UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERIA DE LA COMPUTACIÓN

ROBOTS MOVILES COLABORATIVOS: UN CASO DE ESTUDIO BASADO EN COOPERACION EUSOCIAL

T E S I S

QUE PARA OBTENER EL TÍTULO DE

Maestro en ingeniería de la computación

P R E S E N T A:

EDWIN JAVIER ALDANA BOBADILLA

TUTOR: DR. JESUS SAVAGE CARMONA

MÉXICO, D.F. **2009**

La civilización es, entre otras cosas, el proceso por el que las primitivas manadas se transforman en una analogía, tosca y mecánica, de las comunidades orgánicas de los insectos sociales.

Aldous Huxley

RESUMEN

Este trabajo presenta el diseño de un sistema conformado por varios robots móviles que utilizan el concepto de cooperación eusocial presente en algunas especies de insectos como hormigas, termitas, abejas y avispas, caracterizado por una división del trabajo en busca de un beneficio común, la simplicidad de cada individuo, la ejecución concurrente de tareas y el control descentralizado.

Para lograr esto, el sistema propuesto implementa en cada robot una serie de comportamientos reactivos ante condiciones impuestas por el medio. La interacción entre robots es implícita (a través de la modificación del entorno) usando un algoritmo basado en probabilidades y una red de perceptrones. El sentido de localización en cada robot está dado por un algoritmo que utiliza lógica borrosa que le permite determinar su ubicación sin el uso de uso de modelos geométricos o topológicos del entorno que demandan un alta capacidad de cómputo. Con esto como base se logra generar comportamientos colectivos que permiten realizar tareas de manera conjunta sin que el control del sistema dependa de algún individuo o entidad en particular garantizando escalabilidad del sistema y tolerancia a fallas.

Palabras clave: Agente, Cooperación, Cooperación Eusocial, Lógica Borrosa, Inteligencia Colectiva, Red de Perceptrones, Robot Móvil, Sistemas Multi-Robot.

AGRADECIMIENTOS

El presente trabajo no es únicamente el resultado del esfuerzo personal sino además del esfuerzo y aportes de muchas personas, quienes de manera incondicional siempre estuvieron atentas a brindarme su apoyo y colaboración.

Por tal motivo quiero expresar mis más sinceros agradecimientos al Dr. Boris Escalante, coordinador del posgrado en Computación de la UNAM hasta noviembre del 2008, quien siempre estuvo atento a garantizar las condiciones que hicieran posible el buen desempeño de estudiantes, profesores y funcionarios del posgrado. En particular quiero agradecerle en mi condición de estudiante extranjero, por toda su colaboración para hacer de mi estancia en México una de las mejores experiencias.

Al Dr. Jesús Savage Carmona, por la oportunidad de disponer de su asesoría, consejos y amplia experiencia académica fundamentales para la realización del presente trabajo.

Al Dr. Ángel Kuri Morales, por todos sus conocimientos y experiencias impartidas durante sus cursos.

A la Dirección de Estudios de Posgrado de la UNAM, por el apoyo económico recibido sin el cual no hubiera sido posible la culminación de mis estudios.

Al personal administrativo del posgrado en Computación, en especial a Diana, María de Lourdes, Amalia, Ángeles y Álvaro de quienes siempre recibí su amable atención y colaboración.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ALGORITMOS

GLOSARIO

ACTUADOR: Dispositivo que permite generar fuerza a partir de líquidos, gases o energía eléctrica por lo que se clasifican en hidráulicos, neumáticos y eléctricos respectivamente.

AGENTE: Un agente es una entidad física o abstracta que puede percibir su ambiente a través de sensores, es capaz de evaluar tales percepciones y tomar decisiones por medio de mecanismos de razonamientos sencillos o complejos, comunicarse con otros agentes para obtener información y actuar sobre el medio en el que se desenvuelve a través de *actuadores*.

CANDELA: Es la unidad básica de intensidad luminosa en una dirección dada.

COMPORTAMIENTO: Conjunto de acciones de un individuo desencadenadas a partir de un conjunto de condiciones o restricciones dadas por el medio.

COOPERACIÓN: Es la unión explícita o implícita de operaciones u acciones entre un grupo de individuos.

COORDINACIÓN: Es el mecanismo mediante el cual se disponen metódicamente todos los individuos y componentes del sistema para unir esfuerzos que contribuyan a su objetivo.

ENTOMOLOGIA: Estudio científico de los insectos.

EUSOCIAL: Comportamiento cooperativo presente en algunas especies de insectos caracterizado principalmente por una división del trabajo entre individuos que buscan un beneficio común.

ESTIGMERGIA: Es la comunicación a través del medio físico. En sistemas biológicos, tales como las colonias de hormigas, los diferentes individuos colaboran a través de pautas o hitos dejados en el medio comúnmente denominadas feromonas.

FEROMONA: Sustancia química volátil utilizada por algunos insectos para dejar información de posibles fuentes de alimento o alertar ante peligros a otros individuos.

FIREWIRE: Es un estándar *IEEE* bajo la referencia IEEE 1394 multiplataforma para entrada/salida de datos en serie a gran velocidad. Suele utilizarse para la interconexión de dispositivos digitales como cámaras digitales y videocámaras a computadoras.

FIRMWARE: Es un bloque de instrucciones para propósitos específicos, grabado en una memoria de tipo no volátil, que contiene la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo.

GNU GPL: Acrónimo de *Licencia Pública General de GNU* del inglés *GNU General Public License* que permite proteger la libre distribución, modificación y uso del software.

HUMANOIDE: Robot con morfología humana.

IEEE: Acrónimo de Instituto de Ingenieros Eléctricos y Electrónicos por sus siglas en inglés *The Institute of Electrical and Electronics Engineers*, la cual es una asociación dedicada a la estandarización de tecnología en áreas afines a la electricidad y la electrónicas entre otras cosas.

MICROCONTROLADOR: Circuito integrado constituido por tres elementos fundamentales: Unidad Central de Procesamiento (CPU), Memoria y Unidad de Entrada y Salida.

MONOLÍTICO: En el contexto de este trabajo hace referencia a sistemas integrados por un solo robot.

LUMINANCIA: Magnitud física definida como la *intensidad luminosa* por unidad de área y cuyas unidades se expresan en *candela/m*^{2}.

ODOMETRÍA: Es la estimación de la posición de un robot móvil respecto a una posición inicial durante su navegación. Para realizar esta estimación se usa información sobre la rotación de las ruedas para calcular los cambios en la posición a lo largo del tiempo.

OPENGL: Acrónimo de *Open Graphics Library* que corresponde a un estándar multilenguaje y multiplataforma que permite crear aplicaciones gráficas en 2D y 3D.

OPEN SOURCE: Es el término usado para denominar al software cuyo código es de libre distribución y puede ser modificado y mejorado sin ninguna restricción

PUERTO SERIAL: También denominado puerto serie, es una interfaz de comunicaciones de datos digitales, frecuentemente utilizado por computadoras y periféricos, en donde la información es transmitida bit a bit enviando un solo bit a la vez, en contraste con el puerto paralelo que envía varios bits simultáneamente.

ROBOT MOVIL: Robot que posee la capacidad de moverse en un determinado espacio a través de algunos medios convencionales como piernas, ruedas u orugas.

SENSOR: Dispositivo capaz de transformar magnitudes físicas o químicas, llamadas variables de instrumentación, en magnitudes eléctricas. Las variables de instrumentación dependen del tipo de sensor y pueden ser por ejemplo temperatura, intensidad luminosa,

distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, acidez (pH), etc.

Capítulo 1. INTRODUCCIÓN

La palabra *robot* tiene su origen en el termino checo "*robota*" que significa labor o trabajo y que fue usada por primera vez por Karel Capek en 1921 en su obra teatral *R.U.R (Rossum´s Universal Robots)* haciendo referencia a "seres artificiales" creados por humanos para realizar ciertas labores, quienes se rebelan y asumen el control del mundo (Sabbatini, 1999). Sin embargo, hoy en día la palabra robot se usa para referirse a algún dispositivo o sistema electro-mecánico, dotado de algunos mecanismos que emulan comportamientos animales o humanos para ejecutar una determinada tarea. En muchos casos, el grado de dificultad de ésta determina la complejidad en el diseño del robot por lo que para tareas muy complicadas se tienen robots estructuralmente complejos, costosos y poco versátiles.

Ante este hecho, durante más de dos décadas, ha surgido un creciente interés por el estudio de sistemas que integran varios robots conocidos con el nombre de *Sistemas Multi-Robot* (SMR) y que han demostrado ciertas ventajas (en algunas tareas) que los hacen más eficientes respecto a sistemas monorobot. Durante este tiempo, la investigación en esta área ha progresado en muchos frentes que incluyen arquitecturas, métodos de comunicación, coordinación, control entre muchos otros. Muchas veces bajo la inspiración y soporte de otros campos del conocimiento como la biología, la inteligencia artificial, los sistemas distribuidos entre otros.

En términos generales, un SMR está conformado por varios robots que permiten realizar de manera conjunta una determinada tarea. Dependiendo del dominio de aplicación, cada robot está dotado de ciertas características mecánicas, eléctricas y de cómputo que le permiten mostrar comportamientos específicos. Lo que significa que un SMR puede estar conformado por robots de diferentes tipos como por ejemplo, robots manipuladores para labores en la industria, robots antropomorfos (*humanoides*) usados en labores domésticas y de limpieza o robots en forma de vehículos utilizados en diversos dominios como la exploración de ambientes peligrosos, la vigilancia y control de áreas extensas o la recolección de sustancias nocivas entre otros. Estos últimos también se denominan *robots móviles,* están dotados de ruedas u orugas para desplazarse por terrenos de diferente naturaleza.

El estudio en SMR es naturalmente una extensión de la investigación en sistemas monorobot y hoy por hoy constituye una disciplina como tal (Cao, Fukunaga, & Kahng, 1997), la cual es el resultado de muchos estudios, trabajos y esfuerzos de investigación que como ya se indicó, representan un amplio conocimiento en problemas como la

coordinación, la comunicación, la localización, el aprendizaje y que han generado diferentes enfoques y paradigmas.

Este trabajo ha sido titulado *Robots Móviles Colaborativos: Un caso de estudio basado en cooperación eusocial* porque los robots utilizados durante los planteamientos aquí realizados son robot móviles que implícitamente colaboran entre sí bajo el concepto de la *entomología* denominado *cooperación eusocial* caracterizado por una división del trabajo en busca de un beneficio común, que permite la simplicidad de cada individuo, la ejecución concurrente de tareas y el control descentralizado de estas.

Con esto como fin, el presente trabajo hace básicamente los siguientes tres planteamientos:

1. Cada individuo del sistema implementará un enfoque reactivo o basado en comportamientos, los cuales se activan o desactivan a través de condiciones leídas del entorno mediante sus sensores.
2. La localización de un individuo dentro de su entorno se hará a través de *lógica borrosa* que permita determinar su ubicación de tal manera que no sea necesario, disponer de modelos geométricos o topológicos que demanden demasiados recursos de cómputo.
3. La interacción entre robots se hará de manera implícita, mediante un algoritmo basado en probabilidad y una red de perceptrones.

Al ser implementados se observó la generación de comportamientos colectivos emergentes para realizar un tarea de manera conjunta permitiendo además garantizar la escalabilidad y tolerancia a fallas.

El sistema que aquí se plantea está constituido primordialmente en dos partes:

1. Parte Estructural: Constituida por el conjunto de dispositivos físicos (eléctricos o mecánicos) tales como el hardware de cada uno de los robots y algunos elementos del ambiente o entorno de operación del sistema.

2. Parte Funcional: Constituida por los comportamientos del sistema que incluyen los comportamientos reactivos de cada individuo, el esquema de interacción o comunicación y el mecanismo de localización

En la parte estructural, este trabajo presenta de manera general las características físicas mínimas que debe tener un sistema como el que aquí se plantea. En lo que respecta a la parte funcional, esta constituye el principal problema a resolver, cuya solución se describe a lo largo de este documento.

Todos los planteamientos realizados aquí fueron implementados y probados a través de software, sin embargo al final de este documento se presentan algunas implementaciones en robots físicos.

A parte del presente capítulo, donde se presenta de forma general el propósito y la importancia de este trabajo. El resto del documento ha sido organizado de la siguiente manera:

Capitulo 2: Presenta una revisión del estado del arte de los SMR exponiendo sus ventajas, aplicaciones, características, y campos afines con el objetivo de dar una panorámica que permita abordar y comprender el trabajo presentado.

Capitulo 3: Expone la problemática que motivó a la realización del trabajo y plantea las hipótesis y los objetivos que guían su desarrollo, exponiendo además la relevancia y posibles contribuciones del mismo.

Capitulo 4: Muestra el procedimiento mediante el cual se aborda la solución al problema planteado. En él se presentan las propuestas correspondientes al diseño estructural constituido por el hardware y el ambiente de operación del sistema, así como también el diseño funcional conformado por los comportamientos y las estrategias de comunicación y localización. De manera adicional se muestra el software construido para la simulación y el proceso llevado a cabo para la implementación en robots reales.

Capitulo 5: Presenta todas las pruebas y resultados que permitieron comprobar las hipótesis y los planteamientos realizados en los capítulos anteriores.

Capítulo 6: Aquí se presentan de forma sintetizada los resultados obtenidos contrastados con las hipótesis y los objetivos planteados en el Capítulo 3.

De esta manera, se espera que las propuestas aquí presentadas sirvan como base para la implementación en diferentes dominios aplicación. Además de aportar al enriquecimiento del campo de los SMR y a continuar con esta línea de investigación en la Universidad Nacional Autónoma de México.

Capìtulo 2 SISTEMAS MULTI-ROBOT (SMR)

En este capítulo se presenta una revisión del estado del arte de los SMR exponiendo sus principales conceptos, potenciales ventajas, dominio de aplicación, características, y campos afines con el objetivo de dar una panorámica que permita abordar y comprender el trabajo presentado.

2.1. DEFINICIÓN

Un Sistema Multi-Robot SRM es un grupo de robots homogéneos o heterogéneos que tienen las siguientes características (Verret, 2005):

Cooperación: Es la unión explícita o implícita de operaciones u acciones entre un grupo de individuos.

Conciencia: Es el conocimiento que un individuo tiene de sí mismo y de su entorno.

Comunicación: Es el acto de enviar o transmitir información de manera directa o indirecta a otro u otros individuos.

Coordinación: Es el mecanismo mediante el cual se disponen metódicamente todos los individuos y componentes del sistema para unir esfuerzos que contribuyan a su objetivo.

Se dice que un grupo de robots es homogéneo si cada uno de ellos comparten características y capacidades similares como por ejemplo, capacidad de procesamiento y cómputo, mecanismos de comunicación, dispositivos sensoriales y de actuación etc. En caso contrario será heterogéneo.

2.2. VENTAJAS

Para muchos problemas y aplicaciones los SMR ofrecen mejores soluciones que sistemas de un solo robot. De acuerdo con lo expuesto en (Dias, 2004) existen varias razones por las cuales es preferible utilizar SMR:

1. En muchos casos es posible diseñar un solo robot capaz de ejecutar tareas complejas, sin embargo, su diseño puede ser bastante difícil y costoso por lo que utilizar varios robots podría disminuir dicha dificultad y costo. Por ejemplo, un solo robot puede lograr mover objetos pesados si su diseño es apropiado. Sin embargo, en muchos casos, es más fácil diseñar un equipo de robots que coopere para mover objetos pesados de una forma más eficiente.

2. Un grupo de robots puede completar una tarea dada más rápido y eficientemente que un solo robot a través de la división del trabajo en sub-tareas que se ejecutan concurrentemente en dominios de aplicación donde las tareas pueden ser descompuestas. Dominios de aplicación tales como exploración de áreas desconocidas y búsqueda de objetos requieren un completo cubrimiento de una extensa área. Problemas como estos pueden ser descompuestos de tal forma que un grupo de robot pueda dividir la carga de trabajo y ejecutar partes de la tarea o misión concurrentemente, mejorando el desempeño en comparación con el uso de un solo robot.

3. En un SMR pueden existir robots diseñados para un solo propósito que permiten tener cierto grado de especialización en el sistema.

4. Un SMR tiene un más alto grado de robustez en comparación con un sistema de un solo robot, ya que si un miembro del sistema falla el resto del sistema puede compensar la pérdida y continuar con el cumplimiento de sus objetivos.

2.3. DESVENTAJAS

Desafortunadamente, el principal problema a la hora de implementar un SMR es la coordinación o control que en muchas ocasiones implica resolver problemas de tipo NP. Coordinar un SMR es una tarea más complicada respecto a sistemas de un solo robot no solo por el incremento en el número de robots sino además por la necesidad de que los robots trabajen juntos para cumplir su objetivos eficientemente. Esto sumado a ambientes o entornos dinámicos, comunicación entre robots y múltiples requerimientos del sistema que adicionan complejidad.

2.4. APLICACIONES

Existen muchos dominios de aplicación en áreas como el transporte, la industria, los servicios, investigación, entretenimiento, vigilancia, minería, agricultura entre muchas otras, donde es posible usar SMR eficientemente. A continuación se presentan brevemente algunos posibles dominios de aplicación:

Uso de SMR para tareas en locaciones remotas. Muchas aplicaciones a futuro requerirán de un grupo de robots que ejecute de manera autónoma tareas complejas mientras remotamente algún humano interviene esporádicamente para alterar un procedimiento o remediar alguna situación más allá del alcance y capacidades de los robots. Ejemplos de este tipo de dominio de aplicación son la exploración y construcción interplanetaria o la exploración científica de ambientes peligrosos.

Figura 1. Robot de Exploración Espacial (Kalipedia, 2008)

Uso de SMR para la ayuda en la búsqueda y el rescate. En algunas situaciones como la ocurrencia de desastres naturales se requiere el despliegue rápido de cuerpos de búsqueda y rescate de sobrevivientes. Dependiendo de la magnitud y tipo del siniestro, en muchos casos las labores de estos cuerpos de rescate se ven restringidas por operaciones de alto riesgo, difícil acceso a las áreas de búsqueda o recursos insuficientes en cuyo caso el uso de SMR tendría un gran impacto como apoyo a estas labores.

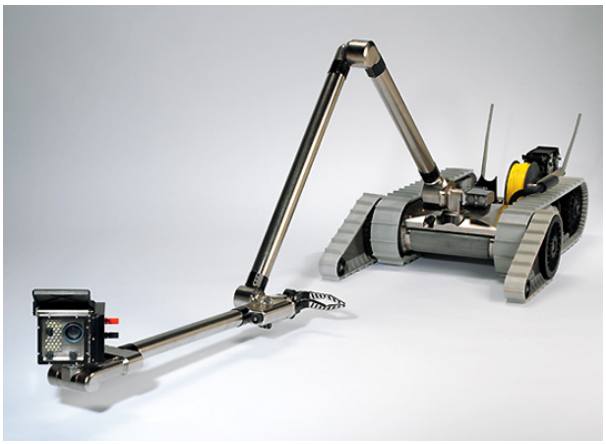


Figura 2. Robot de Rescate (OUNAE, 2008)

Uso de SMR para centros de distribución. La ubicación, el almacenamiento y el transporte de productos dentro de los centros de distribución constituyen tareas que en las cuales los SMR pueden potencialmente contribuir a mejorar sus tiempos y costos de ejecución

Figura 3. Centro de Distribución (Posible aplicación para SMR)

Uso de SMR para educación y entretenimiento. Juguetes, herramientas educativas y sistemas de entretenimiento han alcanzado una gran popularidad que seguirá creciendo en el futuro. Muchos de estos sistemas requieren esfuerzos coordinados por múltiples robots. En lo que respecta a la educación, un claro ejemplo es “Robocup”, una iniciativa de educación e investigación internacional que busca aplicar y proponer técnicas de

Inteligencia Artificial, *Robótica* y áreas afines, con el fin de solucionar un problema estándar como es el juego de fútbol.

Figura 4. Robots futbolistas (Robocup, 2008)

Uso de SMR para líneas de ensamblado y plantas de producción. La automatización es una tendencia latente en las plantas de producción con el fin de minimizar costos, aumentar eficiencia, incrementar la seguridad y proporcionar calidad, lo cual proporciona un escenario para el uso de SMR. Una ventaja de este dominio de aplicación es que el ambiente en el cual operan los robots es generalmente controlado y por tanto más fácil de implementar.

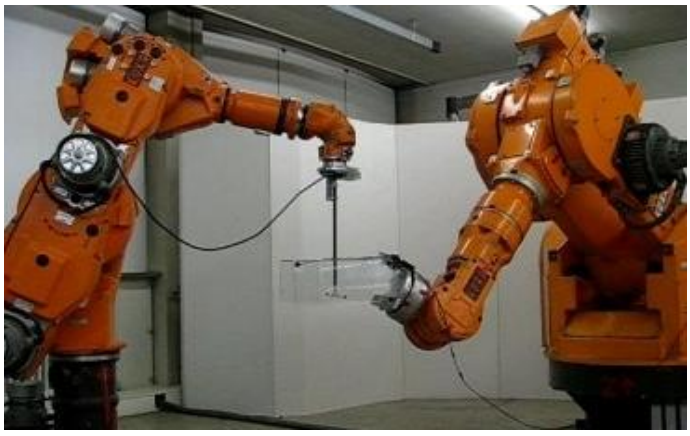


Figura 5. Sistema de Robots Industriales (Power House Museum, 2008)

Uso de SMR para la recolección de residuos tóxicos. Tareas tales como la limpieza y recolección de residuos altamente tóxicos como desechos nucleares, desechos de zonas de desastre etc., pueden ser realizadas de una manera eficiente a través de un SMR evitando el contacto de dichos residuos con humanos.

Figura 6. Robot para limpieza de residuos en el agua (IRobot, 2008)

Uso de SMR en vigilancia y control. En algunas tareas de inspección y vigilancia es necesario tener una amplia área de cobertura y visualización de grandes superficies comerciales, almacenes, laboratorios, edificios de oficinas, estadios e incluso domicilios particulares. Por esta razón, el uso de SMR puede proporcionar soluciones eficientes ya que en muchos casos estos robots incorporan cámaras, sensores infrarrojos, detectores de ruidos y de gases, radar y otros sistemas específicos para la vigilancia y la seguridad capaces de aplicar reconocimiento facial y biométrico a personas y objetos y cubrir extensas áreas de manera concurrente.

Figura 7. Robot de vigilancia OFRO (Robowatch, 2008)

2.5. TAXONOMIA DE LOS SISTEMAS MULTI-ROBOT

El objetivo de una taxonomía o clasificación es dar claridad sobre el alcance, restricciones y complejidad de posibles diseños. Al respecto, muchos autores han propuesto diversas clasificaciones dirigidas por varios aspectos de los SMR como por ejemplo arquitecturas, estrategias de resolución de conflictos, aprendizaje, problemas geométricos, tamaño del sistema (número de robots) , topologia de la comunicación, etc. Al respecto, en (Dudek, Jenkin, Milios, & David, 1997) y (Balch & Parker, 2002) se pueden encontrar varias propuestas de clasificación que consideran estos y muchos otros aspectos.

En este trabajo se presenta una taxonomía dirigida exclusivamente por la arquitectura, la comunicación, la cooperación y la localización con base en los expuesto en (Verret, 2005) y que se ilustra en la Tabla 1. Considerando, que estos aspectos o caracteristicas son más intuitivos e identificables a la hora de proponer un diseño o categorizar un SMR.

Caracteristica	Categorias
Arquitectura	Descentralizado Centralizado
Comunicación	Directa Indirecta
Cooperación	Eusocial Cooperativo Intencional
Localización	Local Global

Tabla 1. Taxonomía de los Sistemas Multi-Robot

2.5.1. ***Arquitectura.*** En el contexto de la robótica, la palabra arquitectura se refiere a la organización, función e interacción de los componentes (hardware y software) que integran un robot. Al respecto, en la literatura se encuentran basicamente tres tipos de arquitecturas.

- Deliberativa: El robot posee una serie de *sensores* y un modelo simbólico del entorno. Con base en el modelo y las condiciones del entorno medidas por los sensores, planifica la toma de decisiones (alcanzar objetivo, seguir trayectoria). Las desventajas principales son el

alto requerimiento de computo y memoria y la dificultad de operar en un ambiente dinámico o desconocido (Hibilisco, 2003).

- Reactiva: El robot dispone de una serie de sensores que permiten medir caracteristicas o condiciones del entorno y apartir de ellas activar *comportamientos* que ejecutan acciones a través de *actuadores*. Por lo tanto, su capacidad es responder a los estimulos del ambiente por lo que no requieren una representación de éste ni la planeacción de acciones o movimientos. Esta arquitectura es denominada basada en comportamiento (Behavior Based).

- Hibrida: En los utimos años ha tenido gran acogida ya que combina la rápida respuesta de la arquitectura reactiva con la toma de decisiones basada en modelos internos de la arquitectura deliberativa (Hibilisco, 2003).

A partir de estas tres arquitecturas básicas, en el área de los SMR se han propuesto arquitecturas más complejas donde los robots en un bajo nivel pueden ser capaces de reaccionar rápidamente a cambios dinámicos del entorno y ejecutar rutinas tales como evadir obstáculos. En un nivel intermedio los robots ejecutan tareas de más largo plazo como la planeación de rutas y la construcción de modelos del ambiente. Finalmente, en un nivel más alto los robots pueden ser capaces de coordinarse con otros ejecutando tareas asíncronas como la búsqueda de objetos o tareas síncronas como transporte cooperativo (Verret, 2005).

La coordinación asegura que un grupo de robots actúen de una manera coherente, lo cual implica sincronización de acciones e intercambio de información entre robots (comunicación). Tanto la sincronización como la comunicación dependen fuertemente de los objetivos del sistema, las características de los robots y el ambiente en el que actúan (Chaimowicz, Kuma, & Campos, 2004). Existen diferentes enfoques de coordinación entre los que se destacan:

- Enfoque Centralizado: Existe un agente que tiene el control de todos los robots los cuales actúan como esclavos, lo cual requiere que este agente tenga un conocimiento global del entorno y un diseño preciso para considerar todos lo posibles estados del sistema.
- Enfoque Descentralizado: Cada uno de los robots perciben el ambiente y pueden tomar decisiones de manera autónoma sin recibir instrucciones de un superior, lider o control central. Este enfoque a su

vez se divide en *jerarquico* y *distribuido*. En la coordinación jerárquica los robots trabajan diferentes niveles de abstracción de un problema. En un mismo nivel se establece una configuración distribuida, si hay más de un robot. Para resolver un problema cada robot divide el problema en sub-problemas que él puede resolver, con la cooperación de los agentes que están al mismo nivel y sub-problemas que sabe que los agentes de niveles inferiores de la jerarquía pueden resolver

Por lo tanto, un SMR puede ser categorizado respecto a la arquitectura en centralizado o descentralizado dependiendo de la estrategia de coordinación aunque esto no implica que un sistema sea totalmente centralizado o descentralizado como señala Cao en (Cao et. al., 1997).

2.5.2. ***Comunicación.*** A través de la comunicación es posible enviar o transmitir información a otro u otros individuos. En el contexto de los SMR esta comunicación puede ser de dos tipos:

- Directa: Es la transmisión y recepción intencional de información. Esta se realiza generalmente con la ayuda de mecanismos o protocolos de comunicación como IEEE 802.11 wireless ethernet, infrarojo o Bluetooth, para lo cual se requiere hardware especial de comunicación. En lo referente a los SMR comúnmente se usa *broadcasting* o *unicasting* para realizar la comunicación. Un robot puede usar broadcasting para anunciar su localización o estado a todo el sistema o podria usar unicasting para comunicarse exclusivamente con otro robot (Zebrowski, 2004).

- Indirecta: También denominada comunicación implícita, es aquella que ocurre como efecto lateral de otras acciones que generan información sin un remitente explícito. En el caso de algunos insectos, la acción de encontrar una fuente de alimento genera la emisión de señales químicas que pueden ser percibidas por otros individuos.

2.5.3. ***Cooperación.*** En la naturaleza existen basicamente dos tipos de comportamientos cooperativos: eusocial y cooperativo intencional (McFarland, 1994).

El comportamiento *eusocial* presente en varias especies de insectos como termitas, abejas, avispas y hormigas está caracterizado por una clara división de tareas de tal forma que un número reducido de individuos se encarga de reproducirse mientras otros ayudan en tareas tales como la alimentación, cuidado de las crías y defensa (Jaffé, 1993). En este tipo de conducta los individuos sacrifican su propio beneficio con el fin de garantizar la supervivencia de toda la colonia. Además dicha conducta se realiza de manera inconsciente ya que la división de tareas y roles esta preestablecida genéticamente. Una buena definición en el contexto de la inteligencia artificial se puede encontrar en (Mataric, 1994) donde se afirma que este comportamiento es el conjunto de acciones de un agente sobre el medio de tal forma que éstas contribuyen a lo objetivos de otros agentes. En resumen el comportamiento *eusocial* está orientado a la supervivencia del grupo en lugar de los individuos.

Por su parte, el comportamiento cooperativo tambien denominado *cooperativo intencional* está presente en algunas especies de animales vertebrados y es el resultado de interacciones entre individuos "egoistas". Contrario al comportamiento eusocial, éste no es motivado por un beneficio conjunto sino por el deseo y la intención explícita de cooperar con el fin de maximizar el beneficio y la utilidad individual (Cao et. al., 1997).

Muchos de los trabajos referetentes a SMR han adoptado o han sido inspirados por este tipo de comportamientos. Los sistemas basados en comportamientos de tipo eusocial hacen parte de una subdisciplina de los SMR denominada *Robótica de Enjambre* de la que se habla más adelante. En consecuencia, un SMR puede ser clasificado respecto al tipo de cooperación que implemente como eusocial o cooperativo intencional.

2.5.4. ***Localización***. Es el proceso de estimar la posición de un robot dentro de un área que determina un sistema de coordenadas. Tanto en ambientes internos como externos los robots necesitan implementar alguna técnica que les permita conocer su posición con el fin de completar satisfactoriamente su objetivo.

Muchas de las técnicas usadas al respecto, utilizan como fuentes de información las observaciones del ambiente o entorno a través de los *sensores* del robot y/o las medidas de *odometría* que permiten registrar los incrementos de posición del robot. Otras técnicas utilizan elementos incorporados al ambiente denominados "marcas" que permiten triangular la posición del robot. En (Borenstein, Everett, & Feng, 1996) se muestran estas técnicas con más detalle.

Generalmente, el problema de localización suele clasificarse en:

- Localización Local: Consiste en determinar la posición del robot a través del conocimiento a priori de su posición inicial, haciendo seguimiento del movimiento del robot (traslacional y rotacional) usando *odometría*. Sin embargo la *odometría* presenta errores acumulativos clasificados como sistematicos y no sistematicos. Los primeros dependen de las propiedades del robot (resolución de encoders, diámetro de las llantas) mientras que los segundos dependen de las condiciones del ambiente como por ejemplo el coeficiente de fricción de la superficie de desplazamiento del robot. Para corregir dichos errores, ésta técnica se suele combinar con el uso de marcas para hacer triangulación y corregir la posición (Borenstein et. al., 1996), (Negenborn, 2003).

