Disponible en www.sciencedirect.com
ScienceDirect
Revista Iberoamericana de Automática e Informática industrial 9 (2012) 57–68
www.elsevier.es/RIAI

# Diseño y desarrollo de un sistema de transporte reconfigurable para entornos hospitalarios

Joaquín López[a], Diego Pérez[a], Roberto Pinillos[b], Salvador Domínguez[b], Eduardo Zalama[c,*], Jaime Gómez-García-Bermejo[c].

[a] *Dep. Ingeniería de Sistemas y Automática, Universidad de Vigo, 36200 Vigo, España.*
[b] *Centro Tecnológico Cartif, Parque Tecnológico de Boecillo parcela 205, 47151 Boecillo, Valladolid, España.*
[c] *Instituto Tecnologías Avanzadas de la Producción, Universidad de Valladolid, Paseo del Cauce 59, 47011 Valladolid, España*

**Resumen**

El objetivo de este artículo es la presentación de un sistema de transporte automático de materiales (comidas, suministros, ropa y medicamentos) en recintos hospitalarios mediante robots móviles. En el artículo se discuten algunos aspectos del desarrollo de la plataforma física, en particular los relacionados con el sistema de control, la arquitectura software, la generación y seguimiento de trayectorias, la planificación de tareas y el sistema de gestión de tráfico, así como los resultados de la integración del sistema global y su aplicación a un sistema de transporte hospitalario. El sistema global está formado por un conjunto de robots móviles con capacidad de navegación independiente y un servidor central encargado de gestionar las actividades que deben realizar los robots y de organizar el tráfico en los puntos críticos. Aunque los robots realizan las tareas de forma autónoma, la interacción con algunos elementos del entorno tales como ascensores, sensores de presencia en zonas de carga y descarga, y puertas, se realiza también a través de dicho servidor central. Los operadores humanos pueden conectarse al servidor central para cambiar las actividades de los robots o supervisar su estado.


*Palabras Clave:*
Robot móvil, sistema de navegación, vehículo autónomo, planificación de trayectorias, sistema de guiado.

## 1. Introducción

En los últimos años, el amplio volumen de actividad asistencial en hospitales, unido a una complejidad cada vez mayor de los procesos que involucra, ha conducido a un notable aumento de la demanda de calidad y eficiencia de los sistemas de salud en los países desarrollados. Uno de los problemas en los que se ha focalizado la atención en los últimos años ha sido la logística hospitalaria y particularmente, el movimiento de materiales en recintos hospitalarios. Los fabricantes de sistemas de transporte en recintos industriales han visto en los sistemas asistenciales de salud un nicho de mercado donde expandir su tecnología de transporte, en especial la basada en robots móviles o vehículos guiados automáticamente (AGVs). Sin embargo, la introducción de sistemas de transporte en recintos hospitalarios presenta en general una mayor complejidad que en recintos industriales: los materiales a transportar pueden ser muy diversos (comida, suministros, ropa, medicamentos), las distancias a recorrer suelen ser mayores, deben operar en diferentes plantas tomando ascensores y deben realizar una mayor interacción con el entorno. Además hay que extremar aún más si cabe la seguridad, puesto que los sistemas deben operar en recintos en los que hay personal sanitario, enfermos y visitantes, sin olvidar que se debe prestar especial atención al desarrollo de interfaces de gestión del sistema que estén adaptadas a las necesidades de los técnicos de mantenimiento hospitalario, muy diferentes a las de los de mantenimiento industrial.

Estas dificultades hacen que pocas empresas hayan hecho desarrollos de sistemas de transporte específicos para entornos hospitalarios. Entre ellos podemos destacar el robot HelpMate (Engelberger 1993) para el transporte de medicamentos en hospitales. Siemens presenta algunos desarrollos de sistemas basados en guiado mediante tecnología magnética (Pérez et al. 2010). Aethon es una empresa afincada en Pittsburgh que desarrolla el robot TUG, un pequeño trolley de (50x18 cm) capaz de arrastrar carros de medicamentos y pequeños suministros, utilizando navegación basada en mapas. Sin embargo, tres son las compañías que realmente compiten a nivel mundial en el mercado del sistema de transporte hospitalario: Ds Automation (Ds Automation, 2011) con una experiencia de más de 300 robots funcionando en el sector hospitalario, que utilizan un sistema de seguimiento de imanes, JBT Corporation, con gran experiencia en

* Autor en correspondencia.
*Correos electrónicos*: joaquin@uvigo.es(Joaquín López), ezalama@cartif.es(Eduardo Zalama).


doi:10.1016/j.riai.2011.11.008

el desarrollo de sistemas de transporte a nivel industrial y que se ha introducido en Europa en varios hospitales, utilizando principalmente sistemas de navegación basados en balizamiento láser, y por último Swisslog (Swisslog, 2011), una empresa de proyección mundial especialista en el desarrollo de sistemas de transporte, que en los últimos años ha centrado su actividad en el sector hospitalario, incluyendo soluciones de transporte neumático y mediante robots basados en tecnología láser.

El objetivo del presente artículo es la descripción de un sistema de transporte automático de materiales (comidas, suministros, ropa y medicamentos) en recintos hospitalarios mediante robots móviles (AGVs). Frente a otros sistemas de transporte logístico hospitalario, el sistema propuesto se caracteriza por una concepción altamente modular. Esta característica facilita su reconfiguración, la puesta en servicio y la integración de diferentes métodos de localización alternativos que pueden ser gestionados en función de las necesidades de precisión y el coste deseado.

En la sección 2 se presenta la arquitectura global del sistema propuesto, describiendo los diferentes módulos que la componen. En la sección 3 se aborda el seguimiento de trayectorias por parte del robot y la realización de maniobras. En la sección 4 se detallan los diferentes tipos de localización utilizados. Un aspecto importante a considerar es la sensorización del entorno y la interacción del robot con los diversos dispositivos como por ejemplo los ascensores, tema que se trata en la sección 5. En la sección 6 se aborda la gestión de tareas mediante la utilización de redes de Petri, que permite de forma rápida y eficaz la instalación y puesta en servicio de un nuevo proyecto tal como se describe en la sección 7. Finalmente en la sección 8 se presentan los resultados y las conclusiones en la sección 10.

## 2. Arquitectura de control de la aplicación

Para la realización de esta aplicación se ha utilizado el entorno de desarrollo RIDE (López et al., 2011) que permite la creación de proyectos multirobot y multiusuario como el mostrado en la Figura 1. RIDE incluye un conjunto de herramientas para desarrollar componentes de forma modular y con interfaces entre los distintos módulos bien definidas.

El desarrollo de módulos se considera a dos niveles distintos:

- **Módulos de bajo nivel**. Son los empleados en el control interno de un único elemento (robot, sistema de automatización del edificio).
- **Módulos de alto nivel**. Son los módulos empleados en el control del sistema multirobot y multiusuario. El conjunto de módulos utilizados en el bajo nivel se consideran como un único módulo de alto nivel. La Figura 2 muestra los principales módulos de control de alto nivel.

Las herramientas de RIDE para el desarrollo de los módulos de bajo y alto nivel son muy similares, aunque aplicadas a distintos niveles, facilitando de esta manera la formación del desarrollador. Las arquitecturas de los dos niveles son modulares y centralizadas y en ambos casos se usa un sistema de comunicación basado en un esquema de publicación/suscripción de mensajes.

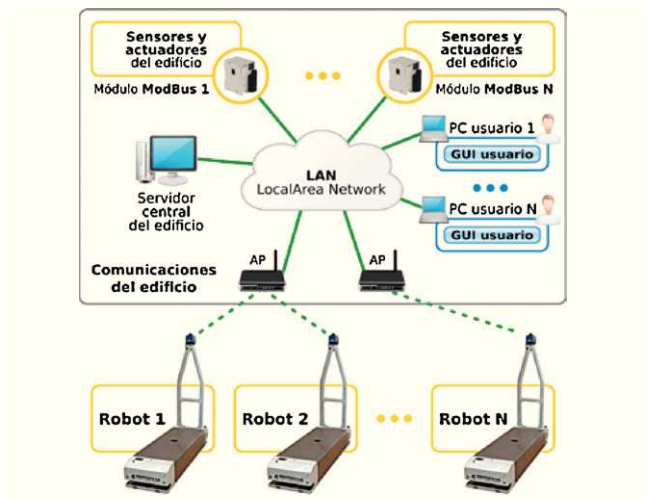


Figura 1: Esquema general del sistema propuesto

El módulo ejecutivo que se encarga de la coordinación de los módulos de cada nivel es el mismo (RoboGraph) (Fernández et al., 2008a). El sistema modular que se presenta en la Figura 2 permite la integración, de forma sencilla, de otros sistemas como pueden ser distintas arquitecturas de navegación. En esta aplicación se han empleado los módulos y herramientas RIDE correspondientes al alto nivel. La integración del esquema de control de bajo nivel (Figura 4) y alto nivel (Figura 2) se realiza a través de un módulo genérico denominado Robot Web Interface que funciona a modo de pasarela para intercambiar información entre el control del robot y el sistema central.

