP *150.214.59.55* y con Red Hat Linux 4.2, que ubicará el simulador del robot.
- pc98te: máquina con dirección IP *150.214.59.98* y con Linux Mandrake 7.0, que ubicará el módulo *Sensores*.
- pc28te: máquina con dirección IP *150.214.59.28* y con Linux Mandrake 7.2, que ubicará el módulo *Mapa*.
- pc93te: máquina con dirección IP *150.214.59.93* y con Windows 98, que ubicará el módulo *Visor*.

El esquema del sistema de prueba es el que se muestra en la Figura 4.

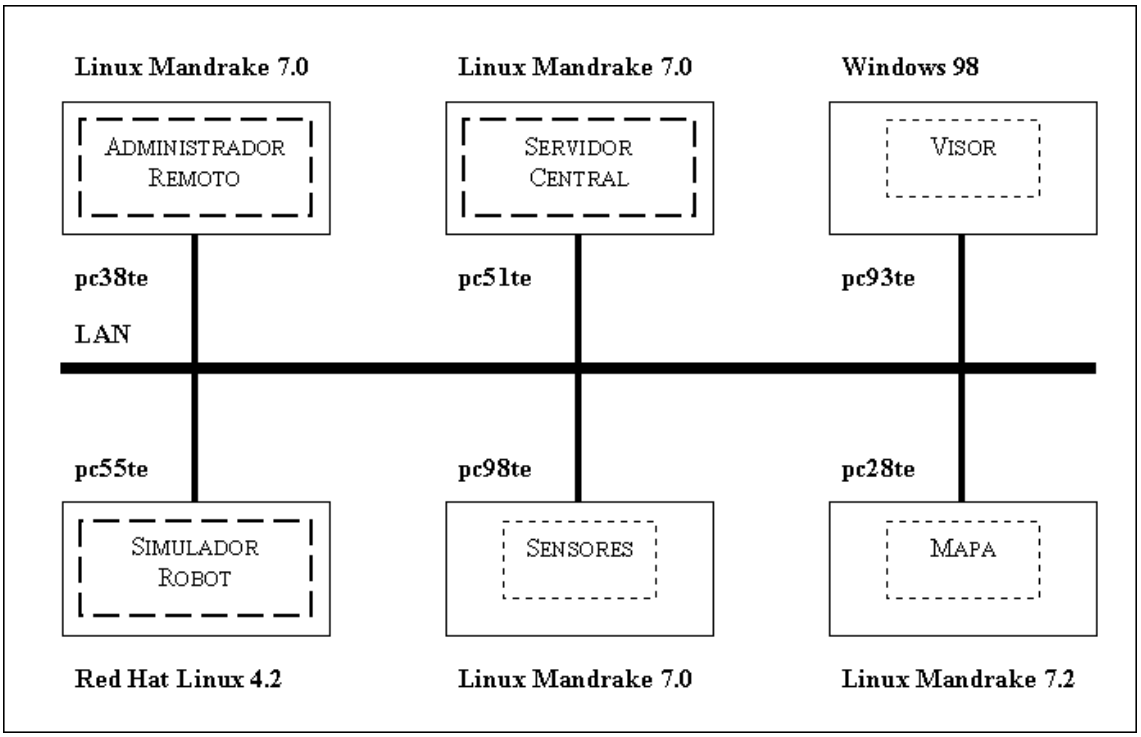

Figura 4: Esquema del sistema de prueba

Es interesante hacer notar que en un sistema final se podrían haber empleado menos máquinas, pues no es necesario que cada módulo se ejecute en una máquina especialmente dedicada para él. Esto se ha hecho para demostrar la flexibilidad y la ilimitada capacidad de distribución de la arquitectura de control implementada respecto al sistema a desarrollar.

El entorno en el cual se va a encontrar el robot, y que será el que deberá representar mediante la contrucción del mapa probabilístico, es el que se muestra en la Figura 5.

