UNIVERSIDAD POLITECNICA DE VALENCIA

Máster de automática e informática industrial.

Departamento de Ingeniería de Sistemas y Automática (DISA)

Departamento de Informática de Sistemas y Computadores (DISCA)

Tesina de Máster

Control distribuido y coordinación de robots mediante sistemas multiagente

Autor: Ángel Soriano Vigueras

Tutores: Ángel Valera Fernández

Marina Vallés Miquel

Instituto Universitario de Automática e Informática Industrial (ai2)

Universidad Politécnica de Valencia

Camino de Vera, s/n

46020 Valencia (España)

Índice:

1. Introducción

1.1 Introducción y justificación

En los últimos años, la investigación en el campo de la robótica móvil cooperativa entre distintos tipos de robots ha crecido notablemente. La heterogeneidad de los sistemas actuales formados por múltiples componentes de diferentes características conectados en red, sugiere el desarrollo de sistemas de control distribuido en los que las distintas funciones de control se implementen a diferentes niveles.

Cuando en este tipo de configuración se incluyen elementos dotados de movilidad (robots, vehículos, helicópteros,…), la capacidad de obtener y procesar información, así como la de generar acciones de control, es variable y en gran medida dependiente del tiempo. Es más, los recursos disponibles en cada momento por cada elemento para realizar las actividades que se le asignen, también variables, son limitados. No obstante, la idea de hacer uso de distintos tipos de robots, ofrece ventaja de poder aprovechar las funcionalidades específicas de cada robot, bien por su tamaño, su velocidad de movimiento, su sensorización, su capacidad de volar, etc. Por ello, es preciso definir una estructura de control jerarquizado que ayude al control y coordinación de forma que se asegure que las acciones de máxima prioridad se ejecuten a nivel local y que el resto (opcionales, refinamientos, relacionadas con la coordinación) se lleven a cabo en niveles superiores con menor prioridad.

Los sistemas jerárquicos tradicionales típicamente tienen una estructura rígida que les impide reaccionar de una manera ágil ante variaciones. Por otra parte, los sistemas heterárquicos tienen un buen desempeño ante cambios, y pueden auto-adaptarse continuamente a su entorno. No obstante, el control heterárquico no provee un sistema predecible y de alto rendimiento, especialmente en entornos heterogéneos complejos, donde los recursos son escasos y las decisiones actuales tienen serias repercusiones en el rendimiento futuro.

Estos conceptos han dado lugar a la teoría de Sistemas Holónicos. Un sistema Holónico es una organización altamente distribuida, donde la inteligencia se distribuye entre las entidades individuales. Se puede comparar con los sistemas distribuidos de la

primera generación, sin embargo, el elemento nuevo de estos sistemas es el hecho de que las entidades individuales trabajan juntas en jerarquías temporales (holarquías) para obtener un objetivo global.

En este trabajo se utilizan modelos basados en agentes para lograr la implementación de un modelo de cooperación o comunicación entre los distintos nodos de control locales, con el fin de lograr un objetivo común. Además se estudia cómo mejorar la odometría de los robots y métodos de localización y posicionamiento.

1.2 Objetivos

El principal objetivo de este proyecto es tratar de llegar a conocer en profundidad todas las opciones que ofrece el uso de sistemas holónicos para llevar a cabo el desarrollo de aplicaciones de coordinación y colaboración de robots.

Una lista de los objetivos que se han marcado en este proyecto:

- Estudio de las opciones de intercomunicación de los robots.
- Estudio de las arquitecturas basadas en sistemas holónicos.
- Investigar sobre las distintas estrategias de control cinemático de robots móviles.
- Analizar la calidad y la fidelidad de la odometría e intentar mejorar o suplir las posibles pérdidas de precisión.
- Calibración de parámetros estructurales de los robots para evitar posibles errores sistemáticos.
- Estudio del uso de sistemas de visión estáticos para localización y posicionamiento de robots.
- Analizar las posibilidades de captura y reconocimiento de formas en imágenes a través de Java.

- Estudio de la geometría de un señuelo que otorgue posición, orientación y distinción entre robots.
- Investigación sobre el posicionamiento desde sistemas de visión inestables o en movimiento, que ofrezcan fidelidad ante variaciones de orientación, altura y movimiento.
- Desarrollo de aplicaciones que hagan uso de los conocimientos adquiridos.

2. Desarrollo Teórico

2.1 Introducción a la robótica.

El hombre durante toda su historia ha convivido con el deseo de construir máquinas capaces de realizar tareas que facilitasen su trabajo. Desde hace cientos de años antes de Cristo, ya se intentaban crear dispositivos, denominados artefactos o máquinas, que tuvieran un movimiento sin fin y que no estuvieran controlados ni supervisados por personas.

Los primeros autómatas que aparecen en la historia son ingenios mecánicos, más o menos complicados, que desarrollaban una tarea de forma continua. Sin embargo, las primeras máquinas construidas no tenían una utilidad práctica, sino que su principal objetivo era entretener a sus dueños. Generalmente funcionaban por medio de movimientos ascendentes de aire o agua caliente. El vertido progresivo de un líquido o la caída de un peso provocaba rupturas de equilibrio en diversos recipientes provistos de válvulas; otros mecanismos se basaban en palancas o contrapesos. Mediante sistemas de este tipo se construían pájaros artificiales que podían “cantar" o "volar", o puertas que se abrían solas.

El origen de los autómatas se remonta al Antiguo Egipto, donde las estatuas de algunos de sus dioses despedían fuego por los ojos, poseían brazos mecánicos manejados por los sacerdotes del templo o emitían sonidos cuando los rayos del sol los iluminaba. Estos ingenios pretendían causar temor y respeto entre la gente del pueblo.

Sin embargo, los primeros datos descriptivos acerca de la construcción de un autómata aparecen en el siglo I. El matemático, físico e inventor griego Herón de Alejandría describe múltiples ingenios mecánicos en su libro *Los Autómatas*, por ejemplo aves que vuelan, gorjean y beben.

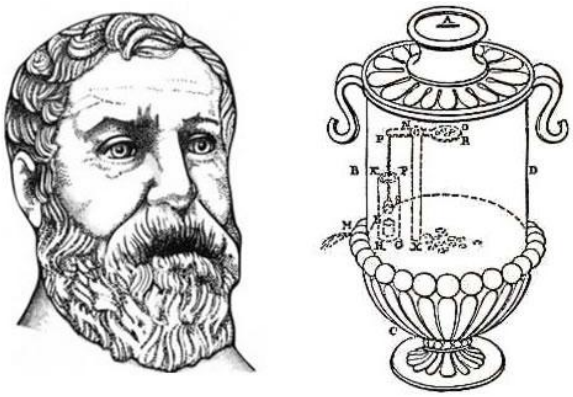


Figura 1: Herón junto a uno de sus inventos, un dispensador de agua sagrada operado por monedas

La Eolípila está considerada como la primera máquina térmica de la historia, pero lamentablemente durante mucho tiempo tan sólo se consideró un juguete sin mayor aplicación. Por otro lado se describen algunos ingenios que sí presentaban una función útil, como en el caso del dispensador de agua por monedas o el de un molino de viento para accionar un órgano (Hydraulis).

Figura 2: La Eolípila

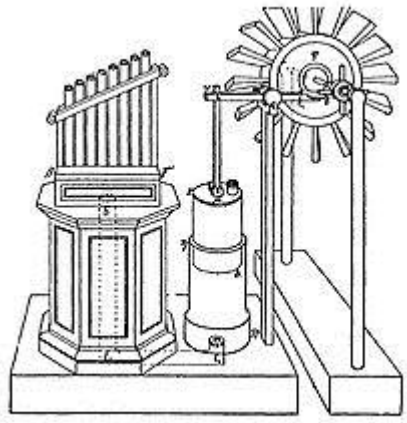


Figura 3: Hydraulis de Ctesibios

Las construcciones de la escuela de Alejandría se extendieron por todo el Imperio Romano y posteriormente por el mundo árabe, siendo estos genuinos aparatos los antecesores de los autómatas actuales.

La cultura árabe heredó y difundió los conocimientos griegos. Al-jazari, uno de los más grandes ingenieros de la historia, fue el creador de muchos inventos de control automático como el cigüeñal o uno de los primeros relojes mecánicos movidos por pesos y agua. Estuvo también muy interesado en la figura del autómata y escribió "El libro del conocimiento de los ingeniosos mecanismos", obra considerada de las más importantes sobre la historia de la tecnología. Cabe destacar su complejo reloj elefante, animado por seres humanos y animales mecánicos que se movían y marcaban las horas o un autómata con forma humana que servía distintos tipos de bebidas.

Figura4: Reloj elefante, creado por Al-jazari

Del siglo XIII son otros autómatas de los que no han llegado referencias suficientemente bien documentadas, como el *Hombre de Hierro* de Alberto Magno o la *Cabeza Parlante* de Roger Bacon. Otro ejemplo relevante de la época y que aún se conserva en la actualidad es el *Gallo de Estrasburgo*, situado en la catedral de esta misma ciudad, que mueve el pico y las alas al dar las horas.

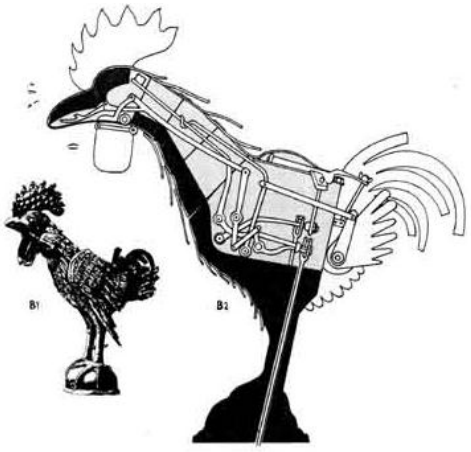


Figura 5: El Gallo de Estrasburgo

Durante los siglos XV y XVI algunas de las figuras más relevantes del Renacimiento también realizaron sus propias aportaciones a la historia de los autómatas. El más famoso quizá sea el *León Mecánico*, desarrollado por Leonardo Da Vinci para el rey Luis XII de Francia, que abría su pecho con la garra y mostraba el escudo de armas del rey. En España, Juanelo Turriano, inventor, arquitecto y relojero real del emperador Carlos V, construyó un autómata de madera con forma de monje conocido como el *Hombre de Palo*, construido con el fin de recolectar limosnas.

En 1738, Jacques de Vaucanson construyó uno de los autómatas más famosos de la historia, el *Pato con aparato digestivo*, considerado su obra maestra. El pato tenía más de 400 partes móviles, y podía batir sus alas, beber agua, digerir grano e incluso defecar. Los alimentos los digería por disolución y eran conducidos por unos tubos hacia el ano, donde había un esfínter que permitía evacuarlos. Vaucanson también construyó otros autómatas, entre los que destaca *El Flautista*, un pastor que tocaba la flauta capaz de tocar hasta 12 melodías diferentes. El ingenio consistía en un complejo mecanismo de aire que causaba el movimiento de todos las diferentes partes del engranaje interno del muñeco. El resultado era una música melodiosa, que tenía el mismo encanto y precisión en las notas que un flautista de carne y hueso.

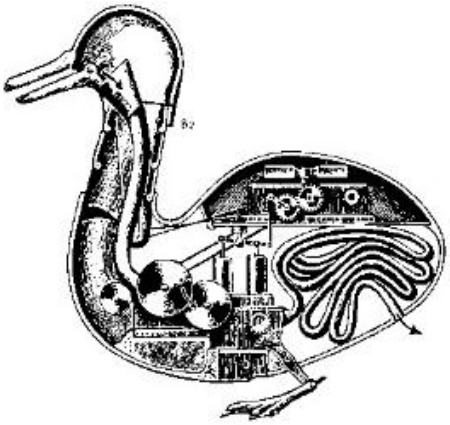


Figura 6: Pato con aparato digestivo

A finales del siglo XVIII y principios del XIX, en plena Revolución Industrial, se desarrollaron diversos ingenios mecánicos utilizados fundamentalmente en la industria textil. Entre ellos se puede citar la hiladora giratoria de Hargreaves (1770), la hiladora mecánica de Crompton (1779), el telar mecánico de Cartwright (1785), el telar de Jacquard (1801), que constituyó con sus tarjetas perforadas los primeros precedentes históricos de las máquinas de control numérico. Más tarde se incorporaron los automatismos en las industrias mineras y metalúrgicas. Pero sin duda, el automatismo que causó mayor impacto por tener un papel fundamental en la Revolución Industrial lo realiza Potter a principios del siglo XVIII, automatizando el funcionamiento de las válvulas en la máquina de vapor atmosférica, creada por el inventor inglés Thomas Newcomen.

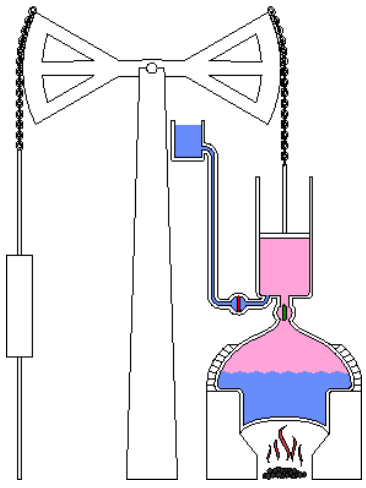


Figura 7: Máquina de vapor atmosférica.

Sin embargo, no fue hasta 1921 cuando se escuchó por primera vez la palabra robot, utilizada por el escritor checo Karel Capek en su obra de teatro *R.U.R (Rossum´s Universal Robots)*. La palabra robot viene del vocablo checo '*Robota*' que significa "trabajo", entendido como servidumbre, trabajo forzado o esclavitud.

Los primeros robots industriales empezaron a producirse a principios de los años 60 y estaban diseñados principalmente para realizar trabajos mecánicos difíciles y peligrosos. Las áreas donde estos robots tuvieron su aplicación fueron trabajos laboriosos y repetitivos, como la carga y descarga de hornos de fundición. En 1961 el inventor norteamericano George Devol patentaba el primer robot programable de la historia, conocido como *Unimate*, estableciendo las bases de la robótica industrial moderna.

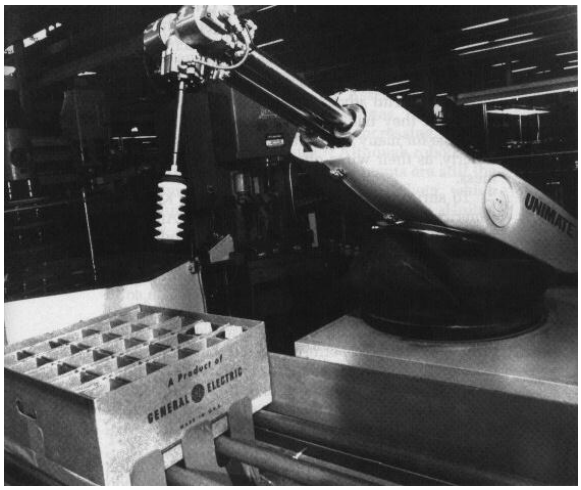


Figura 8: Unimate

Este robot industrial era un manipulador que formaba parte de una célula de trabajo en la empresa de automóviles Ford Motors Company, diseñado para levantar y apilar grandes piezas de metal caliente, de hasta 225 kg, de una troqueladora de fundición por inyección.

Debido a los continuos avances en la informática y la electrónica, a partir de 1970 fueron desarrollados diversos robots programables, siendo de gran importancia en la industria mecánica, tanto en las líneas de ensamblaje como en aplicaciones como la soldadura o pintura.

En los últimos años, los robots han tomado posición en todas las áreas productivas industriales. La incorporación del robot al proceso productivo ha representado uno de los avances más espectaculares de la edad moderna. En poco más de cuarenta años, se ha pasado de aquellos primeros modelos, rudos y limitados, a sofisticadas máquinas capaces de sustituir al hombre en todo tipo de tareas repetitivas o peligrosas, y además, hacerlo de forma más rápida, precisa y económica que el ser humano. Hoy en día, se calcula que el número de robots industriales instalados en el mundo es de un millón de unidades, unos 20.000 en España, siendo Japón el país más tecnológicamente avanzado, con una media de 322 robots por cada 10.000 trabajadores.

2.2 Tipos de robots

No resulta sencillo hacer una clasificación de tipos de robots, puesto que ningún autor se pone de acuerdo en cuántos y cuáles son los tipos de robots y sus características esenciales.

2.2.1 Robots Industriales

La creciente utilización de robots industriales en el proceso productivo, ha dado lugar al desarrollo de controladores industriales rápidos y potentes, basados en microprocesadores, así como un empleo de servos en bucle cerrado que permiten establecer con exactitud la posición real de los elementos del robot y su desviación o error. Esta evolución ha dado origen a una serie de tipos de robots, que se citan a continuación:

2.2.1.1 Manipuladores

Son sistemas mecánicos multifuncionales, con un sencillo sistema de control, que permite gobernar el movimiento de sus elementos de los siguientes modos:

- **Manual:** Cuando el operario controla directamente la tarea del manipulador.
- **De secuencia fija:** cuando se repite, de forma invariable, el proceso de trabajo preparado previamente.

- **De secuencia variable:** Se pueden alterar algunas características de los ciclos de trabajo

Existen muchas operaciones básicas que pueden ser realizadas de forma óptima mediante manipuladores. Por ello, estos dispositivos son utilizados generalmente cuando las funciones de trabajo son sencillas y repetitivas.

Figura 9: Robot manipulador

2.2.1.2 Robots de repetición o aprendizaje

Son manipuladores que se limitan a repetir una secuencia de movimientos, previamente ejecutada por un operador humano, haciendo uso de un controlador manual o un dispositivo auxiliar. En este tipo de robots, el operario durante la fase de enseñanza se vale de una pistola de programación con diversos pulsadores o teclas, o bien de joysticks, o bien utiliza un maniquí, o desplaza directamente la mano del robot. Actualmente, los robots de aprendizaje son los más conocidos en algunos sectores de la industria, y el tipo de programación que incorporan recibe el nombre de "*gestual*".

Figura 10: Robot Pingüino de aprendizaje

2.2.1.3 Robots con control por computador

Son manipuladores o sistemas mecánicos multifuncionales, controlados por un computador, que habitualmente suele ser un microordenador. El control por computador dispone de un lenguaje específico de programación, compuesto por varias instrucciones adaptadas al hardware del robot, con las que se puede diseñar un programa de aplicación utilizando solo el ordenador. A esta programación se le denomina "*textual*" y se crea sin la intervención del manipulador.

Las grandes ventajas que ofrece este tipo de robots, hacen que se vayan imponiendo en el mercado rápidamente, lo que exige la preparación urgente de personal cualificado, capaz de desarrollar programas de control que permitan el manejo del robot.

Figura 11: Robot FANUC

2.2.2 Robots Inteligentes

Son similares a los del grupo anterior, pero tienen la capacidad de poder relacionarse con el mundo que les rodea a través de sensores y de tomar decisiones en función de la información obtenida en tiempo real. De momento, son muy poco conocidos en el mercado y se encuentran en fase experimental, donde grupos de investigadores se esfuerzan por hacerlos más efectivos, al mismo tiempo que más económicamente asequibles. El reconocimiento de imágenes y algunas técnicas de inteligencia artificial son los campos que más se están estudiando para su posible aplicación en estos robots.