### *2.2 Módulos de alto nivel.*

La comunicación entre los módulos de alto nivel se lleva a cabo mediante JIPC (López et al., 2011). Se trata de un sistema de comunicaciones centralizado basado en el mecanismo de publicación/suscripción de mensajes con un interfaz similar a IPC (Simmons 2005). No obstante, JIPC viene a complementar IPC para aplicaciones multirobot y multiusuario como se indica en (López et al, 2011)

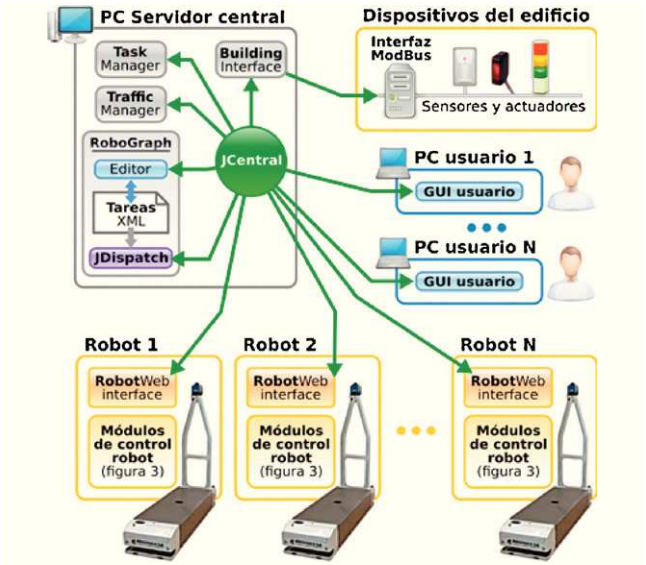


Figura 2: Arquitectura de control de aplicación. Módulos de alto nivel.

Los módulos de alto nivel son:

- **Robot-Web Interface**. Es la interfaz de comunicaciones entre los módulos de control del robot (bajo nivel) y el sistema central. Este módulo es genérico de RIDE. Al arrancar obtiene de un archivo de configuración los tipos de mensajes que han de ser intercambiados entre ambos sistemas. Este archivo de configuración se crea con un sencillo interfaz gráfico donde se van seleccionando entre todos los mensajes existentes en el sistema aquellos que queremos intercambiar.
- **Building Interface**. Este módulo se encarga de obtener la información sobre el estado de los sensores del edificio y publicar mediante mensajes JIPC aquellos cambios que puedan ser importantes para las distintas tareas. Al mismo tiempo, se encarga de actualizar las salidas a los dispositivos conectados como pueden ser ascensores, puertas, etc. Este sistema se describirá con más detalle en la sección 5.
- **Dispatch**. Forma parte de la herramienta RoboGraph (Fernández de al, 2008a) y es el encargado de cargar las tareas y ejecutarlas. Las tareas se definen mediante redes de Petri usando un editor gráfico donde se especifican la secuencia de comandos (publicación de mensajes) y eventos (condiciones sobre la recepción de mensajes) que forman la tarea. El funcionamiento de dicho módulo junto con los dos siguientes se describirá brevemente en la sección 6, mientras que una descripción más completa puede verse en (López et al., 2011).
- **Task manager**. Las redes de Petri son una buena herramienta para modelar y analizar sistemas de eventos discretos. Han sido utilizadas con éxito para describir y analizar tareas que contengan tanto acciones secuenciales como concurrentes en aplicaciones industriales complejas. No obstante, no se adaptan tan bien para la descripción de distintas políticas de gestión de recursos para las cuales se suelen usar lenguajes de programación de alto nivel. Este módulo se encarga de realizar este tipo de gestión y asignación de recursos y tareas.
- **Traffic Manager**. Debido a la existencia de varios robots en el mismo entorno, se hace necesario definir mecanismos para evitar situaciones en las que los robots se queden intentando acceder a la misma zona. Esto se puede dar en situaciones tan habituales como al encontrarse dos robots de frente en un pasillo estrecho por el que sólo cabe uno. El módulo Traffic Manager permite la definición de zonas en las cuales sólo puede haber un robot a la vez. También se encarga de la planificación de caminos.

### *2.3. Módulos de bajo nivel.*

Como se muestra en el esquema de la Figura 3, el control básico de bajo nivel del sistema se realiza utilizando una controladora de diseño propio (GPRMC6LC). Esta controladora se encarga de controlar los dispositivos físicos tales como los drivers de los motores de tracción y dirección y la plataforma de carga. Además proporciona información de estado de dispositivos como el joystick y las setas de emergencia, o el nivel de baterías.

En cuestión de seguridad, la placa dispone de salida redundante, doble, por relé que avisa de posibles problemas en el hardware o de valores críticos en ciertas variables vitales, como por ejemplo el nivel de baterías. El mecanismo que activa los relés es el siguiente. El microcontrolador genera dos señales PWM idénticas que hacen que los dos relés estén cerrados. Estos se abren cuando los PWM dejan de generarse, debido por ejemplo a que la controladora se ha dañado o a que así se ha ordenado desde el ordenador embarcado. Las salidas de los relés están conectadas a sendas entradas del autómata de seguridad del robot que se encarga de desactivar el movimiento ante cualquier alarma de seguridad.

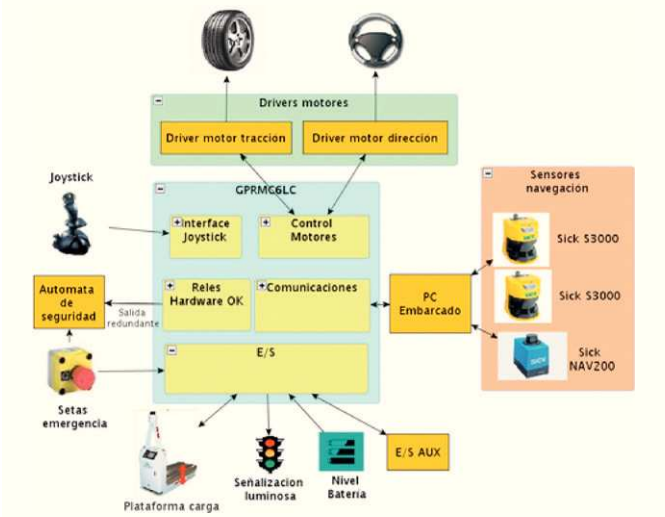


Figura 3: Arquitectura hardware de un AGV.

Las salidas de seguridad son redundantes para minimizar la probabilidad de que la señal permanezca activa por un mal funcionamiento del circuito o porque los contactos del relé se queden pegados, por ejemplo. Esto conllevaría una interpretación por parte del autómata de que todo está funcionando correctamente en la controladora incluso aunque el microcontrolador dejara de funcionar.

En la Figura 1 se muestra un diagrama de bloques de los módulos funcionales que el AGV lleva incorporados.

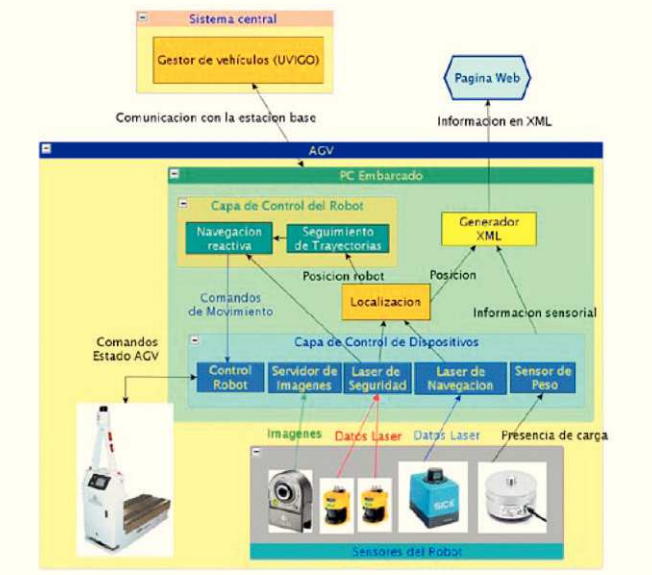


Figura 4: Arquitectura funcional de un AGV.