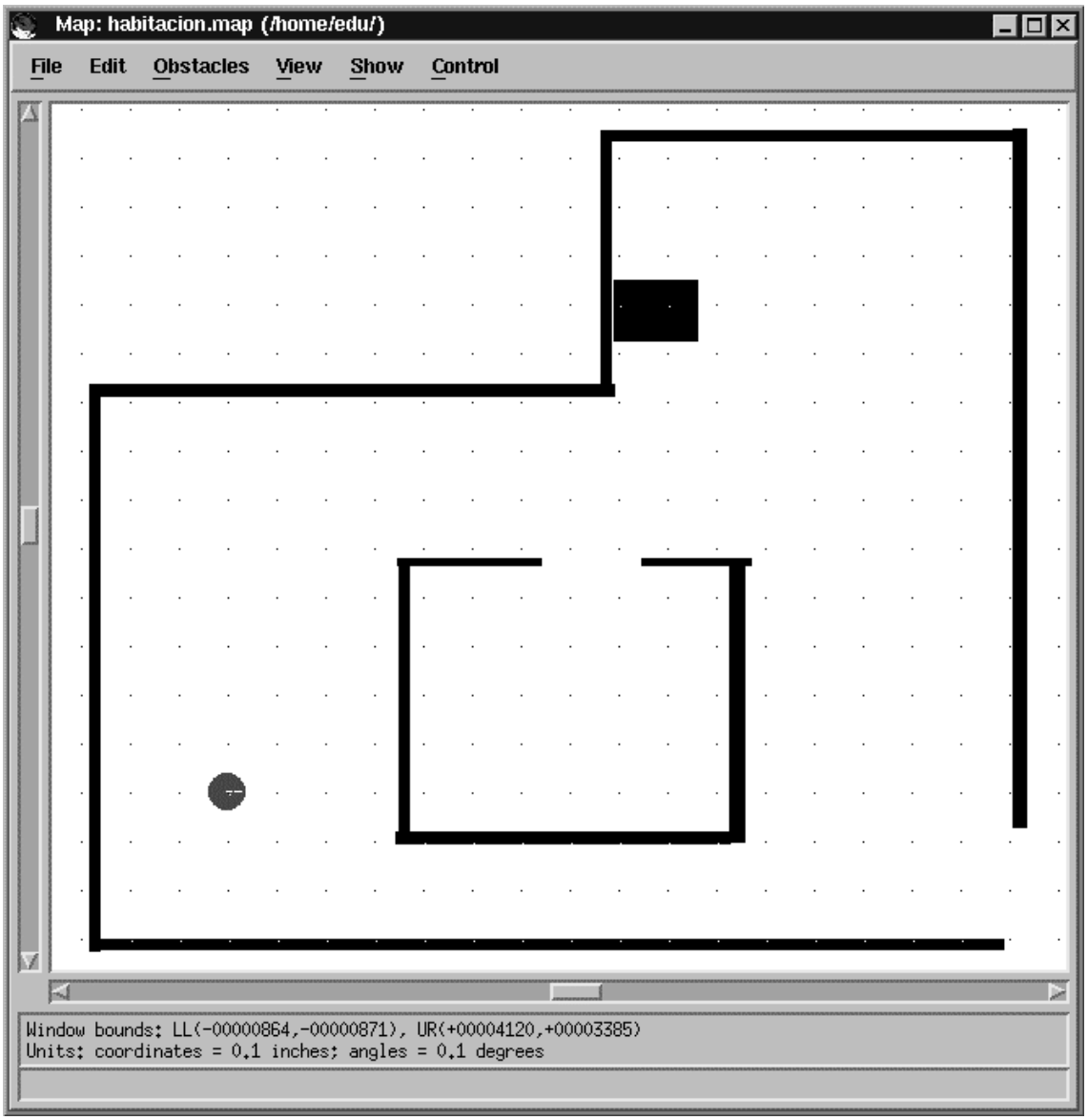

Figura 5: Entorno del robot

Para verificar el correcto funcionamiento de la arquitectura de control implementada hay que comprobar que el intercambio de datos entre los distintos módulos del sistema se efectúa de forma correcta. Esto se puede comprobar verificando que efectivamente el sistema realiza la funcionalidad para la cual estaba diseñado, es decir, la correcta construcción del mapa probabilístico del entorno del robot.

La salida del módulo *Visor* en diferentes instantes de tiempo, que es un interface gráfico que representa el mapa construido y compartido por todo el sistema, es la que se muestra en la Figura 6.