Figura 12: Robot ASIMO de Honda

2.2.3 Micro-robots

Con fines educacionales, de entretenimiento o investigación, existen numerosos robots de formación o micro-robots a un precio muy asequible, cuya estructura y funcionamiento son similares a los de aplicación industrial.

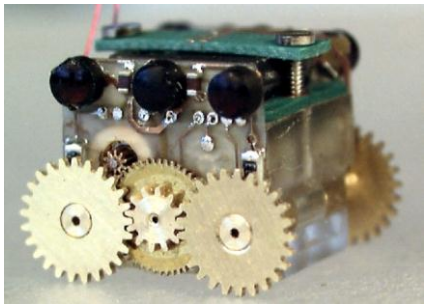


Figura 13: Robot Smoovy

Estos robots que un día se hicieron un hueco en universidades y centros de investigación, puesto que eran una forma económica de experimentar con múltiples tareas robóticas, hoy en día se pueden encontrar en centros docentes de todo tipo, incluidas escuelas de primaria e institutos. El personal de dichos centros ha apostado por este tipo de robots para estimular el interés de sus alumnos por la ciencia y la tecnología y los resultados son como se ha podido observar altamente satisfactorios.

Figura 14: Robot LEGO

2.3 Robótica Móvil

En el apartado anterior se ha realizado un desglose de los diferentes tipos de robots existentes atendiendo a su aplicación, pero más allá de este aspecto práctico hay otro hecho característico de los robots modernos que les confiere un mayor grado de libertad y utilidad. Esta característica es el movimiento en el espacio físico, es decir, la posibilidad de desplazarse por el entorno para observarlo e interactuar con él, y de esta forma emular con mayor fidelidad las funciones y capacidades de los seres vivos.

2.3.1 Clasificación de los robots móviles

De la misma forma que se ha descrito en el apartado anterior en base a diferentes criterios se puede establecer una taxonomía dentro del colectivo de los robots móviles. Si por ejemplo se atiende a sus características estructurales y funcionales se puede establecer la siguiente clasificación:

2.3.1.1 Robots rodantes

Son aquellos que, como su nombre indica, se desplazan haciendo uso de ruedas, generalmente montadas por pares en una configuración 2+2 como las de un vehículo por mera simplicidad. Habitualmente solo dos de sus ruedas presentan tracción y otras dos dirección, de forma que sea posible maniobrar el robot con un solo servomotor.

Figura 15: Robot rodante dotado de 4 ruedas en configuración 2+2

También es frecuente encontrar distribuciones de ruedas montadas en modo triciclo, donde una rueda sirve para la dirección y las otras dos aportan la tracción. Otra opción es que la tercera rueda simplemente sea una rueda 'loca' y las otras dos aporten tanto la tracción como la dirección, mediante el método de las orugas tratado más adelante.

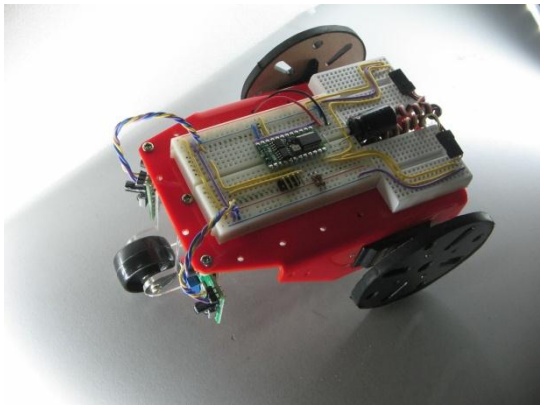


Figura 16: Robot rodante con las ruedas en configuración triciclo

Existen algunos casos especiales en los que se usan otras configuraciones que dotan al robot de mejor adaptación a terrenos difíciles. En estos casos los algoritmos de control de movimiento adquieren una mayor complejidad, proporcional al número de elementos direccionables de forma independiente.

Figura 17: Robot rodante de 6 ruedas con dirección independiente frontal y trasera

Por último cabría considerar a los robots con orugas como un tipo de robot rodante en el que se substituyen las ruedas por un mecanismo de oruga para la tracción. La dirección se consigue parando una de las orugas o haciéndolas girar en sentido contrario.

Figura 18: Robot rodante dotado de orugas

2.3.1.2 Robots andantes

Respecto a los robots construidos a imagen y semejanza humana, con dos piernas, las técnicas de control necesarias son varias, pero todas ellas hacen uso de compLeJOS algoritmos para poder mantener el equilibrio y caminar correctamente. Todos ellos son capaces de caminar bien sobre suelos planos y subir escaleras en algunos casos, pero no están preparados para caminar en suelos irregulares.

Algunos incluso pueden realizar tareas como bailar, luchar o practicar deportes, pero esto requiere una programación sumamente compleja que no siempre está a la altura del hardware del robot y de su capacidad de procesamiento.

Figura 19: Robot humanoide Robonova.

2.3.1.3 Robots reptadores

Una clase curiosa de robots, creados basándose en animales como las serpientes, su forma de desplazarse es también una imitación de la usada por estos animales. Están formados por un número elevado de secciones que pueden cambiar de tamaño o posición de forma independiente de las demás pero coordinada, de forma que en conjunto provoquen el desplazamiento del robot.

Figura 20: Robot reptador

2.3.1.4 Robots nadadores

Estos robots son capaces de desenvolverse en el medio acuático, generalmente enfocados a tareas de exploración submarina en zonas donde no es posible llegar pos ser de difícil acceso o estar a profundidades que el cuerpo humano no tolera.

Figura 21: Robot pez

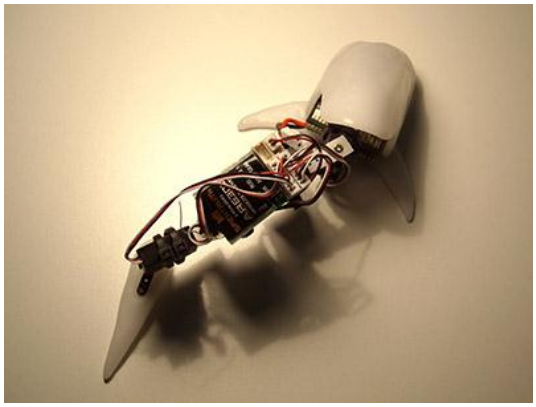


Figura 22: Robot ballena

Aparte de lo puramente anecdótico, se ha demostrado que la estructura corporal de los peces así como el movimiento que realizan durante su desplazamiento en el agua, es uno de los métodos más óptimos de movimiento submarino dado que aprovecha la energía de forma muy eficiente y permite mayor control en la navegación, produciendo mucho menos ruido y turbulencias.

Es por todo esto que se está tendiendo a estudiar y emular en lo posible el comportamiento de estos animales a la hora de crear nuevos robots subacuáticos.

Figura 23: Atún robótico

2.3.1.5 Robots voladores

Conquistados los dominios del mar y la tierra solo queda una meta por alcanzar en el mundo de la robótica, ser capaces de poner robots en el cielo. Para ello y por el momento existen dos aproximaciones, en función de su principio de vuelo y estructura:

- **Helicópteros:** habitualmente helicópteros RC convencionales a los que se les añade la electrónica necesaria para tener visión artificial y capacidad de toma de decisiones autónoma.

Figura 24: Helicóptero robótico

- **Drones o aviones no tripulados:** actualmente en uso por el ejército de EEUU para tareas de logística en operaciones militares, apoyo cartográfico así como tareas de espionaje. Aunque no es habitual pueden estar dotados tanto de contramedidas para repeler agresiones como de armamento para realizar ataques.

Figura 25: Drone (robot volador militar no tripulado)

2.3.2 Componentes de un robot móvil

Vistos los principales tipos de robots móviles que se construyen en la actualidad, a continuación se detallan las partes constituyentes de los robots, tanto estructurales, como mecánicas y electrónicas.

2.3.2.1 Estructura

La estructura es el esqueleto, el soporte fundamental que constituye tanto la forma como la funcionalidad del robot. Sirve de sujeción para toda la electrónica, sensores, actuadores y también es parte del aparato motriz como prolongación de los actuadores. Está formada generalmente por una mezcla de partes rígidas y flexibles, fijas y móviles y entre sus materiales destacan los plásticos, metales y aleaciones resistentes a la par que ligeras como la fibra de carbono o derivados del aluminio.

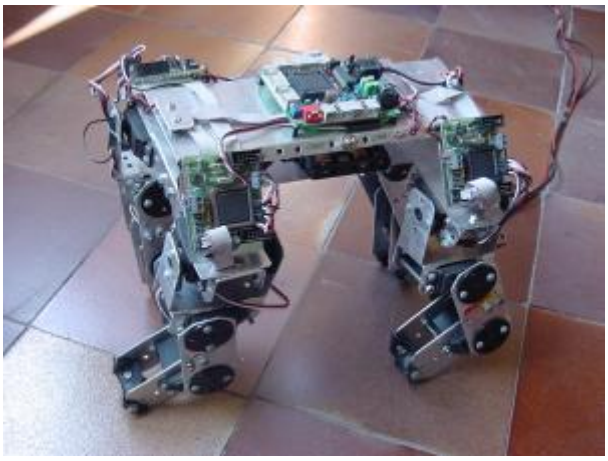


Figura 26: Estructura básica fabricada en aluminio para un robot cuadrúpedo.

Determina el medio en el que se va a poder desenvolver el robot, así como las actividades que será capaz de realizar. También protege partes sensibles de la electrónica de golpes, polvo, agua y otros agentes externos. Como ya se ha comentado, debe ser un compromiso entre resistencia y ligereza, adaptándose de la mejor manera posible al tipo de tareas para las que se diseña el robot que va a poseer dicha estructura.

2.3.2.2 Sensores

Estos elementos son los encargados de adquirir información del entorno y transmitirla a la unidad de control del robot. Una vez esta es analizada, el robot realiza la acción correspondiente a través de sus actuadores.

Los sensores constituyen el sistema de percepción del robot, es decir, facilitan la información del mundo real para que el robot la interprete. Los tipos de sensores más utilizados son los siguientes:

- **Sensor de proximidad:** Detecta la presencia de un objeto ya sea por rayos infrarrojos, por sonar, magnéticamente o de otro modo.

Figura 27: Sensores de proximidad

- **Sensor de Temperatura:** Capta la temperatura del ambiente, de un objeto o de un punto determinado.

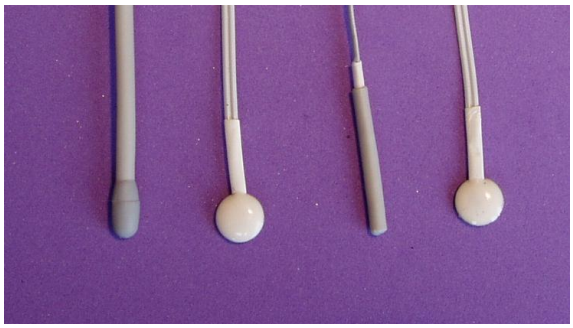


Figura 28: Sensores de temperatura

- **Sensor magnéticos:** Captan variaciones producidas en campos magnéticos externos. Se utilizan a modo de brújulas para orientación geográfica de los robots.

Figura 29: Sensores magnéticos

• **Sensores táctiles, piel robótica:** Sirven para detectar la forma y el tamaño de los objetos que el robot manipula. La piel robótica se trata de un conjunto de sensores de presión montados sobre una superficie flexible.

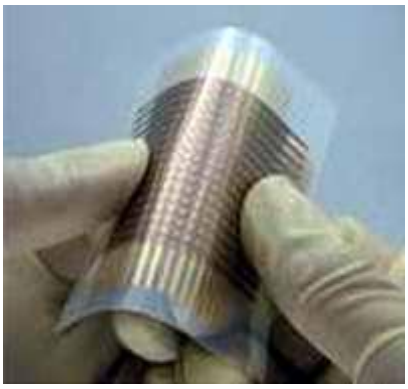


Figura 30: Sensor táctil flexible

• **Sensores de iluminación:** Capta la intensidad luminosa, el color de los objetos, etc. Es muy útil para la identificación de objetos. Es parte de la visión artificial y en numerosas ocasiones son cámaras.

Figura 31: Sensores de luz

• **Sensores de velocidad, de vibración (Acelerómetro) y de inclinación:** Se emplean para determinar la velocidad de actuación de las distintas partes móviles del propio robot o cuando se produce una vibración. También se detecta la inclinación a la que se encuentra el robot o una parte de él.

Figura 32: Sensores de velocidad de reluctacia variable

- **Sensores de fuerza:** Permiten controlar la presión que ejerce la mano del robot al coger un objeto.

Figura 33: Sensor de presión

- **Sensores de sonido:** Micrófonos que permiten captar sonidos del entorno.

Figura 34: Micrófonos

- **Micro interruptores:** Muy utilizados para detectar finales de carrera.

Figura 35: Diferentes tipos de micro interruptores

2.3.2.3 Actuadores

Los actuadores son los sistemas de accionamiento que permiten el movimiento de las articulaciones del robot. Se clasifican en tres grupos, dependiendo del tipo de energía que utilicen:

- **Hidráulicos:** se utilizan para manejar cargas pesadas a una gran velocidad. Sus movimientos pueden ser suaves y rápidos.

- **Neumáticos:** son rápidos en sus respuestas, pero no soportan cargas tan pesadas como los hidráulicos.

- **Eléctricos**: son los más comunes en los robots móviles. Un ejemplo son los motores eléctricos, que permiten conseguir velocidades y precisión necesarias.

Ejemplos de actuadores son motores, relés y contadores, electro válvulas, pinzas, etc.

2.3.2.4 Sistemas de control

El control de un robot puede realizarse de muchas maneras, pero generalmente se realiza por medio de un ordenador industrial altamente potente, también conocido como unidad de control o controlador. El controlador se encarga de almacenar y procesar la información de los diferentes componentes del robot industrial.

La definición de un sistema de control es la combinación de componentes que actúan juntos para realizar el control de un proceso. Este control se puede hacer de forma continua, es decir en todo momento o de forma discreta, es decir cada cierto tiempo. Si el sistema es continuo, el control se realiza con elementos continuos. En cambio, cuando el sistema es discreto el control se realiza con elementos digitales, como el ordenador, por lo que hay que digitalizar los valores antes de su procesamiento y volver a convertirlos tras el procesamiento.

Existen dos tipos de sistemas, sistemas en lazo abierto y sistemas en lazo cerrado.

- **Sistemas en bucle abierto**: son aquellos en los que la salida no tiene influencia sobre la señal de entrada.

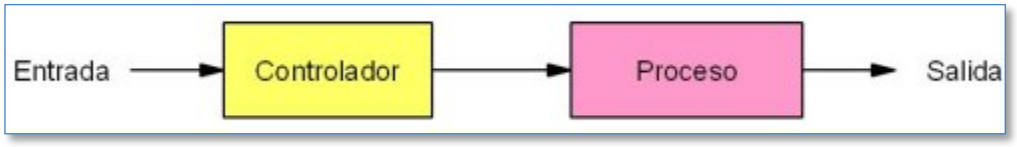


Figura 36: Esquema de un controlador en bucle abierto

- **Sistemas en bucle cerrado:** son aquellos en los que la salida influye sobre la señal de entrada.

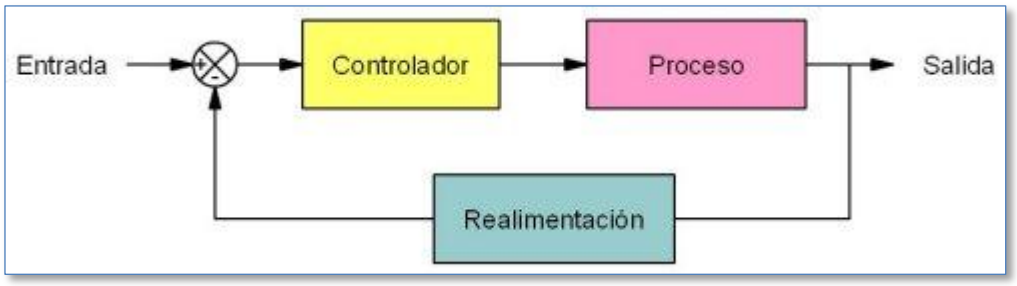


Figura 37: Esquema de un controlador en bucle cerrado

- **Sistemas discretos:** son aquellos que realizan el control cada cierto tiempo. En la actualidad se utilizan sistemas digitales para el control, siendo el ordenador el más utilizado, por su fácil programación y versatilidad.

Generalmente, el control en los robots se realiza mediante sistemas discretos en lazo cerrado, realizados por computador. El ordenador procesa la información captada por los sensores y activa los actuadores en intervalos lo más cortos posibles, del orden de milisegundos.

2.3.2.5 Alimentación

Parte fundamental para garantizar la autonomía del robot, dado que al ser móvil generalmente no va a poder tener energía externa, sólo contará con las reservas internas que pueda transportar, salvo en el caso de que se usen placas solares. Son muchas y muy diversas las formas en que el robot puede almacenar y transportar energía, por lo que veremos las principales:

Sin duda la forma más comúnmente utilizada como fuente de alimentación en todo tipo de robots son las baterías. Son baratas, fiables y se pueden encontrar en cientos de formatos, voltajes y dimensiones.

Figura 38: Diferentes tipos de baterías de uso doméstico

Un aspecto importante a tener en cuenta es el formato y el conexionado de las pilas. Habitualmente encontramos las baterías recargables tanto sueltas como en packs preparados para soldar o para conectar directamente a un PCB. La elección de un formato u otro es más relevante de lo que parece. Por ejemplo, un robot móvil todo terreno que lleve las baterías de forma individual en un porta-pilas no sería nada extraño que con el movimiento alguna pila dejase de hacer contacto o incluso se saliera de su sitio. Y en el otro extremo, si las pilas estuvieran soldadas habría serios problemas para reemplazarlas con facilidad, por no hablar de la dependencia a la hora de cargarlas que deberá ser realizado exclusivamente en el robot, teniendo que añadir los mecanismos necesarios para ello. Parece por tanto que lo más adecuado es usar pilas en packs con conectores, que son las que mayor flexibilidad aportan.

Figura 39: Diferentes tipos de baterías

Si el presupuesto no es un problema se pueden usar tecnologías más avanzadas como las baterías de ion de litio que últimamente están sufriendo una gran expansión gracias a tecnologías móviles de otra índoles como ordenadores portátiles y teléfonos móviles.

Figura 40: Amplio surtido de baterías de Ion de Litio

La diferencia de rendimiento de unos tipos de pilas respecto a otros es notable, así como su tamaño y precio, por lo tanto es importante tener claros los requerimientos del robot respecto a estos aspectos antes de decidir qué tipo de batería se va a utilizar para alimentarlo.