El PC embarcado a bordo del robot soporta SLAX, un sistema operativo Linux portable, ligero y con un enfoque modular. Por su parte el software ha sido desarrollado en los lenguajes de programación C, y C++ para los driver implementados bajo el entorno de robótica Player/Stage (B. Gerkey et al., 2003). A continuación se describen los diferentes módulos de bajo nivel:

- **Control de placa base**. Es el encargado de comunicarse con la placa de control del AGV a través de bus RS232, utilizando tramas de comunicación. Actúa como intermediario entre los módulos funcionales del PC con los que se comunica por IPC y la placa GPRMC6LC. Puede transferir movimiento al robot a partir de los parámetros de dirección y velocidad, controlar la plataforma de carga y descarga y recibir las tramas directamente de la placa base que informen de su estado y de posibles anomalías en el hardware.
- **Driver láser Sick NAV200**. Se ha desarrollado un driver bajo el entorno Player/Stage que permite configurar y controlar los diferentes modos de funcionamiento del láser. El PC se comunica con el dispositivo a través de uno de los puertos USB pudiendo almacenar las coordenadas de las balizas en la memoria interna del láser y obtener la posición en la que se encuentra el robot. El láser Sick NAV200 se trata de un sistema láser de localización preciso basado en balizas pasivas.
- **Driver láser Sick S3000**. Del mismo modo que en el caso anterior, se ha desarrollado un driver que obtiene los datos del rango de escaneo del láser SICK S3000, de tal forma que el funcionamiento sea independientes del tipo de láser utilizado. A partir de esas medidas se pueden detectar posibles obstáculos que se interpongan en el camino del AGV. Además permite la construcción de mapas de entorno y su localización mediante éstos utilizando SLAM.
- **Módulo Readplayer.** Actúa como abstracción del hardware. Es el encargado de comunicar los drivers de Player con el resto de módulos embarcados, mediante mensajes IPC. De este modo el funcionamiento de los módulos que se encuentren por encima de este nivel (por ejemplo el de navegación) es totalmente independiente de los diferentes dispositivos embarcados o del método utilizado para la localización del robot.
- **Control de tareas.** Es el responsable de mantener la comunicación con el sistema de gestión a través del **módulo Robot-Web**, publicando el estado del robot en el instante que se requiera y recibiendo las distintas tareas a realizar. Además controla la activación y el funcionamiento del resto de módulos embarcados en el AGV.
- **Módulo de navegación.** En el momento en el que el sistema de gestión transmite al robot que debe realizar una trayectoria, el módulo de control de tareas recibe dicho recorrido a modo de secuencia de nodos y cede el control al módulo de navegación para que la lleve a cabo. Para realizar el seguimiento necesita conocer los datos de posición y los datos de medida del láser de seguridad proporcionados por el **módulo ReadPlayer**. El software calcula varios puntos intermedios que deberá seguir el robot hasta llegar al siguiente nodo, generando las variables de dirección angular y velocidad que deberá comunicar al **módulo Control de placa base** para generar el movimiento del robot. En cualquier momento de la navegación, si el AGV se encuentra ante un obstáculo detectado por el láser de seguridad detendrá su marcha y no continuará hasta que el camino quede libre. Este sistema de seguridad también es controlado directamente por hardware. El módulo mantendrá comunicación con el sistema de gestión informándole de su estado, posición, presencia de obstáculos, fin de trayectoria, así como de petición de acceso a siguientes nodos ante la posible ocupación por otros robots.

El AGV se comunica con la estación base de gestión de flota mediante conexión Wi-Fi. Se trata de un sistema estándar capaz de realizar roaming entre los diferentes puntos de acceso instalados en el edificio, asegurando la comunicación en todo momento.

El estado del robot es actualizado periódicamente en un archivo XML. Dicho archivo se utiliza para crear la interfaz web de la que dispone cada AGV, desde donde el operador podrá visualizar el estado actual del robot. De este modo, de forma remota se puede comprobar el correcto funcionamiento de los dispositivos y hacer una primera evaluación ante posibles alarmas surgidas. Si fuese necesario, a través de la interfaz web también se pueden controlar los movimientos del robot a modo de joystick.

## 3. Navegación

La funcionalidad más destacable que el AGV ha de desempeñar es la de seguir, lo más precisamente posible, una trayectoria desde un punto de inicio a un punto final pasando por una serie de puntos intermedios. Esta función es desempeñada por el módulo de navegación.

Periódicamente cada robot publica un mensaje de estado, informando al sistema de gestión de su posición, el nivel de baterías, y si está ejecutando alguna tarea o se encuentra disponible para aceptarla. En el momento en que el sistema central adjudica una tarea a un robot, se lo comunica a través de un mensaje JIPC proporcionándole una trayectoria a modo de secuencia de nodos. Cada nodo contiene la información de sus coordenadas, la velocidad del tramo, tipo de nodo (normal, elevador, punto de control ante posible ocupación de zona por otros robots) y el modo de navegación hacia el siguiente nodo (normal o maniobra). Durante todo el recorrido el AGV se comunicará con el sistema central informando de su posición y estado, además de realizar peticiones de acceso a zonas de posible colisión con otros robots o elevadores. Por último, informará de la finalización de la trayectoria, quedando a la espera de nuevas instrucciones.

### *3.1. Arquitectura mecánica del robot.*

La arquitectura elegida para el AGV es de tipo triciclo, con una rueda delantera motriz y directriz, y dos ruedas traseras pasivas. Las variables que controlan al robot son el ángulo de dirección de la rueda delantera, $\varphi$, y su velocidad angular,$\omega$.

Parámetros geométricos útiles para calcular las referencias de control son hl, distancia entre eje delantero y trasero; CIR, centro instantáneo de rotación; y dl, distancia entre las ruedas del eje trasero. Se considera el centro de guiado del robot el punto central del eje trasero, F. Por lo tanto el radio de giro del AGV viene dado por:

$$\rho = \frac{hl}{\tan(\varphi)} \quad (1)$$

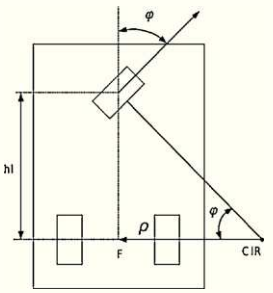


Figura 5: Esquema cinemática AGV.

### 3.2 Navegación reactiva.

Uno de los aspectos más importantes a considerar dentro de los desplazamientos de plataformas robóticas es su comportamiento ante posibles obstáculos que puedan aparecer en el entorno. A partir de los datos obtenidos por los escáneres láser S3000 se detectan dichos obstáculos, lo que permitiría incluir software de navegación reactiva que fuese capaz de recalcular la trayectoria y evitar la colisión continuando con la tarea encomendada.

Estos métodos reactivos son factibles en aplicaciones de robots de pequeño tamaño con baja inercia y gran maniobrabilidad. En este caso los AGVs son de gran tamaño y han de operar en zonas industriales del hospital formadas por corredores, pasillos y ascensores en las que no se contemplan la existencia de obstáculos estacionarios en las zonas de paso del robot. Si se contempla la presencia de personas, por lo que el módulo de navegación reactiva debe atender esta posibilidad pero siempre atendiendo a la seguridad evitando cualquier riesgo de colisión aunque manteniendo la prioridad de la trayectoria frente a las personas. En este caso el módulo de navegación reactivo incluye dos perfiles de distancia de seguridad alrededor del robot y configurables dependiendo la tarea a realizar por el robot (e.g. seguimiento trayectoria o maniobra), Cuando un obstáculo invade el primer perfil de distancia (detectado por el laser Sick S3000) el robot reduce la velocidad y emite señales acústicas, mientras que si se invade el segundo perfil de distancia más próximo al robot el robot se detiene totalmente. En este caso además la parada se realiza independientemente del sistema de control de navegación por lo que la parada se garantiza ante fallos graves del sistema (e.g. fallo del ordenador de a bordo del robot)

Los escáneres láser Sick S3000 incluidos en los robots poseen homologación de seguridad, por lo que se ha incluido un circuito hardware que se encarga de la deshabilitación y parada de los motores del AGV ante la presencia de algún obstáculo en la zona crítica de seguridad.

Aunque en este desarrollo no se ha contemplado la posibilidad de evitación de obstáculos, la arquitectura contempla la inclusión de un módulo de navegación reactiva que incluya esta posibilidad. Este es el caso, de otros desarrollos realizados en (Fernández et al 2008b) en donde el control reactivo puede desviarse temporalmente de la trayectoria inicial para sortear obstáculos siguiendo algún algoritmo como los presentados en (Fox et al 1997) ó (Chaoxia et all 2010).

### 3.3 Seguimiento de trayectoria.

El algoritmo de navegación para el seguimiento de trayectorias (caminos generados por la secuencia de nodos) está basado en la ley de control de persecución pura (Ollero, 2001).

Dicha ley supone que en el intervalo de control la curvatura es constante, describiendo el vehículo un arco de circunferencia que pasa por la posición actual del mismo y por el punto objetivo. El control es proporcional al error lateral (Δx) con respecto al punto objetivo, que se encuentra a una distancia L. La constante de proporcionalidad (ganancia) varía con la inversa del cuadrado de L.