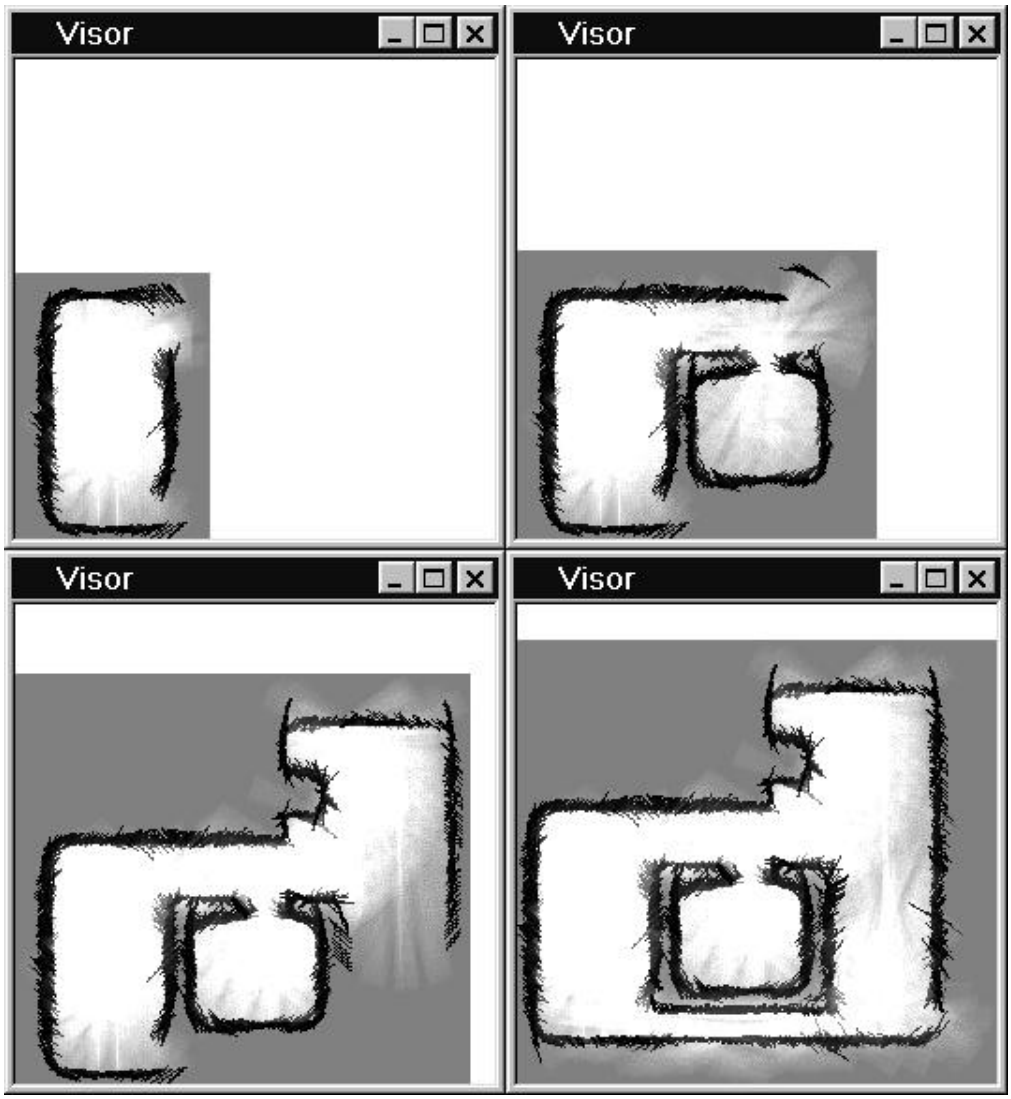

Figura 6: Mapa probabilístico creado

Como se puede observar en la Figura 6, el mapa probabilístico creado se corresponde perfectamente con el entorno del robot de la Figura 5. El que la construcción de este mapa probabilístico sea correcta implica que los datos intercambiados entre los distintos módulos sean correctos, pues para ello los datos del entorno deben ser adquiridos y compartidos correctamente con el resto del sistema.