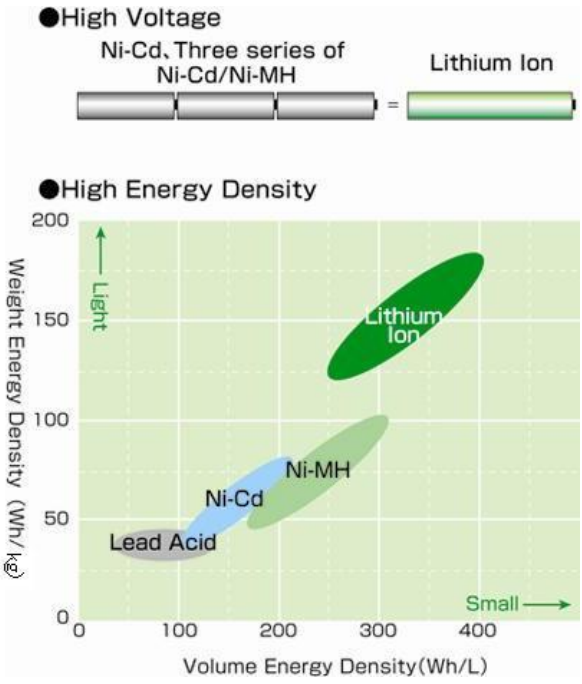


Figura 41: Gráfico comparativo del rendimiento de diferentes tipos de baterías

2.3.3 Comunicaciones

Dentro del campo de las comunicaciones en la robótica móvil, cada fabricante ofrece unas posibilidades de comunicación según las prestaciones y las capacidades de su producto.

A pesar de que en algunos casos se hace uso de conexiones por cable (normalmente USB) para la transferencia de los programas a los robots. Todos los robots utilizados para la realización de la presente tesina disponen de la posibilidad de hacer uso de protocolos de comunicación inalámbrica, lo que les brinda de cierta autonomía y de la ventaja de no necesitar cables.

Es importante conocer a fondo los protocolos de comunicación para poder aprovechar al máximo los tiempos de transmisión y la capacidad del canal, para no limitarse al uso de metodologías de alto nivel donde se desconoce el tratamiento de las comunicaciones lo que puede llevar a un mal aprovechamiento de las mismas. En el siguiente apartado se describen los protocolos de comunicación inalámbrica más usuales en la robótica móvil.

2.4 Comunicaciones inalámbricas

2.4.1 Wifi

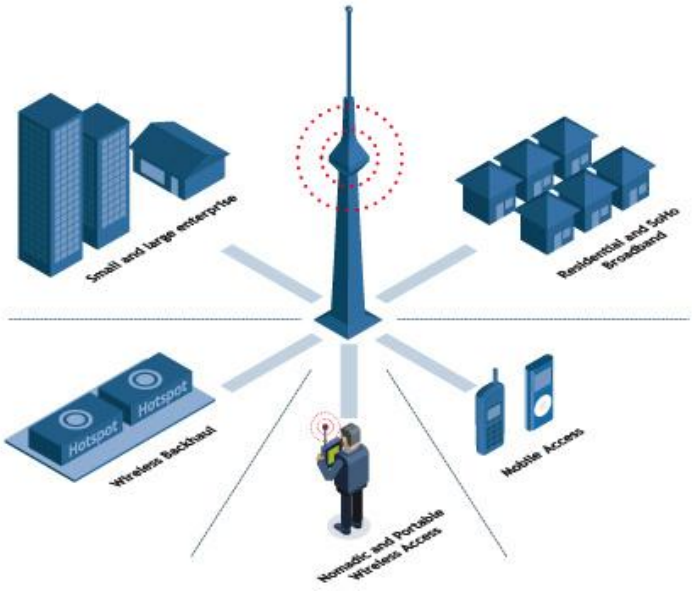


Figura 42: Escenario Wifi

WLAN (Wireless Local Area Network, en inglés) es un sistema de comunicación de datos inalámbrico flexible, muy utilizado como alternativa a las redes LAN cableadas o como extensión de éstas. Utiliza tecnología de radiofrecuencia que permite mayor movilidad a los usuarios al minimizar las conexiones cableadas. Las WLAN van adquiriendo importancia en muchos campos, como almacenes o para manufactura, en los que se transmite la información en tiempo real a una terminal central. También son muy populares en los hogares para compartir el acceso a Internet entre varias computadoras.

Características

- Movilidad: permite transmitir información en tiempo real en cualquier lugar de la organización o empresa a cualquier usuario. Esto supone mayor productividad y posibilidades de servicio.
- Facilidad de instalación: al no usar cables, se evitan obras para tirar cable por muros y techos, mejorando así el aspecto y la habitabilidad de los locales, y reduciendo el tiempo de instalación. También permite el acceso instantáneo a usuarios temporales de la red.
- Flexibilidad: puede llegar donde el cable no puede, superando mayor número de obstáculos, llegando a atravesar paredes. Así, es útil en zonas donde el cableado no es posible o es muy costoso: parques naturales, reservas o zonas escarpadas.

Inicios

Los pioneros en el uso de redes inalámbricas han sido los radioaficionados mediante sus emisoras, que ofrecen una velocidad de 9600 bps. Pero si hablamos propiamente de redes inalámbricas debemos remontarnos al año 1997, en el que el organismo regulador IEEE (Institute of Electronics and Electrical Engineer) publicó el estándar 802.11 (802 hace referencia al grupo de documentos que describen las características de las LAN) dedicado a redes LAN inalámbricas.

Dentro de este mismo campo y anteriormente, en el año 1995, tenemos la aparición de Bluetooth, una tecnología de Ericsson con el objetivo de conectar

mediante ondas de radio los teléfonos móviles con diversos accesorios. Al poco tiempo se generó un grupo de estudio formado por fabricantes que estaban interesados en esta tecnología para aplicarla a otros dispositivos, como PDAs, terminales móviles o incluso electrodomésticos.

Pero el verdadero desarrollo de este tipo de redes surgió a partir de que la FCC, el organismo americano encargado de regular las emisiones radioeléctricas, aprobó el uso civil de la tecnología de transmisiones de espectro disperso (SS o spread spectrum, en inglés), pese a que en un principio lo prohibió por el uso ampliado del espectro. Dicha tecnología ya se usaba en ámbitos militares desde la Segunda Guerra Mundial debido a sus extraordinarias características en cuanto a la dificultad de su detección y su tolerancia a interferencias externas.

A pesar, de que como hemos visto, esta tecnología ya tiene una antigüedad de más de diez años, no ha sido hasta ahora cuando este tipo de redes se ha desarrollado eficazmente debido a la disminución de precios de los dispositivos que la integran. En la actualidad cada vez más se encuentran equipos que pueden competir en precios con los modelos para redes cableadas.

Cómo trabajan las redes WLAN

Se utilizan ondas de radio para llevar la información de un punto a otro sin necesidad de un medio físico guiado. Al hablar de ondas de radio nos referimos normalmente a portadoras de radio, sobre las que va la información, ya que realizan la función de llevar la energía a un receptor remoto. Los datos a transmitir se superponen a la portadora de radio y de este modo pueden ser extraídos exactamente en el receptor final.

A este proceso se le llama modulación de la portadora por la información que está siendo transmitida. Si las ondas son transmitidas a distintas frecuencias de radio, varias portadoras pueden existir en igual tiempo y espacio sin interferir entre ellas. Para extraer los datos el receptor se sitúa en una determinada frecuencia, frecuencia portadora, ignorando el resto. En una configuración típica de LAN sin cable los puntos de acceso (transceiver) conectan la red cableada de un lugar fijo mediante cableado

normalizado. El punto de acceso recibe la información, la almacena y la transmite entre la WLAN y la LAN cableada. Un único punto de acceso puede soportar un pequeño grupo de usuarios y puede funcionar en un rango de al menos treinta metros y hasta varios cientos. El punto de acceso (o la antena conectada al punto de acceso) es normalmente colocado en alto pero podría colocarse en cualquier lugar en que se obtenga la cobertura de radio deseada. El usuario final accede a la red WLAN a través de adaptadores. Estos proporcionan una interfaz entre el sistema de operación de red del cliente y las ondas, mediante una antena. La naturaleza de la conexión sin cable es transparente a la capa del cliente.

Configuraciones de red para radiofrecuencia

Pueden ser de muy diversos tipos y tan simples o complejas como sea necesario. La más básica se da entre dos ordenadores equipados con tarjetas adaptadoras para WLAN, de modo que pueden poner en funcionamiento una red independiente siempre que estén dentro del área que cubre cada uno. Esto es llamado red de igual a igual (peer to peer). Cada cliente tendría únicamente acceso a los recursos del otro cliente pero no a un servidor central. Este tipo de redes no requiere administración o preconfiguración.

Instalando un Punto de Acceso se puede doblar la distancia a la cual los dispositivos pueden comunicarse, ya que estos actúan como repetidores. Desde que el punto de acceso se conecta a la red cableada cualquier cliente tiene acceso a los recursos del servidor y además gestionan el tráfico de la red entre los terminales más próximos. Cada punto de acceso puede servir a varias máquinas, según el tipo y el número de transmisiones que tienen lugar. Existen muchas aplicaciones en el mundo real con un rango de 15 a 50 dispositivos cliente con un solo punto de acceso.

Los puntos de acceso tienen un alcance finito, del orden de 150 m en lugares u zonas abiertas. En zonas grandes como por ejemplo un campus universitario o un edificio es probablemente necesario más de un punto de acceso. La meta es cubrir el área con células que solapen sus áreas de modo que los clientes puedan moverse sin cortes entre un grupo de puntos de acceso.

Esto es llamado roaming, mediante el cual el diseñador de la red puede elegir usar un Punto de Extensión (EPs) para aumentar el número de puntos de acceso a la red, de modo que funcionan como tales pero no están enganchados a la red cableada como los puntos de acceso.

Los puntos de extensión funcionan como su nombre indica: extienden el alcance de la red retransmitiendo las señales de un cliente a un punto de acceso o a otro punto de extensión. Los puntos de extensión pueden encadenarse para pasar mensajes entre un punto de acceso y clientes lejanos de modo que se construye un puente entre ambos.

Uno de los últimos componentes a considerar en el equipo de una WLAN es la antena direccional. Por ejemplo: si se quiere una LAN sin cable a otro edificio a 1 km de distancia. Una solución puede ser instalar una antena en cada edificio con línea de visión directa. La antena del primer edificio está conectada a la red cableada mediante un punto de acceso. Igualmente en el segundo edificio se conecta un punto de acceso, lo cual permite una conexión sin cable en esta aplicación.

Asignación de Canales

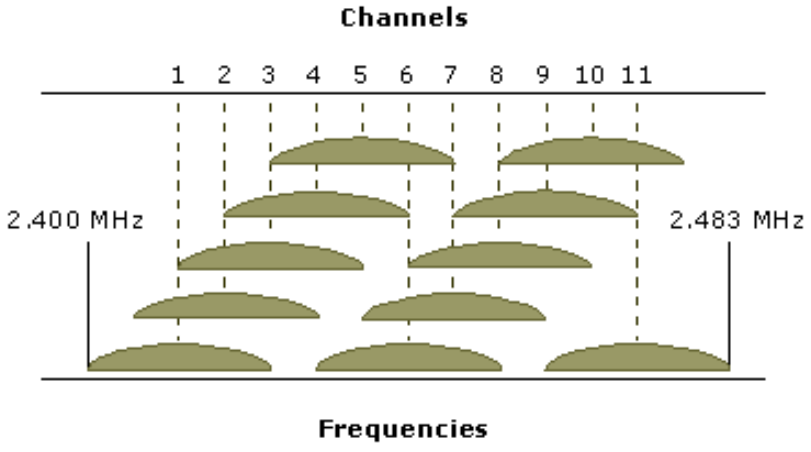


Figura 43: Banda de frecuencias Wifi

Los estándares 802.11b y 802.11g utilizan la banda de 2.4 – 2.5 Ghz. En esta banda, se definieron 11 canales utilizables por equipos WIFI, los que pueden configurarse de acuerdo a necesidades particulares. Sin embargo, los 11 canales no son completamente independientes (canales contiguos se superponen y se producen

interferencias) y en la práctica sólo se pueden utilizar 3 canales en forma simultánea (1, 6 y 11). Esto es correcto para USA y muchos países de América Latina, pues en Europa, el ETSI ha definido 13 canales. En este caso, por ejemplo en España, se pueden utilizar 4 canales no-adyacentes (1, 5, 9 y 13). Esta asignación de canales usualmente se hace sólo en el Access Point, pues los "clientes" automáticamente detectan el canal, salvo en los casos en que se forma una red "Ad-Hoc" o punto a punto cuando no existe Access Point.

Seguridad

Uno de los problemas de este tipo de redes es precisamente la seguridad ya que cualquier persona con una terminal inalámbrica podría comunicarse con un punto de acceso privado si no se disponen de las medidas de seguridad adecuadas. Dichas medidas van encaminadas en dos sentidos: por una parte está el cifrado de los datos que se transmiten y en otro plano, pero igualmente importante, se considera la autenticación entre los diversos usuarios de la red. En el caso del cifrado se están realizando diversas investigaciones ya que los sistemas considerados inicialmente se han conseguido descifrar. Para la autenticación se ha tomado como base el protocolo de verificación EAP (Extesible Authentication Protocol), que es bastante flexible y permite el uso de diferentes algoritmos.

2.4.2 Bluetooth

Bluetooth proviene de la palabra escandinava "*Blåtand"* que significa "hombre de tez oscura" pero en los tiempos que corren el significado original se ha perdido y ahora se asocia a las comunicaciones inalámbricas, un estándar global que posibilita la transmisión de voz, imágenes y en general datos entre diferentes dispositivos en un radio de corto alcance y lo que le hace más atractivo, muy bajo coste.

Los principales objetivos que este estándar quiere lograr son:

- Facilitar las comunicaciones entre equipos.
- Eliminar cables y conectores entre aquéllos.
- Facilitar el intercambio de datos entre los equipos.

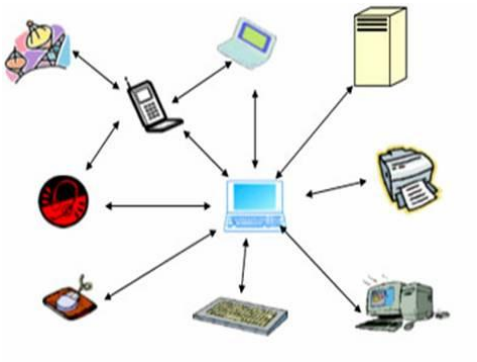


Figura 44: Red Bluetooth

Cómo funciona Bluetooth

Bluetooth funciona bajo radio frecuencias pudiendo atravesar diferentes obstáculos para llegar a los dispositivos que tenga a su alcance.

Opera bajo la franja de frecuencias 2.4 – 2.48 GHz o como también es conocida como "Banda ISM" que significa "Industrial, Scientific and Medical" que es una banda libre para usada para investigar por los tres organismos anteriores. Pero esto tiene sus consecuencias, ya que al ser libre puede ser utilizada por cualquiera y para ello, para evitar las múltiples interferencias que se pudieran introducir (microondas, WLANs, mandos, etc.) Bluetooth utiliza una técnica denominada salto de frecuencias.

El funcionamiento es ir cambiando de frecuencia y mantenerse en cada una un "slot" de tiempo para después volver a saltar a otra diferente. Es conocido que entre salto y salto el tiempo que transcurre es muy pequeño, concretamente unos 625 microsegundos con lo que al cabo de un segundo se puede haber cambiado 1600 veces de frecuencia.

Cuando coinciden más de un dispositivo Bluetooth en un mismo canal de transmisión se forma lo que se llaman "piconets" que son redes donde hay un maestro que es el que gestiona la comunicación de la red y establece su reloj y unos esclavos que escuchan al maestro y sincronizan su reloj con el del maestro.

Dicho alcance puede variar según la potencia a la que se transmite (a mayor potencia mayor consumo y menor autonomía para el dispositivo) y el número de

repetidores que haya por el medio. Así el alcance puede estar entre los 10 y 100 metros de distancia (los repetidores provocan la introducción de distorsión que puede perjudicar los datos transmitidos).

Bluetooth puede conectar muchos tipos de aparatos sin necesidad de un solo cable, aportando una mayor libertad de movimiento. Por esta razón ya se ha convertido en una norma común mundial para la conexión inalámbrica. En el futuro, es probable que sea una norma utilizada en millones de teléfonos móviles, PC, ordenadores portátiles y varios tipos de aparatos electrónicos, como por ejemplo:

- Domótica (activación de alarmas, subida de persianas, etc.).
- Sector automovilístico (comunicación con otros vehículos).
- Medios de pago.

Evolución de las versiones del protocolo

Bluetooth v 1.0 y v 1.0B

Algunas versiones de Bluetooth ya habían sido desarrolladas desde antes de que la tecnología fuese introducida en 1998. Las versiones 1.0 y 1.0B tenían demasiados problemas y restricciones para que los fabricantes lo empezaran a desarrollar con éxito. El principal problema fue la falta de interoperabilidad entre los dispositivos.

Bluetooth v 1.1

Esta versión 1.1 fue el primer éxito de la tecnología Bluetooth. Bluetooth 1.1 corregía muchos de los problemas que se encontraron en anteriores versiones.

Bluetooth v 1.2

Esta versión era completamente compatible con la 1.1. Muchos de los nuevos dispositivos Bluetooth, como los móviles, empezaron a venderse con la nueva especificación 1.2. Esta versión incorporaba una velocidad de transmisión superior a 721kbit/s, técnicas de reducción de interferencias, conexiones síncronas extendidas que reducían la latencia en transferencias de audio e introdujo un control de flujo y de retransmisión por L2CAP.

Bluetooth v 2.0 + EDR

La versión 2.0 + EDR fue anunciada por el SIG en 2004, era totalmente compatible con la versión 1.2 y empezó a aparecer en los dispositivos en 2005. Esta versión introdujo el Enhanced Data Rate (EDR) para mejorar los ratios de transferencia de datos consiguiendo una tasa de transferencia teórica de 3 Mbit/s y en la práctica de 2.1Mbit/s. También era capaz de proporcionar menos consumo de energía a través de un ciclo de trabajo reducido. Esta versión incorporó la posibilidad de la creación de piconets, varias redes de disposivos esclavos conectados a un maestro interconectadas entre sí.

Bluetooth v 2.1 + EDR

En esta versión se mejoró la cabecera de realización del pairing (SSP) aumentando la seguridad entre dispositivos y de nuevo mejoró el consumo de energía en modo de baja potencia.

Bluetooth v 3.0 + HS

Aunque su distribución todavía está en proceso y pocos dispositivos disfrutan de esta versión, la mejora más importante que incorpora es la de soportar el estándar IEEE 802.11 (típicamente de Wi-Fi), lo que permite una velocidad de transferencia de 24 Mbit/s.

Bluetooth v 4.0 (en desarrollo)

Si el mayor avance de la versión 3.0 estuvo en la velocidad (24 Mbit/s frente a los 3 Mbit/s de la 2.0), el nuevo Bluetooth 4.0 pondrá el acento en la reducción del consumo energético. Sobre el papel, la nueva especificación podría conseguir una autonomía medible en años en dispositivos que funcionen con pilas de botón, como relojes, controles remotos y pequeños sensores.