De la Figura 6 se deduce que, siendo r el radio de curvatura del AGV,

$$r = \Delta x + d \qquad d^2 + (\Delta y)^2 = r^2 \quad (2)$$

Eliminando d se obtiene:

$$(r - \Delta x)^2 + (\Delta y)^2 = r^2 \quad (3)$$

El radio de curvatura necesario para que el robot móvil se desplace $\Delta x, \Delta y$ es:

$$r = \frac{(\Delta x)^2 + (\Delta y)^2}{2\Delta x} \quad (4)$$

Por tanto, la curvatura que es necesario suministrar al AGV resulta:

$$\gamma_r = \frac{1}{r} = \frac{-2\Delta x}{L^2} \quad (5)$$

Para aplicar esta ley de control hay que determinar el punto del camino que se encuentra a la distancia L, previamente definida, y calcular el error lateral con respecto al centro de guiado del AGV. Al pertenecer las coordenadas a un sistema global, es necesario tener en cuenta la orientación del vehículo. La eficiencia del controlador de persecución pura depende en gran medida de la buena elección del parámetro L, y presenta limitaciones cuando dicha distancia es muy larga o bien muy corta, donde la acción del controlador resulte insuficiente o pueda ser demasiado brusca para el normal seguimiento de la trayectoria.

#### 3.3.1 Spline de Catmull-Rom.

Para determinar la distancia adecuada al punto objetivo se utilizan los splines de Catmull-Rom. Estas curvas son interpolaciones polinómicas cúbicas de Hermite, por lo que se utilizan cuatro puntos de control ( también llamados nodos) para su cálculo, los dos anteriores y los dos posteriores a la localización actual del robot. La ecuación paramétrica del spline de Catmul-Rom es:

$$P(t) = T \cdot M \cdot V \quad (6)$$

donde los vectores *V* y *T*, y la matriz *M* se definen como:

$$T = \frac{1}{2} \cdot \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \quad (7)$$

$$M = \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \quad (8)$$

$$V = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} \quad (9)$$

Una simplificación de la ecuación (5) es la siguiente:

$$P(t) = b_1 P_1 + b_2 P_2 + b_3 P_3 + b_4 P_4 \quad (10)$$

siendo *b1* a *b4* polinomios cúbicos en función de *t*:

$$b_1 = \frac{1}{2} \cdot \left( -t^3 + 2t^2 - t \right) \quad (11)$$

$$b_2 = \frac{1}{2} \cdot \left( 3t^3 - 5t^2 + 2 \right) \quad (12)$$

$$b_3 = \frac{1}{2} \cdot \left( -3t^3 + 4t^2 + t \right) \quad (13)$$

$$b_4 = \frac{1}{2} \cdot \left( t^3 - t^2 \right) \quad (14)$$

El uso de splines permite la creación de un camino que atraviese los nodos recibidos por el AGV, de tal forma que toda la trayectoria sea una línea continua con transiciones suaves entre curvas y rectas, y que el robot pueda completar fácilmente. Esto presenta una diferencia frente a otros sistemas AGVs del mercado. En otros sistemas comerciales es responsabilidad del operador definir la secuencia pormenorizada de puntos que debe seguir el AGV, lo que convierte la definición de trayectorias en una tarea tediosa y no garantiza la suavidad de la trayectoria. En nuestro caso el operador define la trayectoria mediante una secuencia de puntos de referencia pero es el propio robot el que realiza la interpolación mediante splines entre los puntos de referencia.

Por ejemplo, en la Figura 7 al llegar el AGV al nodo 2 calcula una serie de puntos de control equiespaciados dados por el spline de Catmull-Rom entre los nodos P2 y P3, a partir de los nodos P1, P2 ,P3 y P4. De esta forma el punto objetivo del robot será el punto de control correspondiente que se encuentre a la distancia L definida sobre la trayectoria.

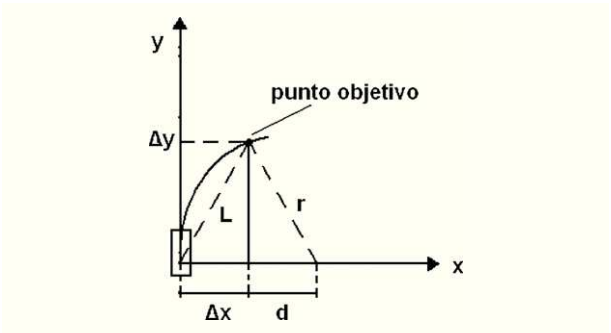


Figura 6: Ley de persecución pura. (Ollero, 2001).

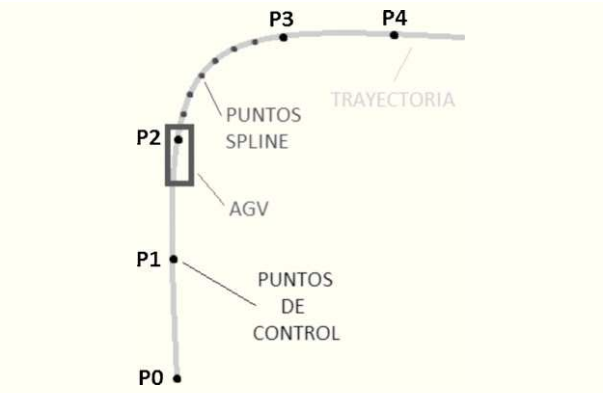


Figura 7: Ejemplo de definición de puntos de control de la trayectoria e interpolación mediante splines entre punto P2 y P 3

### *3.4 Maniobra.*

Durante el seguimiento de una trayectoria el AGV va a encontrase con nodos especiales a los cuales no debe acceder de forma convencional, es decir, desplazamiento hacia adelante. Estos pueden ser puntos de carga o descarga de mercancía, elevadores, puntos de recarga de baterías, etc... Pero además se ha contemplado la posibilidad de que por circunstancias especiales del entorno, la forma de llegar a determinadas zonas difiera también de la habitual, como podría ocurrir por ejemplo en pasillos estrechos donde el robot tuviese que realizar todo el recorrido marcha atrás ante la imposibilidad de maniobrar junto al punto de carga. Por ello cada nodo contiene un parámetro que indica qué tipo de maniobra ha de ejecutar el robot para acceder al siguiente nodo o en su defecto, se tratará de seguimiento de trayectoria simple.

A parte de posibles maniobras que consistan simplemente en navegar marcha atrás desde un nodo a otro, en los que las referencias suministradas al AGV se calculan a partir de las ecuaciones vistas en el apartado 3.3, las maniobras más complejas como el acceso a nodos de carga y descarga de mercancías requieren un tratamiento especial. Dichas maniobras se ejecutan de modo que el AGV se aproxime al nodo perpendicularmente y marcha atrás, para poder introducirse bajo los carros de mercancías de forma precisa. Para ello se descomponen en tres acciones:

1. **Aproximación:** El AGV ha de dirigirse al punto desde el cual comenzará la maniobra marcha atrás.
2. **Maniobra:** Realización de la maniobra hasta situarse en un punto que se encuentra longitudinalmente a una distancia del nodo igual a la longitud del AGV, y con la misma orientación del nodo final.
3. **Finalización:** Acceso lo más perpendicularmente posible al nodo de carga bajo los carros de mercancías.

El cálculo de estos tres puntos vistos en la Figura 8 lo realiza el módulo de navegación a partir del nodo especial etiquetado como final de carga o descarga, conociendo las coordenadas x, y y orientación. De esta forma se simplifica la labor del planificador de las rutas ya que dando únicamente las coordenadas del nodo de carga o descarga y el tipo de maniobra de acceso, el robot será el encargado de hallar los puntos intermedios. El procedimiento para el cálculo es inverso a la secuencia de acciones de las que se compone la maniobra completa.

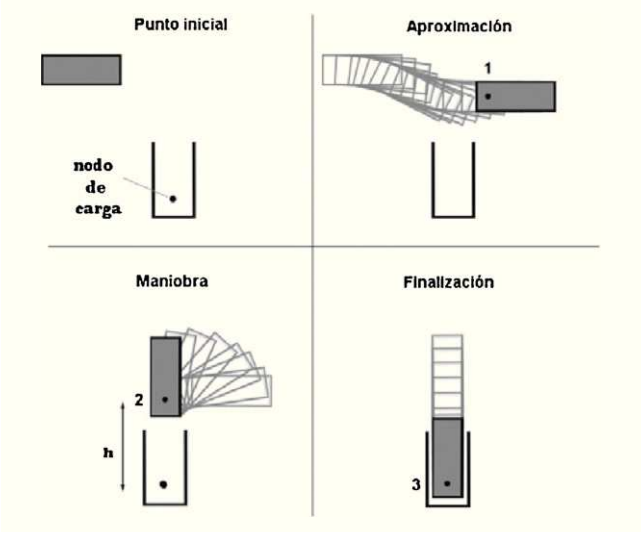


Figura 8: Distintos pasos de maniobra de acceso.

A partir del punto 3 conocido se calculan las coordenadas del punto 2, que se encuentra a una distancia h (longitud del AGV) y en la misma orientación que 3. A partir del punto 2 se halla el punto 1 teniendo en cuenta la geometría del robot, es decir el desplazamiento lineal y angular que necesita el AGV para poder llegar al punto 2 con la misma orientación que el nodo final.