- Localización Global: Permite determinar la posición del robot sin tener como referencia una posición inicial. Para lo cual se manejan hipótesis de la posición utilizando enfoques probabilisticos entre los que se destacan el análisis bayesiano, las cadenas de markov o el filtro de Kalman, en (Negenborn, 2003) y (Gallardo, 1999) se hace una descripción de estas técnicas. En (Martín, Matellan, Barrera, & María, 2006) se presenta una combinación del filtro de Kalman con métodos de Lógica Difusa.

  Muchas de las técnicas de localización mencionadas requieren además, de un modelo o representación del entorno con el fin de comparar las lecturas de los *sensores* del robot con dicho modelo para actualizar su posición. En general, existen dos tipos de modelos del entorno: geométrico y topológico. Los modelos geométricos representan el entorno a través de sus caracteristicas geométricas como distancias, dimensiones, aristas y esquinas de los objetos que lo constituyen (Gallardo, 1999). En la siguiente figura se observa un modelo geometrico en el que resaltan caracteristicas del entorno como esquinas y aristas.

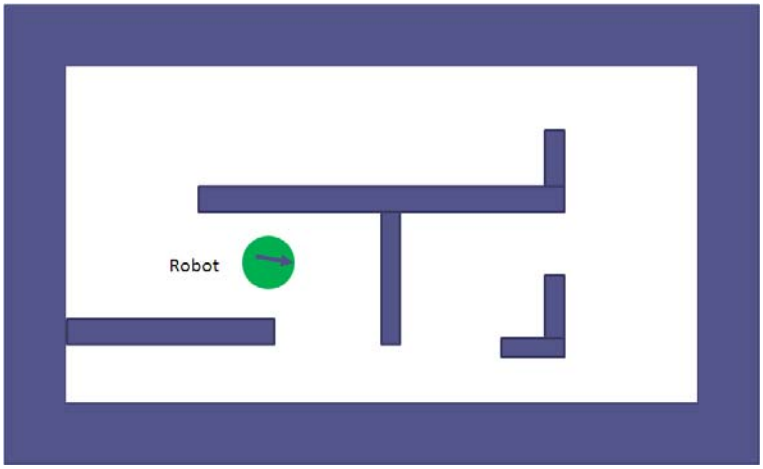


Figura 8. Ejemplo de modelo geométrico del entorno.

Por otro lado los modelos topológicos del entorno utilizan un grafo cuyos nodos representan lugares de interes para el robot y los arcos o aristas representan el camino para llegar a estos. Los nodos generalmente son elementos o marcas en el entorno cuya posición es conocida y son distinguibles a través de los sensores del robot, en (Kuipers, 1999) se presenta una propuesta muy común de este tipo de modelo. En la siguiente figura se presenta un contraste entre el modelo geométrico y el modelo topológico.

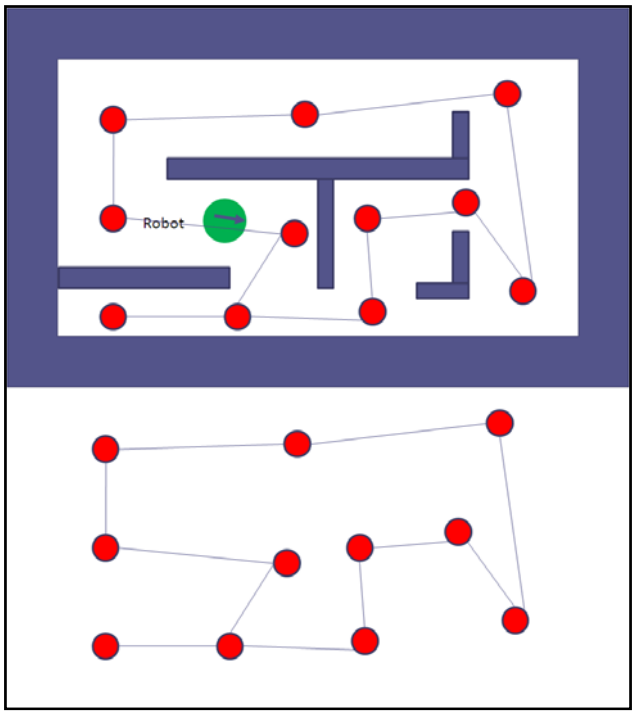


Figura 9. Modelo Topológico vs. Modelo Geométrico

Las técnicas hasta aquí citadas tienen sus limitaciones, por ejemplo, para entornos demasiados complejos, mantener un modelo actualizado y consistente del entorno representa una dificultad enorme que se manifesta en un considerable consumo de recursos de cómputo y que de una u otro forma afectan los tiempos de respuesta del robot. Por otra parte muchas de las técnicas mencionadas han sido enfocadas para ambientes estáticos que asumen que el robot es el único objeto en movimiento, lo cual en un SMR no es nada práctico.

Por estas razones, uno de los planteamientos de este trabajo es un esquema de localización que teniendo en cuenta las limitaciones de cómputo de cada individuo (robot) del sistema, logre determinar su ubicación con base en un enfoque probabilistico sin modelo geométrico o topológico del entorno, permitiendo así dar versatilidad y dinamismo al sistema y a su vez disminuir la complejidad en el procesamiento de cada robot.

2.6. CAMPOS RELACIONADOS

Desde sus orígenes a principios de la decáda de los 80, el campo de los SMR ha tenido un crecimiento continuo a través de numerosos estudios e investigaciones que en muchas ocasiones han sido influenciados e inspirados por otras áreas o campos del conocimiento. En las siguientes secciones se expone de manera descriptiva algunos de los más influyentes campos relacionados con los SMR con el fin de contextualizar el trabajo que en este documento se presenta.

2.6.1. ***Sistemas Biológicos***. Muchos de los trabajos existentes en el campo de SMR específicamente los referentes a la cooperación, citan sistemas biológicos como inspiración o justificación (Balkenius, 1994). Son muy conocidos los comportamientos colectivos de algunas especies animales como hormigas, abejas y termitas que demuestran que simples agentes o individuos pueden realizar complicadas tareas. Es sabido que la capacidad cognitiva de estos insectos es muy limitada y que comportamientos complejos emergen como resultado de la interacción obedeciendo a reglas simples (Cao, et. al., 1997). De igual forma, otras especies de organismos unicelulares tales como bacterias o amibas, presentan comportamientos interesantes de coordinación bajo condiciones normales (suficiencia de alimento, no presencia de antibióticos) que han sido fuente de inspiración a varios trabajos de SMR (Sahin, 2005).

Figura 10. Comportamiento colectivo en hormigas (Low Tech RVing, 2008)

Otros ecosistemas tambien han sido aplicados al desarrollo de SMR, los cuales demuestran comportamientos cooperativos como resultado de la interacción entre individuos que buscan su propio beneficio, por ejemplo, en grupos o manadas de animales como los lobos a la hora de la caza (Parker, 2003). La competencia, presente en algunos animales (incluido el ser humano) también

ha sido inspiración y objeto de estudio para los SMR especificamente para el domino del juego de fútbol (Marsella, Adibi, & Al-Onaizan, 1999).

Figura 11. Cooperación de lobos en la cacería (Galeon, 2008)

De una forma menos directa áreas como las redes neuronales y los algoritmos genéticos cuyos inicios están influenciados por la Biología han sido utilizados en los SMR.

2.6.2. ***Inteligencia Artificial Distribuida.*** La Inteligencia Artificial Distribuida IAD es un campo de la Inteligencia Artificial dedicado al estudio de las técnicas y mecanismos necesarios para la coordinación y distribución del conocimiento y las acciones en un sistema que involucra múltiples entidades. Estas entidades, llamadas generalmente agentes, pueden ser visualizadas colectivamente como una sociedad. Aplicada a los SMR estos agentes son robots que generalmente actúan en ambientes físicos o reales.

En la literatura al respecto, la IAD suele dividirse en dos sub-disciplinas (Balch et. al., 2002):

- Solución Cooperativa de Problemas Distribuidos (SCPD): En esta disciplina el objetivo se centra en cómo un conjunto de entidades (agentes) pueden cooperar e interactuar para dividir y compartir el conocimiento de un problema y desarrollar una solución.
- Sistemas Multi-agente (SMA): En esta disciplina, los agentes son autónomos y generalmente heterogéneos. De manera similar a la SCPD, el objeto de estudio aquí, está dirigido a la coordinación de comportamientos, conocimiento, objetivos, destrezas y planes de acción para resolver un problema a través de los agentes.

Existen algunas diferencias entre estas dos áreas, en (Durfee & Rosenschein, 1994) se presenta un análisis comparativo al respecto. La diferencia fundamental se encuentra en la coordinación de los agentes. En la SCPD las tareas y acciones están predeterminadas, existe un plan centralizado para la solución del problema y un solo miembro realiza el control global del sistema. Por su parte, en los SMA, los agentes tienen un grado de autonomía y capacidad de decisión acerca de que acciones son adecuadas para la búsqueda de la solución u objetivo. Sin embargo, SCPD y SMA son disciplinas mutuamente complementarias (Durfee et. al., 1994).

2.6.2. ***Computación Distribuida.*** El campo de los sistemas distribuidos es una fuente constante de ideas y soluciones aplicadas a los SMR. "Un sistema de cómputo distribuido es una colección de dispositivos de cómputo, los cuales pueden residir en locaciones geográficamente separadas" (Wang & Beni, 1990). Desde esta perspectiva, los SMR son sistemas de cómputo distribuido por lo que en varios trabajos han sido aplicados muchas de las teorías de este campo como por ejemplo el bloqueo mutuo (*deadlock*), el paso de mensajes, la asignación de recursos entre otros. Adicionalmente el campo de los sistemas distribuidos ofrece un rico y amplio conjunto de algoritmos, protocolos, análisis y modelos de desempeño en redes de cómputo que pueden ser utilizados en los SMR.

2.7. ROBÓTICA DE ENJAMBRE

La robótica de enjambre es una subdisciplina de los SMR inspirada en el comportamiento de algunas especies de insectos como hormigas, termitas, abejas y avispas. Su objeto de estudio son los sistemas conformados por un gran número de robots relativamente simples que al interactuar entre ellos y con su entorno logran generar comportamientos complejos (Sahin, 2007). Estos comportamientos colectivos tambien se conocen con el nombre de *inteligencia colectiva*.

Las ventajas más sobresalientes con respecto a otros tipos de SMR básicamente son la flexibilidad con la que cada individuo puede adaptase a un entorno cambiante, extremo y adverso; y la *auto-organización* que se refiere a que las actividades no se controlan centralizadamente.

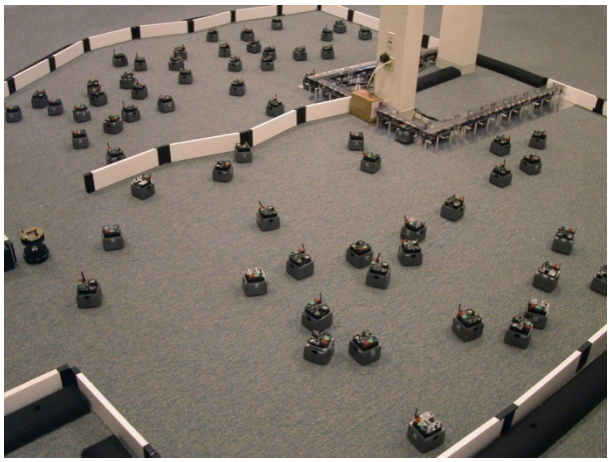


Figura 12. Enjambre de robots compuesto por más de 100 individuos (Schenato, 2007)

La principal caracteristica de un SMR de enjambre es la presencia de *cooperación eusocial* que como se ha mencionado está presente en algunas especies de insectos especialmente hormigas, termitas, abejas y avispas han desarrollado complejas estructuras sociales que facilitan la lucha por la sobrevivencia, haciendo más fácil la búsqueda de alimento, mejorando las oportunidades de defenderse ante depredadores e incluso facilitando el cuidado de las crías y la construcción de nidos y refugios (Jaffé, 1993).

Este tipo de comportamiento tiene las siguientes implicaciones:

2.7.1. ***Simplicidad Individual.*** Cada uno de los individuos tiene capacidades limitadas de memoria, procesamiento, comunicación, localización, aprendizaje etc.

2.7.2. ***Ejecución concurrente de tareas.*** En sistemas de este tipo cada individuo ejecuta sus tareas sin depender de la ejecución de las tareas de otros individuos. En otras palabras cada robot realiza su trabajo de forma asíncrona

2.7.3. ***Control Descentralizado.*** Cada uno de los individuos del sistema perciben el ambiente y pueden tomar decisiones de manera autónoma sin recibir instrucciones de un superior, líder o control central.

2.7.4. ***Comportamientos Colectivos.*** Un comportamiento colectivo es un evento o acción generado de forma espontánea a través de la interacción entre individuos y su entorno. El término tiene su origen en la sociologia a partir de los trabajos de Robert Park referenciados en (Turner, 1967) y Herbert Blumer (Blumer, 1951).

Estas implicaciones se logran a través de mecanismos de interacción implícitos en los cuales la información es el resultado lateral de las acciones de cada uno de los individuos. Este tipo de mecanismos de interacción estan basados en el concepto biológico de *estigmergia* en el cual los individuos del sistema se comunican entre sí indirectamente a través de la modificación de su entorno.

2.8. TRABAJOS RELEVANTES

Desde sus inicios hace más de dos décadas, el campo de los SMR ha experimentado un continuo crecimiento a través de muchos trabajos de investigación en diferentes frentes, por lo que mostrar los trabajos relevantes al respecto no es una tarea fácil. Por tal motivo y con el propósito de una mejor comprensión a la lectura de esta sección, se ha decido agrupar los trabajos de investigación de SMR en dos categorias: Sistemas de Cooperación Eusocial y Sistemas de Cooperación Intencional considerando la clasificación presentanda en la sección 2.4. La razón de esta clasificación está dada por el hecho de cada una de estas categorías tiene características muy particulares en lo que respecta a la arquitectura, comunicación y coordinación que han definido y trazado muchos de los trabajos de investigación en el área. Adicionalmente, tambien se muestran otros trabajos que forman parte del campo de SMR en general.

2.8.1. ***Investigación en SMR de cooperación eusocial.*** Los sistemas de cooperación eusocial han sido objeto de estudio para muchos autores. Theraulaz en (Theraulaz, Goss, Gervet, & Deneubourg, 1990) plantea una estrategia de control cooperativo para un SMR que emula la búsqueda de alimento (*foraging*) a partir del estudio de colonias de avispas. Stell en (Steels, 1990) presenta una simulación que permite generar comportamientos emergentes (derivados de otros comportamientos) a través de una aplicación dirigida al problema de recolección de muestras de rocas en exploración interplanetaria. Drogoul y Ferber en (Drogoul & Ferber, 1994) presenta una simulación aplicada al problema de adquisición o búsqueda de alimento (*foraging*). Mataric en (Mataric, 1994) presenta los resultados de una implementación de un grupo de comportamientos a los que denomina *dispersion, aggregation, flocking* en un grupo de robots físicos. Beni y Wang en (Beni & Wang, 1990) describe métodos y patrones de arbitraje y coordinación. En (Kube & Zhang, 1992) se presentan los resultados de la implementación de una estrategia de control sobre un grupo de robots fisicos que ejecutan tareas de localización y transporte de cajas livianas. En (Arkin, Balch, & Nitz, 1993) se expone una investigación en la que se tratan

problemas de sensado, comunicación y organización social aplicados a *foraging*. Fukuda y Ueyama en (Fukuda & Ueyama, 1994) presentan una propuesta de arquitectura inspirada en la organización celular denominada CEBOT, esta es una arquitectura jerárquica que está basada en células (robots o agentes) maestras que controlan y coordinan subtareas y se comunican con otras células maestras. En (Cai, Fukuda, Arai, Ueyama, & Sakai, 1995) se presenta otra arquitectura jerárquica implementada en robots físicos cuyos comportamientos básicos son pesados o ponderados a través de algunos coeficientes y sumados con el fin de obtener comportamientos compuestos. En (Arnaud, 1999) se propone un algoritmo de aprendizaje para adaptar los coeficientes de estos comportamientos básicos. En (Mitsumoto, Fukuda, Shimojima, & Ogawa, 1995) se presenta un conjunto de robots miniatura denominados MARS (Micro Autonomous Robotic System) que implementan algunas ideas de control basado en la arquitectura CEBOT. En (Jin, Liang, & Beni, 1994) se presenta SWARM, una arquitectura descentralizada que usualmente utiliza robots homogéneos y que busca generar *comportamientos inteligentes* a partir de comportamientos elementales en cada uno de los robots utlizando comunicación implícita.

Entre las referencias más recientes al respecto se tiene (Sugawara, Kazama, & Watanabe, 2004) en las que se propone una implementación virtual de la feromona, a través de gráficas proyectadas en el suelo. En (Purnamadjaja, Iskandar, & Russell, 2007) se realiza una simulación de robots que utilizan feromonas para comunicarse pero limitada a una implementación real.

2.8.2. ***Investigación en SMR de cooperación intencional.*** Los SMR se caracterizan por un número limitado de robots heterogéneos que ejecutan diferentes tareas bajo ciertas restricciones de eficiencia y desempeño, que implican un tipo de cooperación más directa en comparación a sistemas de cooperación eusocial. Usualmente estos sistemas requieren que la mision u objetivo sea dividido en subtareas. Existe un plan que determina que tarea debe ejecutar un robot bajo ciertas condiciones con el fin de maximizar la eficiencia y asegurar la coordinación entre cada uno de los miembros. Al respecto en (Noreils, 1993) se presenta una arquitectura bajo el esquema *sensar-modelar-planear-actuar* la cual incluye tres niveles de control: El nivel de planeación el cual maneja protocolos de coordinación, descompone las tareas dentro de pequeñas subunidades y las asigna a determinados robots, el nivel de control que organiza y ejecuta las tareas asignadas en cada robot y el

nivel de funcionalidad que permite activar los mecanismos de acción controladamente bajo ciertas condiciones. En (Caloud, Choi, Latombe, Le Pape, & Yim, 1990) se describe otra arquitectura bajo el esquema *sensar-modelar-planear-actuar* que incluye un planificador de tareas, un asignador de tareas, un planificador de movimientos y un monitor de ejecución. Cada robot obtiene sus objetivos con base en su estado actual o a través de solicitud de otros miembros del sistema. Por otra parte, Asama y sus colegas en (Asama, Ozaki, Matsumoto, Ishida, & Endo, 1992) presentan un sistema centralizado llamado ACTRESS (*ACTor-based Robot and Equipments Synthetic System*) que permite la comunicación, asignación de tareas y planeación de movimientos entre robots o agentes heterogéneos. En (Caloud et. al., 1990) y (Le Pape, 1990) se presenta una arquitectura denominada GOFER para estudiar la solución de problemas distribuidos usando técnicas de inteligencia artificial tradicional. En GOFER un planificador y agendador central (CTPS, *Central Task Planning Scheduling*) tiene una vista global de la ejecución de tareas y la disponibilidad de cada robot para ejecutarlas. Los robots usan un algoritmo de asignación para determinar sus roles y objetivos. Por último otro trabajo interesante en (Parker, 1998) presenta una arquitectura denominada ALLIANCE para estudiar la cooperación en un grupo de robots heterogéneos, debilmente acoplados. Cada robot implementa una arquitectura reactiva (basada en comportamientos) e implementa un mecanismo que permite que cada uno sienta los efectos de sus propias acciones y las de otros robots a través de percepción y comunicación explícita. Otra arquitectura como AuRA se puede encontrar en (Arkin & Balch, 1997) la cual combina la planeación de una arquitectura deliverativa con la rápida respuesta de una arquitectura reactiva.

2.8.3. ***Otros trabajos importantes.*** Además de los trabajos referenciados anteriormente, existen otros destacados que han permitido organizar y dirigir la investigación y el estudio en SMR. Al respecto se tienen trabajos como (Cao et. al., 1997) donde se presenta un estudio del estado del arte de los SMR y se dan algunas direcciones a futuro. Trabajos más recientes con propósitos similares se encuentran en (Parker, 2003) y (Verret, 2005). Un trabajo de múltiples referencias en el campo es el expuesto en (Dudek, et. al., 1997) en el que se propone una taxonomía de los SMR a partir de ciertos aspectos y características al igual que (Verret, 2005). Algunos trabajos que recopilan los principales conceptos, métodos y técnicas de los SMR se encuentran en (Liu & Wu, 2001) y (Balch et. al., 2002).

Muchos de los trabajos aquí citados proponen SMR similares al que se trata en este documento. Sin embargo, algunos de estos son restrictivos en cuanto a las condiciones y dinamismo del entorno, en otros casos son limitados respecto al número de individuos o demandan demasiado cómputo y en otros son exclusivamente simulaciones que no consideran dificultades técnicas a la hora de una implementación real. Sin embargo, todos constituyen un valioso punto de partida para generar propuestas que permitan sobrepasar estos límites.

Capìtulo 3 PRESENTACIÓN DEL PROBLEMA

Teniendo como preámbulo el capítulo anterior en el que se presentó una panorámica general del campo de los SMR, el presente capítulo tiene como fin exponer la problemática que motivó a la realización del trabajo que aquí se trata y plantear los objetivos y las hipótesis que guiaron su desarrollo, exponiendo además la relevancia y posibles contribuciones del mismo.

3.1. MOTIVACIÓN

Como se ha mencionado, los SMR ofrecen muchas ventajas en diferentes dominios de aplicación. Dependiendo de los requerimientos y restricciones de cada dominio, un SMR puede ser de arquitectura centralizada o descentralizada.

Al respecto, se mencionó que en una arquitectura centralizada existe una entidad o agente que regula y controla la ejecución de las tareas en el sistema lo cual representa algunas potenciales desventajas:

1. En caso de que la entidad o agente de control presente una falla esta se verá reflejada en todo el sistema.

2. La entidad o agente de control requiere procesamiento concurrente que demanda alta capacidad de cómputo y almacenamiento.

3. Un incremento en el número de robots puede afectar el desempeño por lo que el sistema tendría límites de escalabilidad.

Ante estas potenciales desventajas otra alternativa es usar una arquitectura descentralizada en la que cada uno de los individuos del sistema actúa de manera independiente sin recibir instrucciones de un agente o entidad de control central. Si bien es claro que esta alternativa puede presentar un mejor desempeño surgen los siguientes cuestionamientos:

¿Cuál debería ser el esquema de colaboración o cooperación para lograr el objetivo del sistema?

¿Cómo deberían ser las interacciones entre cada individuo con su entorno y otros individuos?

¿Qué tanto debería saber un individuo acerca de los objetivos, el ambiente y otros individuos?

Responder a estas preguntas no es una tarea fácil, ya que depende del dominio de aplicación al cual está dirigido el sistema. En lo que se refiere a la primera pregunta, en algunos SRM es posible que cada uno de los agentes disponga de planes de acción y estrategias de cooperación que permitan el trabajo concurrente, el problema radica en que en muchas ocasiones esto requiere complejos algoritmos para evaluar todos los posibles estados del sistema. Con respecto a la segunda pregunta, es posible tener esquemas de comunicación que utilizan sofisticados protocolos pero que tienen restricciones en cuanto al ancho de banda del canal y al número de agentes que pueden transmitir lo cual representa una clara limitante en la escalabilidad del sistema. Por último, es posible encontrar individuos que tienen un conjunto detallado de información de su posición respecto al ambiente, localización de otros individuos, mapas de navegación, planes de acción etc., que representan un alto requerimiento en cómputo y memoria que no siempre puede ser satisfecho.

Por lo tanto, a pesar de que una arquitectura descentralizada puede presentar mejores resultados de desempeño (en contraste con una arquitectura centralizada) también existen potenciales desventajas que se deben tener en cuenta y las cuales constituyen la motivación del presente trabajo en el que se propone un SMR con las siguientes características:

- Cada robot del sistema implementará un enfoque reactivo (basado en comportamientos) que no demanda altos recursos de cómputo y memoria.
- La interacción entre individuos será implícita a través del cambio en ciertos atributos del entorno.
- La localización se implementará en cada robot mediante un algoritmo que evite almacenar el estado global del ambiente y procesar y almacenar complejas estructuras de datos que afectan el desempeño de cada individuo y del sistema en general.

Todo esto bajo la inspiración del concepto del campo de la entomología denominado cooperación eusocial del cual se hizo referencia en el capitulo anterior.

3.2. OBJETIVO GENERAL

Analizar y diseñar un sistema de robots móviles colaborativos de arquitectura descentralizada basado en el concepto de cooperación eusocial, que pueda ser

implementado en diferentes dominios de aplicación y que ofrezca características de escalabilidad, tolerancia a fallas y bajos requerimientos de cómputo a nivel de cada individuo.

3.3. OBJETIVOS PARTICULARES

Definir comportamientos básicos en cada uno de los robots de tal forma que a partir de estos se puedan generar comportamientos colectivos.

Definir un esquema de interacción implícita entre individuos, a través de características del entorno.

Utilizar un esquema de localización que permita cambios en el entorno y que no demande una gran cantidad de recursos de cómputo.

Realizar diferentes experimentos y simulaciones que permitan comprobar el desempeño del sistema.

3.4. HIPÓTESIS DE INVESTIGACIÓN

Bajo este preámbulo a continuación se formulan las hipótesis que guiaron el trabajo aquí presentado y las cuales constituyeron un factor importante para determinar algunas de las conclusiones finales.

- Es suficiente implementar un enfoque reactivo en cada robot para generar comportamientos colectivos.
- Para algunas tareas, los robots que integran el sistema no necesitan esquemas complejos de localización que demandan considerables recursos de cómputo.
- Se puede implementar un esquema de interacción mediante el cambio que cada uno de los individuos hace sobre el entorno.

3.5. CONTRIBUCIÓN Y RELEVANCIA

Los aportes de este trabajo están orientados en dos frentes: el académico y el social. En lo académico se espera que éste sea una buena referencia para trabajos futuros en el área de los SMR. Se espera además que contribuya a seguir una línea de investigación en la Universidad Nacional Autónoma de México que permita explorar y proponer técnicas, mecanismos de

control, mecanismos de comunicación y enfoques de cooperación que permitan crear sistemas aplicados a dominios reales.

Ahora bien, tal vez a simple vista este trabajo sea solo un ejercicio académico que no representa un impacto social directo, sin embargo, éste puede contribuir al desarrollo de SMR que permitan realizar tareas en diversos dominios y áreas de aplicación como las que se enumeran a continuación:

- Exploración y búsqueda en áreas de difícil acceso.
- Manipulación y recolección de residuos tóxicos.
- Vigilancia y control de áreas extensas.
- Asistencia en la búsqueda y rescate de víctimas ante siniestros naturales.
- Ubicación y manipulación de productos en grandes superficies de almacenamiento como bodegas y almacenes.

Capìtulo 4 METODOLOGÍA

Para abordar el problema planteado, se utilizó una metodología que a grandes rasgos estuvo conformada por los siguientes pasos: el primero fue determinar los requerimientos funcionales y no funcionales del sistema propuesto, con esto como antecedente, se procedió a establecer el diseño estructural y funcional del sistema para satisfacer dichos requerimientos. Seguidamente se implementó una simulación del sistema el fin de corroborar este diseño. Adicionalmente se muestra una aproximación a una implementación real utilizando robots comerciales de propósito general tipo *LEGO®*

2.1. REQUERIMIENTOS FUNCIONALES

Un requerimiento funcional se define como el comportamiento esperado de un sistema desde la perspectiva de un usuario, entendiendo como usuario, a alguna entidad externa al sistema que demanda su funcionalidad (Stellman & Greene, 2005), (Wiegers, 2003).

En un sentido general, muchos de los dominios de aplicación de los SMR demandan la búsqueda y transporte de objetos en un ambiente dinámico o desconocido lo cual, implícitamente incluye tareas como la exploración de una determinada área, la carga y el traslado de dichos objetos hacia un determinado lugar, la detección y evasión de obstáculos entre otros. Adicionalmente estás tareas deben realizarse de manera conjunta entre varios individuos a través de alguna estrategia de cooperación e interacción. Por lo tanto el diseño del sistema que aquí se trata estará orientado a dar cumplimiento a estos requerimientos.

Adicionalmente, existe otra serie de demandas que conciernen al desempeño del sistema y que para efectos de este documento se han denominado requerimientos no funcionales y se presentan en la siguiente sección.

2.2. REQUERIMIENTOS NO FUNCIONALES

Un requerimiento no funcional representa requisitos de operación y desempeño que debe satisfacer un sistema independientemente de su funcionalidad. Para el caso del sistema que aquí se propone se identificaron los siguientes:

- Escalabilidad: Permitir el incremento en el número de individuos del sistema sin afectar el desempeño
- Tolerancia a fallas: Permitir la recuperación del sistema ante posibles fallas de algún componente o individuo del sistema.
- Concurrencia: El sistema permitirá realizar varias tareas de manera concurrente sin que esto afecte el desempeño
- Interacción: El esquema de interacción entre robots deberá evitar posibles cuellos de botella o exclusiones mutuas.
- Sencillez Individual: Cada individuo implementará un enfoque basado en comportamientos y un esquema de localización que no requiera de un alto procesamiento y almacenamiento.

Para dar cumplimiento a estos requerimientos, la siguiente sección presenta la propuesta estructural del sistema.

2.3. DISEÑO ESTRUCTURAL DEL SISTEMA.

2.3.1. ***Hardware***. Cada uno de los individuos serán robots móviles con las siguientes características de hardware.

Actuadores: Cada individuo deberá tener como mínimo 2 actuadores eléctricos para dar movimiento a las llantas izquierda y derecha del robot. Pueden ser servomotores que permitan controlar tanto velocidad como posición, en (Carletti, 2008) se muestra una completa referencia al respecto. La Figura 13 muestra algunos modelos de servomotores.

Figura 1. Diversos tipos de servomotores (Carletti, 2008)

Adicionalmente se requiere de un actuador eléctrico (servomotor) o hidráulico para un brazo mecánico que permita la manipulación de objetos.

Sensores: Los sensores permiten a cada individuo percibir algunas características del medio en el que actúan, para un SMR con los requerimientos funcionales mencionados, se requiere que cada uno de los robots disponga de los siguientes sensores:

- Sensores de luz: Dos sensores de luz (izquierdo y derecho) que permitan medir la incidencia de la luz, generalmente este tipo de sensores son fotorresistencias o fotoresistores cuyo valor es bajo cuando hay luz incidiendo en él y alto en caso contrario. Este tipo de sensores se conoce como LDR por sus siglas en inglés *Light Depended Resistor* el cual se muestra en la siguiente figura.

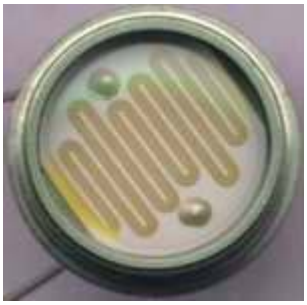


Figura 2. Fotoresistor LDR (Carletti, 2008)

De manera opcional cada individuo puede ser dotado de una cámara que permita la captura de imágenes que puedan ser procesadas con el fin de identificar algunas características de su entorno tales como obstáculos u objetos, o también utilizar sensores de sensibilidad espectral para detectar el color.