Arquitectura

En la figura 45 se muestra la pila de protocolos Bluetooth, que está basada en el modelo de referencia OSI (Open System Interconnect) de ISO (Internacional Standard Organization) para interconexión de sistemas abiertos. En conjunto, estos protocolos permiten el intercambio transparente de información entre aplicaciones diseñadas de acuerdo con dicha especificación y fomentan la interoperatividad entre los productos de diferentes fabricantes.

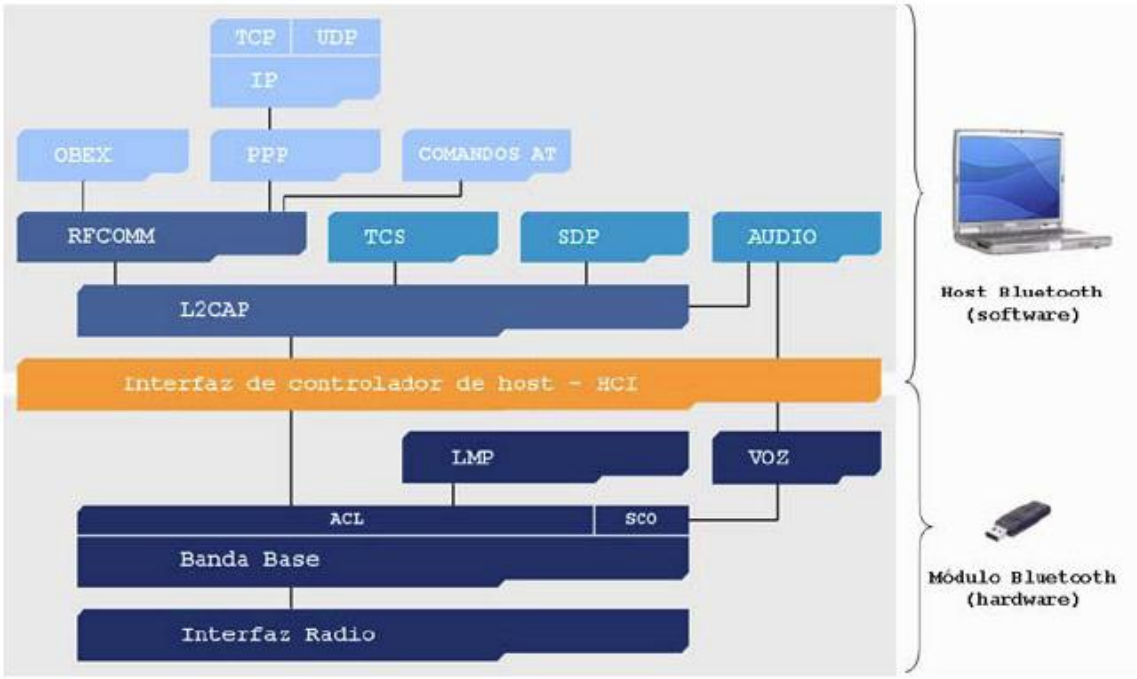


Figura 45. Conjunto de protocolos Bluetooth.

La pila de protocolos Bluetooth se divide en dos, comunicadas por el Interfaz de Controlador de Host (HCI):

- El **módulo Bluetooth** (hardware), encargado de las tareas relacionadas con el envío de información a través del interfaz de radiofrecuencia.
- El **host Bluetooth** (software), encargado de la parte relacionada con las capas superiores de enlace y aplicación.

Sobre la capa de protocolos específicos de Bluetooth, cada fabricante puede implementar su capa de protocolos de aplicación propietarios. De esta forma, la especificación abierta de Bluetooth expande enormemente el número de aplicaciones

que pueden beneficiarse de las capacidades que ofrece esta tecnología inalámbrica. Sin embargo, la especificación Bluetooth exige que, a pesar de la existencia de diferentes pilas de protocolos de aplicación propietarios, se mantenga la interoperatividad entre dispositivos aunque implementen diferentes pilas.

En cuanto al módulo hardware, se compone básicamente en dos partes:

- Un dispositivo de radio encargado de modular y transmitir la señal.
- Un controlador digital, compuesto por una CPU y por un DSP llamado Link Controller.

El LC o Link Controller es el encargado de procesar el BaseBand (la capa más física) y del manejo de los protocolos ARQ y FEC de la capa física. Además se encarga de las funciones de transferencia (asíncronas y/o síncronas), de la codificación de Audio y del cifrado de datos.

La CPU se encarga de atender las instrucciones relacionadas con Bluetooth, para ello implementa un software llamado Link Manager que tiene la capacidad de comunicarse con el resto de dispositivos mediante el protocolo LMP.

Las principales funciones del Link Manager y el Link Controller son el envío y la recepción de Datos, el emparejamiento, la determinación de conexiones, la autentificación, la negociación de tipos de enlace y la determinación del tipo de cuerpo de cada paquete.

Establecimiento de la conexión

Para establecer una conexión Bluetooth entre dos o más dispositivos es necesario que primero se realice un pairing, es decir, un enlace entre los dispositivos que permita conocerse entre ellos. Éste pairing puede realizarse mediante una búsqueda de dispositivos Bluetooth dentro del alcance y requiere una contraseña que bien puede verificarse internamente o bien el usuario debe proporcionar. Una vez enlazados, el Bluetooth almacena la MAC, el nombre y la contraseña dentro de su lista de dispositivos conocidos. Además si el pairing se realiza desde un PC, Bluetooth asignará y mantendrá un mismo puerto COM para cada dispositivo enlazado.

Una vez enlazados ya se puede hacer uso de las diferentes librerías o módulos de controladores Bluetooth existentes para establecer la conexión y enviar o recibir datos.

En algunos experimentos prácticos, incluso se ha conseguido gestionar el envío y recepción de mensajes a través del puerto COM que el PC asigna al dispositivo.

Tipos de datos que se envían

En Bluetooth todos los datos que se envían a través del canal son fragmentados y enviados en paquetes. Además la información se encuentra protegida mediante códigos detectores y correctores de errores. El formato de un paquete de intercambio de información Bluetooth se divide en tres campos: Código de Acceso (72 bits), Cabecera (54 bits) y Datos (hasta 2746 bits).

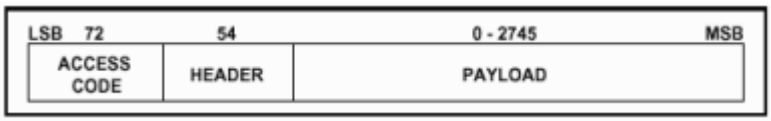


Figura 46. Formato de paquete Bluetooth.

El código de acceso se utiliza para la sincronización, identificación y compensación. Existen tres tipos diferentes de código de acceso:

- Channel Access Code o código de acceso al canal (CAC): identifica una red de dispositivos Bluetooth (piconet).
- Device Access Code o Código de acceso de dispositivo (DAC): utilizado para procesos de señalización especiales.
- Inquiry Access Code o Código de Acceso de Búsqueda (IAC): utilizado para procesos de búsqueda de dispositivos.

La cabecera contiene información de control con seis campos:

- Dirección o AM_ADDR: contiene la dirección temporal de 3 bits que se utiliza para distingue los dispositivos de una piconet.
- Tipo: define el tipo de paquete y cuántos slots va a ocupar.
- Flujo: bit de control que notifica al emisor cuándo el buffer del receptor está lleno (0).
- ARQN: bit de reconocimiento de paquetes recibidos. 1->ACK 0->NAK.
- SEQN: bit que se va invirtiendo para evitar retransmisiones en el receptor.
- HEC: Código de redundancia para comprobar errores en la transmisión.

El último campo tiene una capacidad máxima de 2746 bits y contiene el conjunto de datos que supone la información a transmitir.

En la siguiente figura se muestra el desglose de un paquete en sus distintos campos.

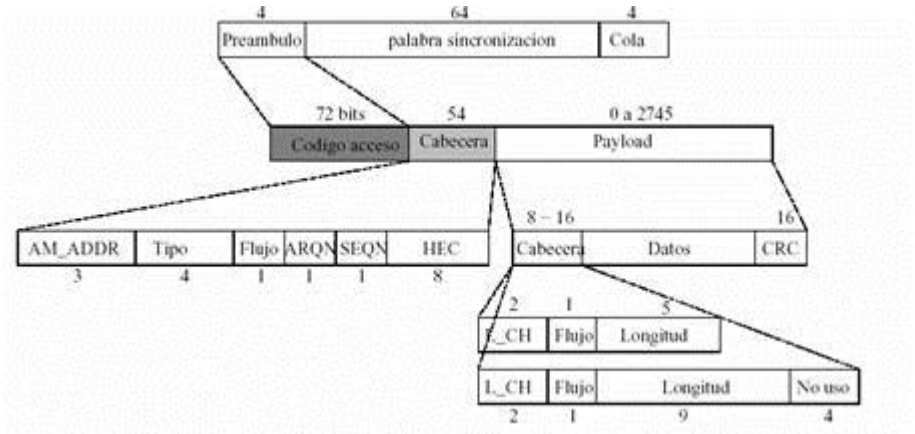


Figura 47. Desglose de paquete Bluetooth.

Las piconets o picoredes

Las *piconets* (o *picoredes*) son la topología de red utilizada por Bluetooth. Todo enlace Bluetooth existe en una de estas redes, que unen dos o más dispositivos Bluetooth por medio de un canal físico compartido con un reloj y una secuencia de saltos única. Si bien un dispositivo puede ser maestro de una única piconet, un dispositivo cualquiera puede pertenecer a varias piconets al mismo tiempo. Este solapamiento se denomina *Scatternet* (*red dispersa*), aunque no se definen capacidades de ruteo por defecto entre ellas.

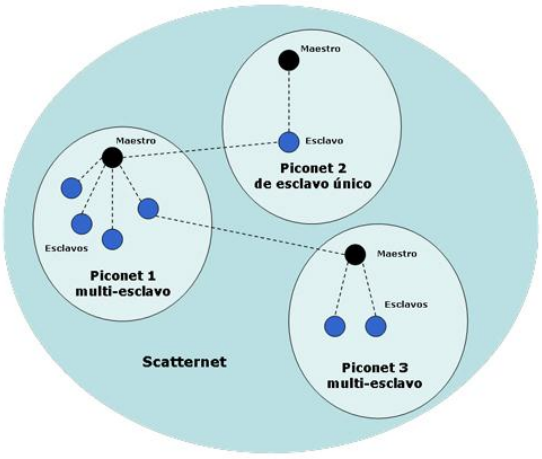


Figura 48. Scatternet

Además, como ya se ha comentado, el paquete Bluetooth tan sólo tiene 3 bits para el direccionamiento de dispositivos, lo que limita el número de dispositivos que pueden formar parte de una misma Piconet a siete. No obstante mediante Scatternets, las dimensiones de la red dispersa pueden ser infinitas.

2.4.3 Otras alternativas

A pesar de que durante el desarrollo de la tesina se han hecho uso de los protocolos Wifi y Bluetooth, hay que tener en cuenta que también existen otros protocolos en el mercado que son capaces de hacerles frente en un futuro no muy lejano.

2.4.3.1 Zigbee

ZigBee es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radios digitales de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (wireless personal area network, WPAN). Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías.

En principio, el ámbito donde se prevé que esta tecnología cobre más fuerza es en domótica, como puede verse en los documentos de la ZigBee Alliance, en las referencias bibliográficas que se dan más abajo es el documento «ZigBee y Domótica». La razón de ello son diversas características que lo diferencian de otras tecnologías:

- Su bajo consumo
- Su topología de red en malla
- Su fácil integración (se pueden fabricar nodos con muy poca electrónica).

Visión general

La relación entre IEEE 802.15.4-2003 y ZigBee es parecida a la existente entre IEEE 802.11 y Wi-Fi Alliance. La especificación 1.0 de ZigBee se aprobó el 14 de diciembre de 2004 y está disponible a miembros del grupo de desarrollo (ZigBee Alliance). Un primer nivel de suscripción, denominado adopter, permite la creación de productos

para su comercialización adoptando la especificación por 3500 dólares anuales. Esta especificación está disponible al público para fines no comerciales en la petición de descarga. La revisión actual de 2006 se aprobó en diciembre de dicho año.

ZigBee utiliza la banda ISM para usos industriales, científicos y médicos; en concreto, 868 MHz en Europa, 915 en Estados Unidos y 2,4 GHz en todo el mundo. Sin embargo, a la hora de diseñar dispositivos, las empresas optarán prácticamente siempre por la banda de 2,4 GHz, por ser libre en todo el mundo. El desarrollo de la tecnología se centra en la sencillez y el bajo coste más que otras redes inalámbricas semejantes de la familia WPAN, como por ejemplo Bluetooth. El nodo ZigBee más completo requiere en teoría cerca del 10% del hardware de un nodo Bluetooth o Wi-Fi típico; esta cifra baja al 2% para los nodos más sencillos. No obstante, el tamaño del código en sí es bastante mayor y se acerca al 50% del tamaño del de Bluetooth. Se anuncian dispositivos con hasta 128 kB de almacenamiento.

En 2006 el precio de mercado de un transceptor compatible con ZigBee se acerca al dólar y el precio de un conjunto de radio, procesador y memoria ronda los tres dólares. En comparación, Bluetooth tenía en sus inicios (en 1998, antes de su lanzamiento) un coste previsto de 4-6 dólares en grandes volúmenes. A principios de 2007, el precio de dispositivos de consumo comunes era de unos tres dólares.

La primera versión de la pila suele denominarse ahora ZigBee 2004. La segunda versión y actual a junio de 2006 se denomina ZigBee 2006, y reemplaza la estructura MSG/KVP con una librería de clusters, dejando obsoleta a la anterior versión. ZigBee Alliance ha comenzado a trabajar en la versión de 2007 de la pila para adecuarse a la última versión de la especificación, en concreto centrándose en optimizar funcionalidades de nivel de red (como agregación de datos). También se incluyen algunos perfiles de aplicación nuevos, como lectura automática,automatización de edificios comerciales y automatización de hogares en base al principio de uso de la librería de clusters.

En ocasiones ZigBee 2007 se denomina Pro, pero Pro es en realidad un perfil de pila que define ciertas características sobre la misma.

El nivel de red de ZigBee 2007 no es compatible con el de ZigBee 2004-2006, aunque un nodo RFD puede unirse a una red 2007 y viceversa. No pueden combinarse routers de las versiones antiguas con un coordinador 2007.

Usos

Los protocolos ZigBee están definidos para su uso en aplicaciones embebidas con requerimientos muy bajos de transmisión de datos y consumo energético. Se pretende su uso en aplicaciones de propósito general con características autoorganizativas y bajo costo (redes en malla, en concreto). Puede utilizarse para realizar control industrial, albergar sensores empotrados, recolectar datos médicos, ejercer labores de detección de humo o intrusos o domótica. La red en su conjunto utilizará una cantidad muy pequeña de energía de forma que cada dispositivo individual pueda tener una autonomía de hasta 5 años antes de necesitar un recambio en su sistema de alimentación.

Futuro

Se espera que los módulos ZigBee sean los transmisores inalámbricos más baratos de la historia, y además producidos de forma masiva. Tendrán un coste aproximado de alrededor de los 6 euros, y dispondrán de una antena integrada, control de frecuencia y una pequeña batería. Ofrecerán una solución tan económica porque la radio se puede fabricar con muchos menos circuitos analógicos de los que se necesitan habitualmente.

2.4.3.2 Infrarrojos

Muy extendido y popular en aparatos electrónicos de pequeño alcance como mandos a distancia o periféricos inalámbricos de ordenador. Se basa en un principio sencillo, modular los datos que transmite en trenes de pulsos de diferente longitud y duración. Es el mismo principio que se usa en el código morse para transmitir letras del alfabeto.

Utiliza generalmente para la emisión diodos LED que emiten en el espectro infrarrojo que resulta invisible para el ojo humano. Para la recepción se usaron en primer lugar fotodiodos u otro tipo de componentes pasivos que fueran capaces de reaccionar ante la luz recibida y que han sido cambiados recientemente por circuitos integrados que incorporan algún componente sensible a la luz como fototransistores o los mismo fotodiodos pero que además incluyen toda la electrónica necesaria para procesar las señales recibidas, simplificando en gran medida los componentes adicionales necesarios para montar un circuito receptor de este tipo de señal.

Su ventaja como se ha descrito, es su sencillez y amplia difusión, añadida al hecho de que los componentes necesarios para su uso son realmente económicos.

Su principal desventaja es su alcance, que sin complejos juegos de lentes difícilmente llega a la decena de metros y más importante aún, la necesidad de tener una línea visual directa y despejada de objetos opacos entre emisor y receptor.

Esta problemática hace este método de comunicaciones poco adecuado para su uso con robots móviles, que con mucha facilidad pueden dejar de cumplir los requisitos de visibilidad emisor-receptor comentado. Pese a esto, ha sido utilizado en algunos robots comerciales, como es el caso del predecesor del LEGO Minstorms NXT, el modelo RCX o del e-puck.

2.4.4 Comparativa entre protocolos

Con el fin de comparar las distintas tecnologías en una misma tabla en la siguiente imagen se puede observar la relación de tasas de transferencia frente al alcance de las tecnologías inalámbricas más usadas como Wifi, Bluetooth, zibBee, red 3G o algunas menos conocidas como Wiimax o Wireless USB.

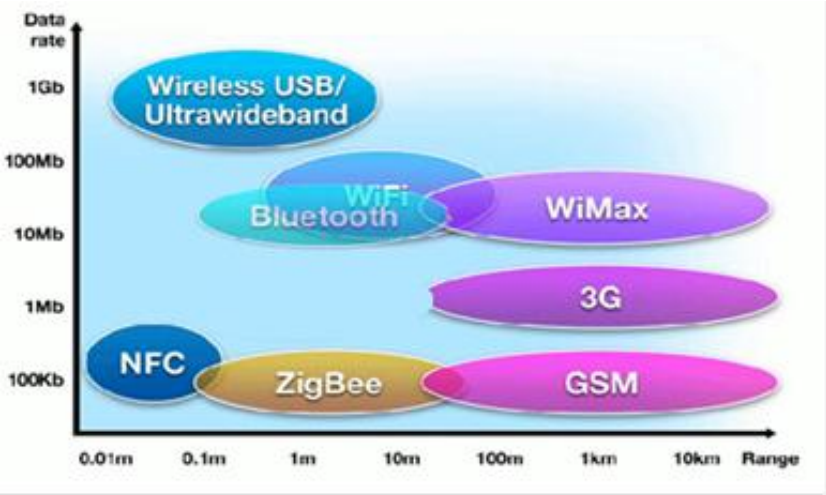


Figura 49. Relación de velocidad y alcance.

Actualmente cada tipo de protocolo intenta hacerse con el control de ciertos tipos de dispositivos según su funcionalidad y sus características. A modo de resumen de diferencias entre los distintos protocolos se adjunta la siguiente tabla que sintetiza las características más importantes de cada protocolo.