Durante la ejecución de la maniobra se utiliza realimentación con información obtenida de los escáneres láser S3000 (zona libre para realizar el tipo de maniobra adecuado, detección del perfil del carro de mercancías para mejorar la precisión, etc…) ajustándose automáticamente la velocidad y control del AGV a las condiciones del entorno. El Módulo de gestión de tareas (sección 6) y más concretamente la red de Petri es la encargada mediante la inclusión de temporizadores y metasensing (Martin, 2011) de verificar si la maniobra se ha llevado a cabo, replanificar en su caso (e.g. salir de la zona de carga y volver a repetir la maniobra) o establecer una alerta para que intervenga el operador (e.g. la carga no está ubicada en el lugar especificado, sensor de presencia de carga no activo).

## 4. Localización

Uno de los puntos críticos para la navegación de AGVs es la localización. Por una parte es necesaria para conocer la posición de los distintos robots en el entorno y para la asignación de tareas al robot más cercano al destino. Por otra parte, es esencial para que el robot sea capaz de seguir trayectorias y ejecutar maniobras de una forma precisa, lo cual es especialmente importante en la carga y descarga de mercancías. Siguiendo con la búsqueda de la versatilidad y modularidad del sistema propuesto, se dispone de diferentes tipos de sistemas de localización para las distintas condiciones que puedan darse en el entorno, de forma que se puede conmutar de un sistema de localización a otro sin que la navegación y la tarea que esté ejecutando en ese momento se vean afectadas. En concreto como sistema principal se ha utilizado el sistema comercial Sick Nav200 (Sick, 2011), basado en láser y balizas pasivas, que proporciona gran precisión. Como sistema de localización alternativo, aprovechando la existencia de los escáneres láser de seguridad Sick S3000 a bordo del robot, se ha utilizado es el mapeado láser o SLAM. A continuación se describen ambos sistemas así como la conmutación entre ellos.

### *4.1 Sistema de localización láser basado en balizas pasivas.*

Este tipo de sistemas de localización están compuestos por un emisor láser situado en un cabezal giratorio y unas balizas posicionadas en el entorno por el que ha de moverse el robot. Conociendo previamente las posiciones de las balizas, almacenadas en la memoria interna del láser, el sistema es capaz de calcular la posición (x, y) y orientación en la que se encuentra. En el caso de que se trate de balizas pasivas, como en el caso del Nav200, es el propio láser el que obtiene la posición exacta en la que se encuentra, a partir del tiempo que tarda la luz emitida en reflejarse en algunas de las balizas y volver al origen. Las balizas son únicamente reflectores y no requieren alimentación. El sistema necesita detectar la luz láser reflejada por al menos 3 balizas para poder calcular su posición mediante triangulación.

Se trata de un sistema muy fiable, con un alcance de hasta 28 metros y una precisión de +/-4mm. Para garantizar dicha precisión las balizas deben posicionarse con la ayuda de algún dispositivo de medida de coordenadas como una estación total.

### *4.2 Sistema de localización basado en mapeado láser.*

El algoritmo de localización adaptativo de Monte-Carlo (AMCL) (Fox,2001) estima la posición del robot mediante distribución probabilística, a partir de los datos de odometría y los escaneos láser, que son comparados con un mapa del entorno predefinido (Figura 9). La distribución probabilística es calculada por medio de un filtro de partículas adaptativo, ya que dinámicamente el número de partículas se ajusta proporcionalmente a la incertidumbre de la posición.

Previamente a la localización mediante este algoritmo, es necesario crear el mapa del entorno a partir de los escaneos láser obtenidos en un recorrido realizado con el AGV de forma manual. Por medio de estos escaneos asociados a cada posición intermedia (ya sea calculada mediante odometría o localización por balizas) se crea el mapa de ocupación del entorno siguiendo el algoritmo iterativo de alineamiento de superficies (MRICP) (Taha T., 2006). Estos datos son los que van a permitir calcular la posición del robot, al ser comparados con los escaneos láser actuales. Este sistema de localización no es tan preciso como el de balizas, por lo que en puntos críticos como las zonas de recogida de carros habría que complementarlo con algún otro método.

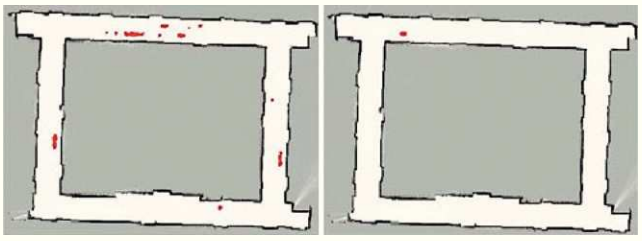

Figura 9: Distribución probabilística. a) t=40s, aprox. 1000 partículas. b)t=80s, aprox. 100 partículas.

### *4.3 Conmutación entre sistemas*

La conmutación entre ambos sistemas puede ser predefinida, asignando a diferentes zonas del entorno el sistema de localización más apropiado, o automática, de modo que cuando por alguna circunstancia anómala la localización por balizas no obtenga una posición válida, el robot se sitúe mediante mapeado láser y si tampoco resulta exitoso continúe la navegación con odometría pura durante una distancia fiable, con el fin de poder

recuperar la localización de alguno de los sistemas. Si superada la distancia considerada como fiable de navegación mediante odometría no se recupera ninguna localización, el AGV detendrá la marcha y comunicará al sistema central de gestión el error pertinente. Toda la gestión de la localización la lleva a cabo el módulo ReadPlayer, de modo que la navegación es totalmente independiente al tipo de sistema utilizado, pudiendo en un futuro sustituir o añadir nuevos sistemas de localización de forma sencilla.

## 5. Integración con el sistema de automatización del edificio

La mayoría de los sistemas de automatización de los edificios conocidos como BAS (Building Automation System), utilizan redes que conectan controladores de alto nivel con controladores de bajo nivel, dispositivos de entradas y salidas e interfaces hombre-máquina (HMI). El bus utilizado puede ser, entre otros, BACne, Ethernet, MODBUS, ARCNET, RS-232, RS-485 o una red inalámbrica (Kim et al., 2006).

En este trabajo se ha utilizado MODBUS TCP/IP que es una variante o extensión del protocolo Modbus que permite utilizarlo sobre la capa de transporte TCP/IP. De este modo, Modbus-TCP se puede utilizar en Internet, lo que facilita la tarea de integración dentro de la arquitectura de control centralizada.

El hecho de usar TCP/IP y basarse en el establecimiento de sockets con los distintos módulos de interfaz MODBUS nos permite programar el módulo Building Interface ( Figura 10) en una gran variedad de lenguajes de programación. Las tramas a enviar son muy sencillas tratándose principalmente de solicitudes de escritura o lectura sobre registros de los módulos de interfaz MODBUS.

El esquema utilizado se representa en la Figura 10. Cada interfaz de red MODBUS integra un conjunto de sensores y actuadores del edificio y a su vez estas interfaces de red están unidas a la red TCP/IP del mismo. En la parte superior podemos ver representados los módulos interfaz de red Modbus que se conectan a la red del edificio a la cual está también conectado el servidor central. En dicho servidor central se ejecuta, entre otros, el módulo Building Interface que, a su vez, está conectado a JCENTRAL de JIPC. El módulo Robot Web Interface se encarga de las comunicaciones entre el robot y el servidor central. Dicho módulo va a bordo del robot y se conecta al proceso IPC del robot y a JIPC del servidor central. Esto hace el sistema más robusto de cara a conexiones y desconexiones de los robots. Esta característica toma especial relevancia en aquellos entornos donde los robots pueden atravesar zonas de baja cobertura Wi-Fi, para lo cual se ha dotado a JIPC con servicios de temporización (López et al., 2009) con el fin de dar de baja y alta los robots cuando sea necesario.

Una vez configurado el sistema, la tarea principal del módulo Building Interface, desde el punto de vista de nuestra aplicación, consiste en:

- Informar mediante la publicación de mensajes JIPC de los cambios en las entradas conectadas a las interfaces de red MODBUS.
- Cambiar los valores de las salidas de las interfaces de red MODBUS de acuerdo con las solicitudes (recepción de mensajes) de otros módulos conectados a IPC.

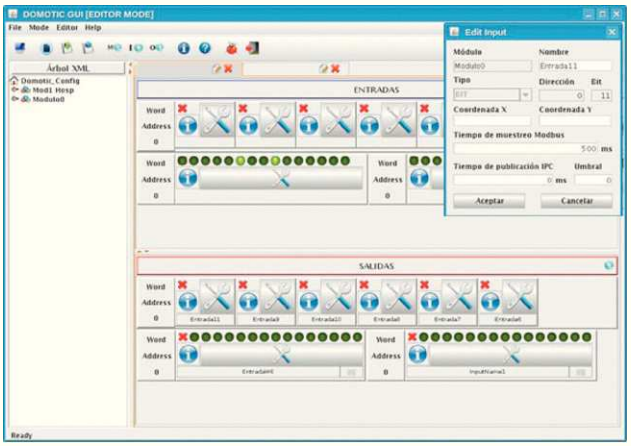


Figura 10: Ventana principal de Building Interface GUI.