- Sensores ultrasónicos: Este tipo de sensores son sistemas de sonar en donde un emisor lanza una señal o tren de pulsos ultrasónicos y espera la reflexión, calculando el tiempo entre la emisión y el retorno de la señal. Para el SMR que se plantea se requiere que cada robot posea un arreglo de sensores ultrasónicos que cubran las partes frontal, lateral (izquierda, derecha) y trasera. Se recomiendan como mínimo 12 sensores ultrasónicos colocados alrededor del robot con espacio de 30 grados entre cada sensor.

Figura 3. Modelo de sensor ultrasónico (Roboworld, 2008)

- Sensores de contacto: Este tipo de sensores se usa para detectar el contacto físico del robot con un obstáculo, generalmente se utilizan interruptores como los que se muestran en la Figura 14, los cuales cierran un circuito al hacer contacto con algún objeto.

Figura 4. Interruptores usados como sensores de contacto (Roboworld, 2008)

De igual manera que los sensores ultrasónicos, se requiere que estos cubran la periferia del robot. Una técnica común es agrupar este tipo sensores en parachoques o *bumpers* como se ilustra a continuación con base lo expuesto por en (Carletti, 2008)

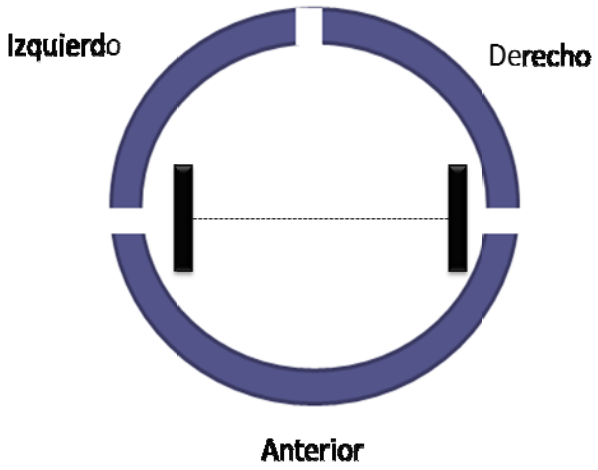


Figura 5. Esquema de parachoques de un robot utilizando sensores de contacto

Por lo tanto, un esquema general de un robot del sistema se puede visualizar en la siguiente figura.

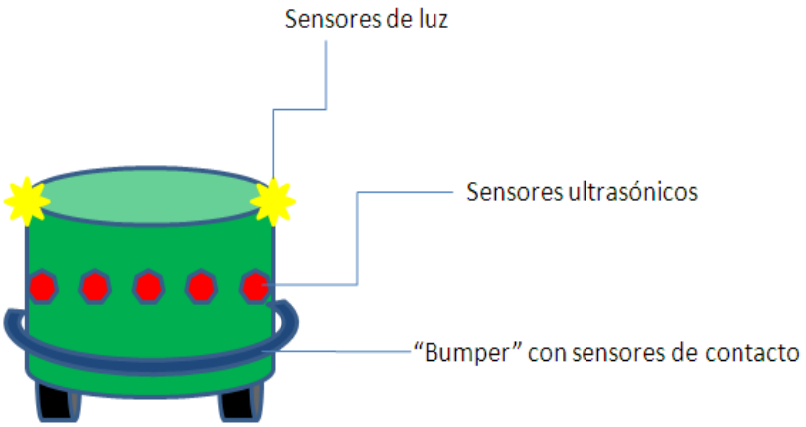


Figura 6. Diagrama general de un robot del sistema

Procesamiento: Cada uno de los robots debe poseer como mínimo una unidad de procesamiento que permita recibir información del entorno a través de los sensores del robot, procesarla y enviar señales a los actuadores del robot dependiendo de algunas condiciones. Para este fin existe una gran gama de *microcontroladores* que disponen de unidad central de

procesamiento, memoria (EPROM, EEPROM, FLASH), puertos de entrada/salida (E/S) y periféricos. Uno de los más celebres fabricantes de estos dispositivos es MICROCHIP® por su familia *PIC* (*Peripheral Interface Controller*) cuyo set de instrucciones es tipo RISC. Si bien existen muchos otros tipos de controladores se recomienda el uso de PIC por su versátil entorno de programación en lenguajes como C y Basic.

Interfaces de Entrada y Salida: Los robots deberán tener varias interfaces que permitan recibir y enviar señales entre sus diversos componentes, como el Microcontrolador, Actuadores y Sensores. Estas pueden ser interfaces estándar como *puerto serial*, *firewire* o *USB.*

Figura 7. Algunas interfaces de E/S estándar (Tienda Digitus, 2008)

2.3.2. ***Ambiente de Operación***. Debido a que el sistema que aquí se plantea ha sido inspirado en el comportamiento de las hormigas, es evidente que el ambiente de operación debería ser un entorno dinámico en el que la incertidumbre con respecto al cambio o evolución de su estado en el tiempo es alta**.**

Sin embargo lograr que un sistema robótico opere en un entorno de este tipo no es una tarea fácil, lo cual implica que el dinamismo de éste tenga sus limitaciones. En lo que respecta a este trabajo, el ambiente de operación debe permitir adicionar o remover objetos o elementos de él así como también incrementar o disminuir el número de individuos (robots) inmersos en éste. La única restricción aquí, sobre el entorno, es que debe tener algún elemento identificable por el robot a través de sus sensores. Esto con el fin de plantear un esquema de localización del que se hablará en la sección 4.3.6 de este capítulo. Para las pruebas realizadas en este trabajo el elemento

identificable que se uso fue una fuente de iluminación que será percibida por cada individuo a través de sus sensores de luz.

La siguiente figura muestra un esquema del ambiente de operación del sistema. El número de obstáculos puede variar sin que esto afecte el desempeño.

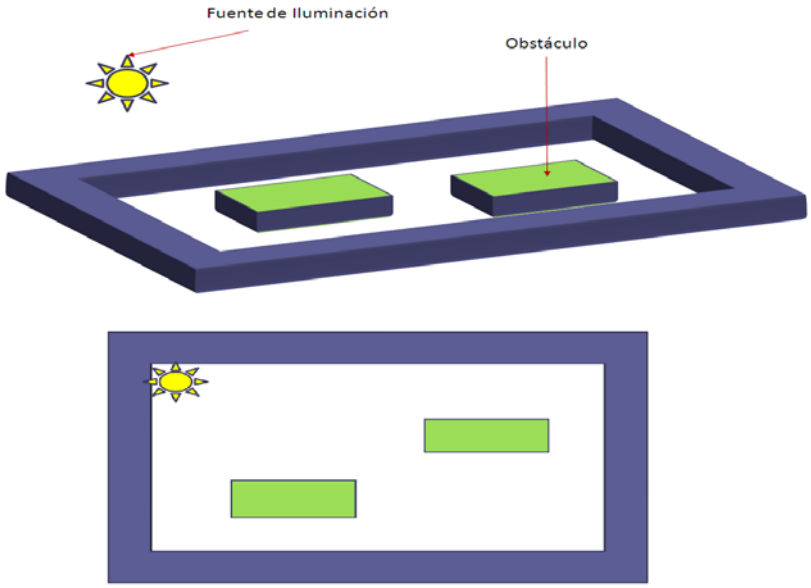


Figura 8. Esquema del ambiente de operación

La siguiente figura muestra lo que podría ser un ambiente físico de operación, en la que se puede observar una lámpara como fuente de iluminación que es utilizada para identificar la base de los robots. Adicionalmente, se pueden ver varios objetos pintados intencionalmente de color negro con el fin de ser detectados a través de los sensores de luz de cada robot. También se distinguen algunos obstáculos en el ambiente que deberán ser evadidos por cada uno de los individuos o robots del sistema.

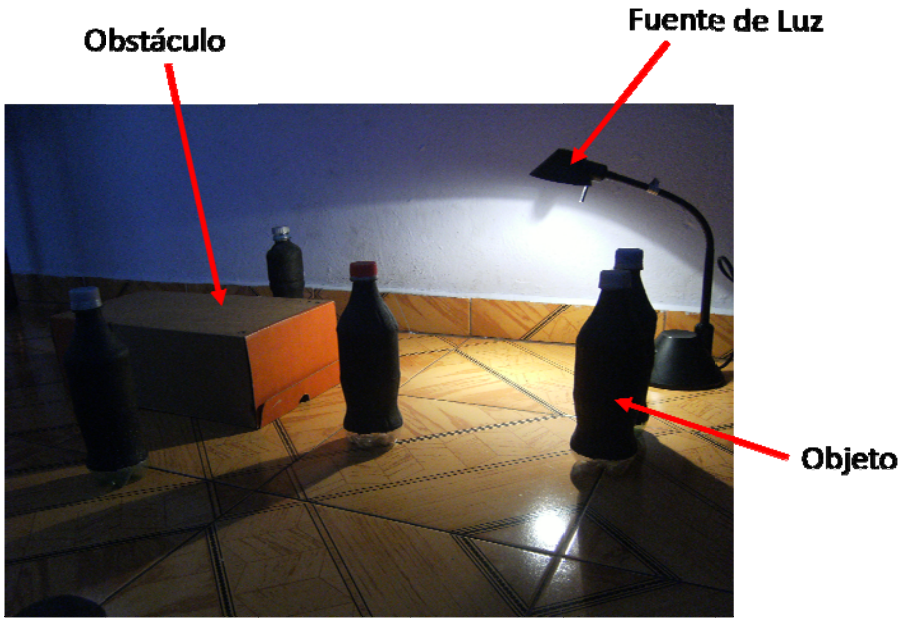


Figura 9. Ambiente físico utilizado para las pruebas del sistema

Para una implementación del sistema planteado en otros contextos, se pueden utilizar algunas marcas identificables por el robot a través de un sistema de visión que incorpore técnicas de reconocimiento y procesamiento de imágenes (Ayache, 1991), (Williams, 2002) aunque esto podría llegar a contradecir la premisa de la simplicidad de cada individuo.

2.4. DISEÑO FUNCIONAL

2.4.1. ***Comportamientos***. Para satisfacer los requerimientos funcionales determinados al inicio de este capítulo, cada uno de los individuos debe presentar ciertas conductas que pueden ser modeladas a través de máquinas de estados. La siguiente figura muestra una máquina de estados para la búsqueda y transporte de objetos. La transición entre cada estado es determinada por ciertas variables cuyo valor es dado en el mayor de los casos por la percepción del robot a través de sus sensores y las cuales se describen en la Tabla 5.

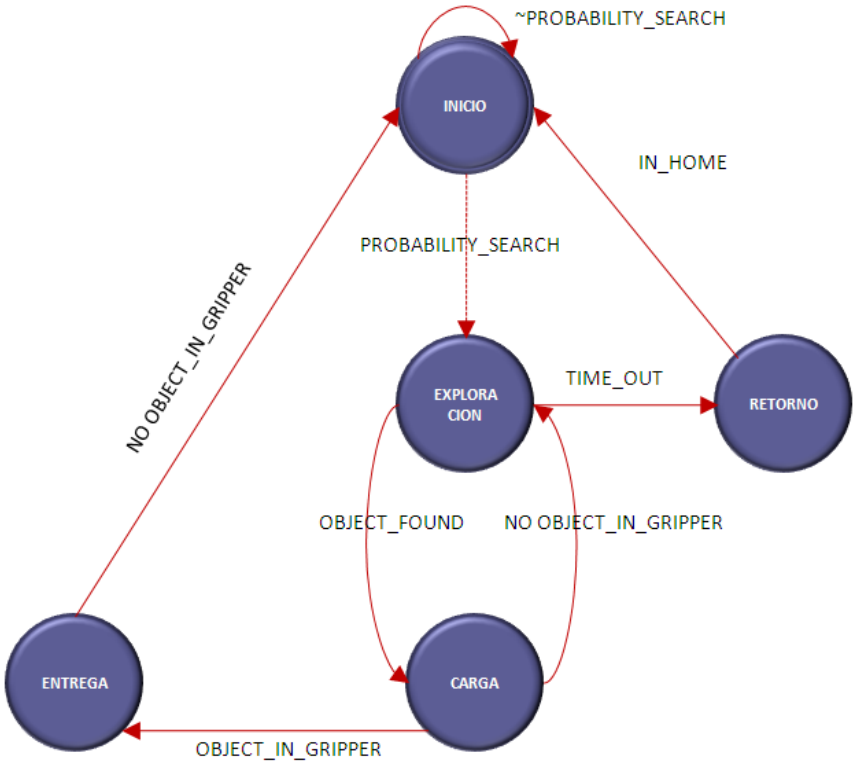


Figura 10. Máquina de estados para la búsqueda y transporte de objetos

La anterior figura muestra que partiendo de un estado de *inicio* se pasa a un estado de *exploración* con cierta probabilidad (de la cual se habla más adelante). Si se encuentra un objeto se pasará a un estado de *carga* donde el robot intentará ponerlo en su brazo o "gripper", si este proceso es exitoso, se hará una transición a un estado de *entrega*, de lo contrario la transición será nuevamente hacia el estado de *exploración*.

Estando en un estado de *entrega* el robot intentará poner el objeto en un lugar determinado y volverá al estado de *inicio*. Por otro lado, si estando en el estado de *exploración* el robot supera el tiempo determinado para tal fin (TIME_OUT) pasará al estado de *retorno* en el que el robot intentará volver a su base o casa y en tal caso hará una transición al estado inicial.

Ahora bien, en la siguiente tabla se muestra una descripción de las variables que determinan la transición de estados del diagrama anterior.

Variable	Descripción
OBJECT_FOUND	Determina si el robot ha encontrado un objeto que debe ser transportado.
OBJECT_IN_GRIPPER	Determina si el robot tiene o no un objeto cargado en su brazo o "gripper".
PROBABILITY_SEARCH	Es la probabilidad que determina el cambio de un estado inicial (reposo) a un estado de exploración en un robot.
TIME_OUT	Permite evaluar si un robot ha alcanzado un tiempo límite en un determinado estado.
IN_HOME	Determina si el robot esta en un área determinada tal como su base o casa.

Tabla 1. Principales variables de estado para un robot del sistema

Cada uno de los estados de la máquina estará conformado por una serie de *comportamientos* que serán activados como respuesta a algunas condiciones dadas por el entorno del robot.

En la siguiente tabla se presentan los comportamientos necesarios para satisfacer los requerimientos del sistema.

Comportamiento	Descripción
Avoidance	Permite evadir un obstáculo
Grip Object	Permite tomar un objeto y cargarlo en el brazo o "gripper" del robot.
Homing	Permite al robot emprender el regreso a su base o casa cuando éste tiene un objeto en su "gripper".
Come Back	Permite al robot regresar a su base después de pasado un determinado tiempo de búsqueda (timeout) sin tener éxito de

	encontrar objetos
Deliver	Permite entregar un objeto cuando el robot ha llegado a su base o *Home.*
Rest	Permite dejar al robot en un estado inicial o de reposo.
Wandering	Permite al robot realizar un recorrido aleatorio en busca de un objeto

Tabla 2. Comportamientos en la máquina de estados para el transporte de objetos

Adicionalmente a las variables de estado del autómata, algunos de los comportamientos citados en la tabla anterior utilizan las siguientes variables:

Variable	Descripción
GOAL_REACHED	Determina si el robot a alcanzado su objetivo
TIME_SEARCH	Tiempo tarda el robot en el estado de exploración hasta que encuentra un objeto.
LUMINANCE	Valor de la luminancia del ambiente, percibida por los sensores del robot.

Tabla 3. Otras variables utilizadas en comportamientos

2.4.2. ***Estructura de un comportamiento.*** De manera formal, un comportamiento es el conjunto de acciones de un individuo desencadenadas a partir de un conjunto de condiciones o restricciones dadas por el medio. De acuerdo con esto, aquí se plantea una implementación en la que un comportamiento está conformado por dos módulos fundamentales: la activación y las acciones, tal como se ilustra en la siguiente figura.

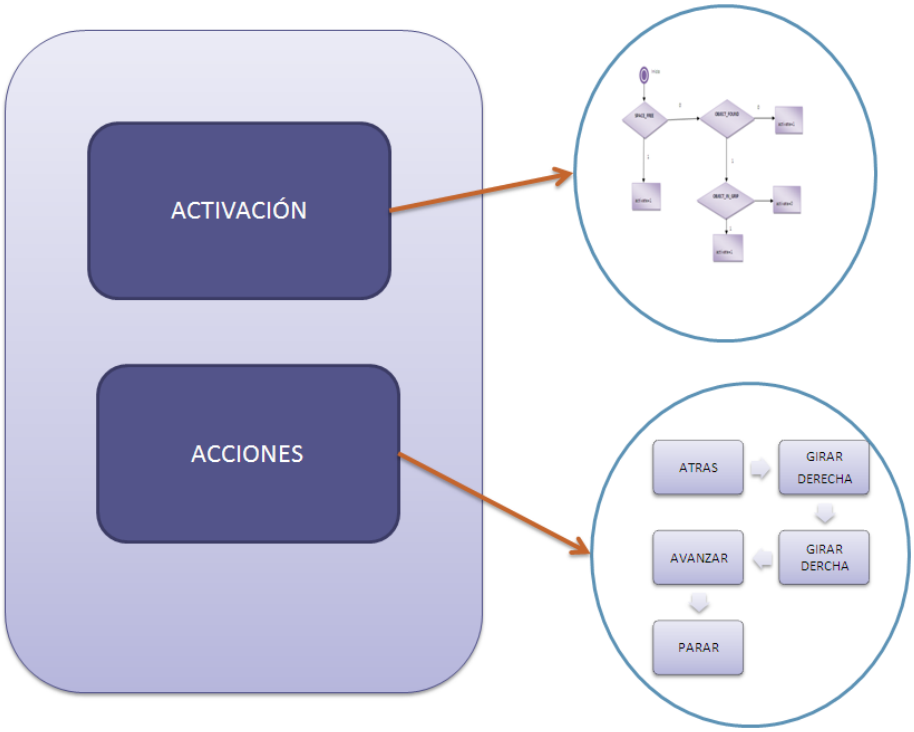


Figura 11. Estructura básica de un comportamiento

El módulo de activación es un algoritmo que evalúa ciertas condiciones para determinar si el comportamiento debe activarse o no. Dichas condiciones están dadas por variables de estado, algunas de las cuales se mostraron en la Tabla 2 y cuyo valor depende de la percepción del entorno por parte del robot a través de los sensores.

En lo que respecta al módulo de acciones, este es un conjunto de instrucciones que se ejecutarán si y solo si el modulo de activación determina que el comportamiento debe activarse. Tomando como ejemplo el comportamiento de evasión de obstáculos denominado *Avoidance*, su módulo de activación puede ser descrito a través del siguiente diagrama de flujo:

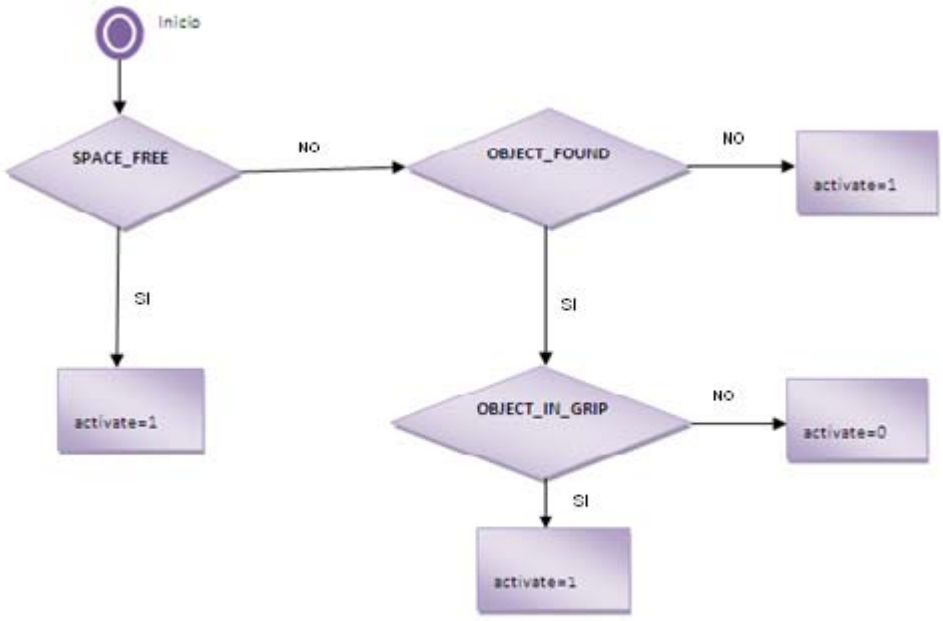


Figura 12. Diagrama de flujo para el comportamiento Avoidance (Evasión Obstáculos)

Seguidamente se muestra la implementación del diagrama de la figura anterior.

```
si(SPACE_FREE==false)
    {
     si(OBJECT_FOUND==true)
     {
            si(OBJECT_IN_GRIPPER)
            {
                  activate= 1;
            }
            sino
            {
                  activate=0;
            }

     }
     sino
     {
         activate=1;
     }
    }
sino
    {
          activate=0;
    }
```

Algoritmo 1. Activación para el comportamiento Avoidance

Si luego de ejecutarse el módulo de activación se llega a que el comportamiento debe activarse (*activate=1*) entonces se ejecutaran todas las tareas que defina el módulo de acciones del comportamiento, para el caso de *Avoidance* estas podrían ser dar marcha hacia atrás *x* metros y girar θ grados en alguna dirección.

A continuación se muestran los diagramas de flujo y las correspondientes implementaciones del módulo de activación, para cada uno de los comportamientos de un robot del sistema. En lo que respecta al módulo de acciones, en el Anexo 3 se presentan sus respectivas implementaciones.

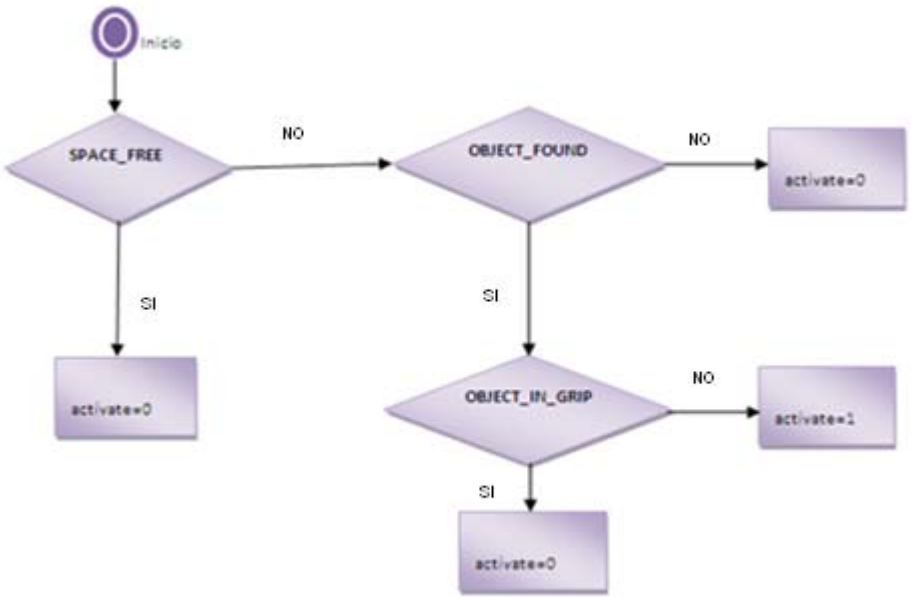


Figura 13. Diagrama de flujo para el comportamiento Grip_Object (Atrapar objeto)

```
si(SPACE_FREE==0)
    {
      si(OBJECT_FOUND)
      {
                si(OBJECT_IN_GRIPPER)
                {
                        active= 0;
                }
                sino
                {
                        active= 1;
                }

      }
      sino
      {
                active= 0;
      }
    }
    sino
    {
              active= 0;
    }
```

Algoritmo 2. Activación para el comportamiento Grip_Object

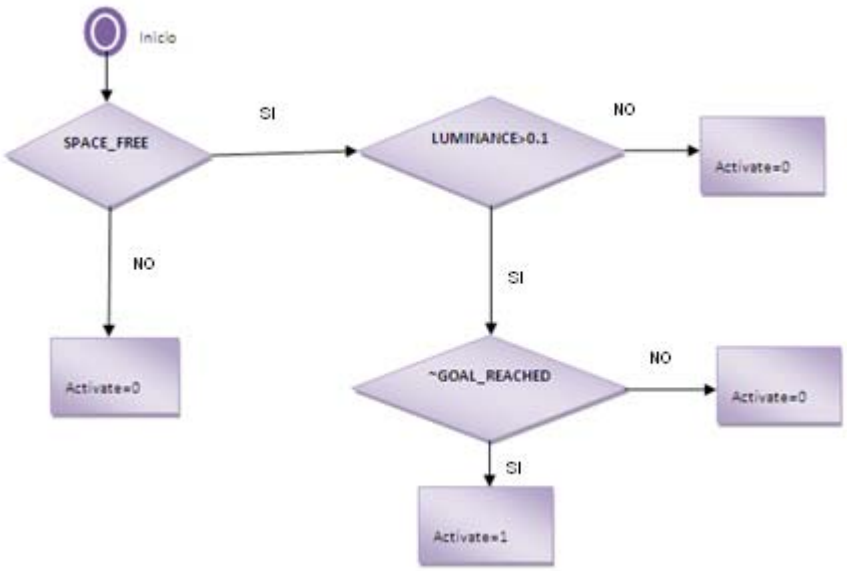


Figura 14. Diagrama de flujo para el comportamiento Homing (Regreso a base con objeto)

```
si(SPACE_FREE)
   {
    si(OBJECT_IN_GRIPPER)
    {
              si(LUMINANCE>0.1 && GOAL_REACHED==false)
                activate= 1;
              sino
                activate= 0;
    }
    sino
    {
            activate= 0;
    }
   }
   sino
   {
            activate= 0;
   }
```

Algoritmo 3. Activación para el comportamiento Homing

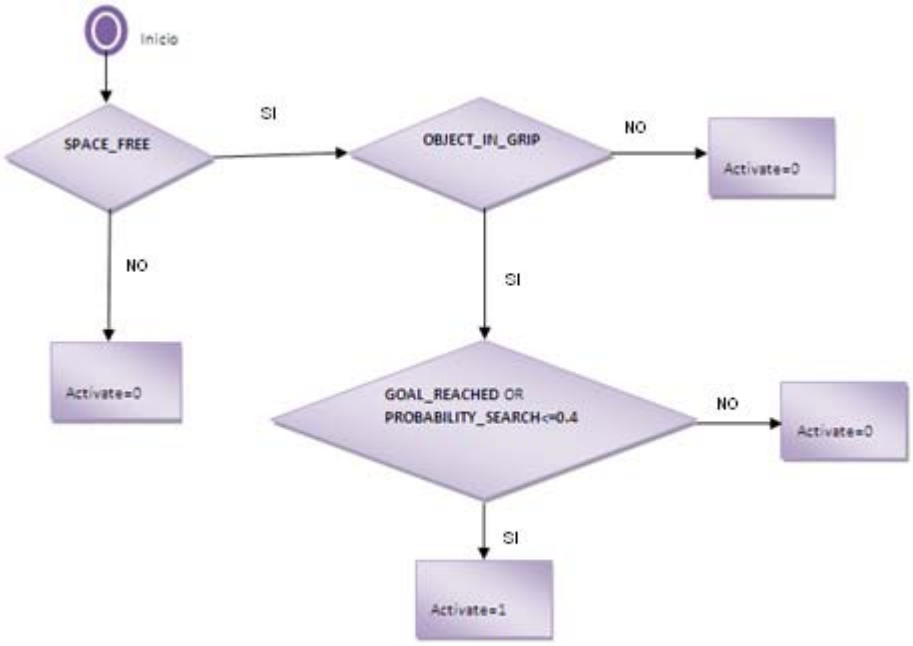


Figura 15. Diagrama de flujo para el comportamiento Rest (Reposo o estado inicial)

```
si(SPACE_FREE)
  {
    si(OBJECT_IN_GRIPPER==false)
    {
            si(GOAL_REACHED OR PROBABILITY_SEARCH<=0.4)
              activate= 1;
            sino
              activate= 0;
    }
    sino
    {
            activate= 0;
    }
  }
 sino
  {
            activate= 0;
  }
```

Algoritmo 4. Activación para el comportamiento Rest

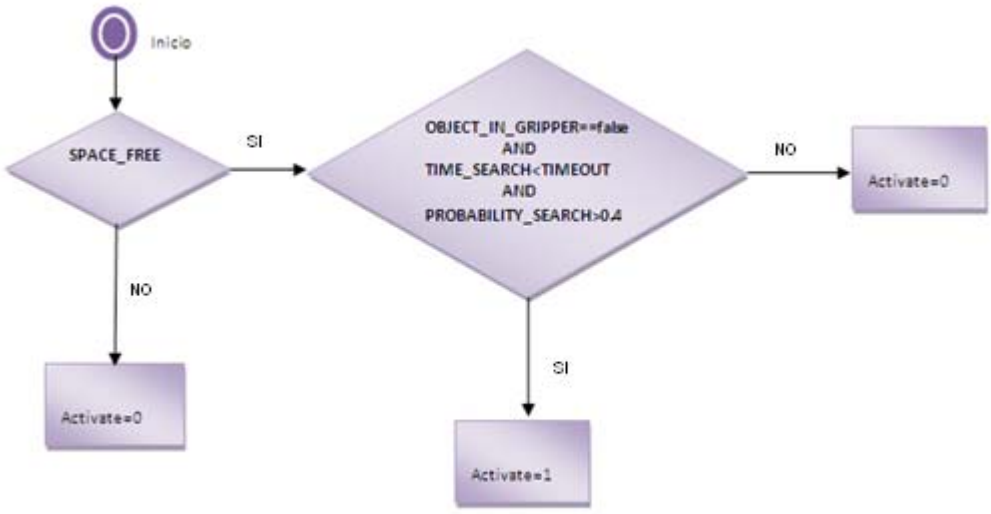


Figura 16. Diagrama de flujo para el comportamiento Wandering (Búsqueda aleatoria de objetos)

```
si(SPACE_FREE)
  {
      si( OBJECT_IN_GRIPPER==false
          AND TIME_SEARCH<TIMEOUT
          AND PROBABILITY_SEARCH>0.4)
       {
              active= 1;

       }
       sino
              active= 0;
  }
  sino
  {
              active= 0;

  }
```

Algoritmo 5. Activación para el comportamiento Wandering

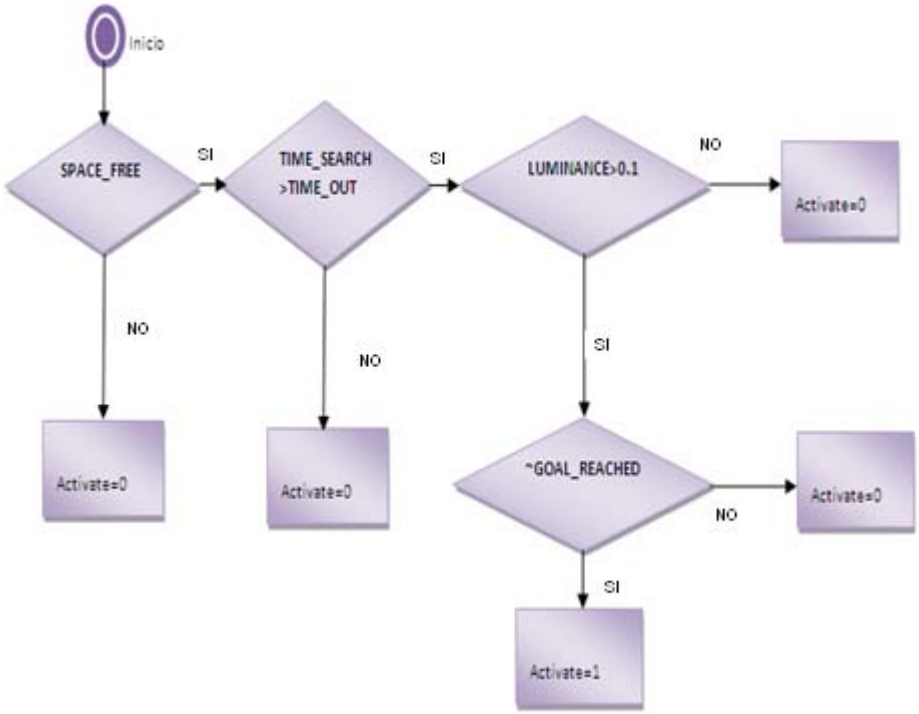


Figura 17. Diagrama de flujo para el comportamiento ComeBack (Regreso por fracaso)

```
si(SPACE_FREE)
 {
  si(TIME_SEARCH>TIMEOUT)
  {
          si(LUMINANCE>0.1 AND GOAL_REACHED==false)
            activate= 1;
          sino
                activate= 0;
  }
  sino
  {
          activate= 0;
  }
 }
 sino
 {
          activate= 0;
 }
```

Algoritmo 6. Activación para el comportamiento ComeBack

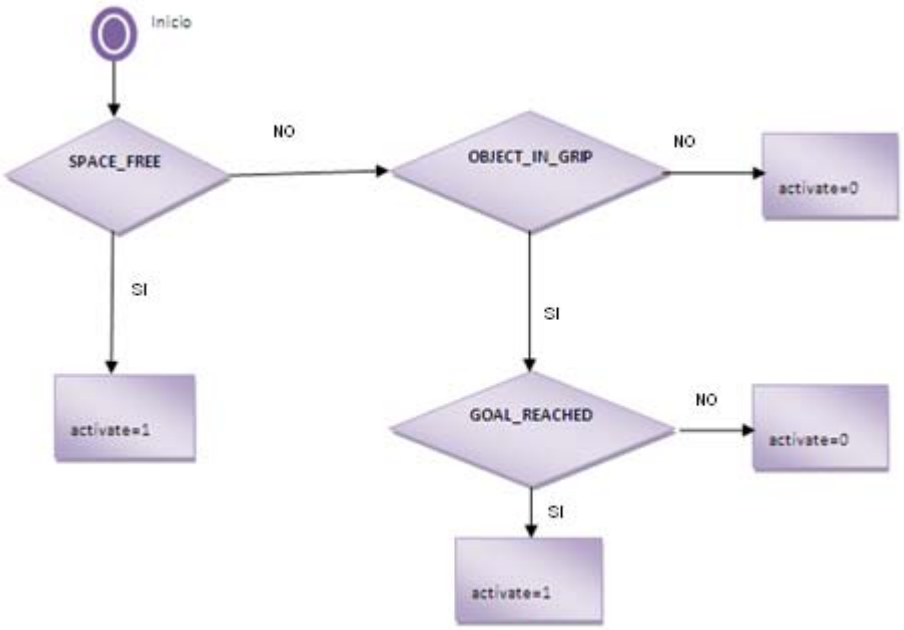


Figura 18. Diagrama de flujo para el comportamiento Deliver (Entrega de objeto)

```
si(robot.SPACE_FREE)
 {
  si(robot.OBJECT_IN_GRIPPER)
  {
            si(robot.GOAL_REACHED)
              activate= 1;
            sino
                    activate= 0;
  }
  sino
  {
          activate=0;
  }
 }
 sino
 {
          activate=0;
 }
```

Algoritmo 7. Activación para el comportamiento Deliver

Los anteriores comportamientos residen en la memoria de cada robot. Para determinar qué comportamiento debe ejecutarse, en la siguiente sección se plantea un mecanismo al que se le denominó *Selector de Comportamientos*.