CATEGORÍA	ZIGBEE	BLUETOOTH	Wi-FI
Distancia	50-1600m	10m	50m
Extensión	Automática	Ninguna	Depende de la red
Consumo (Duración)	Años	Meses	Horas
Complejidad	Simple	Complicada	Muy complicada
Vel. de transferencia	259Kbps	24Mbps	54Mbps
Rango de frecuencias	868MHz, 916MHz, 2.4GHz	2.4GHz	2.4GHz
Nodos posibles	65535	7	50
Tiempo de enlace	30ms	>10s	>3s
Coste	Bajo	Bajo	Alto
Seguridad	128 bits	64bits, 128bits	SSID

Figura 50. Principales diferencias entre protocolos de comunicación inalámbrica.

2.5 Arquitecturas software

En este apartado se describe brevemente las arquitecturas software más generalizadas con el fin de ofrecer una visión general del estado del arte.

2.5.1 Plana

La arquitectura plana integra todos los componentes de la aplicación en el mismo espacio de direccionamiento que el sistema operativo y fue tradicionalmente adoptada por muchos sistemas de tiempo real.

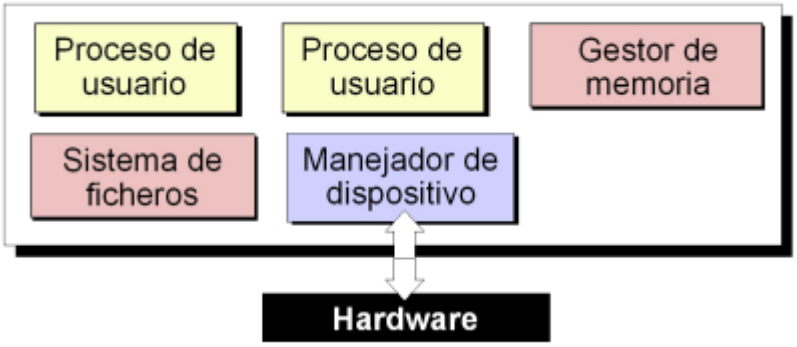


Figura 51. Arquitectura plana.

Es una arquitectura muy utilizada en sistemas empotrados pequeños o procesadores sin facilidades de protección de memoria, como procesadores digitales de señal (DSPs). Su fiabilidad depende de cada componente y aunque un proceso falle, la carencia de protección conlleva a que éste pueda escribir sobre el código o los datos del núcleo, corrompiendo todo el sistema. Además conforme aumenta el tamaño del sistema, la probabilidad de fallo se multiplica.

Los fallos son difíciles de localizar y más aún de aislar. Es preciso que una persona conozca bien todo el sistema completo para modificar un mínimo detalle.

2.5.2 Monolítica

La arquitectura monolítica surge como intento de solución a los problemas de la arquitectura plana. Es una arquitectura rígida de programación en un solo

computador. En esta arquitectura se separan los componentes de gestión de los demás archivos y programas del sistema.

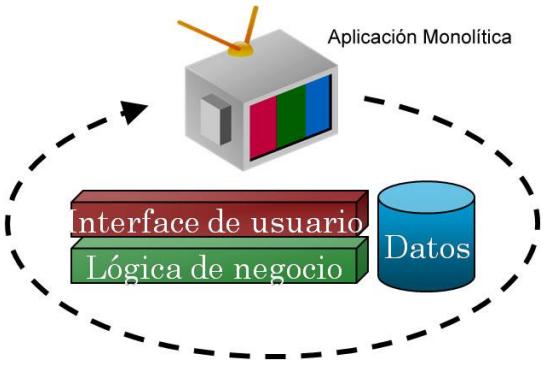


Figura 52. Esquema de arquitectura monolítica.

Es la arquitectura de los primeros sistemas operativos constituidos fundamentalmente por un solo programa compuesto de un conjunto de rutinas entrelazadas de tal forma que cada una puede llamar a cualquier otra. El sistema operativo y todos los servicios fundamentales residen en un monitor al cual se accede a través de un mecanismo de llamada al núcleo. Las llamadas al núcleo hacen de transición del modo aplicación o usuario, al modo supervisor. Así se proporciona cierta protección pudiéndose acceder a los recursos del núcleo únicamente en modo supervisor. Las aplicaciones se ponen en ejecución generalmente como procesos que tienen espacios de direccionamiento separados con protección entre ellos.

Con un monitor monolítico, los servicios fundamentales que requieran acceso a los recursos del núcleo deben residir en éste. De esta forma, la complejidad del núcleo aumenta, aumentando la probabilidad de encontrar errores. Así mismo, el acceso a entrada-salida, al vector de interrupciones, y a la memoria física se puede restringir al núcleo por razones de seguridad, lo que significa que la mayoría de los controladores deben residir en el núcleo.

Generalmente están hechos a medida, por lo que son eficientes y rápidos en su ejecución y gestión, pero por lo mismo carecen de flexibilidad para soportar diferentes ambientes de trabajo o tipos de aplicaciones. Son difíciles de ampliar y de depurar.

2.5.3 Cliente-Servidor

Esta arquitectura consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras. En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

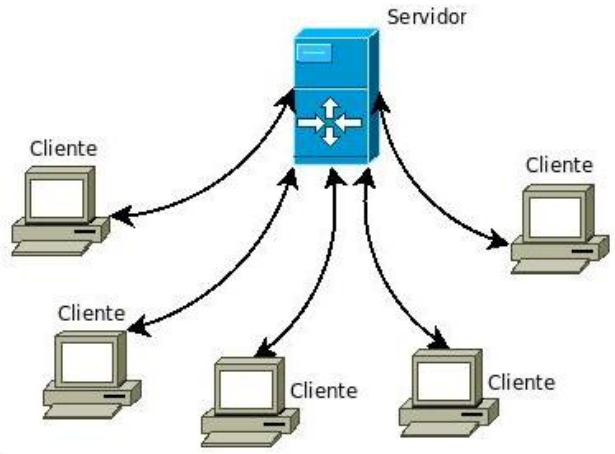


Figura 53. Arquitectura cliente-servidor

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común para esta arquitectura son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes ordenadores aumentando así el grado de distribución del sistema.

La arquitectura cliente-servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico.

2.5.4 Arquitectura multicapa o en tres niveles

Como se ha comentado anteriormente, la arquitectura cliente-servidor ha originado especializaciones como la programación multicapa, cuyo objetivo primordial es la separación de la lógica de negocios de la lógica de diseño.

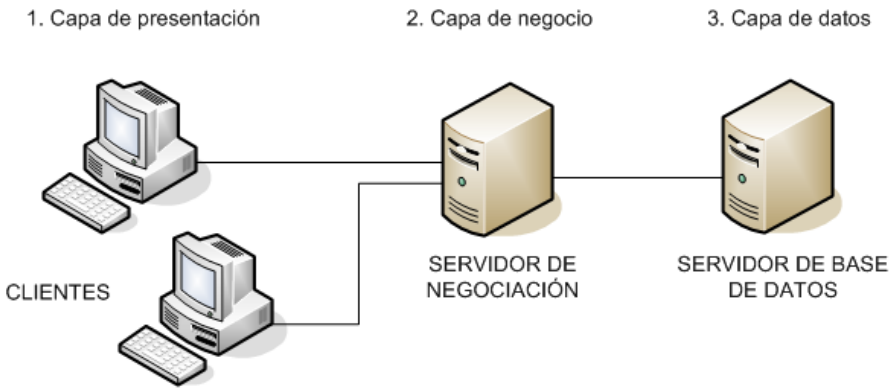


Figura 54. Arquitectura en tres niveles o capas.

La ventaja principal de este tipo de arquitectura es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, únicamente se ataca al nivel requerido sin tener que revisar entre todo el código mezclado.

Además, permite distribuir el trabajo de creación de una aplicación por niveles de forma que cada grupo de trabajo está totalmente abstraído del resto de niveles. Esto da lugar a que esta arquitectura sea fácilmente escalable y por eso sea hoy en día una de las arquitecturas más utilizadas en el diseño de sistemas informáticos.

El diseño más común de esta arquitectura suele ser de tres niveles o capas: una para la presentación (interfaz de usuario), otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento.

2.5.5 Peer to peer

La arquitectura peer to peer consiste en crear una red de computadoras en la que todos o algunos de los nodos de la red funcionan sin ser clientes ni servidores fijos,

sino que se comportan como iguales entre sí. Actúan simultáneamente como clientes y servidores respecto a los demás nodos de la red.

Figura 55. Arquitectura peer to peer.

A pesar de su auge en polémicas aplicaciones de intercambio de archivos por la red, se está utilizando en gran escala para sistemas de ficheros distribuidos o incluso para cálculos científicos que procesen grandes bases de datos. Su alta aceptación y aplicación se debe a su alto nivel de escalabilidad donde lo deseable es que cuantos más nodos estén conectados a la red P2P, mejor será su funcionamiento.

Además la naturaleza distribuida de esta arquitectura incrementa la robustez en caso de haber fallos en la réplica excesiva de los datos hacia múltiples destinos.

La descentralización, la distribución de costes entre los nodos, el anonimato entre los mismos, la escalabilidad y la seguridad que ofrece, hace que sea una de las arquitecturas más deseadas para cualquier sistema distribuido.

2.5.6 Arquitectura en Pipeline

Las arquitecturas paralelas o Pipeline surgen por la necesidad de aumentar la velocidad de procesamiento. Está basada en filtros y consiste en ir transformando un flujo de datos en un proceso comprendido por varias fases secuenciales, siendo la entrada de cada fase, la salida de la fase anterior.

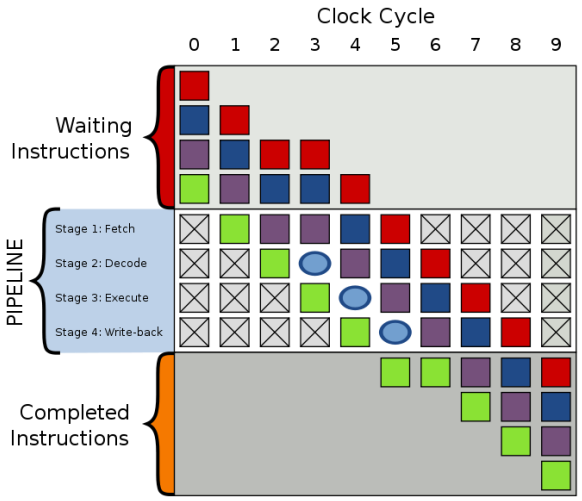


Figura 56. Arquitectura en pipeline.

Esta arquitectura es muy común en el desarrollo de programas para el Shell o el intérprete de comandos con la finalidad de poder concatenar comandos mediante tuberías o pipes. No obstante, tiene un alto uso en el paradigma de la programación funcional dado a que equivale a la composición de funciones matemáticas.

2.5.7 Orientada a servicios (SOA)

Es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requisitos que se solicitan, es decir, es un conjunto de servicios interconectados cuyo objetivo es automatizar uno o varios procesos solicitados.

Aunque existen infinitas clasificaciones de servicios, una de las más simples consiste en dividirlos en Servicios controladores, Servicios de negocio y Servicios de utilidad.

Figura 57. Arquitectura orientada a servicios.

Esta arquitectura permite la creación de sistemas de información altamente escalables que reflejan el negocio de la organización, a su vez brinda una forma bien definida de exposición e invocación de servicios lo cual facilita la interacción entre diferentes sistemas propios o de terceros.

En una arquitectura SOA, los nodos de la red hacen disponibles sus recursos a otros participantes en la red como servicios independientes a los que tienen acceso de un modo estandarizado. Para comunicarse entre sí, estos servicios se basan en una definición formal independiente de la plataforma subyacente y del lenguaje de programación. La definición de la interfaz encapsula las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo.

2.5.8 Dirigida por eventos

Consiste en el diseño e implementación de aplicaciones que transmiten eventos entre componentes de software débilmente acoplados. Existen emisores o agentes y consumidores de eventos que tienen la responsabilidad de reaccionar a los eventos.

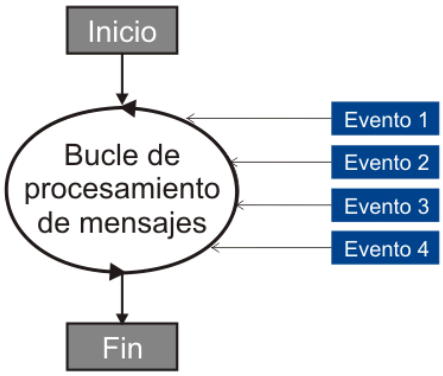


Figura 58. Arquitectura dirigida por eventos.

Esta arquitectura no define un flujo del programa secuencial o estructurado si no que mediante eventos entre los agentes, se va dirigiendo el flujo del programa. Al comenzar la ejecución del programa se llevan a cabo las inicializaciones y demás código inicial y luego el programa queda bloqueado hasta que se produzca algún evento.

El administrador de la arquitectura dirigida por eventos, debe definir los eventos que manejarán su programa y las acciones que se realizarán al producirse cada uno de ellos. Los eventos soportados estarán determinados por el lenguaje de programación utilizado, por el sistema operativo e incluso por eventos creados por el mismo programador.

2.5.9 Sistemas Holónicos

Un Sistema Holónico es una organización altamente distribuida, donde la inteligencia se distribuye entre las entidades individuales. Se la puede comparar con los sistemas distribuidos de la primera generación, sin embargo, el elemento nuevo en estos sistemas es el hecho de que las entidades individuales trabajan juntas en jerarquías temporales (holarquías) para obtener un objetivo global.

Figura 59. Concepto de sistema holónico.

Cada entidad del sistema se llama holón o agente y son capaces de cierto grado de razonamiento local, con capacidad de decisión y reacción ante estímulos o eventos en el entorno, y con la habilidad de comunicarse de manera interactiva con otros holones.

La característica más importante de esta arquitectura es la comunicación entre los agentes y, por lo general, se suelen utilizar uno de los dos métodos que a continuación se detallan.

2.5.9.1 Arquitectura de pizarra

Mediante esta arquitectura el comportamiento básico de cualquier agente consiste en examinar la pizarra, realizar su tarea y escribir sus conclusiones en la misma pizarra. De esta manera, otro agente puede trabajar sobre los resultados generados por otro. También pueden existir agentes con tareas de control específicas sobre la pizarra, o incluso varias pizarras.

Figura 60. Arquitectura de pizarra.

La pizarra tiene un doble papel. Por una parte, coordina a los distintos agentes y por otra facilita su intercomunicación. El estado inicial de la pizarra es una descripción del problema a resolver y el estado final será la solución del problema.

Con este modelo no hay una comunicación directa entre los agentes, toda la información queda centralizada en cada una de las pizarras.

2.5.9.2 Arquitectura directa

La comunicación mediante arquitectura directa consiste en el paso de mensajes entre los agentes. La ventaja que ofrece es la flexibilidad de la comunicación y que no es necesario tener toda la información centralizada, lo cual en algunos casos puede ser no deseable. No obstante en situaciones donde se tenga que informar a todos los agentes sobre algo, la comunicación resulta más engorrosa que con la arquitectura de pizarra.

Figura 61. Comunicación mediante arquitectura directa.

3. Desarrollo práctico

Una vez expuestos los conceptos teóricos que definen el marco donde se desarrolla esta tesina, a continuación se detallan los aspectos prácticos así como el desarrollo de aplicaciones llevado a cabo junto a la problemática encontrada y las soluciones aportadas.

3.1 Arquitectura holónica

Como se ha comentado anteriormente, se ha hecho uso de una arquitectura holónica en sistemas multiagente para aplicaciones de coordinación y comunicación entre distintos tipos de robots móviles. El término agente está cada vez más extendido en el campo del desarrollo software. Pero… ¿qué es exactamente un agente?

3.1.1 ¿Qué es un agente?

Los agentes surgen dentro del campo de la Inteligencia Artificial y, a partir de los trabajos desarrollados en el área de la Inteligencia Artificial Distribuida (DAI), surge el concepto de sistemas multiagente.

Un agente describe una abstracción de software, una idea o concepto, similar a los métodos, funciones y objetos de la programación orientada a objetos. El concepto agente describe una compleja entidad software que es capaz de actuar con cierto grado de autonomía con el fin de cumplir tareas en representación de personas. A diferencia de los objetos que se definen mediante métodos y atributos, un agente software es definido por sus comportamientos.

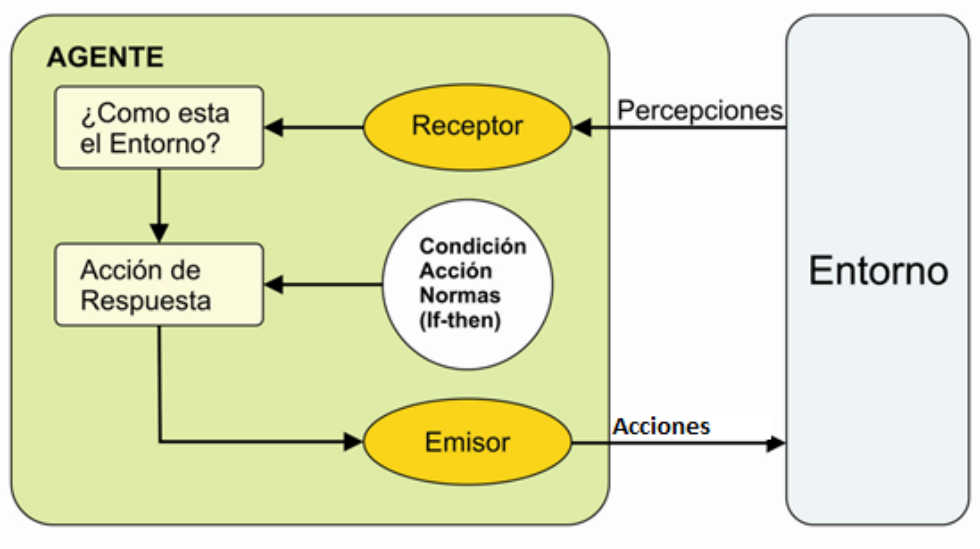


Figura 62. Esquema agente.

Según Franklin y Graesser (1997), las características que definen a cualquier agente son:

- Persistencia: el código no es ejecutado bajo demanda sino que se ejecuta continuamente y decide por sí mismo cuando debería llevar a cabo alguna actividad.

- Autonomía: pueden trabajar sin intervención directa del usuario y tienen cierto control sobre sus acciones y estado interno. Los agentes tienen la capacidad de seleccionar tareas, priorizarlas, tomar decisiones sin intervención humana, etc.

- Capacidad o habilidad social: tienen la habilidad de sincronizarse con personar y otros agentes, a través de coordinación y comunicación, para colaborar en la consecución de una determinada tarea.

- Reactividad: pueden percibir su entorno (que puede ser el mundo físico, un usuario detrás de una interfaz gráfica o vocal, aplicaciones en la red, u otros agentes) y responder oportunamente a cambios que se produzcan en el mismo.

- Iniciativa: el comportamiento de los agentes esta determinado por los objetivos (metas) que persiguen y por tanto pueden producir acciones no sólo como respuesta al entorno.

- Continuidad temporal: persistencia de su identidad y estado durante largos periodos de tiempo.