El módulo Building Interface es genérico y no es necesario reprogramarlo para nuevos proyectos. Esto se consigue guardando la configuración específica del proyecto en un archivo XML que contiene los parámetros de las interfaces de red y sus configuraciones que incluye direcciones IP, puertos, entradas a muestrear con los períodos de muestreo y registros donde se almacenan. Para editar dicho archivo se ha desarrollado la interfaz gráfica denomina Building Interface GUI cuya ventana principal se puede ver en la Figura 10.

Building Interface GUI puede trabajar en dos modos distintos que son modo Editor y modo Monitor. En modo Editor permite, de forma fácil e intuitiva, añadir nuevas interfaces de red y definir sus parámetros de configuración, así como los dispositivos conectados. De esta forma se puede crear una configuración de un proyecto nuevo, o bien editar una configuración existente.

En modo Monitor muestra el estado de los dispositivos conectados al módulo Building Interface vía MODBUS. Para ello, se conecta a JIPC y se subscribe a los mensajes que publica Building Interface. Cada vez que hay un cambio de estado en una entrada o salida cambia la forma de visualización de la misma.

## 6. Gestión de tareas.

Para la programación, análisis, depuración y monitorización de las tareas se ha desarrollado RoboGraph. Esta herramienta se compone de dos programas distintos. El primero de ellos (Dispatch) se encarga de la ejecución de las tareas, no tiene interfaz gráfico y debe ejecutarse siempre que el sistema esté funcionando. El segundo (Editor) se utiliza para la edición de las tareas, su análisis y seguimiento. RoboGraph utiliza redes de Petri jerarquizadas e interpretadas para coordinar la actividad de los distintos módulos implicados en la tarea.

### *6.1 Definición de tareas*

La edición de tareas se lleva a cabo con el módulo Editor de RoboGraph ( Figura 11). El sistema es lo suficientemente flexible para facilitar la definición de cualquier secuencia. Las tareas se definen mediante estructuras basadas en una red de Petri. Estas estructuras las definirá el administrador y se almacenan en un archivo XML. Añadir una tarea nueva consistirá simplemente en definir una nueva red de Petri utilizando un editor gráfico en el que se van seleccionando y añadiendo los distintos elementos de la red de Petri.

A modo de ejemplo, veamos cómo se define la tarea encargada de ir a por un carro de la zona de cocina. Esta tarea comenzará cuando se activa un sensor indicando que hay un nuevo carro a transportar en una zona de cocina. En este caso la tarea consistirá en enviar un robot al nodo del entorno donde se ha activado el sensor, ejecutar la maniobra de enganche, leer el destino del carro (tag rfid), transportar el carro al punto de destino y realizar la maniobra de desenganche.

Como veremos más abajo, será Task Manager el encargado de identificar que se debe realizar la tarea al activarse el sensor y solicitar al módulo Dispatch la ejecución de la red de Petri correspondiente, que en este caso se muestra en la Figura 11. Con esta solicitud se enviarán los parámetros que se muestran en la parte superior izquierda y que corresponden a la posición donde se tiene que ir a recoger el carro (posición del sensor activado) y el robot asignado por Task Manager.

Analizando la red, el primer lugar y la primera transición es de inicialización. Cuando el marcado alcanza el segundo lugar, Dispatch publica un mensaje JIPC solicitando el cálculo de la trayectoria desde donde se encuentra el robot asignado a la posición de recogida. El módulo Traffic Manager está subscrito a este mensaje y cuando lo recibe, calcula dicha trayectoria y publica un nuevo mensaje. Este mensaje con la trayectoria es recibido por Dispatch que hará evolucionar la red bien al lugar P2 enviando un mensaje al robot para que siga la trayectoria calculada o bien reportando el error correspondiente y finalizando la red en caso que no exista trayectoria (NO_PATH).

Una vez el robot llegue al final de la trayectoria reportará este evento publicando un mensaje (Goal Reached) que será recibido por Dispatch. La llegada de este mensaje implica la evolución del marcado al lugar etiquetado como Pick que tiene asignada la acción de publicar otro mensaje JIPC solicitando al robot la ejecución de la maniobra de enganche del carro. Una vez el carro este enganchado, se procederá de forma similar para la solicitud de lectura de la posición destino (Request Position), cálculo (Request Path) y seguimiento (P8) de la trayectoria al lugar de entrega y desenganche (Drop) del carro.

A un lugar de la red se puede asociar la publicación de cualquier mensaje, incluido el de solicitud de ejecución de una red de Petri. En ese caso será Dispatch quien publique y reciba el mensaje. De esta forma, la ejecución de una red puede implicar la ejecución de otras redes.

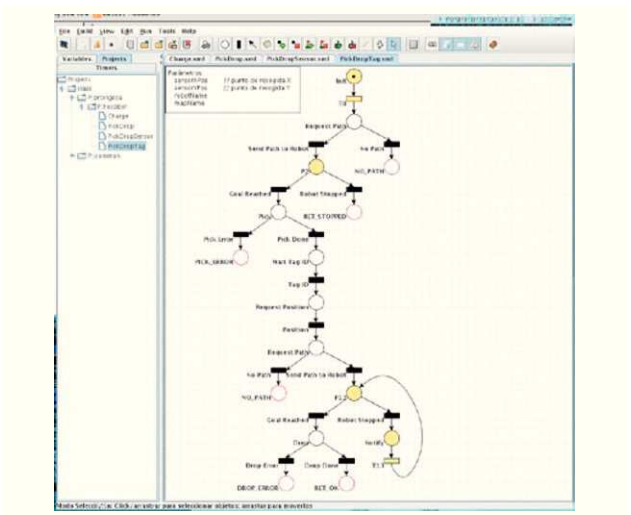

Figura 11: Instantánea de RoboGraph Editor.

Entre otras herramientas también es importante mencionar los temporizadores que, aunque no se muestran en la Figura 11 permiten de forma sencilla temporizar determinadas acciones de la tarea.

Una vez definida la red de Petri, el Editor tiene herramientas para analizar algunas de las propiedades de la misma (López et al., 2011). La edición de nuevas redes de Petri puede ser necesaria si algún proyecto requiere realizar tareas específicas, distintas a las que ya están definidas. No obstante, en la mayor parte de los proyectos es suficiente con las tareas existentes. En el peor de los casos, lo único restante es definir los parámetros para la creación de distintas instancias de estas tareas, como se verá en la sección 7.

Además de trabajar en modo edición, RoboGraph Editor puede trabajar en modo monitorización y en modo depuración. En modo monitorización visualiza el estado (marcado) de todas las redes de Petri que está ejecutando Dispatch, lo que facilita el seguimiento de la ejecución de las tareas y la identificación del malfuncionamiento de algún módulo. En modo depuración permite localizar errores del sistema fuera de línea, utilizando archivos de registro de actividad almacenados por Dispatch, lo cual resulta conveniente en algunos momentos en que el número de redes en ejecución es elevado y su seguimiento en tiempo real resulta problemático.

### *6.2 Ejecución de tareas*

Como ya se mencionó, el módulo Dispatch de RoboGraph se encarga de la ejecución de tareas. Aunque desde el interfaz de usuario también se puede solicitar la ejecución de alguna tarea, es el módulo Task Manager quien realiza la mayor parte de las solicitudes mediante la publicación de los mensajes JIPC correspondientes.

Además de la ejecución de estas tareas, Dispatch publica un mensaje de cambio de estado cada vez que evoluciona el marcado de alguna de las redes en ejecución y lo guarda en un archivo. Esto permite el modo de funcionamiento monitor del editor descrito anteriormente así como el modo depuración.

### *6.3 Gestión de tareas*

El módulo Task Manager se encarga de arrancar las tareas y asignar recursos a las mismas. El inicio de una tarea lo determinan los arrancadores, donde cada uno de ellos tiene definido una serie de condiciones bajo las cuales se debe iniciar una tarea, como puede ser la activación de un sensor del entorno que indica que hay un carro para transportar. Estas condiciones pueden además estar restringidas a intervalos temporales. Por ejemplo, la activación de un sensor en la zona de cocina puede significar la creación de una tarea de transporte de un carro de comida a las plantas solamente a las horas de las comidas, mientras que se descarta si se activa fuera de este horario. También se pueden configurar tareas automáticas en determinados instantes sin necesidad de que se produzca ningún evento.

Cuando finaliza una tarea, Task Manager tiene conocimiento por el mensaje de fin de tarea que publica Dispatch de RoboGraph. En ese momento Task Manager puede asignarle al robot otra tarea si existen algunas pendientes, o bien arrancar la tarea de recargar batería.

Otra función de este módulo es la asignación de los robots a las distintas tareas. Esta asignación se define mediante un conjunto de criterios que se puede cambiar de un proyecto a otro

usando la interfaz de configuración. En cuanto a las políticas definidas para esta aplicación, se considera que todas las tareas tienen los mismos requerimientos y todos los robots son iguales. Por ello, cualquier robot puede realizar cualquier tarea y el criterio de asignación de una nueva tarea entre los robots disponibles, consiste en asignar el robot más cercano al punto de recogida. Si hay varias tareas pendientes y varios robots disponibles, para cada tarea se va seleccionando el robot más cercano. El conjunto de robots disponibles lo forman aquellos que tienen suficiente carga de batería y que no tengan ninguna tarea asignada y por tanto estarán en la estación de recarga o de camino a ella.