2.4.3. ***Selector de Comportamientos.*** El Selector de Comportamientos es un algoritmo que almacena en una estructura de datos secuencial (lista o arreglo) varios comportamientos. En un bucle o ciclo ejecuta el módulo de activación de cada comportamiento y si encuentra que alguno está activo ejecuta el módulo de acciones de este. El algoritmo estará permanentemente en ejecución mientras el robot esté encendido.

La siguiente figura ilustra lo dicho anteriormente:

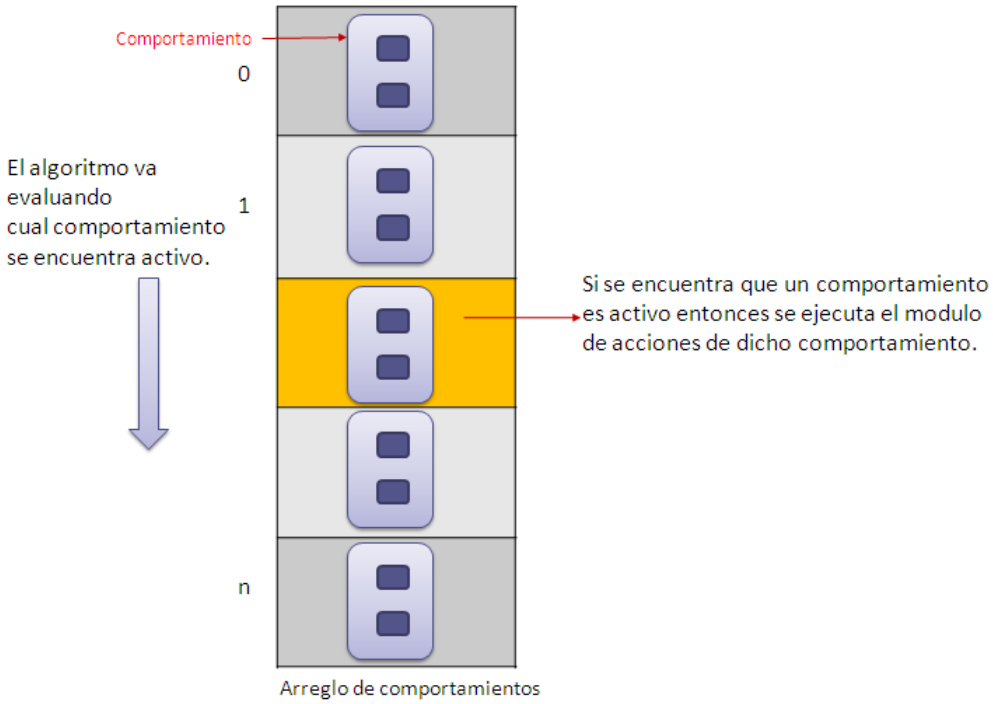


Figura 19. Selector de comportamientos

Para evitar un posible conflicto de activación, los comportamientos se almacenan en la estructura de datos de acuerdo con una prioridad de manera descendente, que garantiza que si existen dos o más comportamientos activados solo se ejecutará el de mayor prioridad. Un posible algoritmo se muestra a continuación:

```
while(true)
{
 int indice=0;
 while(indice<n)
 {
   Comportamiento cmp=array[indice];
    if(cmp.activate)
     {
        cmp.acciones();
        break;

     }
  indice++;
 }

}
```

Algoritmo 8. Selector de comportamientos

2.4.4. ***Interacción.*** Un esquema de interacción permite a varios individuos obtener información necesaria para cumplir un determinado objetivo, en los sistemas biológicos eusociales, la forma más común de interacción es a través de sustancias químicas volátiles denominadas feromonas que permiten dar avisos de alarma ante posibles invasores de la colonia, definir el territorio o proporcionar información acerca de posibles fuentes de alimento. En este tipo de interacción denominado *estigmergia* cada individuo sigue el gradiente de feromona a través de sofisticados y complejos mecanismos sensoriales cuya implementación en sistemas robóticos representa algunos desafíos técnicos.

En este trabajo no se busca emular el gradiente de feromona, en su lugar se plantea un esquema de interacción indirecta donde la información obtenida por cada individuo es el resultado de sus acciones y las de otros individuos sobre el entorno. En otras palabras dicha información guarda la historia de las acciones individuales de todos los miembros del sistema.

En el contexto de este trabajo y con base en la máquina de estados de la Figura 22, cada robot deberá decidir si inicia o no un proceso de exploración para la búsqueda y captura de objetos colocados aleatoriamente en el ambiente. Esta decisión depende de una probabilidad denominada de aquí en adelante probabilidad de búsqueda y es representada en la máquina de estados con el nombre de PROBABILITY_SEARCH. Dicha probabilidad determina el grado de certeza de que un robot encuentre un objeto y su valor será calculado de manera individual de acuerdo con la percepción del entorno a través de los sensores del robot y sus experiencias.

A medida que los robots exploran el entorno el número de objetos por descubrir y capturar cambiará en el tiempo dependiendo de esta probabilidad, como lo muestra la siguiente expresión matemática:

$$\frac{dO}{dt} = -K_o\left(\prod_{i=1}^{R} p_i\right) \qquad (1)$$

Donde:

p_i=Probabilidad de búsqueda del i-ésimo robot

K_0=Número inicial de objetos

R=Número de robots

Si bien esta probabilidad es calculada de manera individual por cada robot, su valor estará implícitamente relacionado con las acciones de otros robots ya que si un robot encuentra y captura un objeto, está acción disminuirá la riqueza en objetos del entorno lo que se reflejará en el valor de las probabilidades de los demás.

En las siguientes secciones se plantea una forma de calcular la probabilidad de búsqueda en cada robot.

2.4.5. ***Cálculo de la probabilidad de búsqueda.*** En un principio cada robot tiene una probabilidad de búsqueda igual a uno ya que inicialmente ningún robot tiene un conocimiento a priori del entorno. Dependiendo del éxito del robot en la exploración del entorno, el valor de dicha probabilidad cambiará como se describe a continuación.

Definición: Un robot tiene éxito si durante un proceso de búsqueda encuentra un objeto y lo lleva satisfactoriamente a su base o *home*. De lo contrario el robot tendrá un fracaso.

Por lo tanto:

- Si el robot tiene éxito en un tiempo t_s, entonces su probabilidad de búsqueda aumentará en un valor Δ_e.
- Por el contrario, si el robot fracasa en un tiempo t_s entonces su probabilidad de búsqueda disminuirá en un valor Δ_f.

Tras varios procesos de exploración es posible que la probabilidad de búsqueda tome valores fuera del intervalo [0,1], lo cual implicaría un error. Para evitarlo, se propone lo siguiente:

Sean *pr_max* y *pr_min* valores pertenecientes al intervalo [0,1] correspondientes al valor máximo y mínimo de la probabilidad de búsqueda. Por lo tanto:

- Si al aumentar la probabilidad de búsqueda en un valor Δ_e ésta es mayor que 1 entonces su valor es igual a *pr_max*.
- Si al aumentar la probabilidad de búsqueda en un valor Δ_f ésta es menor que 0 entonces su valor es igual a *pr_min*.
- En otro caso la probabilidad de búsqueda es igual a su valor actual aumentado o disminuido en un valor Δ_e o Δ_f respectivamente.

Este proceso se aplicará siempre que el robot esté en el estado de exploración. Al final de un determinado tiempo se espera que la probabilidad de búsqueda en cada robot tienda a cero, lo que significaría que la misión de todos los robots ha terminado ya que no existen objetos para buscar y capturar.

En la siguiente sección se plantea un procedimiento para realizar el cálculo del valor de Δ_e y Δ_f.

1.4.6. ***Determinación del valor de Δ_e y Δ_f.*** Sea *Ri* el i-ésimo robot, t_s el tiempo empleado por un robot para la exploración y búsqueda de objetos y T el tiempo límite preestablecido para esta misma labor, entonces:

$$\Delta_e = \begin{cases} 1 & para\ 0 < t_s \leq T \\ \frac{1}{t_s}\ T & para\ T < t_s < \infty \end{cases} \qquad (2)$$

Lo cual significa que si R_i tiene éxito en un tiempo t_s menor al tiempo preestablecido su probabilidad de búsqueda aumentará a uno. Si R_i tiene éxito en un tiempo de búsqueda superior a T entonces su probabilidad aumentará en proporción inversa a dicho tiempo, tal como lo ilustra la siguiente figura para un T=30s

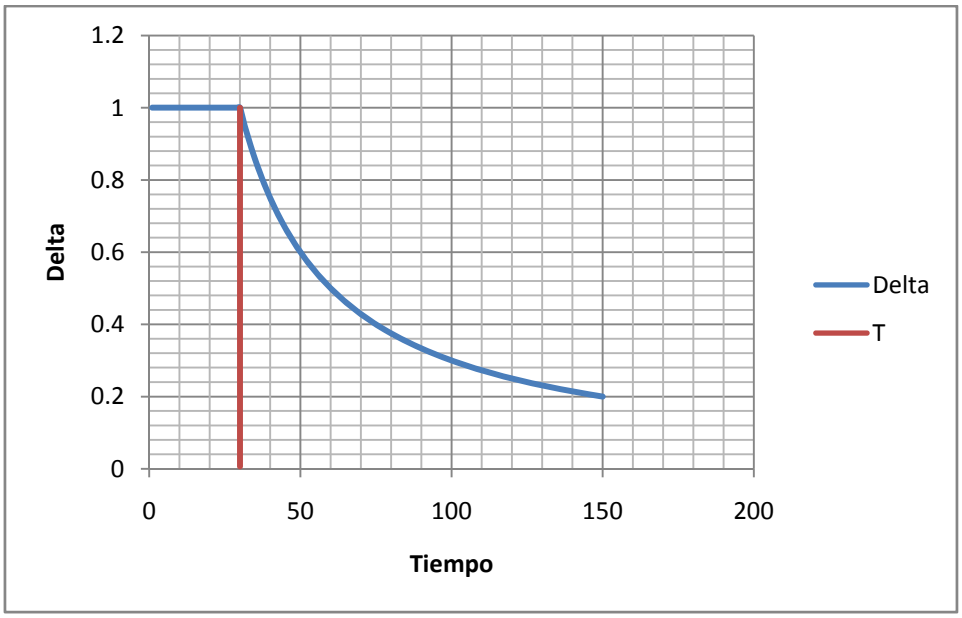


Figura 20 Función delta cuando el robot tiene éxito para un T=30s

Por su parte cuando R_i ha fracasado se cumple que $t_s > T$ lo cual significa que el robot no ha encontrado un objeto durante el tiempo preestablecido y deberá disminuir su probabilidad de búsqueda en Δ_f, cuyo valor será proporcional al tiempo durante el cual ha estado buscando. Para reflejar este hecho se plantea la siguiente expresión:

$$\Delta_f = 1 - e^{-t_s+T} \; para \; t_s, T > 0 \; y \; t_s > T \tag{3}$$

Para un valor de *T=30*, en la siguiente gráfica se observa que el valor de Δ_f tiene un rápido crecimiento que en pocos segundos después de *T* converge a 1.

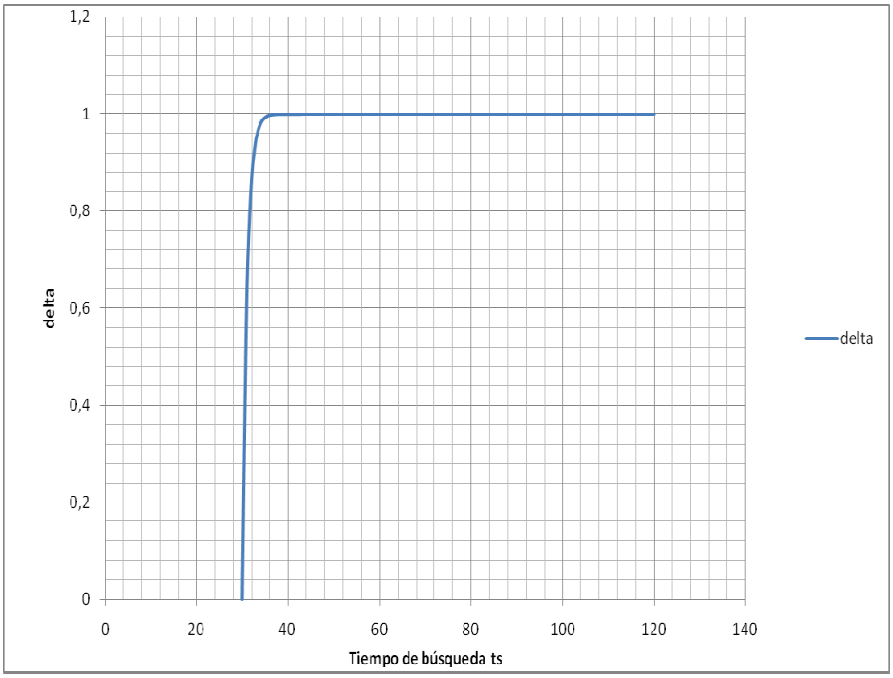


Figura 21 Función delta cuando el robot fracasa, T=30 s

Este hecho no es conveniente ya que a los pocos segundos después de superado el *timeout*, el robot verá reducida su probabilidad de búsqueda en un valor cercano a 1 y no podrá seguir explorando, ya que pasará a un estado de reposo. Para evitar este fenómeno se introduce un factor en el exponente de la expresión anterior que disminuye la velocidad de crecimiento de la función y que depende del valor de T, por lo que finalmente la expresión es:

$$\Delta_f = 1 - e^{\frac{1}{4T}(-t_s+T)} \; para \; t_s, T > 0 \; y \; t_s > T \tag{4}$$

Y cuya grafica se presenta en la siguiente figura:

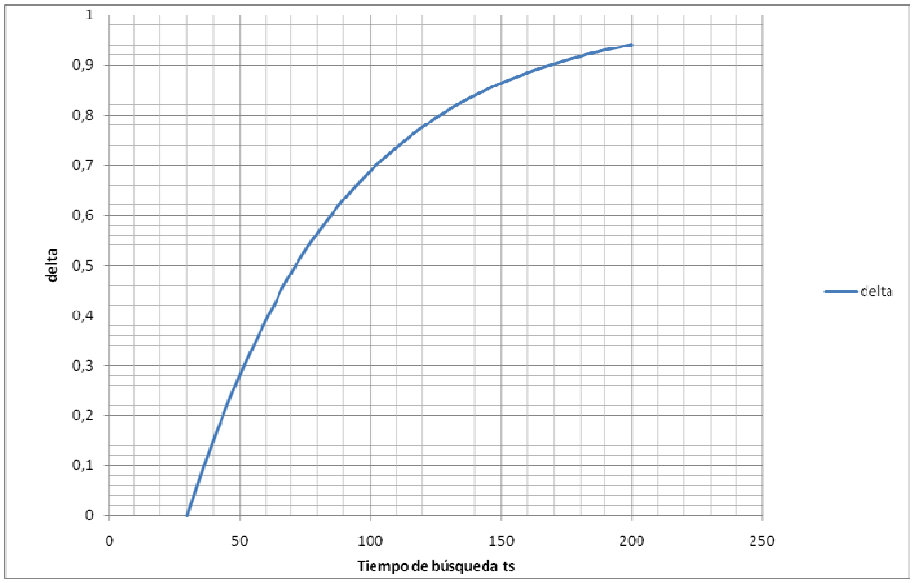


Figura 22. Ajuste de la función delta cuando el robot fracasa

A continuación se muestra este procedimiento implementado en un algoritmo en donde t_s, T, Δ_e, Δ_f han sido renombradas como TIME_SEARCH, TIME_OUT, DELTA_E, DELTA_F respectivamente. El indicador de que un robot tuvo éxito durante su exploración, será la variable nombrada como OBJECT_IN_GRIPPER sobre la cual se hizo referencia en la Tabla 2.

```
si(OBJECT_IN_GRIPPER)
 {PROBABILITY_SEARCH=PROBABILITY_SEARCH+DELTA_E;
  si(PROBABILITY_SEARCH>PROBABILITY_SEARCH_MAX)
    PROBABILITY_SEARCH=PROBABILITY_SEARCH_MAX;
 }
sino
 {
      si(TIME_SEARCH>TIMEOUT)
   {
     PROBABILITY_SEARCH=PROBABILITY_SEARCH-DELTA_F;
     si(PROBABILITY_SEARCH<PROBABILITY_SEARCH_MIN)
        PROBABILITY_SEARCH=PROBABILITY_SEARCH_MIN;
   }
 }
```

Algoritmo 9. Implementación del cálculo de la probabilidad de búsqueda

Adicionalmente en el algoritmo anterior se observan dos variables que constituyen el valor máximo y mínimo de la probabilidad de búsqueda cuyos

valores utilizados en las pruebas realizadas fueron 0.99 para la probabilidad máxima y 0.1 para la mínima. Como se indicó en páginas anteriores estos valores son usados para acotar la probabilidad de búsqueda en el intervalo [0,1] después de aumentar o disminuir su valor.

Mediante el algoritmo que se describió anteriormente se logra cuantificar en cierta medida la percepción individual del entorno a través de la probabilidad de búsqueda de cada robot. El inconveniente es que dicha percepción es altamente susceptible de sesgos como se puede apreciar en la siguiente figura donde dos robots tienen percepciones diferentes del mismo entorno al cabo de cierto tiempo con un *timeout* de 30 segundos.

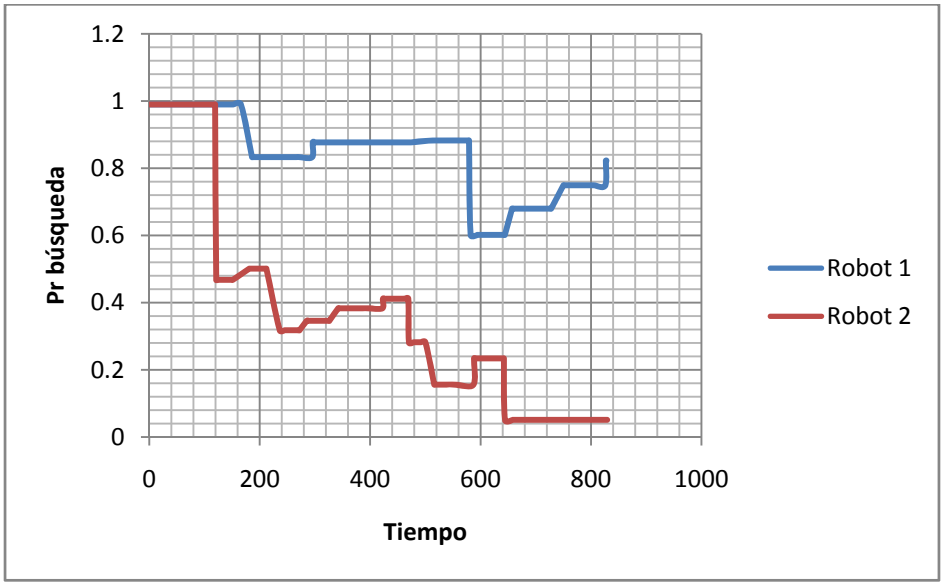


Figura 23. Percepciones sesgadas del entorno

Lo cual no contribuye ni incentiva al cumplimiento del objetivo global del sistema. Para corroborar que efectivamente cada robot tiene percepciones diferentes del entorno se realizaron varios experimentos cuyas características se presentan en la siguiente tabla:

Experimentos			
No. Experimento	**No. Robots**	**Timeout**	**No. Objetos**
1	4	30	20
2	6	30	20
3	8	30	20
4	10	30	20

Descripción:

En cada experimento los robots se encuentran inicialmente en su base o *home*. Cada robot implementa la máquina de estados de la Figura 22 con una probabilidad de búsqueda de 0.99. Esto permite al robot iniciar un proceso de exploración o búsqueda aleatoria a través de todo su entorno. La probabilidad de búsqueda es recalculada por cada robot de acuerdo con sus experiencias durante la exploración.
Los objetos son colocados de manera aleatoria en el ambiente.

Tabla 4. Características de los experimentos realizados

Los datos obtenidos a partir de estos experimentos mostraron que efectivamente cada robot percibe de manera diferente su entorno ya que existe una divergencia entre sus probabilidades de búsqueda. Este hecho se ilustra en las siguientes gráficas

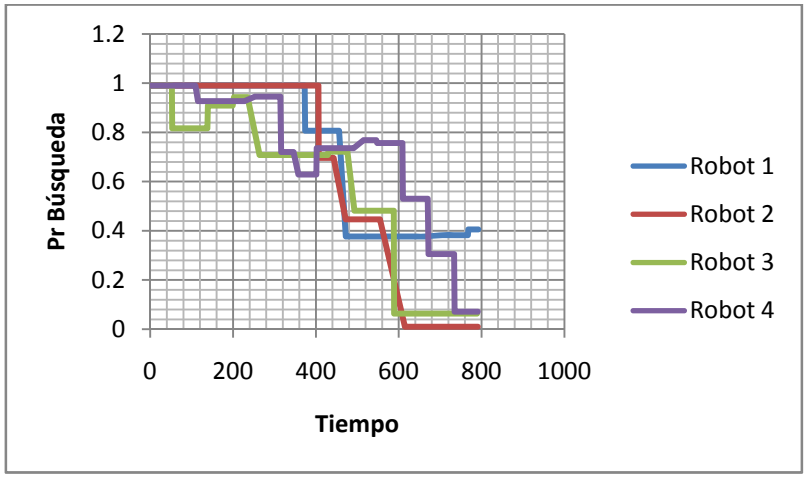


Figura 24. Probabilidad de búsqueda para 4 robots con T=30s

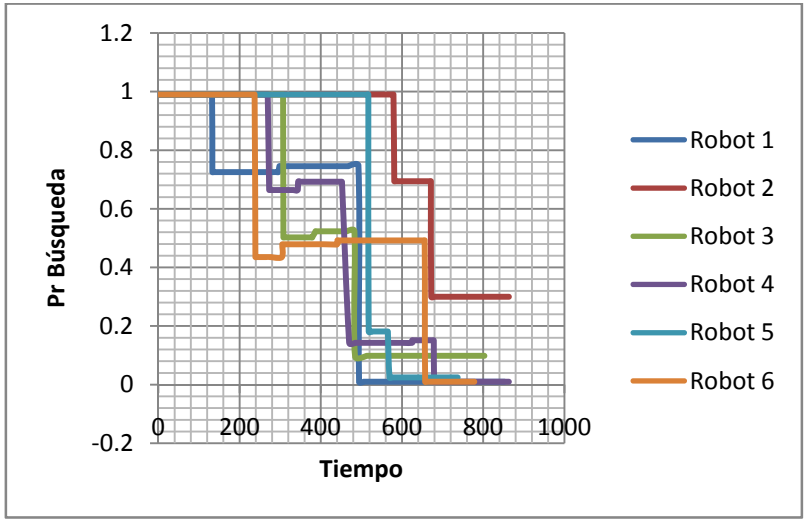


Figura 25. Probabilidad de búsqueda para 6 robots con T=30s

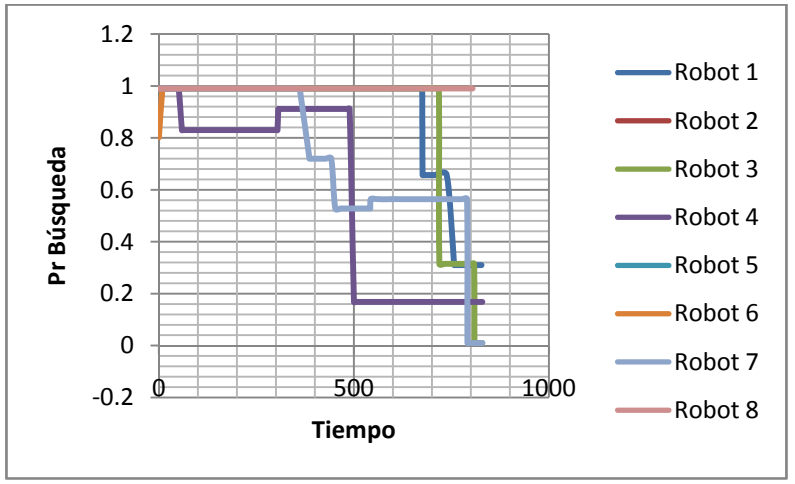


Figura 26. Probabilidad de búsqueda para 8 robots con T=30s

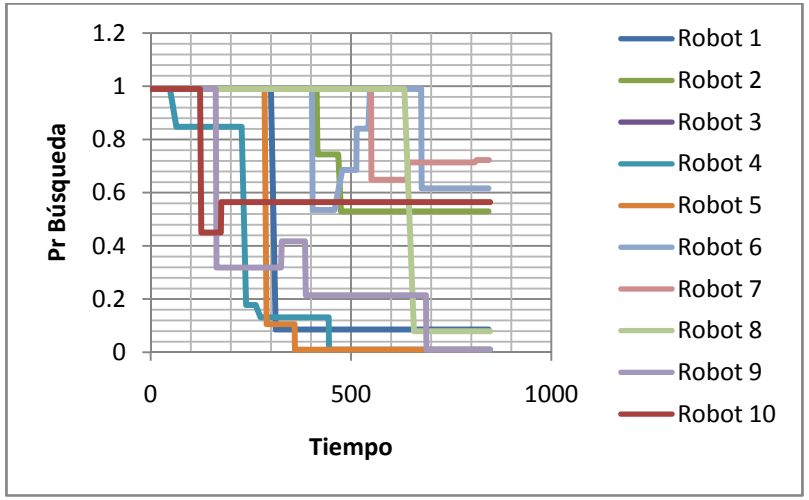


Figura 27. Probabilidad de búsqueda para 10 robots con T=30s

Adicionalmente, se realizaron otros experimentos cuyas características se presentan en la siguiente tabla.

Experimentos			
No. Experimento	**No. Robots**	**Timeout**	**No. Objetos**
1	4	15	20
2	4	30	20
3	4	60	20

Descripción:
Al igual que en los experimentos anteriores, la misión de cada robot fue buscar y recolectar una serie de objetos colocados de manera aleatoria en el ambiente. La diferencia radica en que en este caso el *timeout* fue cambiado en cada experimento y el número de robots fue en todos los casos el mismo.

Tabla 5. Experimentos adicionales

Los datos obtenidos a partir de estos experimentos se presentan en las siguientes gráficas.