- Adaptabilidad: deben ser capaces de aprender e improvisar a partir de su experiencia.

- Movilidad: deben ser capaces de desplazarse de una posición a otra de forma autodirigida.

3.1.2 Comunicación entre agentes

A parte de los dos modelos de comunicación descritos en el desarrollo teórico (arquitectura de pizarra y comunicación directa), existen unas organizaciones de estandarización que se encargan de la redacción y aprobación de las normas que rigen cuál debe ser el formato de la información que fluye en un sistema holónico. Principalmente hay tres destacables para el caso de agentes software:

3.1.2.1 OMG

La OMG (Object Manager Group) es una asociación de empresas e instituciones como IBM, RTI o SPARX Systems, formada a finales de los 80. Su principal objetivo era la reutilización, portabilidad e interoperabilidad de sistemas distribuidos de componentes software orientados a objetos. Sus logros más destacables son UML (Unified Modeling Languaje), CORBA (Common Object Requies Broker Architecture) y MASIF (Mobile Agent System Interoperability Facility).

Figura 63. Logotipo OMG.

3.1.2.2 KSE

La iniciativa de formar KSE (Knowledge Sharing Effort) fue tomada por varias empresas a las cuales interesaba el software agente: ARPA (Advanced Research Projects Agency), ASOFR (Air Force Office of Scientific Research), NRI (Corporation for National Research Initiative) y NSF (National Science Foundation).

Su objetivo principal fue facilitar la compartición y reutilización de bases y sistemas basados en conocimiento. Además, para conseguir una interacción eficaz de agentes software eran necesarias tres componentes fundamentales:

1. Un lenguaje común
2. Una comprensión común del conocimiento intercambiado
3. Una habilidad para intercambiar todo lo relativo al lenguaje y la comunicación.

De sus trabajos surgen varios lenguajes como resultado, los cuales son:

- KIF (Knowledge Interchange Format): lenguaje para el intercambio de conocimiento.
- Ontolingua: Lenguaje para la definición de ontologías
- KQML (Knowledge Query and Manipulation Language): Lenguaje para la comunicación entre agentes e interoperabilidad entre agentes en un entorno distribuido.

3.1.2.3 FIPA

En el área de los agentes, la FIPA (Foundation for Intelligent Physical Agents) es la organización encargada de producir especificaciones para la interacción de agentes y sistemas de agentes heterogéneos.

Una plataforma de agentes FIPA se define como el software que implementa un conjunto de especificaciones FIPA. Para que se considere que sigue las normas de FIPA, una plataforma debe implementar al menos la especificación sobre la gestión de agentes y las relativas al lenguaje de comunicación de agentes (ACL). La primera se ocupa del control y gestión de agentes dentro de y a través de plataformas de agentes.

Las relativas al ACL se encargan del formato de los mensajes, los protocolos de interacción y de intercambio de mensajes entre agentes, la descripción de actos comunicativos que definen la semántica de los mensajes intercambiados y los diferentes lenguajes para expresar el contenido de un mensaje (lenguaje de contenido).

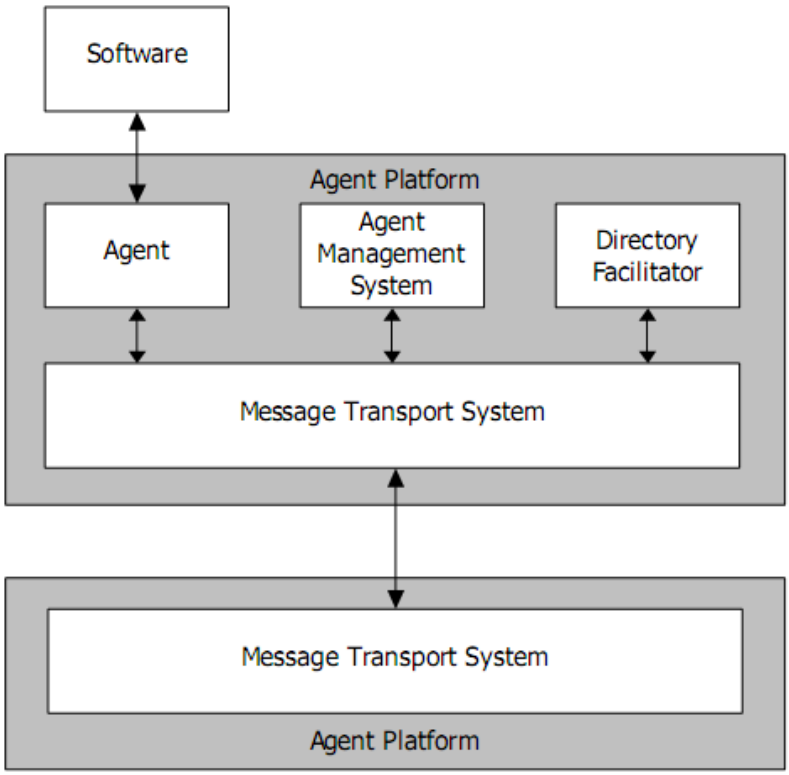


Figura 64: Intercambio de mensajes en FIPA

El objetivo de la especificación de gestión de agentes es ofrecer un marco de trabajo estándar donde los agentes FIPA existan y operen, estableciendo un modelo de referencia lógico para la creación, registro, localización, comunicación, migración y baja de agentes. Este modelo de referencia se compone de un conjunto de entidades que ofrecen diferentes servicios. Estas entidades son:

- Agente: un agente software es un proceso computacional que implementa la funcionalidad autónoma y comunicativa de la aplicación, ofreciendo al menos un servicio. Los agentes se comunican a través de un ACL. Un agente debe tener al menos un propietario y un identificador de agente (AID) que lo identifica de forma unívoca. Además, un agente puede disponer de varias

direcciones de transporte a través de las cuales puede ser contactado. Dependiendo de la implementación de la plataforma un agente puede tratarse de un componente Java o un objeto CORBA.

- DF (Directory Facilitator): un facilitador de directorio que ofrece un servicio de páginas amarillas para localizar agentes. Los agentes deben registrarse previamente en este servicio para ser localizados como proveedores de un servicio, y recurren a él en busca de agentes que ofrezcan un determinado servicio.

- AMS (Agent Management System): un sistema de gestión de agentes que controla el acceso y uso de la plataforma de agentes. Sólo puede existir un agente AMS por plataforma, y se encarga de generar AID's válidos además de ofrecer un servicio de búsqueda de agentes por nombre.

- MTS (Message Transport Service): un servicio de transporte de mensajes ofrece un servicio de comunicación entre agentes, encargándose del transporte de mensajes entre agentes. FIPA especifica para este componente un modelo de referencia y varias especificaciones sobre distintos protocolos y mecanismos de transporte que favorece la interoperabilidad entre plataformas.

- AP (Agent Platform): la plataforma de agentes ofrece una infraestructura física sobre la que desplegar los agentes y está compuesta de un software de soporte, los componentes para la gestión de agentes (DF, AMS, y MTS) y los agentes.

Además, FIPA también define y especifica una serie de elementos y conceptos importantes necesarios para conseguir la interoperabilidad. Estos conceptos son: un servicio (para definir un conjunto de acciones incluidas en una ontología), un ACL (lenguaje para construir mensajes y actos comunicativos) y un AID (identifica un agente incluyendo denominaciones, roles y direcciones).

3.2 Software multiagente: JADE.

El entorno de desarrollo que se ha escogido para la programación de los sistemas multi-agentes basados en robots móviles que se presentan en este trabajo es Java Agent DEvelopment Framework (o JADE). Se trata de una plataforma software para el desarrollo de agentes implementada en Java que ha estado en desarrollo desde 2001. La plataforma JADE soporta la coordinación de múltiples agentes FIPA (Foundation for Intelligent Physical Agents) y proporciona una implementación estándar del lenguaje de comunicación de agentes FIPA-ACL.

Figura 65. Logotipo JADE

JADE fue desarrollado originalmente por Telecom Italia y se distribuye como software libre siendo completamente compatible con Java Development Kit (JDK) 1.4 o superiores, incluyendo la funcionalidad necesaria para la creación básica de agentes, la programación del comportamiento de los agentes, la implementación de la especificación FIPA ACL para el envío y la recepción de mensajes, las clases útiles para la programación de protocolos FIPA, el manejo de información usando ontologías, etc. Además, proporciona también la plataforma FIPA (AMS, Facilitador de directorio y MTS) de forma que puede ejecutarse en una o varias Java Virtual Machine (JVM) donde cada JVM es vista como un entorno donde los agentes pueden ejecutarse concurrentemente e intercambiar mensajes, organizándose en contenedores.

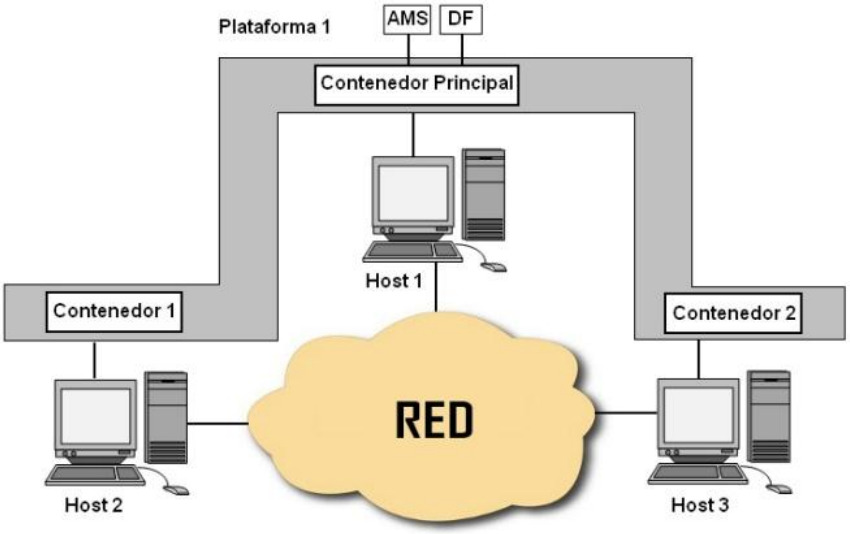


Figura 66. Estructura plataforma JADE

El trabajo que tiene que realizar cada uno de los agentes se encuentra dentro de sus comportamientos (*behaviours*). Un comportamiento representa una tarea que un agente puede llevar a cabo, implementada como un objeto que extiende de la clase *jade.core.behaviours.Behaviour*. A fin de que un agente ejecute una tarea implementada en un comportamiento es suficiente con que el agente añada su comportamiento a la pila de comportamientos. Además, un comportamiento puede ser añadido en cualquier momento, cuando se crea el agente o desde dentro de otros comportamientos.

Un agente puede ejecutar varios comportamientos concurrentemente. Sin embargo, conviene recordar que la planificación de comportamientos en un agente no es preventiva, sino cooperativa. Esto significa que cuando un comportamiento es planificado para ejecutarse, su método *action*() es ejecutado hasta que termina. Por lo tanto, es el programador quien decide cuándo un agente cambia de la ejecución de un comportamiento a la ejecución de otro.

El servicio de directorio (o servicio *yellow pages*) permite a los agentes publicar uno o más servicios que ellos proveen, para que otros agentes puedan buscar por un determinado servicio. El servicio de directorio en JADE (de acuerdo a la

especificación FIPA) es previsto por un agente llamado DF (Directory Facilitator). Cada plataforma FIPA dispone de un agente DF por defecto, aunque se pueden activar más agentes DF con el fin de facilitar un catálogo de páginas amarillas distribuido por toda la red. Aunque es posible interactuar con el agente DF a través del intercambio de mensajes ACL como con cualquier otro agente, JADE simplifica estas interacciones mediante el uso de la clase jade.domain.DFService, con la que es posible publicar, modificar y buscar distintos servicios. De esta forma, un agente que desea buscar un determinado servicio, debe proporcionar al DF una plantilla de descripción y como resultado de la búsqueda obtendrá una lista con todas las descripciones que concuerdan con lo buscado.

Partiendo de la descripción de los agentes que proporcionan un determinado servicio, se puede acceder a su nombre y enviar un mensaje para activar un determinado comportamiento. Una de las características más importantes que los agentes JADE poseen es la habilidad para comunicarse. El paradigma de comunicación adoptado es el paso de mensajes asíncrono. Cada agente tiene una pequeña pila de entrada de mensajes donde el *runtime* de JADE almacena los mensajes enviados por otros agentes. Así mismo, cada vez que un mensaje es añadido a la cola de mensajes, el agente que lo recibe es notificado. Sin embargo, el instante en el que el robot coge el mensaje de la cola y lo procesa depende únicamente del programador.

Un agente puede obtener un mensaje de su propia cola de mensajes utilizando el método *receive*(). Este método devuelve el primer mensaje en la cola de mensajes (y lo elimina) o devuelve *null* en caso de que la cola de mensajes esté completamente vacía. Del mismo modo, se dispone de una función bloqueante (*blockingReceive()*) para la recepción de mensajes, que detendrá el comportamiento hasta que un mensaje con las características necesarias sea recibido.

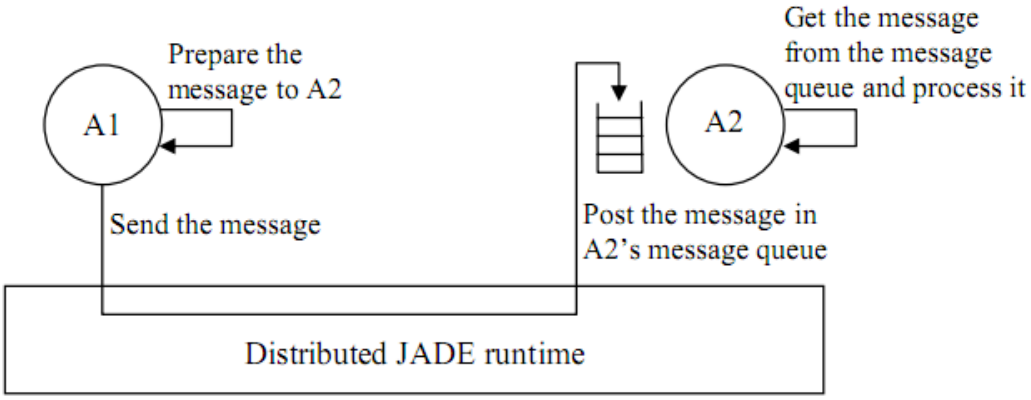


Figura 67. Mensajes en JADE.

3.2.1 Instalación y configuración de JADE.

Para poder utilizar el framework de JADE es necesario instalar y configurar el equipo. Para ello, lo primero que hay que hacer es descargar las librerías necesarias desde la página web del proyecto (http://jade.tilab.com). La última versión disponible es la 4.0.1, aunque, por motivos de estabilidad, en este trabajo se ha utilizado la versión 3.4.

Para configurar el equipo, es necesario añadir la siguiente línea a la variable de entorno CLASSPATH:

CLASSPATH=/ruta/jade/lib/jade.jar:/ruta/jade/lib/iiop.jar:/ruta/jade/lib/http.jar

La ruta debe ser la ruta donde se han descomprimido las librerías de JADE que se han descargado.

Para comprobar que funciona correctamente, se abre un terminal y se ejecuta el siguiente código:

java jade.Boot –gui

y debe aparece la siguiente ventana:

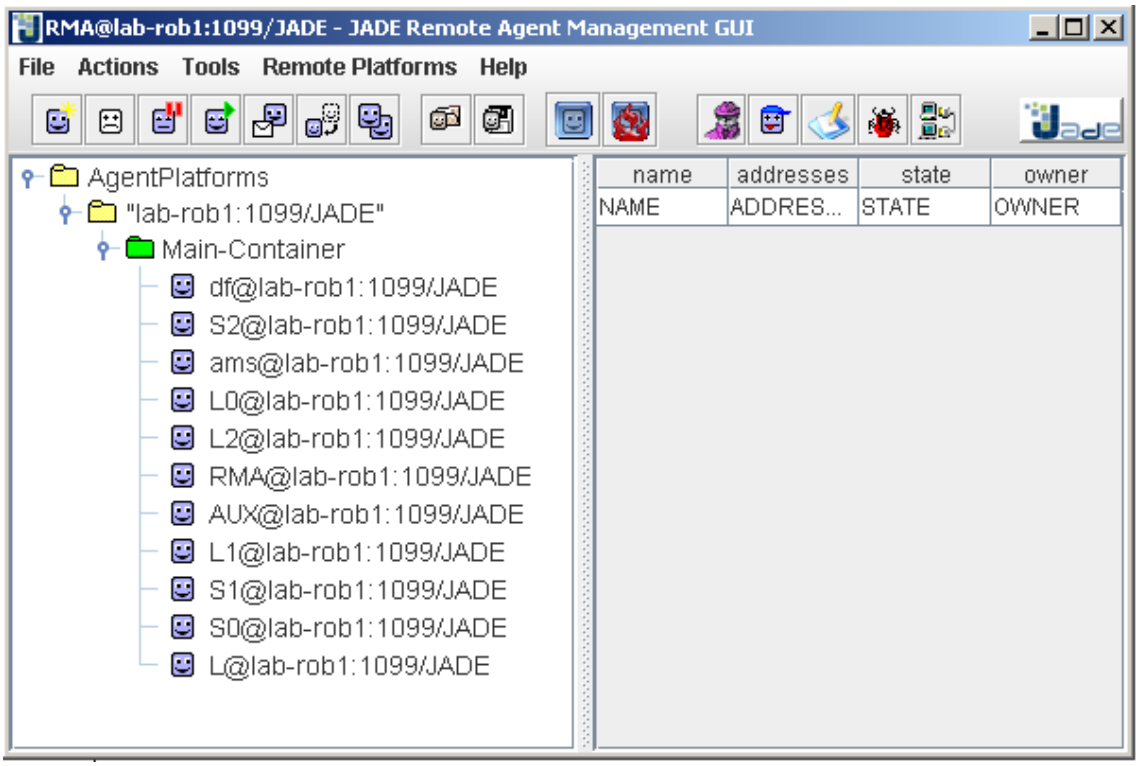


Figura 68: Interfaz de administración JADE

Esta es la interfaz de administración de JADE. En ella es posible realizar distintas acciones como consultar los agentes en ejecución, lanzar nuevos agentes, consultar sus estados, depurar en tiempo de ejecución, enviar mensajes ACL, etc.

Además, como se puede observar, al lanzar la interfaz ya aparecen creados los dos agentes DF y AMS descritos en el apartado anterior.

3.2.2 Compilación y ejecución de agentes JADE

JADE puede ser ejecutado bajo Eclipse como una aplicación Java cualquiera. Sin embargo, es necesario llevar a cabo la configuración necesaria, así como incluir las librerías JADE que utilizarán los agentes.

En primer lugar, y antes de configurar la interfaz para la ejecución de JADE, se debe incluir las librerías de JADE en el proyecto. Para ello, hay que ir a "Project"->"Properties"->"Java Build Path"->"Libraries". Presionar el botón "Add External JARs" y seleccionar las librerías de JADE que se han descargado previamente y una vez estén incluidas, botón "OK".