Aunque en este caso todas las tareas son iguales, el sistema contempla la posibilidad de priorizar las tareas en una escala de 1 a 10 (Fernández et al., 2008a).

## 7. Instalación y ejecución de un nuevo proyecto

El objetivo de este trabajo no se limita a la creación de un único proyecto, sino a la creación de una aplicación genérica con herramientas para generar distintos proyectos de forma sencilla por personal, que puede formarse en un período corto de tiempo. Esto es posible gracias al interfaz gráfico de configuración de proyectos.

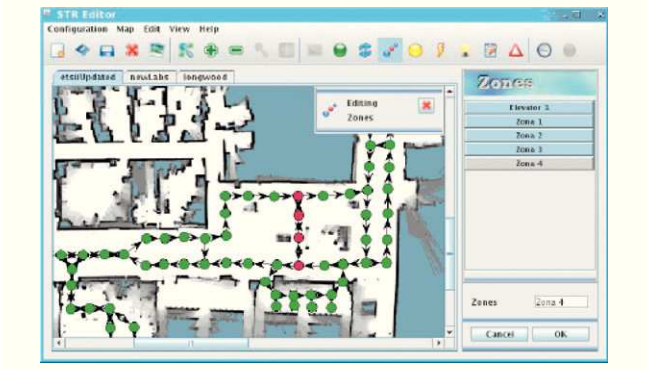


Figura 12: Editor en modo de edición de zonas.

El sistema completo puede ser instalado y programado en un corto período de tiempo gracias a una herramienta gráfica (HospBot Editor) que permite definir de forma sencilla todos los parámetros del proyecto. Mediante este programa se obtendrá un fichero de configuración único para todo el sistema, evitando de este modo posibles incoherencias en la información manejada por los distintos módulos. Dentro de la arquitectura de control RIDE, los módulos que utilizan el fichero obtenido son:

- **Building Interface**: Utiliza la información relativa a los módulos distribuidos de señales de entrada y salida. Esta parte también puede ser configurada mediante el programa Building InterfaceGUI para posteriormente ser importada desde HospBot Editor.
- **Traffic Manager**: Utiliza la configuración relativa a la geometría del entorno de los robots móviles. Esta información incluye los nodos que forman parte de las rutas, las maniobras para desplazarse entre los nodos y las zonas con acceso restringido.
- **Task Manager**: Información relativa a las tareas y planificadores de ejecución.

Muchos de estos parámetros, como pueden ser los nodos, zonas, sensores o elevadores, están asociados a posiciones en el plano del edificio. Por eso el Editor permite trabajar sobre uno o varios mapas de forma que la selección de estas posiciones sea más sencilla. Aunque la aplicación trabaja con un formato de mapas específico, optimizado para su tratamiento y transferencia a través de la red, este mapa se puede generar de forma muy distinta dependiendo del sistema de localización que utilicen los robots. Por ejemplo, si se usa un escáner láser, los propios robots pueden generar mapas como el de la Figura 12, pero también se puede trabajar sólo con mapas de balizas. Se está, además, añadiendo la capacidad de que puedan incorporarse mapas en formato DXF de Autocad para poder programar y simular los proyectos antes de tener el edificio acabado. En edificios con varias plantas se debe utilizar un mapa por planta.

En general, para especificar completamente un proyecto es necesario definir una gran variedad de parámetros entre los que podemos destacar:

- **Nodos**. Los nodos son los puntos del entorno por los que el robot puede pasar. Es necesario además definir los arcos entre los nodos para indicar los posibles caminos que los robots pueden realizar. A estos arcos se le asocian las maniobras descritas en el la sección 3. El módulo Traffic Manager será el encargado de calcular las trayectorias entre el nodo origen de donde parte el robot al nodo destino.
- **Zonas de exclusión**. Estas zonas son grupos de nodos en las cuales sólo puede haber un robot a la vez. Al igual que en el caso anterior, es el módulo Traffic Manager quien conoce en todo momento en qué nodo se encuentran los distintos robots y por tanto también se encarga de otorgar o denegar el acceso a las distintas zonas.
- **Elevadores**. Se encargan de "unir" nodos que se encuentran en distintos mapas. La configuración de un elevador tiene asociados un grupo de nodos que se encuentran en plantas distintas de los edificios así como las señales necesarias para operarlo, tanto para controlar las puertas como el desplazamiento entre niveles.
- **Sensores**. Los sensores son componentes del sistema utilizados para controlar estados del entorno y reportar eventos para ejecutar acciones. La información asociada a un sensor está formada por una señal de entrada y por un nodo, que determina la posición en la que se encuentra el sensor.
- **Estaciones de carga**. Son puntos del sistema donde se encuentran instalados los dispositivos de recarga de baterías de las plataformas móviles. La configuración de una estación de carga depende básicamente del sensor que tenga asociado.
- **Tags**. Los *tags* RFID se utilizan en el sistema para especificar puntos de destino de los robots en función de la carga que transporten. Estos *tags* tendrán un nodo del sistema asociado.
- **Schedulers**. Son planificadores temporales que se utilizan para generar elementos por tiempo. Se pueden definir planificadores diarios o semanales.
- **Tareas**. Las tareas en este punto deben estar en realidad ya creadas con la herramienta RoboGraph. Lo que aquí se define es el nombre de la red de Petri y los parámetros que han de pasar para su ejecución.

- **Arrancadores**. Son lanzadores de tareas que se encargan de supervisar el estado del sistema y determinan si este reúne las condiciones necesarias para el lanzamiento de las tareas que tiene asociadas. La configuración de un arrancador está compuesta por un nombre, un tipo relativo a los eventos que activan el arrancador, un posible *scheduler* que es opcional, sensores también opcionales, con los cuales se pueden configurar condiciones de activación. Un ejemplo típico seria la programación de un arrancador con la condición: "si se activa un sensor en la zona de cocina de las 12:00 a las 14:00 se debe arrancar una tarea "llevar carro de cocina a planta" ".

## 8. Resultados

Para verificar el funcionamiento del sistema, y antes de su instalación en Hospitales, se han realizado pruebas intensivas en las instalaciones de la empresa Proingesa de Valladolid (España), donde se ha reproducido fielmente el entorno de un Hospital, incluyendo zonas de recogida de carros de transporte, zona de entrega, zona de recarga, sensores de presencia y ascensores (ver Figura 13).

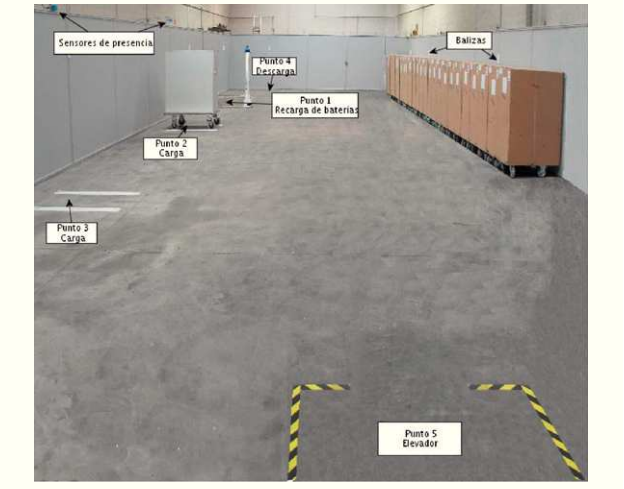


Figura 13: Entorno de verificación en instalaciones Proingesa.

En la Figura 14 se muestran las dos plantas del entorno de operación con dos puntos de recogida de carros (puntos A y B de la Figura 14) y un punto de entrega (punto D). Se requirió menos de un día de trabajo para definir la configuración del nuevo proyecto en las instalaciones de Proingesa (sin tener en cuenta la instalación de sensores del entorno), lo cual prueba la versatilidad de la aplicación y facilidad de configuración en la creación nuevos proyectos.

El experimento muestra dos plantas. En realidad se trata de la misma planta física, pero se han creado dos plantas virtuales con configuración distinta como se puede apreciar en la Figura 24.

El paso de una a otra planta se lleva a cabo a través del ascensor. El ascensor se representa con el nodo inferior que une las dos plantas. Si bien en la empresa no existe un ascensor real, todas las señales del mismo se han implementado en un módulo MODBUS (Schneider OTB 1E0DM9LP) con las entradas y salidas que tendría el ascensor. En concreto se usan salidas para abrir/cerrar puertas e indicar la planta a la que se desea mover el ascensor. En cuanto a entradas, además de los sensores fin de carrera para detectar si las puertas están abiertas o cerradas, hay un sensor por cada planta. Para simular el funcionamiento del ascensor se ha programado un PLC que lee las salidas del módulo MODBUS y activa las entradas de dicho módulo, de la misma forma que lo haría el ascensor, mostrando al mismo tiempo en un display la planta en la que se encuentra.