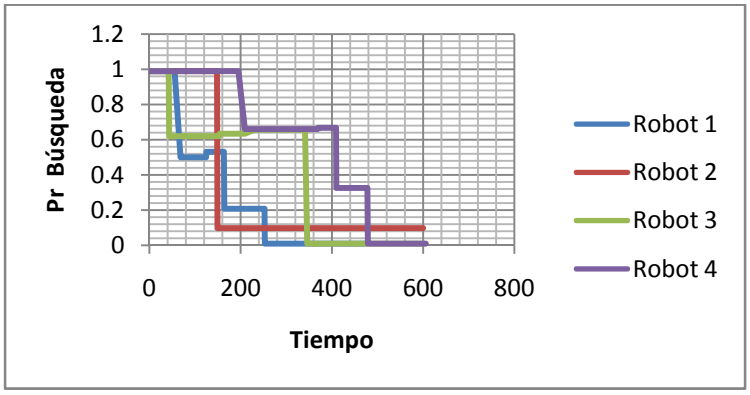


Figura 28. Probabilidad de búsqueda para T=15

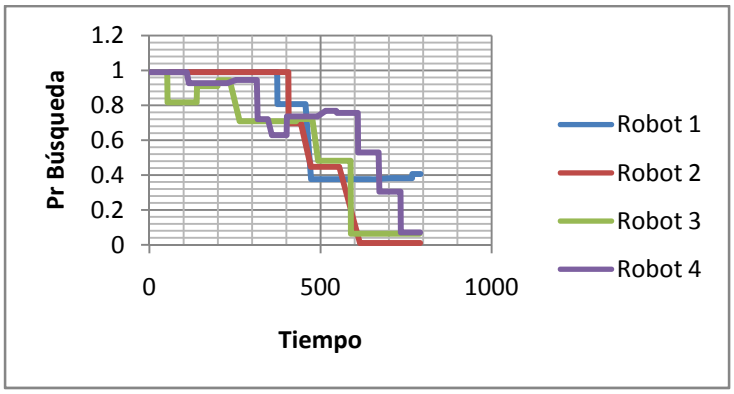


Figura 29. Probabilidad de búsqueda para T=30

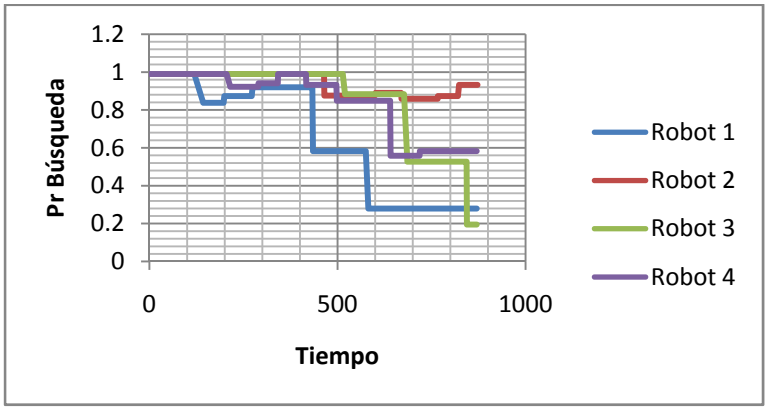


Figura 30. Probabilidad de búsqueda para T=60

Donde se observa igualmente una divergencia en la probabilidad de búsqueda de cada robot que no contribuye al objetivo global, ya que cada uno actúa de manera independiente de acuerdo con sus propias experiencias y percepciones permitiendo afirmar que no hay trabajo en equipo.

Con el fin de unificar las percepciones de cada robot respecto al entorno en la siguiente sección se plantea un procedimiento para unificar el cálculo de la probabilidad de búsqueda. El objetivo de esta propuesta es lograr la cooperación entre robots.

2.4.6. ***Unificación de la probabilidad de búsqueda***. De los gráficos mostrados con anterioridad se puede observar directamente que la probabilidad de búsqueda en cada robot depende de los siguientes factores:

- Tiempo del robot: Representa el tiempo desde que el robot empieza a realizar su misión en el sistema. A medida que este tiempo avanza la

probabilidad de búsqueda presenta cambios cuya tendencia es generalmente a la baja.

- El tiempo límite para realizar una búsqueda (*timeout*): Entre mayor sea el tiempo límite se observa que la velocidad de descenso de la probabilidad de búsqueda en cada robot es menor.
- El número de robots en el sistema: La probabilidad de búsqueda de cada robot disminuye más rápidamente cuando existen más robots en el sistema.

Adicionalmente se considera que la probabilidad de búsqueda también depende de otros factores menos evidentes como:

- Tiempo de búsqueda de un objeto: La probabilidad de búsqueda aumenta o disminuye en proporción al tiempo empleado por un robot para encontrar un objeto.
- El resultado de la exploración (acierto o fracaso): Si un robot logra tener éxito durante su proceso de exploración, su probabilidad de búsqueda se incrementa. En caso contrario su valor disminuye. Entiéndase como éxito el hecho de encontrar un objeto y llevarlo a la base o *home* de los robots.

Por lo tanto se requiere encontrar una expresión para la probabilidad de búsqueda de cada robot de la forma:

$$pr_{busqueda} = f(t, T, n, r, t_s,) \quad (5)$$

Donde:
t=tiempo de vida del robot
T=Tiempo límite de búsqueda (timeout)
n=Número de robots
r=resultado de la exploración (acierto o fracaso)
t_s=Tiempo de búsqueda

Que permita a todos los robots calcular su probabilidad de búsqueda de una forma uniforme de tal manera que ésta represente una percepción global del entorno. Para este fin, y de acuerdo con la anterior expresión, se entrenó una red neuronal con una muestra de datos obtenida experimentalmente y estructurada como se ilustra en la siguiente figura:

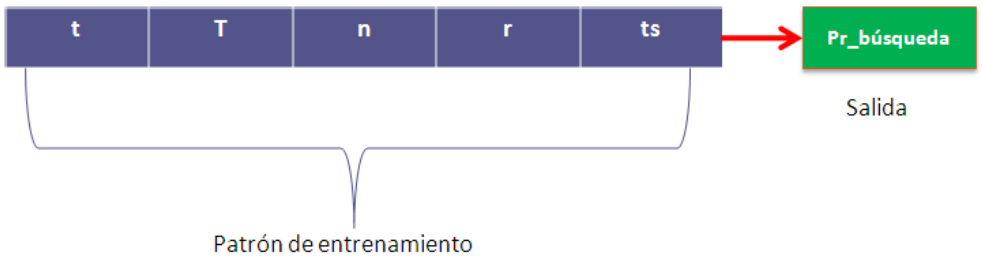


Figura 31. Estructura de los datos utilizados para el entrenamiento de la red

Para obtener los datos de entrenamiento, se realizaron varios experimentos en los que cada robot generaba información acerca de su estado de exploración y el valor de su probabilidad de búsqueda calculada a partir del procedimiento descrito en la sección 4.4.5.

Un subconjunto de los datos de entrenamiento utilizados, se presenta en la siguiente tabla donde cada fila representa un instante del estado de exploración de un robot; por ejemplo en la primera fila se tiene un robot cuyo tiempo de vida (inicio de operación) transcurrido es de 57 s, con un *timeout* de 60 s, 10 robots en el sistema y una exploración no exitosa que hasta el momento lleva 37 s de iniciada por lo que su probabilidad aún no ha sido afectada.

t	T	n	r	ts	pr búsqueda
57	60	10	0	37	0,99
249	60	10	0	71	0,945997482
166	60	6	1	27	0,99
582	30	8	0	59	0,558364977
642	30	8	0	60	0,338789953
704	30	8	0	62	0,106315638
133	30	2	1	33	0,99
140	15	10	0	50	0,547105862
124	15	6	0	84	0,01
332	60	10	0	73	0,638428309
287	60	8	0	287	0,379003606
243	60	4	1	26	0,99
213	30	4	0	64	0,458324803
39	15	4	1	11	0,99
91	15	4	0	51	0,536074427
238	80	5	1	31	0,99
72	15	6	1	11	0,99

Tabla 6. Subconjunto de datos de entrenamiento

El conjunto completo de los datos de entrenamiento puede verse en el Anexo 1 al final de este documento. Cada uno de estos datos fue

normalizado en el intervalo [0,1] para evitar efectos negativos en el entrenamiento de la red debido a la diferencia de unidades o magnitudes de los valores de cada variable, utilizando la siguiente fórmula.

$$x_{normalizado} = \frac{X - X_{min}}{X_{max}} \quad (6)$$

Adicionalmente, se realizó un análisis de correlación para evitar información redundante. La siguiente tabla muestra la matriz de correlación de las variables utilizadas en la red, donde se observa una correlación poco significativa que permite deducir que los datos utilizados para el entrenamiento de la red no presentan redundancia.

	tiempo	timeout	núm. robots	éxito	tiempo búsqueda
Tiempo	1,00	0,14	-0,02	-0,24	0,36
Timeout	0,14	1,00	-0,03	0,18	0,06
núm. Robots	-0,02	-0,03	1,00	-0,07	0,17
Éxito	-0,24	0,18	-0,07	1,00	-0,46
tiempo búsqueda	0,36	0,06	0,17	-0,46	1,00

Tabla 7. Matriz de correlación de variables utilizadas en el entrenamiento

Para definir la arquitectura y algunas características de la red se realizaron algunos experimentos en los que se variaron el número de capas ocultas, el número de neuronas en dichas capas, la función de transferencia en cada neurona y el factor de aprendizaje. Obteniendo así, una red con un error de aprendizaje alrededor del 4% y cuyas características se muestran en la siguiente tabla.

Arquitectura	Neuronas en capa de entrada:5 Número de capas ocultas: 2 Neuronas en capa oculta 1:8 Neuronas en capa oculta 2:4 Neuronas en capa de salida:1 Función de Transferencia: $Sigmoide$
Método de aprendizaje	Backpropagation Factor de aprendizaje=0.001
Entrenamiento	Numero de patrones: 2413 Número de épocas: 10000
Prueba	Numero de patrones: 168

Tabla 8. Características de la red neuronal.

Gráficamente, la red neuronal del problema se visualiza en la siguiente figura:

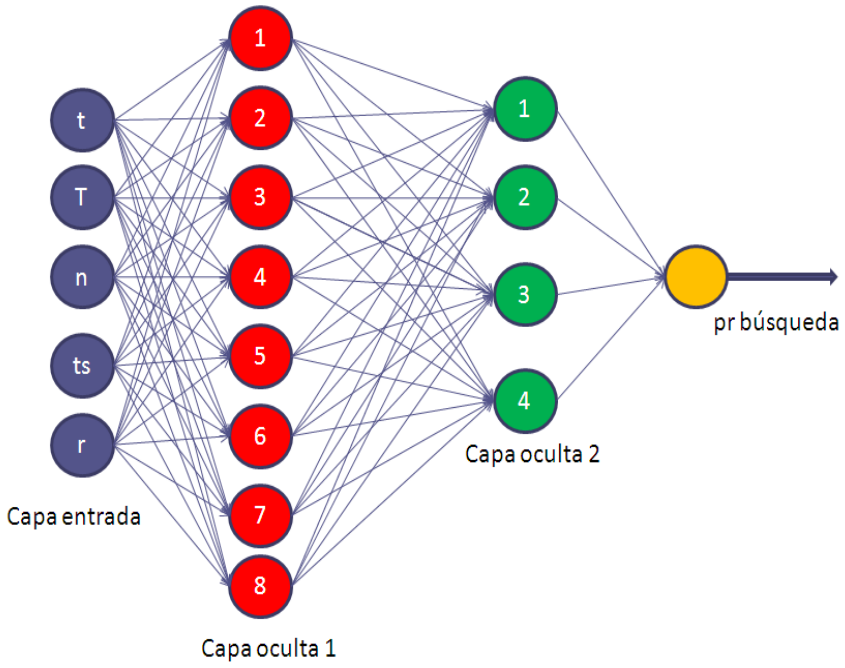


Figura 32. Arquitectura de la red neuronal

Como se indica en las características de la red, a cada neurona perteneciente a una capa oculta se le aplicó una función denominada *función de transferencia* de tipo sigmoide representada por la siguiente expresión:

$$f(s) = \frac{1}{1+e^{-s}} \quad (7)$$

Lo cual puede visualizarse a través de la siguiente figura:

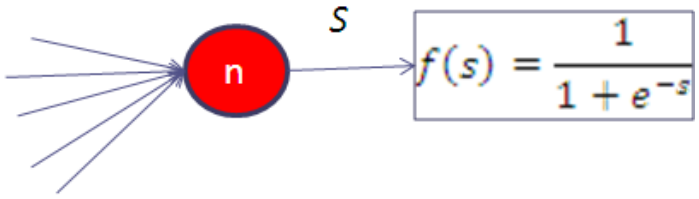


Figura 33. Función de transferencia para una neurona de capa oculta

Para medir la calidad del proceso de aprendizaje de la red (que incluyó 100000 épocas o iteraciones) se calcularon dos tipos de error: el *error medio cuadrad*o que representa la frecuencia de equivocación de la red y el *error máximo* que representa cuánto o con qué desfase se equivoca.

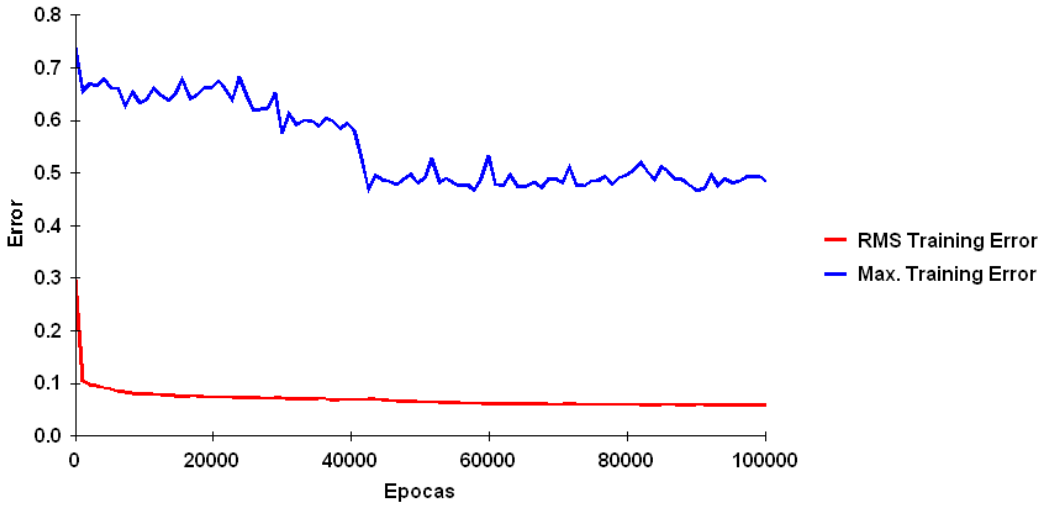


Figura 34. Error mínimo cuadrado durante el proceso de entrenamiento de la red

Así, al final del proceso de aprendizaje se obtuvo un *error medio cuadrado* de 0.04 que permite afirmar que la red tiene una precisión aproximada del 96% y un *error máximo de 0.48* que indica que cuando la red se equivoca lo hace con un desfase del 48%.

Los datos de entrenamiento y los pesos obtenidos de cada una de las conexiones de la red se pueden ver en el Anexo 1 y Anexo2 respectivamente.

De esta manera, cada individuo del sistema implementa la red neuronal con los parámetros obtenidos esperando lograr una probabilidad de búsqueda más o menos homogénea que represente la percepción global de todos los individuos respecto a su entorno. Para comprobarlo se realizaron varios experimentos cuyo contexto se describe en la siguiente tabla.

Experimentos			
No. Experimento	**No. Robots**	**Timeout**	**No. Objetos**
1	2	30	20
2	4	30	20
3	4	60	20
4	8	60	20

Descripción:
Al igual que en los experimentos anteriores, la misión de cada robot fue buscar y recolectar una serie de objetos colocados de manera aleatoria en el ambiente. La diferencia fue que en estos experimentos cada robot implementa la red neuronal planteada para calcular la probabilidad de búsqueda.

Tabla 9. Experimentos con la implementación de la red neuronal en cada robot

Los datos obtenidos a partir de estos experimentos se muestran en las siguientes gráficas:

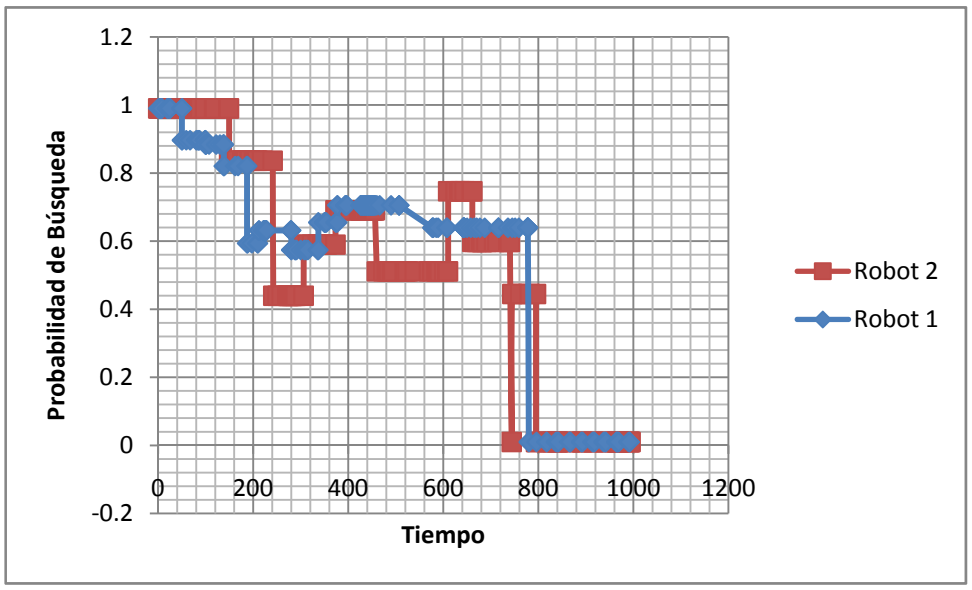


Figura 35. Probabilidad de búsqueda calculada a partir de la red neuronal en dos robots con *timeout* de 30s

En la anterior gráfica donde se muestran dos robots que en el transcurso del tiempo van calculando su probabilidad de búsqueda con base en sus percepciones, donde se observa una tendencia similar en la probabilidad de búsqueda permitiendo deducir que los robots tienen una percepción similar del entorno.

De igual manera en la siguiente gráfica se puede observar una tendencia similar, sin embargo alrededor del segundo 400, los robots 2 y 3 marcan una ligera diferencia que al cabo de algunos segundos disminuye.

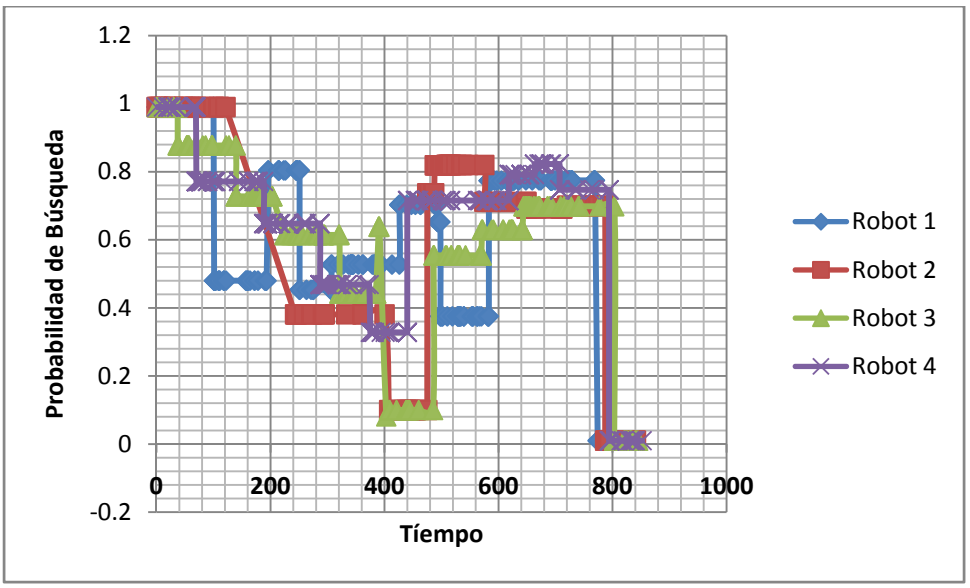


Figura 36. Probabilidad de búsqueda calculada a partir de la red neuronal en 4 robots con *timeout* de 30s

Incrementando el *timeout* a 60s la tendencia uniforme de la probabilidad de búsqueda es más notable tal como se puede observar en la siguiente gráfica.

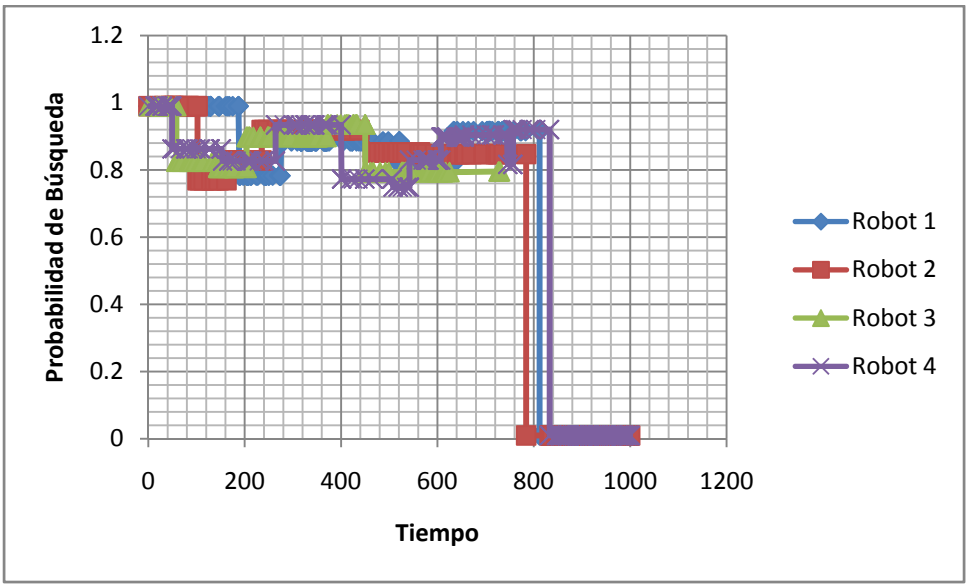


Figura 37. Probabilidad de búsqueda calculada a partir de la red neuronal en 4 robots con *timeout* de 60s

Por último se incrementó el número de robots a 8 de donde se obtuvo la siguiente gráfica que conserva la tendencia de las gráficas anteriores.

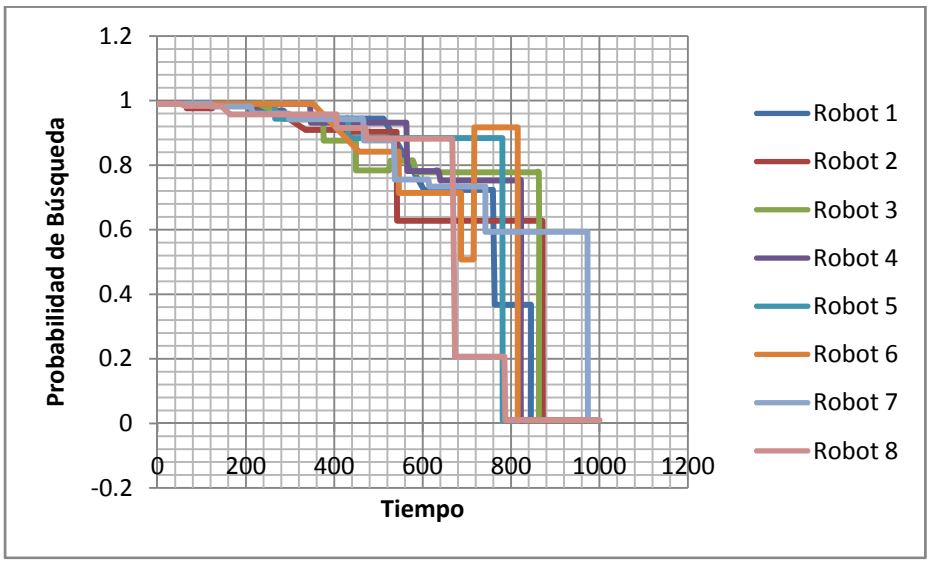


Figura 38. Probabilidad de búsqueda calculada a partir de la red neuronal en 8 robots con timeout de 60s

Si se mide la percepción del ambiente de un robot a través de su probabilidad de búsqueda entonces las gráficas indican que cada robot percibe el ambiente de una manera similar a sus homólogos. Como lo que se quiere es que esto represente un impacto sobre el objetivo global del

sistema, en el Capítulo 5 se muestran algunos experimentos adicionales que permiten evaluar el desempeño del sistema bajo estos planteamientos.

2.4.7. ***Esquema de Localización*.** Teniendo en cuenta las limitaciones de cada uno de los robots del sistema, a continuación se plantea un esquema de localización que no requiere del uso de modelos geométricos o topológicos como los que se mencionaron en el capítulo 2 de este documento.

En el caso del sistema propuesto los robots no requieren conocer su posición en coordenadas x, y sino una medida gradual o aproximada que les permita inferir donde se encuentran. Para esto se usaron algunos conceptos de la lógica borrosa, la cual a diferencia de lógica clásica está conformada por predicados que encierran vaguedad o incertidumbre.

Para el caso de la localización de un robot que aquí se propone, se requiere que éste determine su posición para saber si después de un proceso de exploración ha llegado a su base *o home*. Para lograrlo, en los siguientes párrafos se presenta un planteamiento basado en lógica difusa.

De acuerdo con el ambiente de operación que se planteó al inicio de este capítulo, se tiene una fuente de luz que representa la colonia, base o home de los robots del sistema. Cada uno de los robots debe determinar cuando está lejos o cerca de allí a través de uso de sus sensores de luz que le informan el nivel de *luminancia* (candelas/m^2) del área en que se ubica.

De lo anterior surge la pregunta de ¿cómo saber en dónde se encuentra el robot si sus sensores perciben una *luminancia* x? cuya respuesta se presenta a continuación:

La *luminancia* del ambiente percibida por un robot puede ser definida en términos de los conjuntos borrosos: bajo, medio y alto ilustrados en la siguiente figura. Los valores dados por los sensores dan un valor normalizado en el intervalo [0,1] siendo 0 la mínima *luminancia* y 1 la máxima.

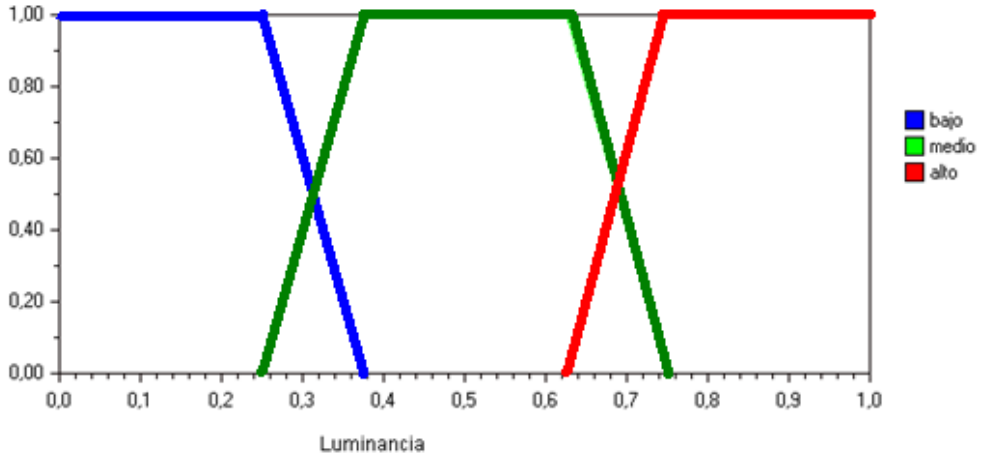


Figura 51. Conjuntos Borrosos para la variable Luminancia

La pertenencia a cada conjunto borroso de la figura está determinada por el cálculo de una fúnción de pertenencia, definida como:

$$\mu A(x): S \rightarrow [0,1] \tag{8}$$

Como cada uno de los conjuntos de la Figura 51 están definidos a través de un trapecio, la función de pertenencia $\mu A(x)$ se denomina trapezoidal la cual se define formalmente como:

$$\mu_A(x) = \begin{cases} 0, & \text{si } (x < a) \text{ ó } (x > d) \\ \dfrac{x-a}{b-a}, & \text{si } a \le x \le b \\ 1, & \text{si } b \le x \le c \\ \dfrac{d-x}{d-c}, & \text{si } c \le x \le d \end{cases} \tag{9}$$

Donde a, b, c y d representan algunos valores de la base del trapecio como se muestra en la siguiente figura:

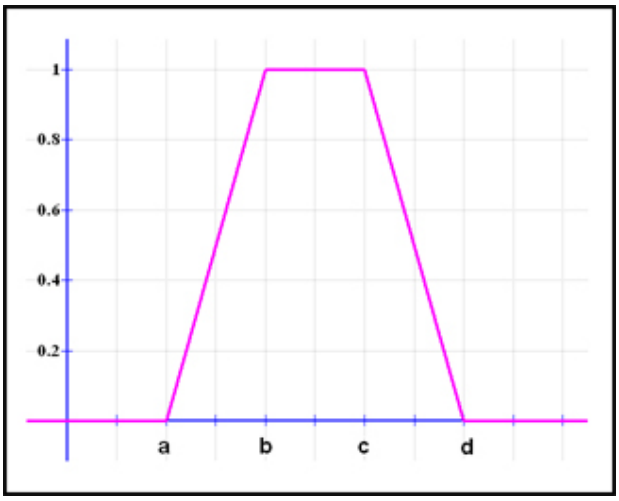


Figura 52. Función de pertenencia trapezoidal

Por lo tanto para cada uno de los conjuntos borrosos definidos en la Figura 51 los valores de a, b, c y d son:

Conjunto	Valores
Bajo	b=0, c=0.3, d=0.4
Medio	a=0.25, b=0.38, c=0.62, d=0.75
Alto	a=0.62, b=0.75, c=1

Tabla 10. Valores de a, b, c y d

De acuerdo con lo anterior, cada robot deberá calcular la función de pertenencia para cada lectura de sus sensores, determinando de esta manera que nivel de luminosidad percibe (bajo, medio, alto) tal como se muestra en el siguiente ejemplo.

Supóngase que el robot en un tiempo t está determinando que la luminancia promedio del ambiente es 0.74 candelas/m^2. De acuerdo con los datos de la tabla anterior y a la ecuación de la función de pertenencia trapezoidal se tiene que:

$$\mu\, bajo(x) = 0$$

$$\mu\, medio(x) = \frac{d-x}{d-c} = \frac{0.75-0.74}{0.75-0.62} = \frac{0.1}{0.13} = 0.83$$

$$\mu\, alto(x) = \frac{x-a}{b-a} = \frac{0.74-0.62}{0.75-0.62} = \frac{0.12}{0.13} = 0.9$$

Ahora bien, a partir de este tipo de conjuntos es posible definir reglas de la forma:

Si x es P entonces y es Q (10)

Donde P y Q son variables lingüísticas que se definen a partir de conjuntos borrosos. Para el caso en cuestión un ejemplo de regla seria:

Si luminancia es ***alta*** entonces robot está en home.