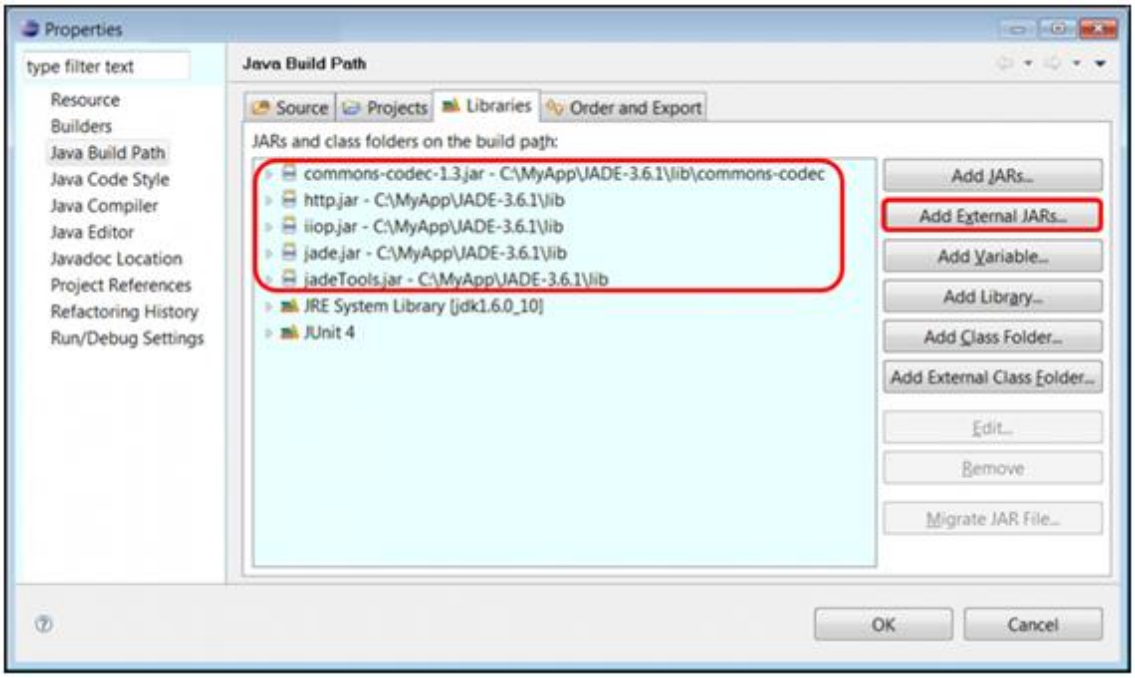


Figura 67: Inclusión de las librerías en Eclipse

El siguiente paso es configurar Eclipse para la ejecución de aplicaciones basadas en agentes. Para ello:

1. Clic en el icono “Run”->”Run Configurations”->”Java Aplication”
2. En la pestaña “Main” hay que introducir el nombre del proyecto, en “Main Class” hay que escribir jade.Boot y asegurarse de tener activada la opción “include libraries when searching for a main class”.
3. En la pestaña “argument”->“Program Arguments” se deben introducir los argumentos que deben ser usados en la línea de comandos cuando se ejecuta JADE siguiendo este esquema:

   -gui jade.Boot [nombre_agente:paquete_java.nombre_clase]

Donde *nombre_agente* es el nombre del agente y que se mostrará en la GUI, *paquete_java* es el paquete donde se encuentra declarado el agente y *nombre_clase* es el nombre de la clase que define al agente.

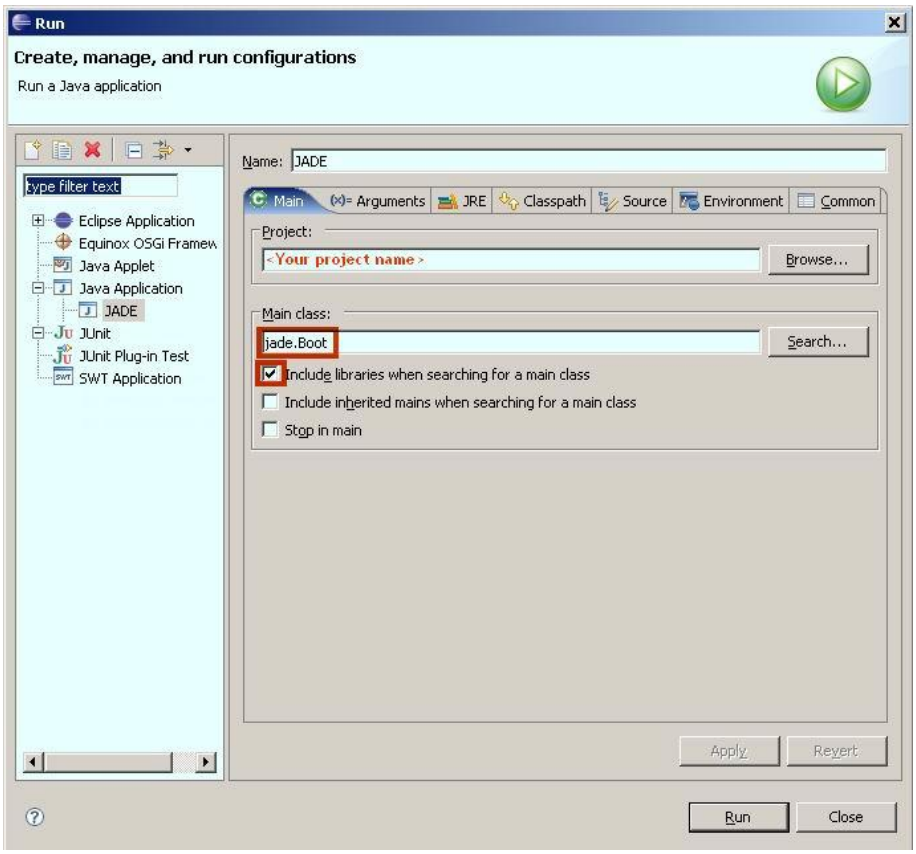


Figura 68: Configuración Eclipse y JADE 1

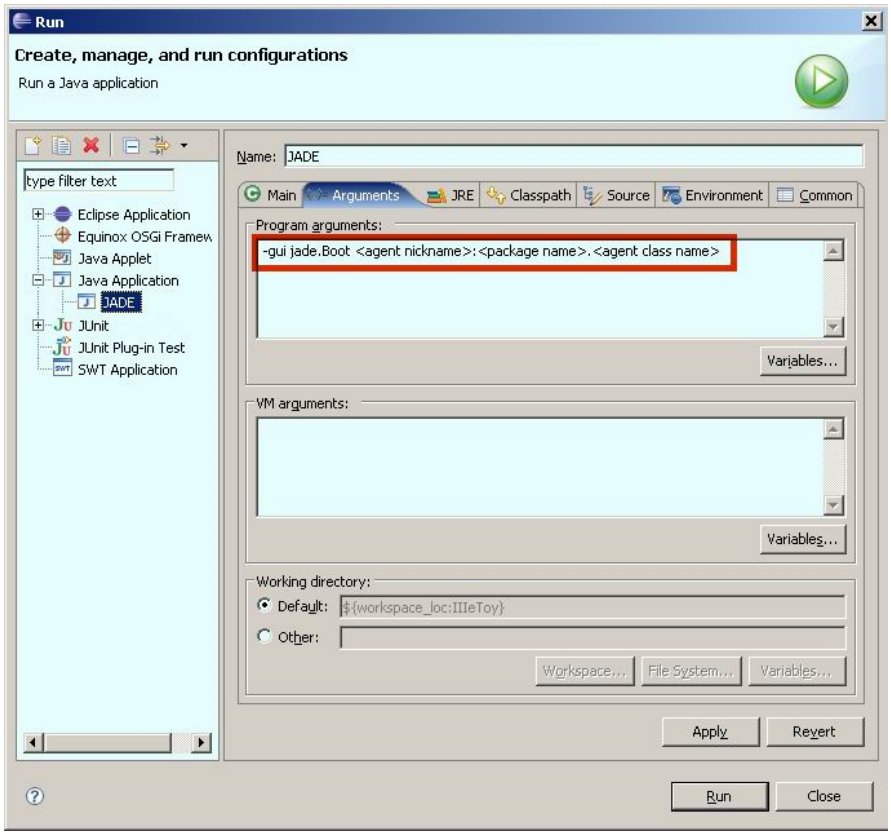


Figura 69: Configuración Eclipse y JADE 2

Una vez realizados estos pasos, si se desea lanzar a ejecución la aplicación sólo hay que pulsar el botón “Run” y se crearían los agentes, se ejecutaría su método *setup()* y se mostraría la ventana de administración de JADE.

3.3 Desarrollo de aplicaciones multiagente para control de robots móviles

A continuación se describen todos los elementos que intervienen en las aplicaciones multiagente desarrolladas en este trabajo, así como los procedimientos y las estrategias estudiadas para su desarrollo.

3.3.1 Hardware utilizado

En este apartado se detalla el hardware que ha intervenido en el desarrollo del proyecto.

3.3.1.1 Cámara

Se instalaron dos cámaras cenitales en el laboratorio con el fin de cubrir el mayor área posible, modelo Logitech Webcam Pro 9000.

Figura 70. Cámara Logitech Webcam Pro 9000.

La cámara ofrece una Óptica Carl Zeiss con enfoque automático y un sensor nativo de alta resolución de 2 megapíxeles. Además es capaz de capturar vídeo en alta resolución (hasta 1600 x 1200 pixeles) y de obtener hasta 30 frames por segundo a la máxima resolución, lo cual resulta muy adecuado para la monitorización de un escenario en tiempo real.

Incorpora además un software muy versátil que permite cambiar desde parámetros típicos como brillo o contraste, hasta la saturación o el balance de blancos, lo que permite el recalibrado rápido mediante su interfaz y evita tener que estar modificando en el código los filtros del procesamiento software que se realiza.

3.3.1.2 Koala

El Koala es un robot móvil de tamaño medio (dimensiones de 32x32x20 cm), que permite desarrollar, desde aplicaciones relacionadas con la vigilancia, transporte de objetos y planificación de trayectorias a seguir, hasta aplicaciones más complejas como la Telemanipulación o la detección y reconocimiento de objetos.

Tal y como se muestra en la Figura, se trata de un robot de 6 ruedas todo-terreno, lo que le posibilita para trabajar desde entornos de oficina hasta terrenos más irregulares de hasta una inclinación máxima de 43 grados. Para ello cuenta con dos servomotores de corriente continua con encoders incrementales. Además, gracias a sus 16 sensores infrarrojos de proximidad distribuidos por todo el robot, se puede obtener una precepción real de los objetos que hay en su entorno. Además, tal y como se comentará posteriormente, en el robot se ha instalado una tarjeta adicional de control y una tarjeta de comunicaciones inalámbricas.

Figura 71. Robot Koala (K-Team)

Una característica importante del Koala es su gran potencia de cálculo, puesto que cuenta con un procesador “Motorola 68331 @ 22 MHz” con 1 Mbyte de RAM y ROM. Además, permite introducirle una gran cantidad de expansiones, mediante las cuales podemos dotarle de aún más características de las que ya tiene.

Para la programación y el control del robot se tienen diversas opciones. Las más simples es mediante la utilización del puerto RS232 disponible en el robot. Mediante este puerto y una aplicación de hiperterminal se pueden utilizar entornos como el LabView o MATLAB. En este caso el algoritmo de control se ejecuta en el PC conectado con el robot. De esta forma el PC puede leer información del robot (como el valor de los encoders, de los sensores de distancia, del nivel de baterías etc.). A partir de esta información el PC puede enviarle por ejemplo las referencias de velocidades a las ruedas o el avance que el robot debe realizar.

La segunda opción consiste en la ejecución embebida de los algoritmos de control. Para ello el fabricante proporciona un entorno de desarrollo y un compilador que permite programar las aplicaciones en lenguaje C, de forma que permite la ejecución del programa compilado y almacenado previamente almacenado en la memoria ROM del robot.

En este trabajo, para poder dotar al robot de un sistema de comunicaciones abierto y flexible que pueda integrarse en una red de computadores se ha optado por añadir al robot una tarjeta de control adicional, encastrando sobre ésta una tarjeta de comunicaciones Wi-Fi.

La tarjeta de control incorporada es la Korebot II. Es una plataforma embebida potente de dimensiones muy reducida basada en el *gumstix Verdex PRO* con un microprocesados Intel XSCALE PXA-270 a 600MHz" con 128 Mbytes de RAM y 32 Mbytes de Flash que incorpora su propio sistema operativo (Linux 2.6.x, OpenEmbedded Ansgtröm Distribution).

La Korebot II tiene disponible varios conectores, de los que podemos destacar un puerto USB (que permite conectar cualquier tipo de dispositivo USB directamente sobre la placa, como por ejemplo, una vídeo cámara USB, ratón o teclado) y un slot Compact Flash. En este último se puede conectar cualquier extensión de pocket PC, como tarjetas de almacenamiento flash, bluetooth, etc.

La tarjeta Korebot II funciona con un sistema operativo Linux. Para la programación de ésta el fabricante proporciona un compilador cruzado y una librería de funciones que permite el control del robot móvil. Las aplicaciones de control se programan en un PC, transfiriéndose el programa ejecutable a la Korebot II vía comunicación serie.

La ventaja de añadir la Korebot II al robot no sólo radica en que tiene mayor memoria y potencia de cálculo, sino que además, permite conectar al robot diferentes dispositivos. En esta aplicación, para dotar al robot de comunicaciones inalámbricas se ha conectado en el slot Compact Flash la tarjeta Wireless Lan Compact Flash Ambicom WL5400G-CF. Se trata de una tarjeta Wireless Lan 802.11 b/g con unas dimensiones de (56x42x5 mm) y un peso de 10 gramos que permite conectarse a cualquier red inalámbrica sin necesidad de cables con una velocidad de transferencia de datos desde 1 hasta 54Mbps.

3.3.1.3 Robotino

Robotino es un robot móvil desarrollado por FESTO dotado de un sistema de 3 ruedas suecas montadas a 120º que le permite desplazarse de forma omnidireccional lo que le permite moverse en todas las direcciones independientemente de su orientación y gracias a las cual es capaz de rotar sobre sí mismo.

Cada una de las unidades que permiten el desplazamiento consiste en un motor de corriente continua unido a una reductora de ratio 16:1, una rueda sueca, una correa dentada y un encoder incremental.

Respecto a la sensorización, está equipado con una webcam, así como con varios sensores, como un sensor de colisión, un anillo de nueve sensores infrarrojos para la detección de distancias, un acelerómetro, dos sensores ópticos digitales, un sensor inductivo analógico, etc. Además, también puede hacer uso del sistema de *Festo NorthStar* para un posicionamiento (absoluto) más preciso a partir de proyecciones en infrarrojo realizadas en el techo.

Figura 72. Robotino de Festo.

Robotino, está controlado por un PC empotrado en el que se ejecuta con una versión de RT Linux. Para acceder al terminal del sistema operativo Linux instalado se puede conectarse al Robotino a través de la red inalámbrica haciendo uso de un cliente de Secure Shell (SSH).

Para la programación y control del Robotino se disponen de varias opciones. Una de ellas es programarlo con el software *Robotino View* en un PC a través de una red de área local inalámbrica. *Robotino View* es un entorno de desarrollo gráfico que es capaz de transmitir señales al controlador del motor, así como visualizar, cambiar y evaluar valores de los sensores. Además, Festo pone a disposición diferentes API's que permiten programar con otras herramientas como Matlab y Labview.

En el caso de no querer depender de un PC externo para establecer el control se puede hacer uso de otro juego de APIs proporcionadas por Festo para programar, compilar y ejecutar las aplicaciones (escritas en C++, Java o .Net) directamente en el PC empotrado del robot.

Para este trabajo se ha programado sobre el mismo PC empotrado dentro del robot a través de un cliente SSH y en lenguaje C++.

3.3.1.4 Lego Mindstorms NXT

El robot LEGO Mindstorms NXT es la 2ª versión del sistema desarrollado en colaboración entre LEGO y el MIT, presentado en enero de 2006 en el International Consumer Electronics Show. La nueva versión además de otros cambios menores en los sensores electrónicos y las piezas de construcción, incorpora una unidad de control nueva: el NXT.

Combinando los bloques de construcción, la fácil programación del NXT y su interfaz de entrada y salida se puede obtener un sistema de prototipado rápido para el desarrollo de una gran variedad de actividades, lo que ha permitido que este sistema haya sido ampliamente aceptado como una herramienta para la investigación y la educación universitaria como se demuestra con la publicación de ediciones especiales de revistas como IEEE Robotics and Automation Magazine o IEEE Control Systems Magazine.

La unidad de control, definida como ladrillo inteligente NXT, está basada en el ARM7, un potente microcontrolador de 32 bits, con 256 Kbytes FLASH y 64 Kbytes de memoria RAM. Para la programación y las comunicaciones, el NXT está equipado con un puerto USB 2.0 y con un dispositivo inalámbrico Bluetooth clase II, V2.0. Así mismo, el NXT dispone de 4 entradas (una de ellas incluye una expansión IEC 61158 Type 4/EN 50 170 para usos futuros) y 3 salidas analógicas. La Figura 73 muestra un ejemplo de robot que se puede montar con el LEGO NXT.

Figura 73. Robot LEGO configuración diferencial

La nueva versión del LEGO proporciona 4 tipos de sensores electrónicos: de contacto, de luz, de sonido y de distancia. Además, existen varias compañías que han desarrollado numerosos sensores compatibles con el NXT, como giróscopos, acelerómetros, buscadores de infrarrojos, cámaras de visión artificial, etc.

De entre los distintos disponibles, en este trabajo se ha optado por utilizar el firmware LeJOS, el cual permite la programación de los NXT en lenguaje Java, ofreciendo librerías orientadas a la navegación, a la comunicación Bluetooth, al uso de trigonometría y funciones matemáticas complejas, a la programación multithreading, al soporte de programación orientada a objetos... Además proporciona distintos niveles de abstracción a la hora de programar los movimientos de los NXT, desde funciones de alto nivel que definen rectas y curvas, al tratamiento del movimiento de los servos de corriente continua a nivel del voltaje que se les aplica.

En cuanto al entorno de programación, LeJOS permite la elección de cualquier entorno de desarrollo en Java como Eclipse o Netbeans. En este trabajo se ha optado por Eclipse. Se trata de un entorno muy utilizado por cualquier programador de lenguaje Java que fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge. Hoy en día, lo desarrolla la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

3.3.1.5 e-puck

La última clase de robots móviles que se utilizan en este trabajo son los e-pucks. Fueron desarrollados por la Federal Swiss Institute of Technology in Lausanne. Se trata de un robot de reducido tamaño (75mm de diámetro) equipado con sensores, actuadores e interfaces que le otorgan gran flexibilidad a un bajo coste. La Figura 74 muestra el robot utilizado en este trabajo.