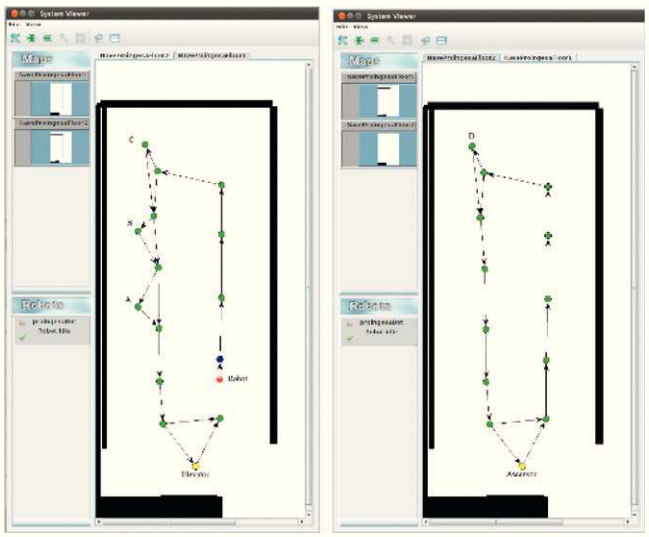

Figura 14: Configuración del entorno de operación

En la Tabla 1 se muestran los resultados de los tiempos empleados en distintas partes de una tarea, realizada de forma repetida durante tres jornadas completas (144 repeticiones). La tarea consiste en situar un carro debajo del sensor del punto A de la planta 1. El carro es detectado por el sensor y se activa la tarea de transportar carro, en la cual el robot ha de recogerlo y llevarlo al punto de entrega situado en el punto D de la planta 2. En cada columna de dicha tabla se representan el valor medio, mínimo, máximo y la varianza del tiempo total de la tarea, en la primera columna; el tiempo de cálculo y publicación de trayectoria, en la segunda; el tiempo empleado en ir desde la estación de carga de batería (punto C) al nodo de enganche (punto A), en la tercera; el tiempo de enganche del carro (sin incluir el tiempo de maniobra), en la cuarta; el de ir desde el punto de recogida (A) al punto de entrega (D) (incluyendo el tiempo empleado en el ascensor), en la quinta; y el de desenganche (sin incluir el tiempo de maniobra) en la sexta.

Durante el experimento se produjeron 5 fallos (3,5%), todos ellos en la operación de enganche del carro. Fueron debidos a pequeños errores de orientación del robot con respecto al carro. Estos errores van a ser resueltos incrementando ligeramente la anchura de los carros (que actualmente solo permite un error lateral de ±2cm).

Tabla 1: Tiempos medios empleados en acciones

| | Total tarea | Planificación trayectoria | Ir al punto recogida | Tiempo enganche | Ir de recogida a entrega | Tiempo desenganche |
|---|---|---|---|---|---|---|
| Media | 409046 ms. | 555.8 ms. | 109338 ms. | 2692 ms. | 297105 ms. | 2817 ms. |
| Máximo | 446627 | 627 | 134201 | 2782 | 311540 | 3280 |
| Mínimo | 393896 | 522 | 101800 | 2270 | 284487 | 2526 |
| Varianza | 17785 | 34 | 27889 | 188 | 9669 | 225 |

## 9. Conclusiones

En este artículo se ha presentado el desarrollo de un sistema de transporte de asistencia hospitalaria. El desarrollo de un sistema como el propuesto ha supuesto el esfuerzo de transferencia de la investigación y desarrollo de una arquitectura escalable que integre sistemas de localización, control de robot, navegación, seguridad, gestión de tráfico y gestión de tareas.

El sistema ha sido desarrollado de forma que presenta una total independencia del sistema físico de transporte, así como de los sistemas de localización utilizados. Además, como consecuencia de su concepción altamente modular, presenta una marcada escalabilidad, mantenibilidad y reusabilidad de los distintos módulos que lo componen. El resultado se ha implantado sobre un prototipo preindustrial de carro robot, desarrollado conjuntamente con la empresa de ingeniería objeto de la transferencia. Los ensayos realizados han resultado plenamente satisfactorios, tanto en términos de integración con la arquitectura de control de bajo nivel del carro, como de precisión en el seguimiento de trayectorias, conformidad de la realización de tareas e integración con la infraestructura del edificio. El personal de la empresa tiene plena capacidad de reconfiguración y utilización del sistema gracias a las características anteriormente mencionadas. En la actualidad se está trabajando en la industrialización del sistema tanto para entornos hospitalarios, como industriales y de servicios.

## English Summary

**Design and Development of a Reconfigurable Transport System for Hospital Environments.**

### Abstract

The aim of this paper is the presentation of an automatic transport system of materials (food, supplies, clothing and medicine) in hospital environments using mobile robots. This article discusses some aspects of the physical platform development, in particular those related to the control system, software architecture, the generation and path tracking, job scheduling and traffic management system, as well as results of the integration of the global system and its application to a hospital transport system. The global system consists of a set of mobile robots capable of independent navigation and a central server which manages the activities to be performed by robots and organize traffic at critical points. Although robots perform tasks independently, interacting with environmental elements such as elevators, occupancy sensors in areas of pick and drop, and doors, is also done through the central server. Human operators can connect to the central server to change the activities to be performed by robots or monitor its status..

*Keywords:*

Mobile robots, navigation system, autonomous vehicle, path planning, guidance system.

## Agradecimientos

Agradecemos a la empresa Proingesa y en especial a Ángel de Miguel su colaboración en el desarrollo del proyecto.

Este trabajo ha sido realizado parcialmente gracias al apoyo de la Agencia Nacional de Comisión Interministerial de Ciencia y Tecnología, bajo el Proyecto DPI2008-06738 así como el Centro para Desarrollo Tecnológico Industrial (CDTI) y Agencia de Desarrollo Económico (ADE) de la Junta de Catilla y León.

## Referencias

Chaoxia Shi, Yanqing Wang, and Jingyu Yang. 2010. A local obstacle avoidance method for mobile robots in partially known environment. Robot. Auton. Syst. 58, 5 (May 2010), 425-434.

Ds Automotion . http://www.ds-automotion.com/

Engelberger Joseph (1993). Health-care robotics goes commercial: the 'HelpMate' experience. Robotica, 11, pp

Fernández, J.L., R. Sanz, E. Paz and C. Alonso (2008) Using hierarchical binary Petri nets to build robust mobile robot applications: RoboGraph. *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA,* pp. 1372-1377.

Fernández, J.L., D. Perez, R. Sanz (2008). Enhancing Building Security Systems with Autonomous Robots. Proceedings of The 2008 IEEE International Conference on Technologies for Practical Robot Applications. pp:19-25. Woburn, MA, USA 2008.

Fox, D. and Burgard, W. and Thrun, S. 1997. The Dynamic Window Approach to Collision Avoidance. Robotics & Automation Magazine, IEEE 4 (1): 23–33.

Fox D. (2001). KLD-Sampling: Adaptative Particle Filter. Advances in Neural Information Processing Systems 14.MIT Press.

Gerkey B., Stoy K.,Vaughan R. T.(2003) The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In Proceedings of the 11th International Conference on Advanced Robotics. (ICAR 2003) http://playerstage.sourceforge.net/

JBT Corporation (2011). http://www.jbtcorporation.com/

Kim, Y., H. Kim, S. Yoon, S. Lee, K. Lee (2006), Home security robot based on Sensor Network, *SICE-ICASE International Joint Conference 2006*, pp: 5977-5982

López, J., Manuel Álvarez, Miguel Cacho, Enrique Paz and Diego Pérez (2009) Creating wireless coverage maps for mobile robot applications. *Proceedings of the Workshop en Agentes Físicos (WAF 2009),* pp. 17-25.

López, J., D. Perez (2011) and E. Zalama (2011). A framework for building mobile single and multi-robot applications. Robotics and Autonomous Systems, vol 59, pp. 151-162. Also in http://webs.uvigo.es/vigobot/

517-523

Martin, Fred. Robotic Explorations: A Hands-on Introduction to Engineering. Prentice Hall, 2001.Ollero, A., (2001). Robótica y manipuladores móviles, Ed. Marcombo.

Pérez, José Ramón. Suárez, Mª Soledad, Lemes, José Manuel (2010). Transporte robotizado en el Hospital Universitario de Gran Canaria Dr. Negrín. Ingeniería Hospitalaria. Feb. 2010.

Sick (2011). NAV200: Laser Positioning system for navigational support. http://www.sick.com/

Simmons, R., (2005). Inter Process Communication (IPC). Carnegie Mellon University.

http://www-2.cs.cmu.edu/afs/cs/project/TCA/www/ipc/ipc.html.

Swisslog (2011). http://www.swisslog.com/

Taha, Tarek, (2006). The mricp driver (map reference iterative closest point). http://playerstage.sourceforge.net/doc/Player2.1.0/player/group__ComponentNavigator.html