Al igual que en la lógica clásica P y Q pueden estar compuestos de otras variables lingüísticas unidas por operadores de tipo Y, O y NEGACION los cuales se definen respectivamente como:

$$\mu Y(x) = \max\left(\mu A(x), \mu B(x)\right) \quad (11)$$

$$\mu O(x) = \min\left(\mu A(x), \mu B(x)\right) \quad (12)$$

$$\mu \sim A(x) = 1 - \mu A(x) \quad (13)$$

La regla expresada en (10) es una implicación cuyo valor de verdad se determina a través de una función de pertenencia dada por:

$$\mu \rightarrow (x, y) = \min\left(\mu P(x), \mu Q(x)\right) \quad (14)$$

Esta expresión se denomina *implicación de Mandani* pero existen otras propuestas para determinar el valor de verdad de una implicación que se pueden encontrar en (Aja, 2008)

El conjunto de reglas de este tipo se denomina base de conocimiento con el cual es posible utilizar un mecanismo de inferencia para determinar cuándo una regla aplica o no. El más conocido es el *modus ponens* que en el contexto de la *lógica borrosa* se denomina *modus ponens generalizado* y está definido como:

Regla: *Si x es P entonces y es Q*

Hecho: x es *P'*

Conclusión: y es *Q'*

Nótese que a diferencia del *modus ponens* clásico, *P* no necesariamente debe ser igual a *P'*. El grado de verdad de la conclusión está dado por la expresión:

$$\mu P'(x) = max_{x \in U}\lceil \min\left(\mu P'(x), \mu \rightarrow (x, y)\right)\rceil \quad (15)$$

Donde U es el universo del discurso de P. Esta expresión se denomina regla composicional de interferencia.

En lo que respecta al sistema de localización se definieron las siguientes reglas:

No.	Regla
1	Si luminancia es alta entonces robot en home con certeza de 1
2	Si luminancia es media entonces robot en zona de exploración con certeza de 0.9
3	Si luminancia es baja entonces robot en zona límite con certeza de 0.7

Tabla 11. Reglas utilizadas para la localización de un robot

De tal manera que si ocurre un hecho de la forma:

Hecho: Luminancia es baja

El robot determinará en qué zona se encuentra realizando inferencia a través de *modus ponens,* llegando a una conclusión de la forma:

Conclusión: Robot en zona limite con certeza p

Que le permitirá activar algún comportamiento por ejemplo regresar a su base o redireccionar la búsqueda.

Las tres reglas mencionadas fueron utilizadas para la implementación tanto a nivel de simulación como en robots reales. Las pruebas y resultados del anterior planteamiento se muestran en el siguiente capítulo.

Como se ha indicado, para medir el nivel de luminancia el robot utiliza dos sensores (izquierdo y derecho) que dan un valor normalizado en el intervalo [0,1]. Estos valores además de servir como fuente de información para determinar la posición del robot se utilizan para lograr un efecto direccional en el desplazamiento de éste a través del siguiente algoritmo:

```
//Obtener los valores de luminancia de los sensores
  float llum = getLeftLuminance();
  float rlum = getRightLuminance();

//Mover el robot  x distancia y theta grados

  mov(x, (llum - rlum) *  Math.PI/4);
```

Algoritmo 10. Efecto direccional basado en los sensores de luz

Este algoritmo hace parte del módulo de acción de los comportamientos *Homing* y *ComeBack* señalados en secciones anteriores de este capítulo.

Es de aclarar que en lo que respecta a este trabajo se utilizó la variable de *luminancia,* pero es posible definir otro tipo de variables dependiendo de los sensores, como por ejemplo el color medido en escala RGB, la temperatura medida en grados Celsius, etc. Todas estas variables son útiles siempre y cuando permitan definir sobre ellas conjuntos borrosos.

2.5. SIMULACIÓN

Con el fin de simular y comprobar los planteamientos realizados anteriormente, se utilizó una librería bajo licencia *GNU GPL* denominada "simbad". Dicha librería está basada en el estándar *OpenGL* que permite crear gráficamente las estructuras básicas de un sistema robótico, tales como objetos, ambientes y por supuesto robots, con el fin de simular algoritmos de control e inteligencia artificial en el contexto de la robótica y los agentes (Hugues & Bredeche).

De esta forma se logró implementar todos los planteamientos hechos durante este capítulo, que van desde el esquema de comportamientos de cada robot hasta los esquemas de interacción y localización del sistema.

Para facilitar este proceso, se creó una interfaz gráfica de usuario que permite bajo ciertos parámetros generar los siguientes tipos de simulación:

- Robots basados en comportamientos: Es una simulación que implementa un robot reactivo basado en comportamientos de acuerdo con lo planteado durante este capítulo en la sección 4.3.1.2.
- Robots sin interacción: Es una simulación que involucra varios robots reactivos que no tienen ningún tipo de interacción entre ellos. La misión de cada robot es buscar y recolectar una serie de objetos colocados de manera aleatoria en el ambiente.
- Robots con interacción implícita: Al igual que el tipo de simulación anterior involucra varios robots reactivos que tiene como objetivo recolectar una seria de objetos ubicados de forma aleatoria. La diferencia radica en que ésta implementa el enfoque de interacción planteado en este capítulo que incluye dos partes: en la primera (robots con percepción individual) cada robot calcula su probabilidad de búsqueda con base en sus percepciones individuales mientras que la segunda (robots con percepción global) realiza este cálculo de acuerdo con una red neuronal previamente entrenada que permite representar una percepción generalizada a través del cálculo unificado de la probabilidad de búsqueda.

La siguiente figura muestra las opciones dadas por la interfaz gráfica para generar estas simulaciones.

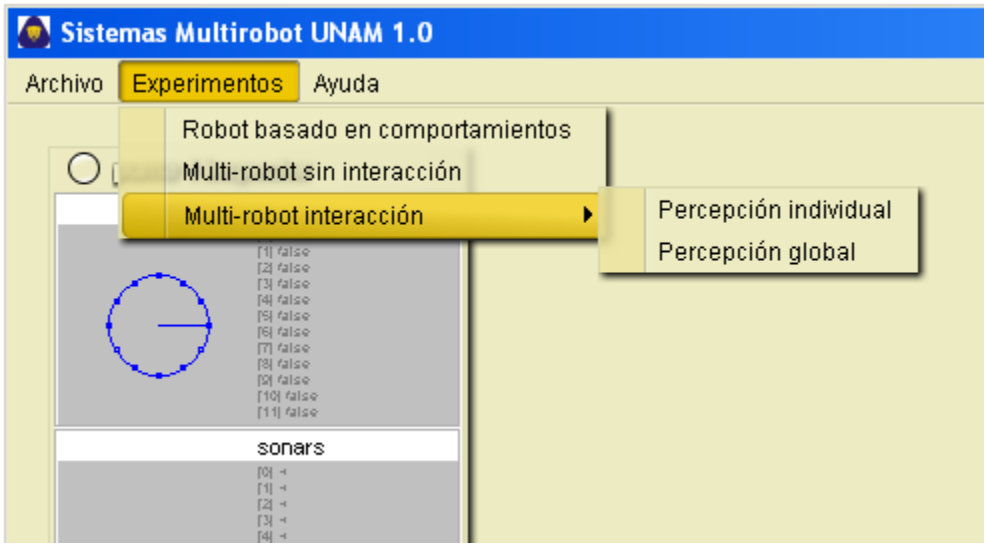


Figura 39. Opciones de la interfaz gráfica para generar simulaciones

Adicionalmente cada proceso de simulación permite guardar los datos obtenidos para su posterior análisis. Para visualizar dichos datos se implementó una funcionalidad a la cual se puede acceder a través de la opción *Archivo/Ver datos simulaciones* como lo ilustra la siguiente figura.

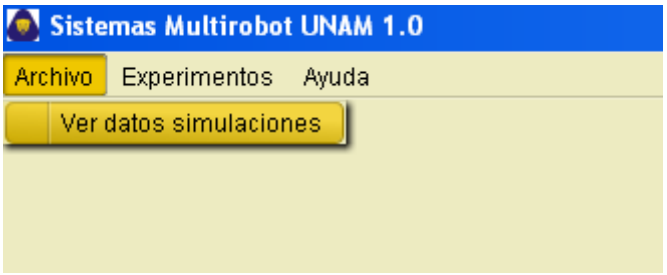


Figura 40. Opción para ver los datos generados por la simulación

Al seleccionar está opción se obtiene una ventana que permite buscar los archivos de datos de cada simulación en un directorio denominado *logs.*

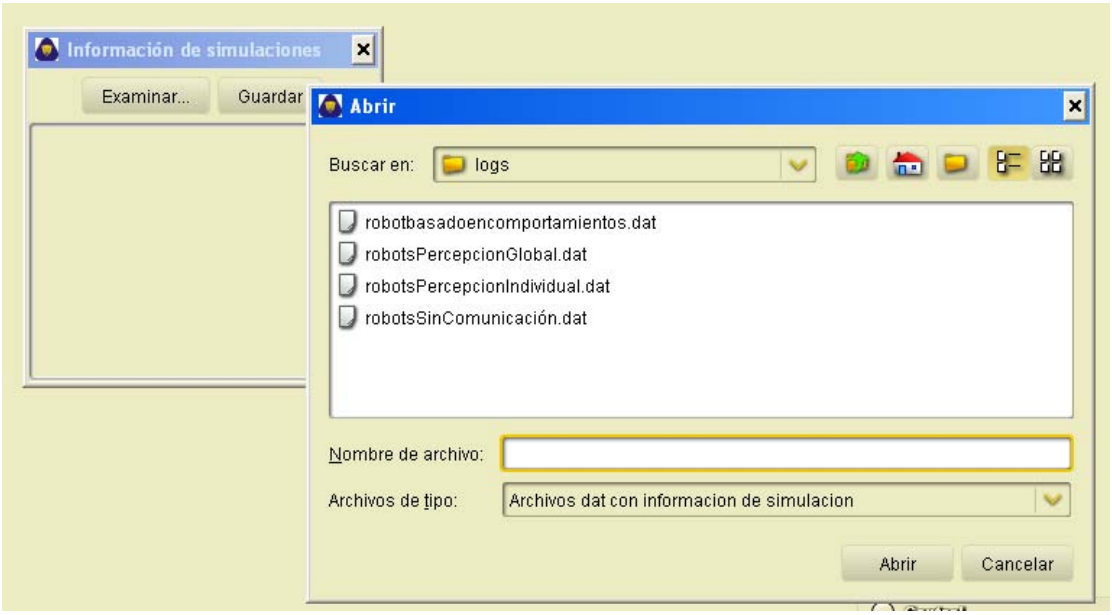
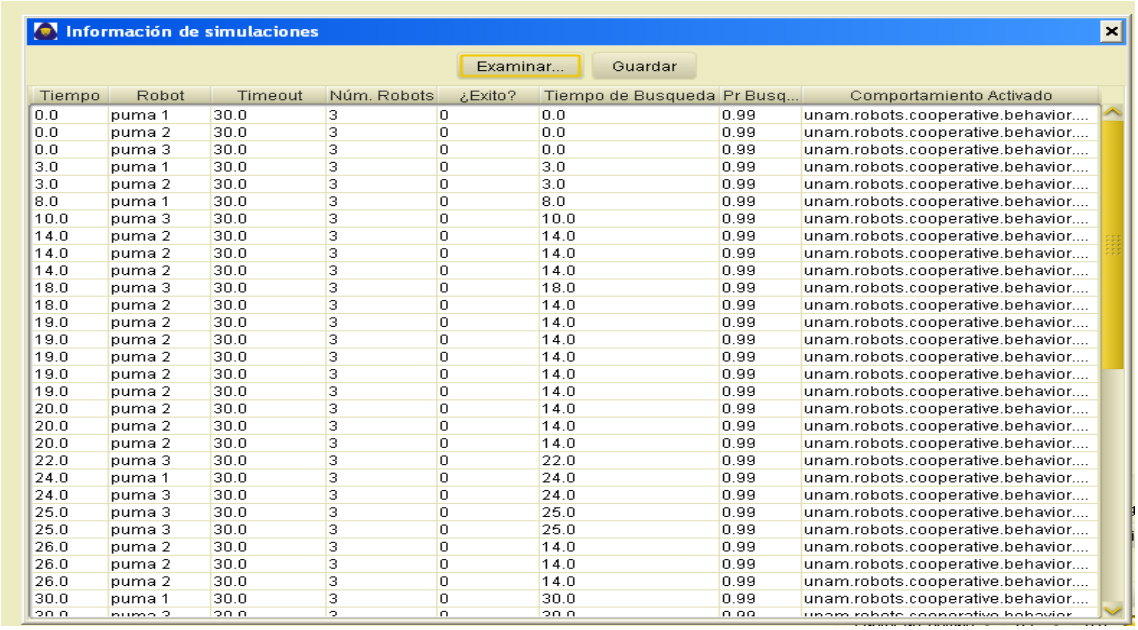


Figura 41. Archivos de simulación

Finalmente al seleccionar uno de los archivos, su contenido se despliega de la siguiente forma:



Tiempo	Robot	Timeout	Núm. Robots	¿Exito?	Tiempo de Busqueda	Pr Busq...	Comportamiento Activado
0.0	puma 1	30.0	3	0	0.0	0.99	unam.robots.cooperative.behavior....
0.0	puma 2	30.0	3	0	0.0	0.99	unam.robots.cooperative.behavior....
0.0	puma 3	30.0	3	0	0.0	0.99	unam.robots.cooperative.behavior....
3.0	puma 1	30.0	3	0	3.0	0.99	unam.robots.cooperative.behavior....
3.0	puma 2	30.0	3	0	3.0	0.99	unam.robots.cooperative.behavior....
8.0	puma 1	30.0	3	0	8.0	0.99	unam.robots.cooperative.behavior....
10.0	puma 3	30.0	3	0	10.0	0.99	unam.robots.cooperative.behavior....
14.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
14.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
14.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
18.0	puma 3	30.0	3	0	18.0	0.99	unam.robots.cooperative.behavior....
18.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
19.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
19.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
19.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
19.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
19.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
20.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
20.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
20.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
22.0	puma 3	30.0	3	0	22.0	0.99	unam.robots.cooperative.behavior....
24.0	puma 1	30.0	3	0	24.0	0.99	unam.robots.cooperative.behavior....
24.0	puma 3	30.0	3	0	24.0	0.99	unam.robots.cooperative.behavior....
25.0	puma 3	30.0	3	0	25.0	0.99	unam.robots.cooperative.behavior....
25.0	puma 3	30.0	3	0	25.0	0.99	unam.robots.cooperative.behavior....
26.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
26.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
26.0	puma 2	30.0	3	0	14.0	0.99	unam.robots.cooperative.behavior....
30.0	puma 1	30.0	3	0	30.0	0.99	unam.robots.cooperative.behavior....

Figura 42. Despliegue de los datos del archivo seleccionado

Cada uno de los procesos de simulación presenta tres ventanas. La primera denominada *inspector* es un panel que permite desplegar algunos atributos de un robot y las lecturas de sus sensores, la segunda, llamada *world* permite visualizar la ejecución de cada uno de los robots y su interacción con el entorno. La tercera es una ventana de control que permite cambiar de perspectiva la visualización de la ventana *world* y controlar la ejecución de la simulación con opciones como *pausa*, *inicio*, *reset* etc.

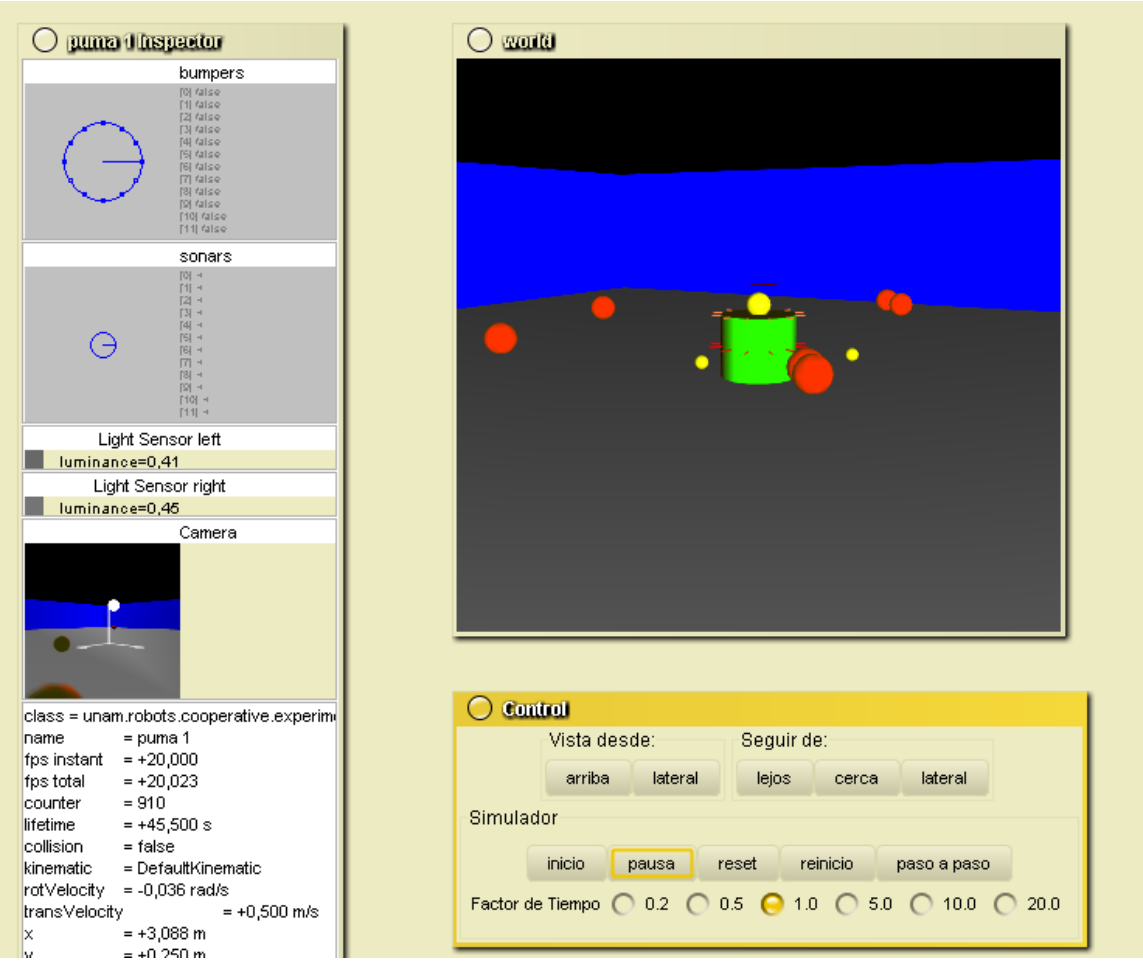


Figura 43. Componentes principales para visualizar una simulación

Con base en esta aplicación se hicieron varias simulaciones que proporcionaron los datos para corroborar los planteamientos hechos con anterioridad en este capítulo. Adicional al proceso de simulación, también se realizó una aproximación a una implementación física que se ilustra en la siguiente sección.

2.6. IMPLEMENTACIÓN FISICA

Después del proceso de simulación se procedió a realizar una sencilla implementación en robots reales. Para lo cual se utilizaron robots NXT de la marca LEGO®, los cuales están constituidos básicamente por un bloque de control, actuadores y sensores como se muestra en la siguiente figura.

Figura 44. Principales componente de un robot NXT (Lego Mindstorms, 2008)

En la siguiente tabla se muestran los detalles de cada uno de estos componentes:

	Nombre	Funcionalidad
1	Bloque de control (*brick*)	Es el cerebro o control del robot que contiene: Microcontrolador de 32-bit ARM7 256 Kbytes FLASH, 64 Kbytes RAM Microcontrolador AVR de 8-bit 4 Kbytes FLASH, 512 Byte RAM Bluetooth Puerto USB (12 Mbit/s) 4 puertos de entrada, cable 6-vias digital platform 3 puertos de salida, cable 6-vias digital platform 100 x 64 pixel LCD graphical display
2	Sensores de Contacto	Permiten detectar cuando el robot ha hecho contacto con algún elemento externo.
3	Sensor de sonido	Permite cuantificar en dB (decibeles) el sonido del ambiente
4	Sensor de luz	Este tipo de sensor cuantificar la luminosidad del ambiente lo cual permite al robot distinguir entre varios niveles de gris.
5	Sensor de ultrasonido	Permite al robot visualizar y detectar objetos u obstáculos , este sensor mide la distancia en centímetros (0cm-255cm) entre el sensor y un determinado objeto con una precisión de +/- 3cm.
6	Servo Motores	Permiten dar al robot movilidad o ejecutar acciones ante algunas condiciones.

Tabla 12. Principales componentes del robot NXT

Adicional a los componentes anteriormente citados, cada NXT está acompañado de una serie de piezas como ruedas y bloques plásticos que permiten configurar un robot de acuerdo con necesidades particulares. En lo que respecta a este trabajo se utilizaron varias piezas para dar forma a un robot que permitiera explorar una determinada área, en busca de varios objetos y llevarlos a un sitio en particular. En las siguientes figuras se muestran algunos de los elementos construidos para tal fin.

Figura 45. Gripper para la recolección de objetos

Figura 46. Chasis del robot compuesto por dos ruedas traseras y una rueda de giro libre en la parte de adelante.

En la siguiente figura se muestra un robot totalmente ensamblado, donde es posible visualizar dos sensores de luz (izquierdo y derecho) para la detección de objetos y localización. También se puede observar un sensor de sonar que fue utilizado para permitir la evasión de obstáculos.

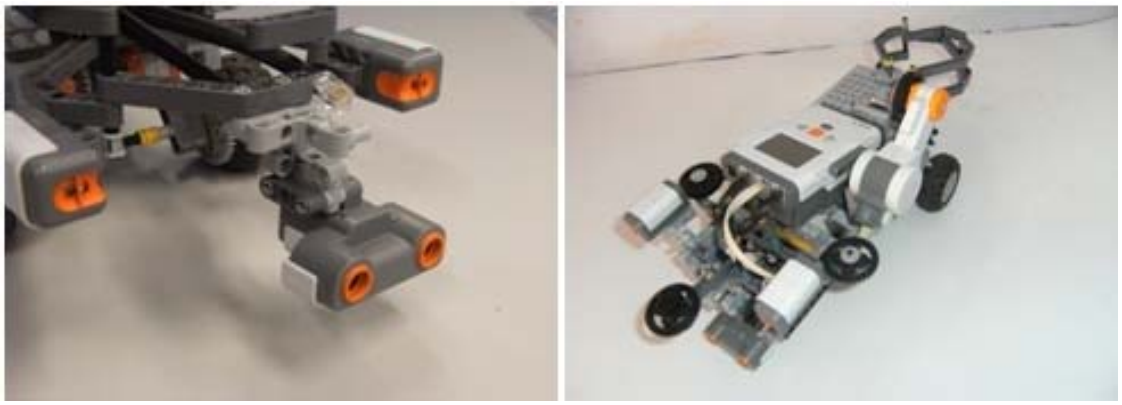


Figura 47. Robot totalmente ensamblado

En cuanto a la programación del robot existen varios ambientes de desarrollo entre los que se destacan los siguientes:

- NXC es un lenguaje de alto nivel similar a C. Utiliza el *firmware* original de LEGO y está disponible para Win32, Mac OSX y Linux.
- LEJOS NXJ permite la programación con Java. Se trata de un completo *firmware* que sustituye el oficial de LEGO que funciona tanto en Windows como en Linux.
- NXT_Python que permite controlar el *Brick* del NXT a través de sentencias Python sin necesidad de modificar el *firmware* original de Lego.

- RobotC Este es un software dirigido a educación desarrollado por Robotics Academy (Carnegie Mellon University) que permite programar en C.

Para la implementación que aquí se presenta, inicialmente se utilizó RobotC, pero debido a algunas limitaciones respecto a su licencia de uso, se optó por utilizar LEJOS NXJ por ser un ambiente de código abierto.

De esta manera se implementaron los planteamientos realizados durante este capítulo, tales como el esquema de comportamientos, comunicación y localización. Posteriormente se adaptó un ambiente adecuado para probar el sistema, conformado por objetos, obstáculos y un elemento identificable (fuente de luz) de acuerdo con lo señalado en la sección 4.3.2 y el cual se muestra en las siguientes figuras.

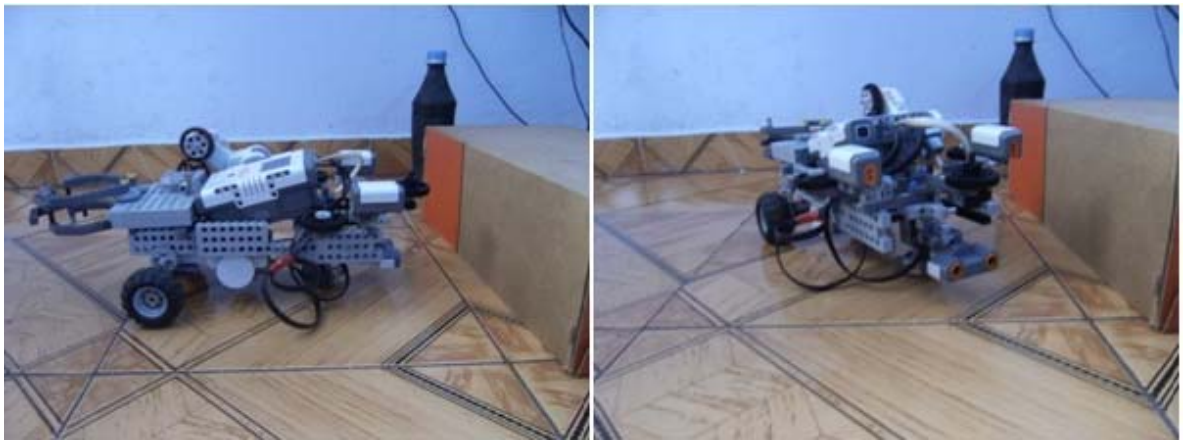


Figura 48. Detección y evasión de obstáculos

Figura 49. Detección y captura de un objeto

Como los sensores solo detectan variaciones de gris, los objetos utilizados son de color negro, los cuales deberán ser llevados a un sitio de alta luminosidad dada por una fuente de luz (lámpara) como se muestra a continuación:

Figura 50. Llegada del robot al home y entrega del objeto

En el siguiente capítulo se presentan algunas de las pruebas realizadas para poder obtener los resultados esperados, para lo cual se presentaron algunas dificultades que también se mencionan allí.

Capìtulo 5 PRUEBAS Y RESULTADOS

En este capítulo se presentan las pruebas realizadas durante este trabajo que permitieron corroborar los planteamientos realizados en el capítulo anterior. A su vez, para cada una de las pruebas se muestran los resultados obtenidos.

5.1. VERIFICACIÓN DE COMPORTAMIENTOS

De acuerdo con los planteamientos respecto al esquema de comportamientos de cada robot, esta prueba consistió en verificar la máquina de estados (definida en el capítulo anterior) en la que se activan o desactivan comportamientos de acuerdo con las lecturas realizadas por sus sensores. Para esto se utilizó el archivo de *log* generado por cada robot, un ejemplo de datos se muestran en la siguiente figura:

Tiempo	Robot	Timeout	¿Exito?	Tiempo de Bu...	Pr Busqueda	Comportamiento Activado
0.0	puma 1	60.0	0	0.0	0.99	unam.robots.cooperative.behavior.basic.Wandering
17.0	puma 1	60.0	0	17.0	0.99	unam.robots.cooperative.behavior.basic.Avoidance
18.0	puma 1	60.0	0	17.0	0.99	unam.robots.cooperative.behavior.basic.GripObject
18.0	puma 1	60.0	0	17.0	0.99	unam.robots.cooperative.behavior.basic.Avoidance
26.0	puma 1	60.0	0	17.0	0.99	unam.robots.cooperative.behavior.basic.Homing
58.0	puma 1	60.0	0	17.0	0.99	unam.robots.cooperative.behavior.basic.Avoidance
58.0	puma 1	60.0	0	17.0	0.99	unam.robots.cooperative.behavior.basic.Homing
58.0	puma 1	60.0	0	17.0	0.99	unam.robots.cooperative.behavior.basic.Avoidance
59.0	puma 1	60.0	0	17.0	0.99	unam.robots.cooperative.behavior.basic.Homing
59.0	puma 1	60.0	0	17.0	0.99	unam.robots.cooperative.behavior.basic.Avoidance
61.0	puma 1	60.0	1	17.0	0.99	unam.robots.cooperative.behavior.basic.Deliver
61.0	puma 1	60.0	0	0.0	0.99	unam.robots.cooperative.behavior.basic.Wandering
77.0	puma 1	60.0	0	15.0	0.99	unam.robots.cooperative.behavior.basic.Avoidance
85.0	puma 1	60.0	0	24.0	0.99	unam.robots.cooperative.behavior.basic.Wandering
96.0	puma 1	60.0	0	34.0	0.99	unam.robots.cooperative.behavior.basic.Avoidance
96.0	puma 1	60.0	0	34.0	0.99	unam.robots.cooperative.behavior.basic.GripObject
96.0	puma 1	60.0	0	34.0	0.99	unam.robots.cooperative.behavior.basic.Avoidance
116.0	puma 1	60.0	0	34.0	0.99	unam.robots.cooperative.behavior.basic.Homing
143.0	puma 1	60.0	0	34.0	0.99	unam.robots.cooperative.behavior.basic.Avoidance
143.0	puma 1	60.0	0	34.0	0.99	unam.robots.cooperative.behavior.basic.Homing
143.0	puma 1	60.0	0	34.0	0.99	unam.robots.cooperative.behavior.basic.Avoidance
144.0	puma 1	60.0	0	34.0	0.99	unam.robots.cooperative.behavior.basic.Homing
144.0	puma 1	60.0	0	34.0	0.99	unam.robots.cooperative.behavior.basic.Avoidance
146.0	puma 1	60.0	1	34.0	0.99	unam.robots.cooperative.behavior.basic.Deliver

Figura 1. Archivo de *log* generado por la simulación

Como se observa en la figura el comportamiento de *Wandering* se activa inicialmente permitiendo deducir que el robot partiendo de un estado de **inicio,** pasa a un estado de **exploración** ya que la probabilidad de búsqueda en ese momento es de 0.99, posteriormente evade algún obstáculo activando el comportamiento de *Avoidance*. Inmediatamente encuentra un objeto por lo que igualmente se puede deducir que hace una transición al estado de **carga** activando su comportamiento de *GripObject*. Con el

objeto cargado pasa a un estado de **entrega** donde activa los comportamientos de *Homing, Avoidance y Deliver.* Este último, provoca que el robot quede nuevamente en un estado de **inicio** e inmediatamente después pase a un estado de **exploración** para emprender la búsqueda de más objetos ya que la probabilidad de búsqueda sigue siendo 0.99. Lo anterior verifica la máquina de estados del robot planteada en el capitulo anterior.

Dicha máquina de estados también se comprobó a través de la implementación en robots reales, aunque con algunos inconvenientes debidos a la precisión de lectura en algunos sensores especialmente los de luz.

5.2. CONVERGENCIA DEL SISTEMA

Definición: Sea *n* el número de objetos dispuestos aleatoriamente en el ambiente y *m* el número de objetos capturados por los individuos del sistema. Se dice que dicho sistema converge a una solución cuando *n* es igual a *m*.

Para verificar esto, se tomaron como base algunos datos generados a través de simulación. En el primer caso se simuló un sistema que involucra robots reactivos sin ningún tipo de interacción donde se evidenció que al cabo de cierto tiempo el sistema converge a una solución. En la figura se muestran los resultados de una serie de nueve experimentos en los que el *timeout* y el número de robots fueron variados, la misión fue la misma en todos los casos: buscar y recolectar 20 objetos colocados de manera aleatoria.

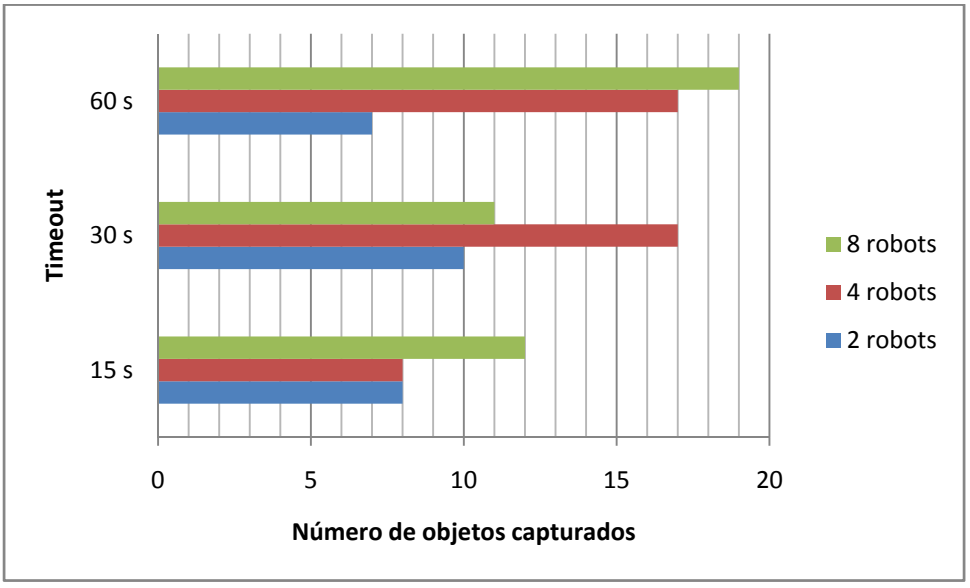


Figura 2. Número de objetos recolectados por robots sin comunicación en varios experimentos

Posteriormente se realizaron pruebas bajo similares condiciones (*timeout,* número de robots, número de objetos) con robots que implementaban el esquema de comunicación planteado, que como se ha señalado, está constituido en dos partes:

- Robots con percepción individual: donde cada robot calcula su probabilidad de búsqueda con base en sus percepciones individuales.
- Robots con percepción global: donde el cálculo de la probabilidad de búsqueda se realiza de acuerdo con una red neuronal previamente entrenada, que permite representar una percepción más generalizada y uniforme logrando un efecto similar al producido por la concentración de *feromona* de los sistemas biológicos de algunos insectos ya mencionados.

En los dos casos, el sistema debe converger a una solución cuando cada uno de sus individuos, queda en un estado de inicio o reposo indefinidamente, después de haber explorado el ambiente. Esto sucede cuando la probabilidad de búsqueda en cada robot es mínima (para la pruebas realizadas la probabilidad mínima fue de 0.01) lo cual a su vez implica que de acuerdo con la definición de convergencia *m* tiende a *n*. Las siguientes graficas muestran los resultados de los experimentos realizados.

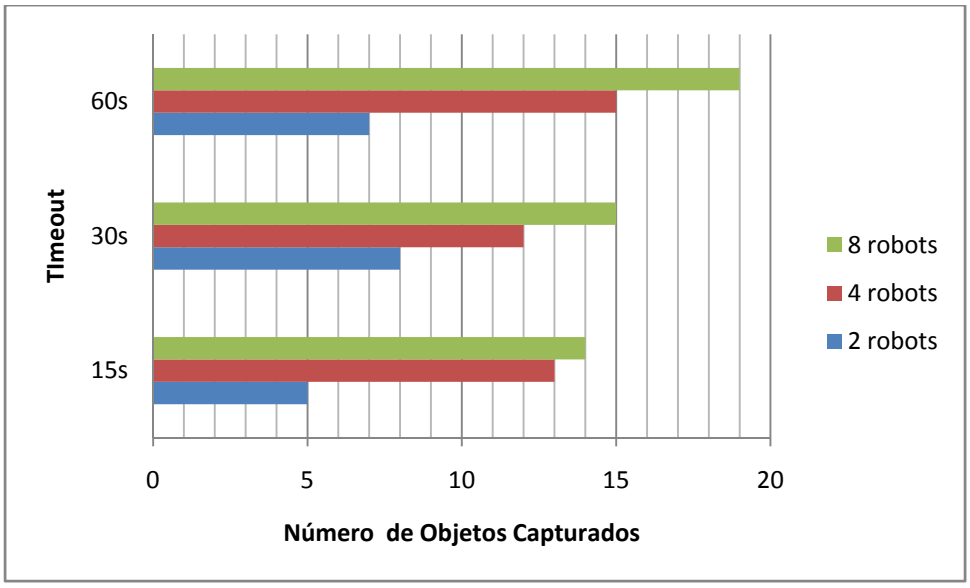


Figura 3. Número de objetos capturados por robots con percepción individual en varios experimentos

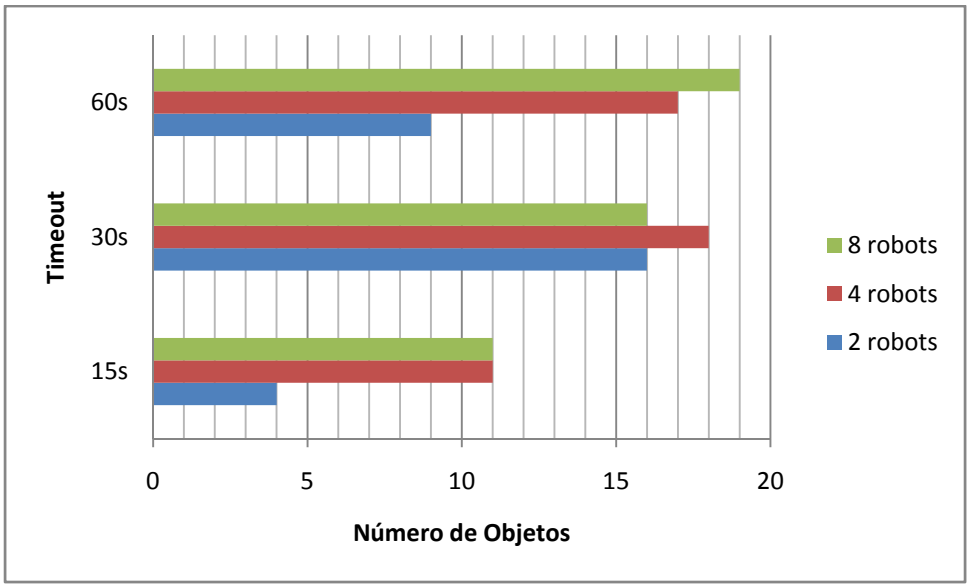


Figura 4. Número de objetos capturados por robots con percepción global en varios experimentos

En los tres casos se encuentra una tendencia similar en cuanto al número de objetos capturados. Entonces, ¿cuál es la diferencia en estos tres esquemas?

Si bien los anteriores gráficos demuestran que cada uno de ellos converge a una solución, la diferencia radica en el tiempo en que lo hacen. La siguiente gráfica muestra el promedio del tiempo de ejecución para alcanzar las soluciones anteriormente ilustradas.

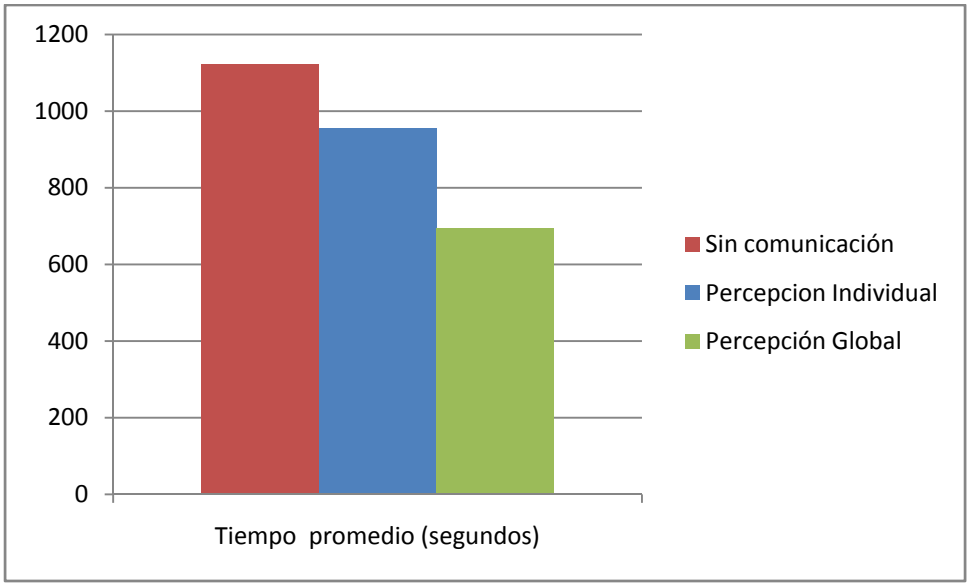


Figura 5. Tiempo de convergencia promedio.

De donde se observa que el sistema de comunicación implícita basado en percepción global, converge a una solución más rápidamente que los otros dos casos. Sin embargo, la pregunta ahora es: ¿qué tan buena es esta solución?

La siguiente sección hace un análisis del desempeño del sistema contrastando estos tres esquemas, con el fin de resolver esta pregunta.

5.3. DESEMPEÑO DEL SISTEMA

Para medir el desempeño del sistema se propone un índice al que se le denominó *índice de captura promedio ICP* que permite determinar el promedio de objetos capturados en una unidad de tiempo por los robots del sistema y cuya expresión matemática es:

$$ICP = \frac{num\ objetos\ encontrados}{tiempo * num\ robots} \tag{16}$$

De acuerdo con esto a continuación se muestran algunas gráficas comparativas del desempeño del sistema obtenidas a partir de varias pruebas con robots que implementan los planteamientos del capítulo anterior. En primer lugar se puede observar una gráfica que muestra el desempeño en robots reactivos que no implementan ningún esquema de interacción. En segundo lugar se tiene el desempeño de robots reactivos con un esquema de interacción implícita basado en la percepción individual del entorno. En último lugar se tiene el desempeño de robots reactivos con interacción implícita basados en la percepción global del entorno. En los tres casos se hicieron pruebas para 2, 4 y 8 robots con *timeout* de 15, 30 y 60 segundos cuya misión en todos los casos fue la búsqueda y captura de 20 objetos colocados de manera aleatoria en el ambiente.

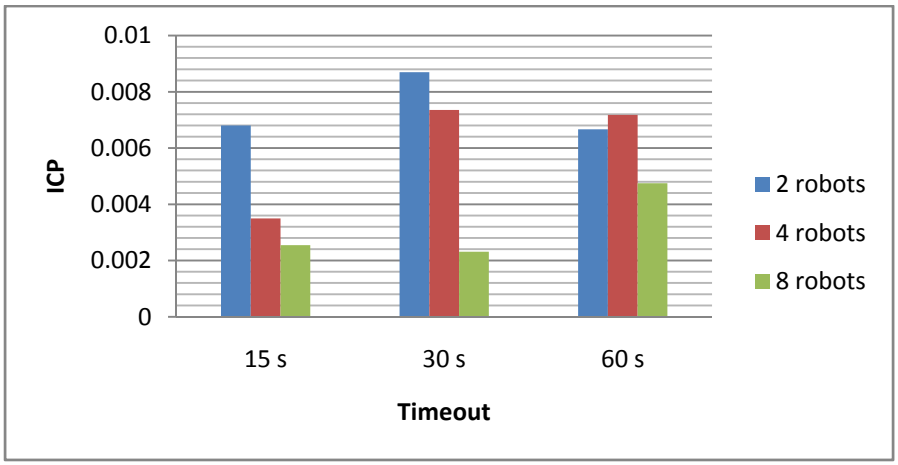


Figura 6. Robots sin interacción

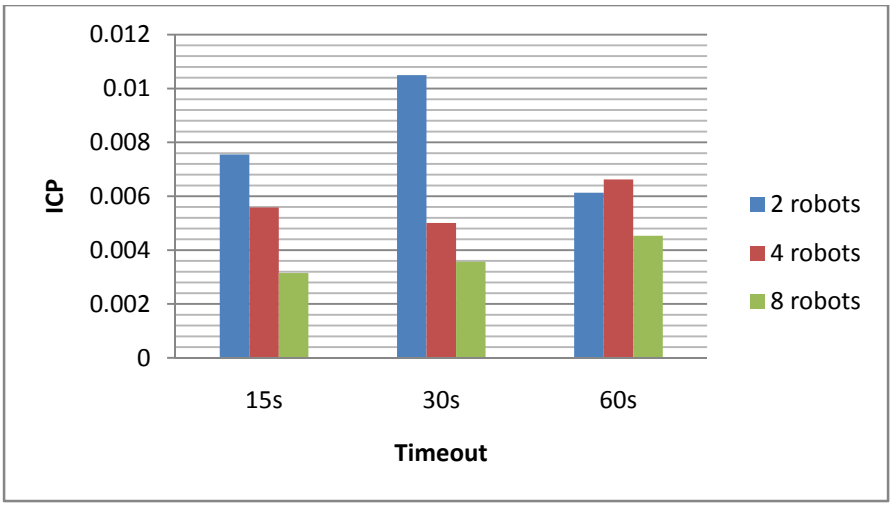


Figura 7. Robots con percepción individual

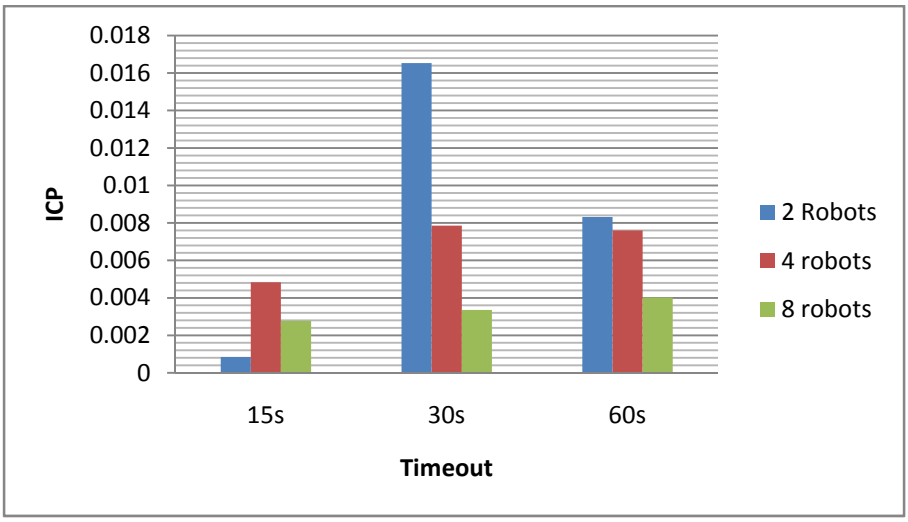


Figura 8. Robots con percepción global

En un principio se puede pensar que entre más objetos recolectados mejor es el sistema, pero hay que considerar los recursos empleados para ello, en este caso el tiempo y el número de robots. Es por eso que en los tres casos se observa un máximo nivel de desempeño para un *timeout* de 30 segundos con dos robots. Contrastando el máximo nivel de desempeño en cada uno de los tres casos se obtiene que el sistema es mucho más eficiente con un esquema de interacción basado en percepción global del entorno tal como lo ilustra la siguiente gráfica.

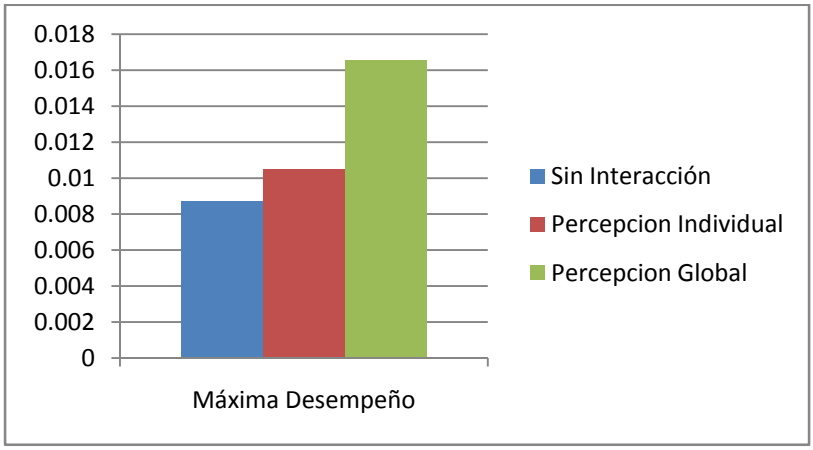


Figura 9. Comparación de desempeño

Aquí surgen dos preguntas: ¿cuál debería ser el valor óptimo del *timeout*? y ¿cuál es el número óptimo de robots?

En el caso del *timeout* no se sabe, ya que este valor depende de muchas variables implícitas en la naturaleza y condiciones del entorno. Por ejemplo, si se tiene un ambiente cuyos objetos se encuentran a una distancia considerable respecto a la base de los robots, es altamente probable que con un *timeout* pequeño no se alcance a realizar una completa exploración del entorno y el desempeño no sea el esperado. En caso contrario (*timeout*

grande) es posible que el proceso de exploración sea excesivo lo cual igualmente se reflejará en el desempeño del sistema.

En el caso del número de robots es posible aplicar una técnica de optimización que maximice el desempeño en función de este número.

5.4. PRECISIÓN DEL ESQUEMA DE LOCALIZACIÓN

El esquema de localización basado en lógica borrosa, se verificó a través de dos formas: en la simulación y utilizando robots reales. En los dos casos se utilizó un solo robot para el cual se tomaron varias lecturas correspondientes a la luminancia del ambiente y se realizó el proceso de inferencia para las tres reglas señaladas en el capítulo anterior aplicando la definición del modus ponens generalizado.

Las siguientes gráficas muestran los resultados de dicha inferencia para 130 lecturas realizadas para el robot simulado y el robot real.

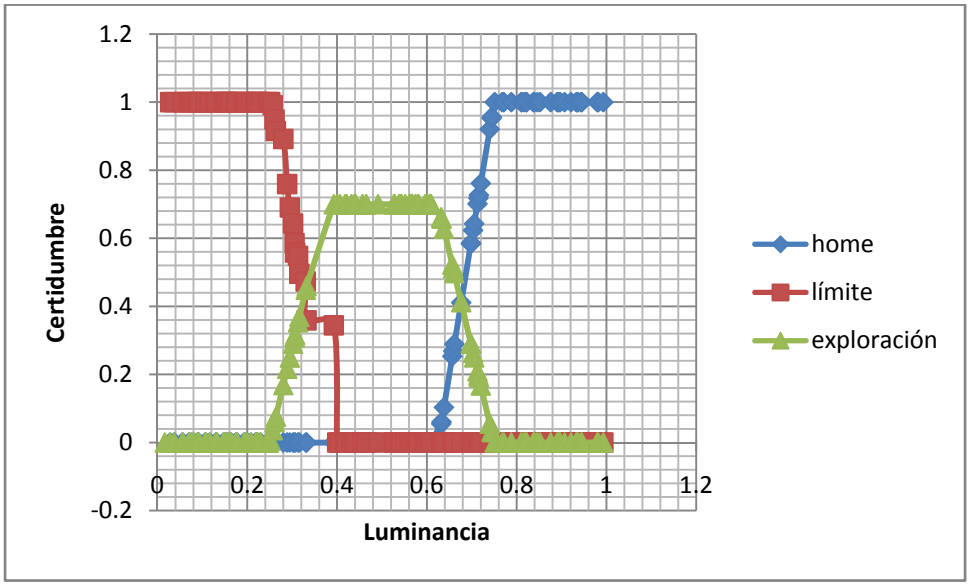


Figura 10. Inferencia de la posición para un robot real

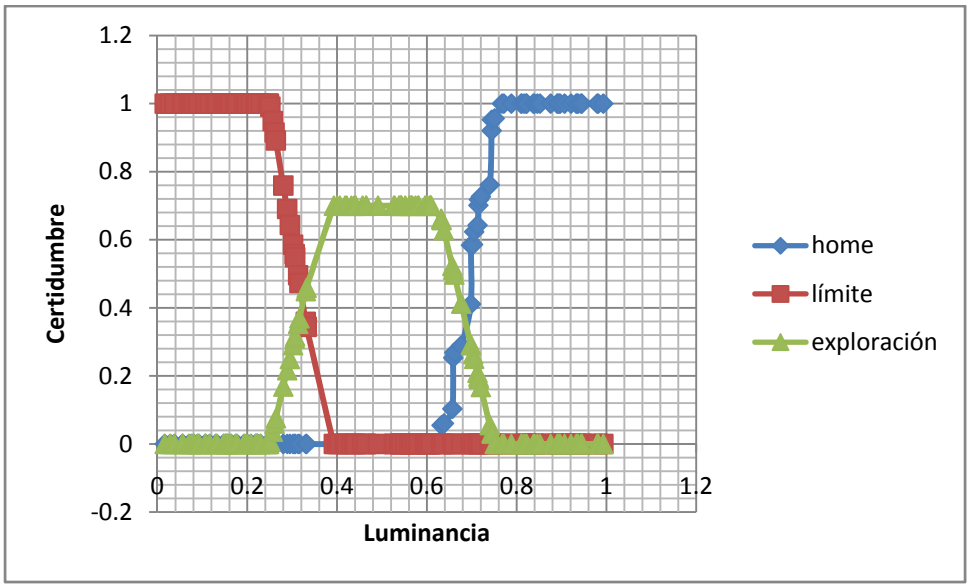


Figura 11. Inferencia de la posición para un robot simulado

Aparte de la similitud en los resultados obtenidos para los dos tipos de robot, se observa que el proceso de inferencia conserva una lógica en cuanto al contexto del problema, ya que para valores altos de luminancia se obtiene con una alta certidumbre que el robot probablemente está ubicado en el *home*. Lo cual corrobora que el esquema de localización planteado funciona adecuadamente

Capìtulo 6 Conclusiones

De acuerdo con las hipótesis planteadas al inicio de este trabajo se logró comprobar experimentalmente que un grupo de robots que implementa un enfoque reactivo o basado en comportamientos, genera conductas colectivas que permiten obtener una solución a determinado problema o realizar una determinada tarea.

Sin embargo también se comprobó que esto no es suficiente ya que si bien se obtiene una solución, el tiempo de convergencia de ésta es indeterminado. Por lo tanto se propuso un esquema de interacción que permitió acelerar el tiempo de convergencia y obtener una solución más eficiente.

Dicho esquema consistió en un modelo que permitió cuantificar la percepción de cada robot respecto a su entorno para tomar decisiones. Inicialmente se obtuvieron considerables diferencias en la percepción de cada robot que no contribuían al objetivo global del sistema. Razón por la cual se propuso una red neuronal entrenada con información relevante acerca del ambiente, la cual permitió que los individuos del sistema tuvieran percepciones del entorno homogéneas que lograran activar o desactivar comportamientos contribuyendo considerablemente al objetivo global.

El inconveniente bajo este esquema es el cálculo en tiempo real de salida de la red que contradice de cierta forma la premisa de la sencillez relativa en cada individuo. Por lo que para trabajos futuros se plantea el uso de un *algoritmo genético* que fuera de línea permita obtener el valor de los parámetros que afectan la probabilidad de búsqueda.

Adicionalmente se verificó que es posible usar un esquema de localización que contrario a otras técnicas, no requiere almacenar y actualizar en tiempo real mapas geométricos o topológicos del ambiente permitiendo obtener características de versatilidad y dinamismo en el sistema.

Algunos de los planteamientos no solo se comprobaron en el campo de la simulación sino además utilizando robots reales. Sin embargo se tuvieron que sortear varios inconvenientes técnicos y logísticos los cuales se mencionan a continuación:

- Por razones de costos no se contó con un número adecuado de robots (por lo menos 4) reales para hacer las pruebas.

- Los robots utilizados presentan demasiadas restricciones técnicas, especialmente sus sensores, cuyas lecturas no tienen mucha precisión.
- Debido a la baja precisión de los sensores fue necesario disponer de un entorno o ambiente controlado lo cual en muchas ocasiones no fue del todo posible.

Aunque los resultados obtenidos fueron satisfactorios como trabajo a futuro se espera poder hacer una implementación con robots reales construidos de conformidad con el diseño estructural planteado en este documento.

BIBLIOGRAFIA

Aja, F. S. (2008). *Biblioteca Virtual Miguel de Cervantes.* Recuperado el 02 de 11 de 2008, de http://descargas.cervantesvirtual.com/servlet/SirveObras/01305008611682844756802/014462_3.pdf

Arkin, R. C., & Balch, T. (1997). AuRA: Principles and Practice in Review. *Journal of Experimental & Theoretical Artificial Intelligence*, (págs. 175-188).

Arkin, R. C., Balch, T., & Nitz, E. (1993). Communication of behavioral state in multi-agent retrieval tasks. *International Conference on Robotics and Automation*, (págs. 588-594).

Arnaud, P. (1999). *Application d'une nouvelle architecture de contrôle réactive à des déplacements collectifs de robots mobiles.* Lausanne: PhD Thesis, DI-EPFL.

Asama, H., Ozaki, K., Matsumoto, A., Ishida, Y., & Endo, I. (1992). Development of task assignment system using communication for multiple autonomous robots. . *Journal of Robotics and Mechatronics*, (págs. 122-127).

Ayache, N. (1991). *Artificial Vsion for Mobile Robots.* MIT Press.

Balch, T., & Parker, L. E. (2002). Robot Teams: From Diversity to Polymorphism. Natick, Massachusetts: AK Peters.

Balkenius, C. (1994). *Biological Learning and Artificial Intelligence.*

Beni, G., & Wang, J. (1990). On cyclic cellular robotic systems. *Japan USA Symposium on Flexible Automation*, (págs. 1077-1083). Kyoto.

Blumer, H. (1951). Collective Behavior. *A. M. Lee, ed., Principles of Sociology* (págs. 67-121). New York: Barnes & Noble.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm Intelligence from Natural to Artificial Systems.* New York: Oxford Univesity Press.

Borenstein, J., Everett, B., & Feng, L. (1996). *Navigating Mobile Robots: Systems and Techniques.* Wellesley, MA: A. K. Peters.

Cai, A., Fukuda, T., Arai, F., Ueyama, T., & Sakai, A. (1995). Hierarchical Control Architecture for Cellular Robotic System - Simulation and Experiments. *Proc. of the IEEE Int. Conference on Robotics and Automation*, (págs. 1191-1196).

Caloud, P., Choi, W., Latombe, J.-C., Le Pape, C., & Yim, M. (1990). Indoor automation with many mobile robots. *In Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*, (págs. 67-72).

Cao, Y. U., Fukunaga, A. S., & Kahng, A. B. (1997). Cooperative Mobile Robotics: Antecedents and Directions. En *Autonomous Robots* (págs. 7-27).

Carletti, E. J. (25 de julio de 2008). *Servos Caracteristicas básicas.* Recuperado el 22 de septiembre de 2008, de http://robots-argentina.com.ar/MotorServo_basico.htm

Chaimowicz, L., Kuma, V., & Campos, M. F. (2004). A paradigm for Dynamic Coordination of Multiple Robots. *Autonomous Robots* , 7-21.

Dias, M. B. (2004). *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments.* Pittsburgh, Pennsylvania: The Robotics Institute Carnagie Mellon University.

Drogoul, A., & Ferber, J. (1994). From Tom Thumb to the Dockers: Some experiments with foraging robots. *Proceedings of the Second International Conference on Simulation of Adaptive Behavior* , 451-459.

Dudek, G., Jenkin, M. R., Milios, E., & David, W. (1997). *A Taxonomy for Multi-Agent Robotics.* Boston: Kluwer Academic Publisher.

Durfee, E. H., & Rosenschein, J. (1994). *Disitributed Problem Solving and Multiagent System: Comparisons and Examples.*

Fukuda, T., & Ueyama, T. (1994). Cellular Robotics and Micro Robotic Systems. *World Scientific Series in Robotics and Automated Systems.* Singapore.

Galeon. (2008). Recuperado el 23 de 06 de 2008, de http://www.galeon.com/gilamu/lobo/lobo.html

Gallardo, L. D. (1999). *Aplicación del muestreo bayesiano en robots móviles: estrategias para localización y estimación de mapas del entorno.* Universidad de Alicante.

Halva, L. T. (2003). *Prey Retrieval by Swarm of Robots.* Bruselas.

Hibilisco, S. (2003). *Concise Encyclopedia of Robotics.* McGraw-Hill.

Hugues, L., & Bredeche, N. (s.f.). *Simbad : an Autonomous Robot Simulation.* Recuperado el 15 de 11 de 2008, de http://www.lri.fr/~bredeche/MyBiblio/SAB2006simbad.pdf

IEEE. (2001). *The CLARAty Architecture for Robotic Autonomy.* IEEE, Big Sky, MT.

IRobot. (2008). Recuperado el 01 de 06 de 2008, de http://store.irobot.com/

Izquierdo, E. (2004). *Collective Intelligence in Multi-Agent Robotics: Stigmergy, Self Organization and Evolution.* Brighton: University of Sussex.

Jaffé, K. (1993). *El mundo de las hormigas.* Maracaibo: Editorial Equinoccio, Universidad Simon Bolivar.

Jin, K., Liang, P., & Beni, G. (1994). Stability of synchronized distributed control of discrete swarm structures. *IEEE ICRA*, (págs. 1033-1038).

Kalipedia. (2008). Recuperado el 1 de 12 de 2008, de http://mx.kalipedia.com/matematicas-geometria/tema/robots-exploradores.html?x1=20070821klpinginf_95.Kes&x=20070821klpinginf_96.Kes

Klir, G., & Yuan, B. (1995). Fuzzy Sets and Fuzzy Logic: Theory and Applications.

Kube, C. R., & Zhang, H. (1992). Collective robotic intelligence. *Proceedings of the Second International Workshop on Simulation of Adaptive Behavior*, (págs. 460-468).

Kuipers, B. (1999). *The Spatial Semantic Hierarchy.* Austin, Texas: University of Texas.

Le Pape, C. (1990). A combination of centralized and distributed methods for multi-agent planning and scheduling. *IEEE ICRA*, (págs. 488-493).

Lego Mindstorms. (2008). Recuperado el 13 de 11 de 2008, de http://mindstorms.lego.com/eng/india_dest/Default.aspx

Liu, J., & Wu, J. (2001). *Multi- Agent Robotic Systems.* Boca Raton: CRC Press.

Low Tech RVing. (2008). Recuperado el 25 de 06 de 2008, de http://rvtravel.com/blog/lowtech/labels/ants.html

Marsella, S., Adibi, J., & Al-Onaizan, Y. (1999). On being a teammate:experiences acquired in the design of RoboCup teams.

Martín, F., Matellan, V., Barrera, P., & María, C. J. (2006). *Localización basada en Lógica Difusa y Filtros de Kalman para robot con patas.* Madrid: Grupo de Robótica, Universidad Rey Juan Carlos.

Mataric, M. J. (1994). *Interaction and Intelligent Behavior.* PhD Thesis, Massachusetts Institute of Technology.

McFarland, D. (1994). *Towards robot cooperation.* Cambridge MA: The MIT Press.

Mitsumoto, N., Fukuda, T., Shimojima, K., & Ogawa, A. (1995). Micro Autonomous Robotic System and Biologically Inspired Immune Swarm Strategy as a Multi Agent Robotic System. *Proc. of the IEEE Int. Conf. on Robotics and Automation*, (págs. 2187-2192).

Negenborn, R. (2003). *Robot Localization and Kalman Filters.* Utrecht University.

Noreils, F. R. (1993). Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment. *The International Journal of Robotics Research*, (págs. 79-98).

OUNAE. (2008). Recuperado el 1 de 12 de 2008, de http://www.ounae.com/2007/04/25/boeing-e-irobot-juntan-esfuerzos-para-crear-un-robot-de-reconocimiento/

Parker, L. E. (1998). ALLIANCE: An Architecture for Fault Tolerant. *IEEE Transactions on Robotics and Automation* , 220-240.

Parker, L. E. (2003). Current research in Multirobot Systems. *Artif Life Robotics* .

Power House Museum. (2008). Recuperado el 01 de 06 de 2008, de http://www.powerhousemuseum.com/freeradicals/?m=200711

Purnamadjaja, A., Iskandar, J., & Russell, R. (2007). Pheromone Communication Simulation for Mobile Robots Using Java 3D. *ICIS 2007. 6th IEEE/ACIS International Conference*, (págs. 261 - 266). 2007.

Quintero, A., Rueda Rodriguez, S., & Ucrós, M. E. (s.f.). *Grupo de Investigación HIDRA, Universidad de los Andes*. Recuperado el 07 de 07 de 2008, de http://agamenon.uniandes.edu.co/yubarta/agentes/agentes.htm

Robocup. (2008). Recuperado el 01 de 06 de 2008, de www.robocup.org/

Robowatch. (2008). Recuperado el 23 de 06 de 2008, de http://www.robowatch.de/

Roboworld. (2008). Recuperado el 04 de 07 de 2008, de http://www.roboworld.com.sg/roboshop/product_list.aspx?Category=Robot+Eyes+n+Sensors

Sabbatini, R. (1999). *Brain and Main.* Recuperado el 02 de 07 de 2008, de http://www.cerebromente.org.br/n09/historia/robots_i.htm

Sahin, E. (2007). *Swarm Robotics*. Recuperado el 01 de 08 de 2008, de http://www.swarm-robotics.org/

Sahin, E. (2005). *Swarm Robotics: From Sources of Inspiration to Domain Application.* Ankara: Middle East Technical University.

Schenato, L. (2007). Recuperado el 23 de 06 de 2008, de http://www.dei.unipd.it/~schenato/

Steels, L. (1990). Cooperation between distributed agents through self-organization. *Yves Demazeau and Jean-Pierre Muller editors,Decentralized A.I. Elsevier Science* .

Stellman, A., & Greene, J. (2005). Applied Software Project Management. Cambridge, MA: O'Reilly Media.

Sugawara, K., Kazama, T., & Watanabe, T. (2004). Foraging behavior of interacting robots with virtual pheromone. *2004 IEEE/RSJ International Conference, Intelligent Robots and Systems.*

Theraulaz, G., Goss, S., Gervet, J., & Deneubourg, J.-L. (1990). Task diferentiation in Polistes wasp colonies: a model for self-organizing groups of robots. *In Proceedings of the First International Conference on Simulation of Adaptive Behavior 1990* , 346-355.

Tienda Digitus. (2008). Recuperado el 12 de 02 de 2009, de http://www.tiendadigitus.com/

Turner, R. H. (1967). *Robert E. Park: On Social Control and Collective Behavior.* Chicago: University of Chicago Press. (an anthology of Park's writings).

Verret, S. (2005). *Current State of the Art in Multirobot System.* Defense Research and Development Canada Suffield.

Wang, J., & Beni, G. (1990). Distributed computing problems in cellular robotic systems.

Wiegers, K. E. (2003). Software Requirements 2: Practical techniques for gathering and managing requirements throughout the product development cycle. Redmond: Microsoft Press.

Williams, O. (2002). *System, Guidance for a Robot Mobile using Computer Vision over a Distributed.* Recuperado el 11 de 11 de 2008, de http://research.microsoft.com/users/olliew/pubs/meng.pdf

Zebrowski, P. (2004). *Communication In Multi-Robot Systems* . Recuperado el 08 de 07 de 2008, de http://www.sfu.ca/~pzebrows/cmpt816/implicit.htm

Anexo 1. Datos de entrenamiento de la red neuronal

t	T	n	r	ts	Pr. búsqueda
57	60	10	0	37	0,99
178	60	10	1	36	0,99
249	60	10	0	71	0,945997482
123	60	8	1	43	0,99
189	60	8	0	66	0,966733277
213	60	8	1	68	0,99
390	60	8	1	152	0,99
530	60	8	0	140	0,707128669
710	60	8	0	173	0,330830204
100	60	6	1	80	0,99
166	60	6	1	27	0,99
280	60	6	1	30	0,99
427	60	6	0	145	0,690881348
103	60	6	0	101	0,833137691
87	60	4	1	11	0,99
156	60	4	1	39	0,99
281	60	4	0	125	0,752902531
408	60	4	1	94	0,76352388
482	60	4	0	74	0,707055879
559	60	4	0	77	0,638284934
66	60	2	1	33	0,99
194	60	2	1	28	0,99
276	60	2	1	56	0,99
407	60	2	0	131	0,734997065
483	60	2	1	15	0,801886697
557	60	2	1	18	0,858066473
622	60	2	1	41	0,882279548
672	60	2	1	31	0,914589725
143	30	10	1	14	0,99
353	30	10	1	12	0,99
407	30	10	0	33	0,966936784
478	30	10	0	70	0,681976875
602	30	10	0	124	0,137147316
618	30	10	0	16	0,137147316
125	30	8	1	11	0,99
462	30	8	1	274	0,99
522	30	8	0	60	0,770424976
582	30	8	0	59	0,558364977
642	30	8	0	60	0,338789953

704	30	8	0	62	0,106315638
48	30	4	0	48	0,85358179
104	30	4	1	25	0,89366195
156	30	4	1	26	0,93242164
542	30	4	0	386	0,01
133	30	2	1	33	0,99
208	30	2	0	75	0,679296799
220	30	2	0	12	0,679296799
311	30	2	1	29	0,714384518
373	30	2	1	33	0,744550431
438	30	2	1	29	0,778680123
517	30	2	1	38	0,804687926
575	30	2	0	56	0,612574722
667	30	2	0	92	0,21082266
719	30	2	0	51	0,047137612
373	15	10	1	296	0,99
50	15	6	0	46	0,585512242
103	15	6	0	53	0,116774226
193	15	6	1	19	0,168188108
251	15	6	0	58	0,01
47	15	4	0	45	0,592500571
84	15	4	1	9	0,708107507
132	15	4	1	11	0,795064029
159	15	4	0	26	0,621334317
223	15	4	0	62	0,077454019
55	15	2	0	53	0,524370069
106	15	2	1	31	0,557049808
264	15	2	0	157	0,01
178	80	5	1	36	0,99
251	80	5	0	72	0,99
342	80	5	1	38	0,99
452	80	5	1	82	0,99
602	80	5	0	150	0,793648134
739	80	5	1	50	0,813809424
103	60	10	1	32	0,99
199	60	10	0	96	0,851964091
222	60	10	0	23	0,851964091
515	60	10	1	31	0,88464383
609	60	10	0	91	0,76365442
112	30	10	1	11	0,99
189	30	10	0	77	0,668186244
338	30	10	1	93	0,678921617
98	15	10	0	93	0,261398606
174	15	10	0	76	0,01
62	60	10	1	9	0,99

152	60	10	1	30	0,99
272	60	10	0	120	0,770099867
106	60	8	1	51	0,99
282	60	8	1	140	0,99
92	60	8	1	37	0,99
155	60	8	1	37	0,99
396	60	8	0	241	0,459521275
593	60	8	0	194	0,03061135
78	60	6	1	24	0,99
271	60	6	1	17	0,99
361	60	6	0	90	0,871394471
414	60	6	1	23	0,914684514
159	60	6	1	30	0,99
236	60	6	1	48	0,99
291	60	6	1	32	0,99
359	60	6	0	68	0,956611779
81	60	4	0	81	0,907747176
227	60	4	1	25	0,947747176
348	60	4	0	121	0,7234713
428	60	4	1	38	0,749479102
714	60	4	1	58	0,766825156
84	60	2	0	84	0,894460481
183	60	2	1	68	0,909209744
238	60	2	1	35	0,937699772
331	60	2	1	42	0,961537674
348	60	2	0	17	0,961537674
449	60	2	1	45	0,983612729
528	60	2	1	16	0,99
541	60	2	0	12	0,99
580	60	2	0	38	0,99
662	60	2	0	82	0,901459304
295	30	10	1	84	0,99
489	30	10	0	193	0,246767741
594	30	10	1	16	0,31005888
158	30	8	1	24	0,99
265	30	8	1	47	0,99
470	30	8	1	94	0,99
550	30	8	0	74	0,681598288
560	30	8	0	7	0,681598288
41	30	4	1	14	0,99
143	30	4	1	30	0,99
572	30	4	1	263	0,99
619	30	4	0	47	0,860808374
720	30	4	1	32	0,892453944
75	30	2	1	28	0,99

116	30	2	0	41	0,905455674
238	30	2	1	48	0,926310731
307	30	2	0	69	0,649139201
406	30	2	0	98	0,214429053
491	30	2	1	31	0,246896585
618	30	2	0	126	0,01
140	15	10	0	50	0,547105862
380	15	10	0	238	0,01
38	15	6	0	38	0,676144751
124	15	6	0	84	0,01
68	15	4	0	68	0,402714173
137	15	4	1	5	0,591393419
210	15	4	1	6	0,746432178
297	15	4	0	87	0,04964106
48	15	2	0	47	0,574206946
97	15	2	0	49	0,139261309
144	15	2	0	46	0,01
208	80	5	1	80	0,99
322	80	5	1	95	0,99
111	60	10	1	14	0,99
258	60	10	0	145	0,69292857
332	60	10	0	73	0,638428309
509	60	10	0	177	0,252204455
598	60	10	0	87	0,146546777
707	60	10	0	98	0,01
287	60	8	0	287	0,379003606
110	60	8	1	15	0,99
266	60	8	1	56	0,99
368	60	8	1	25	0,99
624	60	8	0	248	0,447261428
83	60	6	1	40	0,99
252	60	6	1	67	0,99
331	60	6	0	79	0,914656156
463	60	6	0	132	0,656555525
73	60	6	1	24	0,99
136	60	6	1	44	0,99
254	60	6	1	43	0,99
387	60	6	1	63	0,99
139	60	4	1	55	0,99
243	60	4	1	26	0,99
328	60	4	0	72	0,942022446
390	60	4	1	11	0,99
473	60	4	0	83	0,900131061
611	60	4	1	44	0,922628249
696	60	4	0	82	0,834087553

211	30	10	1	23	0,99
261	30	10	0	50	0,833313363
319	30	10	0	58	0,627515979
388	30	10	0	68	0,356393187
475	30	10	0	87	0,01
56	30	8	1	16	0,99
182	30	8	1	29	0,99
256	30	8	1	25	0,99
450	30	8	1	169	0,99
504	30	8	1	27	0,99
574	30	8	0	67	0,721920532
78	30	4	1	28	0,99
149	30	4	0	70	0,70474222
213	30	4	0	64	0,458324803
272	30	4	0	58	0,249225126
328	30	4	0	54	0,068297088
118	15	10	1	19	0,99
191	15	10	1	15	0,99
66	15	6	1	33	0,99
39	15	4	1	11	0,99
91	15	4	0	51	0,536074427
162	15	4	0	71	0,01
116	80	5	1	41	0,99
238	80	5	1	31	0,99
296	80	5	1	34	0,99
526	80	5	0	229	0,616860501
643	80	5	0	115	0,512414732
130	60	10	1	20	0,99
251	60	10	0	109	0,804817052
230	60	8	1	37	0,99
124	60	8	1	108	0,99
232	60	8	1	72	0,99
310	60	8	0	78	0,918516928
411	60	8	1	67	0,933554522
478	60	8	1	36	0,961487483
546	60	8	0	67	0,930922662
645	60	8	0	99	0,781647394
68	60	6	1	44	0,99
217	60	6	1	35	0,99
387	60	6	1	33	0,99
53	60	6	1	13	0,99
109	60	6	1	35	0,99
276	60	6	1	42	0,99
379	60	6	0	102	0,828408355
76	60	4	0	66	0,965513123

145	60	4	1	37	0,99
232	60	4	0	87	0,885087918
329	60	4	1	43	0,908479731
411	60	4	1	32	0,939926272
495	60	4	1	26	0,978912626
651	60	4	0	155	0,650770584
228	30	10	1	114	0,99
347	30	10	0	119	0,465724361
421	30	10	1	34	0,495179442
528	30	10	1	42	0,519189046
48	30	8	0	48	0,852502987
119	30	8	0	66	0,591779443
136	30	8	0	17	0,591779443
234	30	8	0	93	0,18358134
429	30	8	1	51	0,203344186
65	30	4	0	58	0,782219589
196	30	4	0	130	0,215913327
238	30	4	0	42	0,117363959
371	30	4	1	71	0,131518241
505	30	4	0	134	0,01
259	15	10	1	38	0,99
72	15	6	1	11	0,99
235	15	4	1	197	0,99
284	15	4	0	49	0,560733255
336	15	4	1	33	0,590763285
385	15	4	1	34	0,620480967
437	15	4	0	51	0,165191912
493	15	4	1	28	0,200779101
539	15	4	1	6	0,377770251
593	15	4	1	29	0,412134512
645	15	4	1	28	0,447912688
694	15	4	0	49	0,015799398
38	80	5	1	11	0,99
166	80	5	1	28	0,99
242	80	5	1	28	0,99
353	80	5	1	94	0,99
407	80	5	1	21	0,99
539	80	5	1	119	0,99
669	80	5	0	129	0,847754372
75	60	10	1	49	0,99
305	60	10	0	229	0,484623539
485	60	10	1	76	0,497851052
684	60	10	0	198	0,061025037
81	60	8	1	41	0,99
272	60	8	0	188	0,57676845

128	60	8	1	45	0,99
228	60	8	1	53	0,99
406	60	8	0	178	0,601733632
600	60	8	0	194	0,174372505
734	60	8	1	55	0,192670767
53	60	6	1	26	0,99
127	60	6	1	21	0,99
239	60	6	1	23	0,99
419	60	6	0	179	0,598682598
86	60	6	1	53	0,99
196	60	6	1	46	0,99
331	60	6	0	130	0,737640274
399	60	6	1	24	0,778792538
212	30	10	1	112	0,99
270	30	10	0	58	0,783210483
395	30	10	0	123	0,244682744
463	30	10	0	65	0,01
316	30	8	0	316	0,082550578
185	15	10	1	80	0,99
266	15	10	0	81	0,325097638
60	15	6	1	26	0,99
115	15	6	0	55	0,502562137
174	15	6	0	45	0,109092796
278	15	6	0	104	0,01
78	80	5	1	11	0,99
159	80	5	1	43	0,99
310	80	5	1	53	0,99
370	80	5	1	43	0,99
515	80	5	0	144	0,80885869
635	80	5	0	120	0,691079855
698	80	5	1	45	0,713252804
121	60	10	0	118	0,776627861
202	60	10	1	20	0,825527617
275	60	10	1	33	0,855830647
467	60	10	0	184	0,453206775
616	60	10	0	146	0,15117386
728	60	10	0	112	0,01
40	60	8	1	9	0,99
304	60	8	1	21	0,99
45	60	8	1	14	0,99
117	60	8	1	34	0,99
170	60	8	1	24	0,99
262	60	8	0	86	0,886207477
433	60	8	0	171	0,516965272
498	60	8	1	27	0,554071024

189	60	6	1	43	0,99
446	60	6	1	88	0,99
267	60	6	1	124	0,99
366	60	6	1	41	0,99
69	30	10	0	44	0,882852993
236	30	10	0	162	0,216696368
469	30	10	1	19	0,268509839
540	30	10	0	71	0,01
52	30	8	0	52	0,825270211
210	30	8	1	48	0,845974145
369	30	8	1	37	0,873037745
480	30	8	0	110	0,387311272
537	30	8	0	57	0,188159896
44	15	10	0	41	0,634036421
333	15	10	1	28	0,670268305
86	15	6	1	72	0,99
183	15	6	1	34	0,99
216	15	6	0	33	0,733291735
309	15	6	1	64	0,7488802
390	15	6	1	54	0,767364488
412	15	6	0	21	0,66544101
507	15	6	0	95	0,01
50	60	10	1	12	0,99
362	60	10	1	46	0,99
475	60	10	0	84	0,895025945
792	60	10	1	71	0,909190251
93	60	8	0	90	0,872680775
278	60	8	1	56	0,890617995
83	60	8	1	47	0,99
418	60	8	0	333	0,310686372
633	60	8	0	215	0,01
174	30	10	1	37	0,99
254	30	10	0	80	0,646773107
314	30	10	0	60	0,427848709
549	30	10	0	229	0,01
95	30	8	1	20	0,99
151	30	8	0	48	0,852143686
201	30	8	0	49	0,702868418
443	30	8	1	49	0,723090865
530	30	8	0	85	0,357274535
669	30	8	0	130	0,01
60	15	10	0	60	0,463154487
216	15	10	1	114	0,471891927
261	15	10	0	44	0,085540216
88	60	10	1	30	0,99

146	60	10	1	34	0,99
230	60	10	1	48	0,99
300	60	10	0	70	0,948789878
373	60	10	0	72	0,901407524
477	60	10	0	70	0,861396639
68	60	8	1	33	0,99
298	60	8	1	35	0,99
162	60	8	1	55	0,99
304	60	8	0	139	0,709673002
374	60	8	0	69	0,673068107
470	60	8	0	93	0,544602457
643	60	8	0	171	0,175360252
95	30	10	0	95	0,573720309
200	30	10	1	16	0,635639814
498	30	10	0	293	0,01
171	30	8	1	39	0,99
308	30	8	1	61	0,99
498	30	8	0	190	0,253377565
716	30	8	1	32	0,284385317
26	15	10	0	21	0,893330612
169	15	10	1	43	0,916613499
71	60	10	1	17	0,99
161	60	10	1	72	0,99
342	60	10	1	27	0,99
552	60	10	0	206	0,534370608
646	60	10	0	90	0,418339565
187	30	10	1	6	0,99
286	30	10	1	31	0,99
368	30	10	1	21	0,99
616	30	10	0	247	0,153313546
160	15	10	0	149	0,09699213

Anexo 2. Pesos de conexiones red neuronal

origen	destino	peso
Bias	Capa escondida 1 neurona 1	-2,880144029
Bias	Capa escondida 1 neurona 2	15,94330557
Bias	Capa escondida 1 neurona 3	8,326188772
Bias	Capa escondida 1 neurona 4	-4,311880559
Bias	Capa escondida 1 neurona 5	19,14395563
Bias	Capa escondida 1 neurona 6	5,503387699
Bias	Capa escondida 1 neurona 7	15,33756459
Bias	Capa escondida 1 neurona 8	-27,73972385
Bias	Capa escondida 2 neurona 1	-7,914315989
Bias	Capa escondida 2 neurona 2	12,54774492
Bias	Capa escondida 2 neurona 3	4,915285118
Bias	Capa escondida 2 neurona 4	-1,30082889
Bias	Neurona de Salida	-15,21836722
Neurona de entrada 1	Capa escondida 1 neurona 1	14,96070822
Neurona de entrada 1	Capa escondida 1 neurona 2	-14,17686471
Neurona de entrada 1	Capa escondida 1 neurona 3	-36,17417368
Neurona de entrada 1	Capa escondida 1 neurona 4	2,502224214
Neurona de entrada 1	Capa escondida 1 neurona 5	-20,46913111
Neurona de entrada 1	Capa escondida 1 neurona 6	-35,33015922
Neurona de entrada 1	Capa escondida 1 neurona 7	34,60565034
Neurona de entrada 1	Capa escondida 1 neurona 8	13,68858803
Neurona de entrada 2	Capa escondida 1 neurona 1	-45,27129903
Neurona de entrada 2	Capa escondida 1 neurona 2	12,67200381
Neurona de entrada 2	Capa escondida 1 neurona 3	22,17487704
Neurona de entrada 2	Capa escondida 1 neurona 4	-6,337828796
Neurona de entrada 2	Capa escondida 1 neurona 5	-19,62222141
Neurona de entrada 2	Capa escondida 1 neurona 6	19,79465738
Neurona de entrada 2	Capa escondida 1 neurona 7	-13,48844513
Neurona de entrada 2	Capa escondida 1 neurona 8	3,080019848
Neurona de entrada 3	Capa escondida 1 neurona 1	-21,46137835
Neurona de entrada 3	Capa escondida 1 neurona 2	-7,072369864
Neurona de entrada 3	Capa escondida 1 neurona 3	8,913012329
Neurona de entrada 3	Capa escondida 1 neurona 4	2,42519253
Neurona de entrada 3	Capa escondida 1 neurona 5	-14,23278701
Neurona de entrada 3	Capa escondida 1 neurona 6	4,722856018
Neurona de entrada 3	Capa escondida 1 neurona 7	-19,10164939
Neurona de entrada 3	Capa escondida 1 neurona 8	14,39776984
Neurona de entrada 4	Capa escondida 1 neurona 1	1,10546932
Neurona de entrada 4	Capa escondida 1 neurona 2	-11,22027373
Neurona de entrada 4	Capa escondida 1 neurona 3	-0,998453516

origen	destino	peso
Neurona de entrada 4	Capa escondida 1 neurona 4	-16,70038419
Neurona de entrada 4	Capa escondida 1 neurona 5	-18,62710632
Neurona de entrada 4	Capa escondida 1 neurona 6	4,361892969
Neurona de entrada 4	Capa escondida 1 neurona 7	-10,72344726
Neurona de entrada 4	Capa escondida 1 neurona 8	4,252356869
Neurona de entrada 5	Capa escondida 1 neurona 1	-7,453659324
Neurona de entrada 5	Capa escondida 1 neurona 2	-2,36622287
Neurona de entrada 5	Capa escondida 1 neurona 3	-1,088370842
Neurona de entrada 5	Capa escondida 1 neurona 4	15,99068345
Neurona de entrada 5	Capa escondida 1 neurona 5	11,43831174
Neurona de entrada 5	Capa escondida 1 neurona 6	-22,44161905
Neurona de entrada 5	Capa escondida 1 neurona 7	-32,31357766
Neurona de entrada 5	Capa escondida 1 neurona 8	-5,514000894
Capa escondida 1 neurona 1	Capa escondida 2 neurona 1	8,341692598
Capa escondida 1 neurona 1	Capa escondida 2 neurona 2	0,605055197
Capa escondida 1 neurona 1	Capa escondida 2 neurona 3	-4,589461785
Capa escondida 1 neurona 1	Capa escondida 2 neurona 4	-12,01717695
Capa escondida 1 neurona 2	Capa escondida 2 neurona 1	6,3951654
Capa escondida 1 neurona 2	Capa escondida 2 neurona 2	-7,38567688
Capa escondida 1 neurona 2	Capa escondida 2 neurona 3	-18,42250105
Capa escondida 1 neurona 2	Capa escondida 2 neurona 4	8,072949262
Capa escondida 1 neurona 3	Capa escondida 2 neurona 1	-0,976552241
Capa escondida 1 neurona 3	Capa escondida 2 neurona 2	-2,370366461
Capa escondida 1 neurona 3	Capa escondida 2 neurona 3	-7,834637525
Capa escondida 1 neurona 3	Capa escondida 2 neurona 4	-0,888151511
Capa escondida 1 neurona 4	Capa escondida 2 neurona 1	-15,00019865
Capa escondida 1 neurona 4	Capa escondida 2 neurona 2	-10,24013654
Capa escondida 1 neurona 4	Capa escondida 2 neurona 3	-9,917910748
Capa escondida 1 neurona 4	Capa escondida 2 neurona 4	-3,583367493
Capa escondida 1 neurona 5	Capa escondida 2 neurona 1	-3,317234082
Capa escondida 1 neurona 5	Capa escondida 2 neurona 2	-1,190054492
Capa escondida 1 neurona 5	Capa escondida 2 neurona 3	10,43748283
Capa escondida 1 neurona 5	Capa escondida 2 neurona 4	-1,961514927
Capa escondida 1 neurona 6	Capa escondida 2 neurona 1	1,376147165
Capa escondida 1 neurona 6	Capa escondida 2 neurona 2	-5,466489065
Capa escondida 1 neurona 6	Capa escondida 2 neurona 3	9,962175238
Capa escondida 1 neurona 6	Capa escondida 2 neurona 4	0,886538381
Capa escondida 1 neurona 7	Capa escondida 2 neurona 1	-0,619183558
Capa escondida 1 neurona 7	Capa escondida 2 neurona 2	-8,304271221
Capa escondida 1 neurona 7	Capa escondida 2 neurona 3	-0,802124476
Capa escondida 1 neurona 7	Capa escondida 2 neurona 4	-1,321033444
Capa escondida 1 neurona 8	Capa escondida 2 neurona 1	9,338180986
Capa escondida 1 neurona 8	Capa escondida 2 neurona 2	-7,893817128
Capa escondida 1 neurona 8	Capa escondida 2 neurona 3	3,703246242
Capa escondida 1 neurona 8	Capa escondida 2 neurona 4	-5,70795643

origen	destino	peso
Capa escondida 2 neurona 1	Neurona de Salida	11,03738232
Capa escondida 2 neurona 2	Neurona de Salida	6,325007347
Capa escondida 2 neurona 3	Neurona de Salida	10,27598324
Capa escondida 2 neurona 4	Neurona de Salida	16,38745741

Anexo 3. Algoritmos de los módulos de acción de los comportamientos de un robot

A continuación se presentan los módulos de acción de cada uno de los comportamientos de los robots del sistema propuesto. Cada uno de estos comportamientos realizan una acción a través de los actuadores del robot como por ejemplo moverse cierta distancia con un determinado ángulo (*función mov()*) o abrir o cerrar el brazo del robot para capturar o entregar un objeto.

Avoidance: Evasión de Obstáculos

- **Evadir un obstáculo detectado por los sensores de contacto**

```
if (robot.getSensors().getBumpers().oneHasHit()) { //Si el robot ha golpeado con algo

   int hitLeft=robot.getSensors().getBumpers().getRightQuadrantHits(); //Leer el cuadrante izquierdo
   int hitRight=robot.getSensors().getBumpers().getLeftQuadrantHits(); //Leer el cuadrante derecho
   int hitFront=robot.getSensors().getBumpers().getFrontQuadrantHits();//Leer el cuadrante frontal
   if((hitLeft > 0) || (hitRight > 0) || (hitFront >0))
   {   if(hitLeft>0&&hitRight>0)
           {
            if(Math.random()<0.5){
             translationalVelocity =2.0;
             rotationalVelocity = 0.0;
            }

            else{
             translationalVelocity =-1.0;
             rotationalVelocity = 0.0;
            }

           }else{

             if(hitRight>0)
                   {
                    translationalVelocity =0.0;
                    rotationalVelocity = -Math.PI/4;
                   }

                   if(hitLeft>0)
                   {
                    translationalVelocity =0.0;
                    rotationalVelocity = Math.PI/4;
                   }
           }

           if(hitFront>0)
           {
            translationalVelocity =-0.5;
            rotationalVelocity = Math.PI/2;

           }
   }
   else {
      rotationalVelocity = 0;
      translationalVelocity = 0.6;
   }

}

mov(rotationalVelocity,translationalVelocity);
```

Algoritmo 1. Primera parte módulo acción de *Avoidance*

- **Evadir un obstáculo detectado por los sensores de sonar**

```
if (sonars.oneHasHit()) {
        // Leer los tres cuadrantes de los sensores de sonar
        double left = sonars.getFrontLeftQuadrantMeasurement();
        double right = sonars.getFrontRightQuadrantMeasurement();
        double front = sonars.getFrontQuadrantMeasurement();
        // Si el obstáculo está cerca
        if ((front < 0.7) || (left < 0.7) || (right < 0.7)) {
           double maxRotationalVelocity = Math.PI / 4;
           if (left < right)
              rotationalVelocity = -maxRotationalVelocity
                    - (rotationalVelocityFactor * Math.random());
           else
              rotationalVelocity = maxRotationalVelocity
                    - (rotationalVelocityFactor * Math.random());
           translationalVelocity = 0;
        } else {
           rotationalVelocity = 0;
           translationalVelocity = 0.6;
        }
     }
mov(translationalVelocity, rotationalVelocity);
```

Algoritmo 2. Segunda parte módulo acción de *Avoidance*

GripObject: Atrapar Objeto

```
double translationalVelocity = 0.3;
double rotationalVelocity = 0;

 openGrip(); //Abrir el brazo
 closeGrip(); //Cerrar Brazo
 busyGrip(true);//Brazo ocupado

mov(translationalVelocity, rotationalVelocity);
```

Algoritmo 3. Módulo de acción de *GripObject*

Homing: Regresar a casa con un objeto.

```
//Obtener los valores de luminancia de los sensores
  float llum = getLeftLuminance();
  float rlum = getRightLuminance();

//Mover el robot  x distancia y theta grados

  mov(x, (llum - rlum) *  Math.PI/4);
```

Algoritmo 4. Módulo de acción de *Homing*

ComeBack: Regresar a casa sin un objeto.

```
//Obtener los valores de luminancia de los sensores
  float llum = getLeftLuminance();
  float rlum = getRightLuminance();

//Mover el robot  x distancia y theta grados

  mov(x, (llum - rlum) *  Math.PI/4);
```

Algoritmo 5. Módulo de acción de *ComeBack*

Deliver: Entrega de un objeto.

```
robot.OBJECT_IN_GRIPPER=false;
robot.OBJECT_FOUND=false;
robot.GOAL_REACHED=false;

openGrip();
mov(0.0,PI/2);
closeGrip();
busyGrip(false);

robot.setProbability();
mov(0,0);
```

Algoritmo 6. Módulo de acción de *Deliver*

Rest: Entrar en estado de reposo después de llegar a la base sin un objeto.

```
robot.OBJECT_IN_GRIPPER=false;
robot.OBJECT_FOUND=false;
robot.GOAL_REACHED=false;
robot.applyStigmergy(this);
robot.TIME_SEARCH=0.0;

if(robot.PROBABILITY_SEARCH>0.1)
  {
      if(Math.random()<0.5)
          robot.INIT_DIRECTION=-1;
        else
          robot.INIT_DIRECTION=1;

    robot.INIT_ANGLE=Math.PI*Math.random();
    robot.rotate(robot.INIT_DIRECTION*robot.INIT_ANGLE);

  }

 mov(0,0);
```

Algoritmo 7. Módulo de acción de *Rest*

Wandering: Exploración del ambiente en búsqueda de objetos

```
double WANDERING_PROBABILITY = 0.01;
int direction;

//Determinar el tiempo de busqueda de un objeto
//que es=Tiempo de Vida del robot-Tiempo inicial de la busqueda

robot.TIME_SEARCH=robot.getLifeTime()-robot.TIME_INIT_SEARCH;

if(Math.random()>0.5)
 direction=-1;
else
 direction=1;

if((random.nextDouble() < WANDERING_PROBABILITY))
    mov(0.8, direction*random.nextDouble() * 2 * Math.PI);
else
    mov(0.8, 0.0);
```

Algoritmo 8. Módulo de acción de *Wandering*