Figura 74. Robot móvil *e-puck*

La unidad de control del robot está basada en un microcontrolador dsPIC de Microchip que combina un procesador de 16 bits con una unidad de procesamiento digital de señales (DSP), proporcionando un procesamiento de datos muy eficiente. El microcontrolador dispone de 8kB de memoria RAM y 144kB de memoria flash. Funciona a una frecuencia de 64 MHz y proporciona hasta 16 MIPS.

Por otro lado, el e-puck proporciona una gran variedad de sensores. En su configuración básica cuenta con una anillo de 8 sensores de proximidad, un acelerómetro 3D, 3 micrófonos para capturar el sonido y una cámara de visión en color CMOS.

Los actuadores que proporciona el robot son principalmente un par de motores paso-a-paso con una resolución de 1000 pulsos por revolución. Además, también dispone de un altavoz y 8 LEDs situados en anillo alrededor del chasis más un LED frontal adicional situado junto a la cámara.

Por último, el robot contiene varios dispositivos para poder interactuar con el usuario y para comunicarse con otros dispositivos, como una interfaz serie RS232, un enlace de radio Bluetooth 1.1, un receptor de control remoto infrarrojo y un conmutador rotatorio de 16 posiciones que permite, por ejemplo, seleccionar cuál va a ser el programa a ejecutar. La Figura 75 muestra la arquitectura del robot.

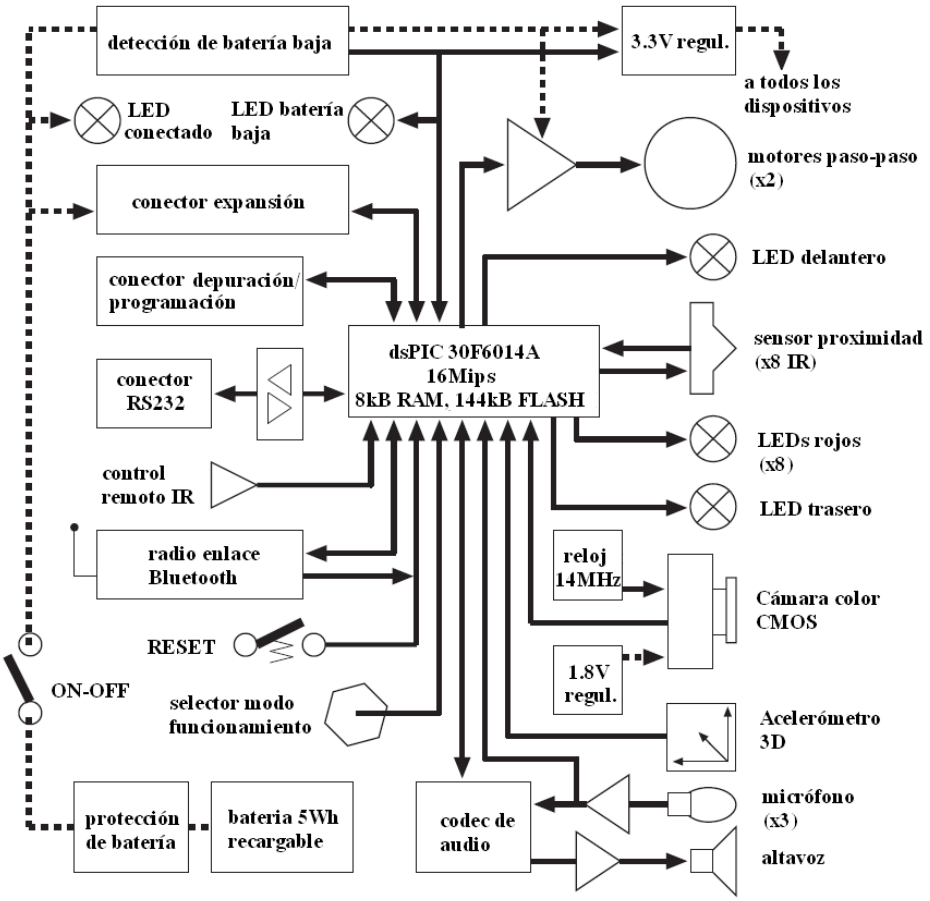


Figura 75. Arquitectura hardware del robot e-puck.

El robot e-puck proporciona una librería de bajo nivel para facilitar el uso del hardware, conteniendo funciones para el control de los motores, la lectura de la cámara, el envío de mensajes a través del Bluetooth, etc. La librería se enlaza estáticamente con la aplicación de usuario en el instante de la compilación.

En este trabajo se ha utilizado el entorno de desarrollo RT-DRUID (que está basado en Eclipse) para la programación de las aplicaciones del robot. Éste, a su vez se sirve de MPLAB (MPLAB Integrated Development Environment, 2010) para la compilación y descarga de las aplicaciones al e-puck.

MPLAB, consiste en un conjunto gratuito de herramientas integradas para el desarrollo de aplicaciones embebidas para microcontroladores de Microchip. Entre ellas, un compilador específico para el microcontrolador del robot. (MPLAB C Compiler for Academic Use, 2010) que es gratuito y proporciona librerías para los microcontroladores de las familias PIC18, PIC24, dsPIC DSC y PIC32.

Por último para transferir el programa al robot se utiliza el programador ICD 2 de Microchip. Se trata de un dispositivo con el que se pueden programar diferentes familias de microcontroladores, como por ejemplo los PIC16F, PIC18F y dsPIC30.

3.3.1.6 AR Drone

El último robot utilizado, el AR Drone, es muy distinto al resto ya que no pertenece a la familia de robots móviles con ruedas sino que es un cuadricóptero volador.

Figura 76. Cuadricóptero AR Drone de Parrot.

Desarrollado por la empresa Parrot, el AR Drone tiene unas dimensiones de 52,5 x 51,5 cm y una estructura hecha a base de tubo de carbón que le otorga un peso de 400 gramos. Lo que le permite elevarse mediante cuatro hélices de fibra de carbono accionadas por cuatro motores eléctricos de tipo Brushless capaces de girar a 3500 rpm y hacer que alcance los 18km/h.

El cuadricóptero dispone de cuatro sensores: un sensor de altitud basado en ultrasonidos con una frecuencia de 40kHz y un rango máximo de seis metros, un acelerómetro de tres ejes, un giroscopio de dos ejes y otro giroscopio de un eje. Todo gobernado por un procesador ARM9 RISC de 32bits a 468MHz, con una memoria de 128MB DDR, una tarjeta Wi-Fi incorporada y manejado por un Linux embebido.

Además posee dos cámaras de vídeo, una frontal VGA con una resolución de 640 x 480 píxeles, con chip CMOS, una lente angular de 93 grados y una frecuencia de 15

frames por segundo; y otra cámara vertical o cenital QCIF de resolución 176 x 144 píxeles, con una lente de 64 grados y una frecuencia de 60 fps.

La batería de litio de tres celdas tiene una capacidad de 1000mAh y otorga una autonomía de vuelo de 12min. Por otro lado, la red Wi-Fi generada por el cuadricóptero alcanza los 50 metros al aire libre y unos 20 metros en un espacio cerrado. También va equipado con un puerto USB.

A pesar de que el Linux integrado está completamente cerrado, la comunicación entre el AR Drone y la plataforma de control (ya sea un Iphone, un ipad, un PC, etc) se realiza a través de una SDK que gestiona el envío de paquetes UDP y TCP, lo que permite establecer el control desde distintos lenguajes de programación como JAVA, o .NET, o C++.

En el presente trabajo se ha utilizado un PC conectado al AR Drone bajo una aplicación JAVA que gestiona el envío de paquetes así como la recepción del video capturado por las cámaras.

3.3.2 Odometría

3.3.2.1 Definición, importancia y uso.

El concepto de odometría se define como el estudio de la estimación de la posición de vehículos con ruedas durante su navegación. En robots móviles se utiliza para estimar su posición relativa a su localización inicial. Es bien sabido que dada una buena aproximación respecto a su localización inicial, la odometría proporciona una buena precisión a corto plazo, no obstante, la odometría es la integración de información incremental del movimiento a lo largo del tiempo, lo cual conlleva inevitablemente a la acumulación de errores.

La odometría se basa en ecuaciones sencillas de implementar, las cuales hacen uso del valor de los encoders de las ruedas del robot para traducir las revoluciones de las ruedas a un desplazamiento lineal relativo al suelo. Este concepto básico puede llevar a imperfecciones y a cálculos erróneos si por ejemplo las ruedas patinan, o se produce una sobre-aceleración (errores no sistemáticos). La eficacia de la estimación

también depende directamente de la exactitud de la localización y orientación inicial del robot.

Situar un robot a mano, en una posición determinada con una precisión de milímetros, puede no resultar excesivamente complicado, pero cuando se trabaja con varios robots los cuales deben compartir las coordenadas absolutas de un escenario, puede resultar realmente complejo e imperfecto. Además, el correcto funcionamiento de la odometría depende también de errores sistemáticos como la medición de los diámetros de las ruedas, su alineamiento, la resolución discreta del encoder o la distancia de separación entre las ruedas. En el siguiente apartado se describe una aplicación para la calibración de la odometría de un Lego Mindstorms NXT con el fin de evitar, o intentar reducir al máximo, el error de odometría producido por errores sistemáticos.

A pesar de sus limitaciones, el uso de la odometría en investigación es uno de los pilares más importantes del sistema de navegación de un robot.

3.3.2.2 Aplicación para la calibración de un Lego Mindstorms NXT

3.3.2.2.1 Descripción

La aplicación para la calibración de la odometría de un Lego Mindstorms está basada en el libro Cyberbotics' Robot Curriculum iniciado por Olivier Michel y ampliado por Fabien Rohrer y Nicolas Heiniger, donde se detalla mediante cuatro sencillos pasos cómo calibrar la odometría de un e-puck.

Las cuatro pruebas son sencillas y el resultado de cada prueba depende del parámetro que en ese momento se está calibrando, por ejemplo, se introduce un valor de separación entre ruedas y se realiza una trayectoria concreta que debería acabar con el robot en una posición determinada y prevista. Si no acaba en esa posición, habrá que variar el valor de la separación entre ruedas porque no es el adecuado.

Al fin y al cabo se trata de un método de prueba y error. Así pues, para no tener que estar cambiando el código del programa cada vez que se modifica un parámetro y volver a ejecutar, se ha desarrollado una interfaz donde se puede ir manipulando los valores de calibración y éstos se actualizan directamente en el robot mediante una

conexión Bluetooth, de manera que se pueden repetir las pruebas las veces que haga falta sin necesidad de ir modificando el código de la aplicación.

Los parámetros a calibrar son los encoders de las ruedas, la distancia entre las ruedas, el diámetro de las ruedas y el factor de escala entre lo que el robot cree haber recorrido y lo que realmente recorre. En el apartado de sistema de archivos, interfaz y ejecución se explica detalladamente los pasos a seguir para realizar los test correctamente.

3.3.2.2.2 Metodología y desarrollo de la aplicación

La aplicación se ha desarrollado en lenguaje JAVA para un Lego NXT con el firmware LeJOS versión 0.9 instalado.

La metodología del programa está basada en una interfaz sencilla a base de ventanas donde primero se realiza la conexión al robot y luego mediante la gestión de eventos por botones se va navegando entre las distintas opciones y ejecutando las distintas pruebas.

Por defecto el programa que se carga en el NXT contiene las cuatro pruebas predefinidas con los parámetros a calibrar definidos bien el estándar que dicta su especificación o bien por una medición previa realizada a mano con un metro o una regla. Desde la interfaz de la aplicación del PC es desde donde estos parámetros se pueden ir modificando y actualizando en el Lego, no obstante se deben apuntar manualmente cuando ya estén calibrados porque al cerrarse la aplicación, los parámetros vuelven a su valor por defecto.

3.3.2.2.3 Sistema de archivos, interfaz y ejecución

El sistema de archivos de la aplicación consta de dos proyectos. Calibrar-NXT es el proyecto que hay que cargar en el Lego NXT y Calibrar-PC, el proyecto a ejecutar en el PC.

Instalando el plugin para Eclipse de la nueva versión del firmware LeJOS 0.9, el proyecto se configura automáticamente si con el botón derecho se selecciona la opción de "convert to leJOS NXJ project". Así las librerías necesarias se cargan automáticamente en el Java Build Path.

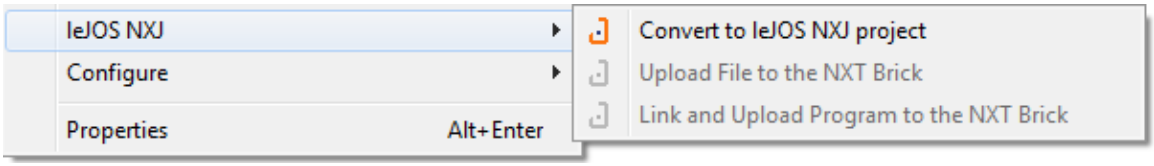


Figura 77. Convertir el proyecto a leJOS NXJ project

De todos modos, si se desea hacer manualmente, para el proyecto del NXT únicamente hay que incluir el classes.jar, mientras que para el proyecto del PC, se debe incluir el bluecove.jar, pccomm.jar y pctools.jar.

Cada proyecto consta de tres ficheros .java como se puede ver en la figura 78. Las clases con el nombre Principal.java, son las que contienen el main que debe ejecutarse.

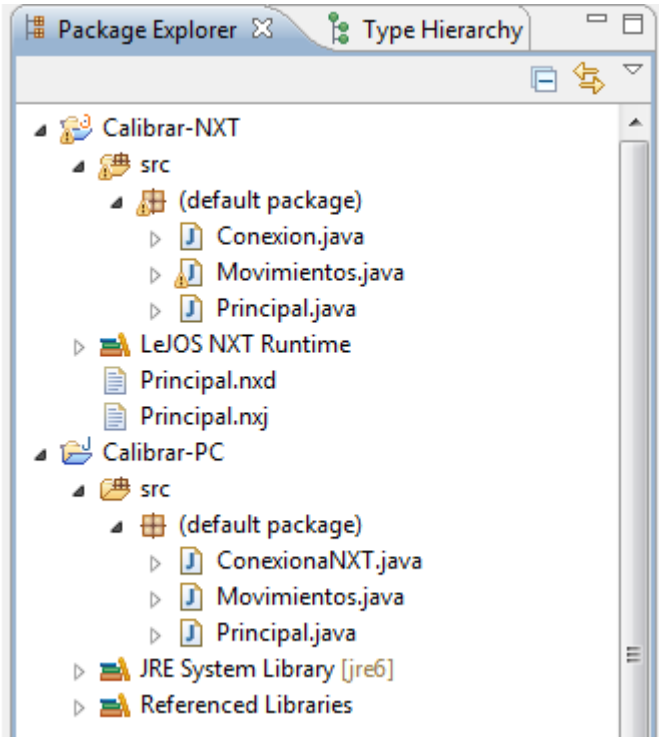


Figura 78. Sistema de archivos.

Las clases de conexión contienen las funciones de conexión y desconexión entre el NXT y el computador. Las clases *Movimientos* contienen todas las funciones que implementan los test a realizar. Y por último la clase Principal extiende de la clase frame para crear la interfaz encargada de ir llamando a las funciones según va navegando el usuario.

Una vez cargado el programa en el NXT, al ejecutar la aplicación del PC, aparece lo siguiente:

Figura 79. Ventana de bienvenida.

Figura 80. LCD NXT waiting.

Si está todo listo, al pulsar Conectar, se producirá la conexión Bluetooth entre el Lego y el computador. Si la conexión se realiza con éxito en el LCD del NXT se leerá *Connected to PC*. En este momento el robot se encuentra esperando la selección de la prueba que se desea realizar. En la interfaz del programa aparece el menú principal del programa con cuatro botones con el nombre de los test a realizar.

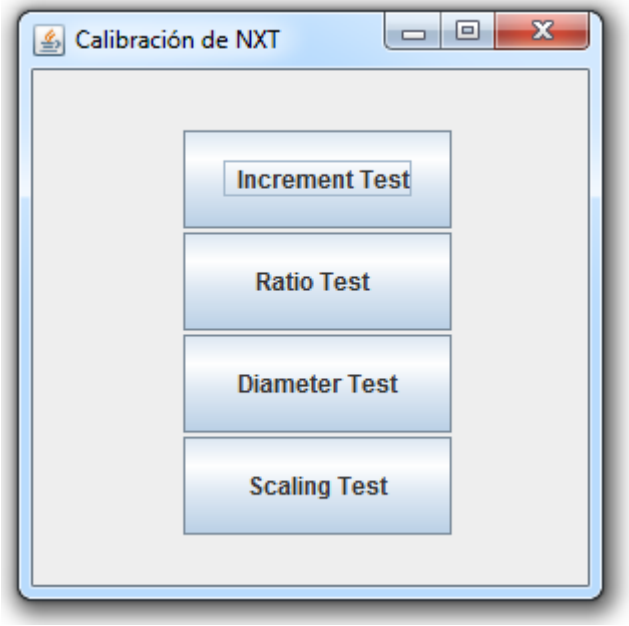


Figura 81. Pantalla principal.

Increment Test calibra los enconders de las ruedas. Radio Test calibra la distancia de separación entre ruedas. Diameter Test calibra el diámetro de las ruedas. Y Scaling Test calibra la relación entre la distancia que cree haber recorrido el robot y la que realmente ha recorrido.

El menú que aparece al pulsar en Increment Test es el siguiente:

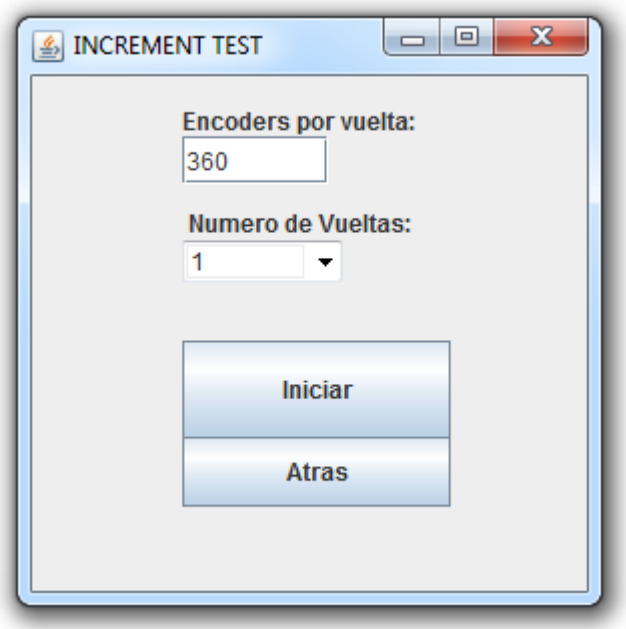


Figura 82. Increment test

En este test se calibran los encoders de las ruedas. La resolución de las ruedas del Lego son de 360 encoders por vuelta. La idea del test es marcar una rueda de alguna forma o fijarla en una posición conocida y pulsar en iniciar. Si se elige una vuelta, el Lego avanzará hasta haber contado 360 encoders y se parará. Habra que comprobar visualmente que efectivamente cada rueda ha dado una vuelta completa justa. Debido a que el Lego no tiene sistema de frenado, es normal que visualmente avance un poco más de la cuenta debido a la inercia, pero por ello se muestra en el LCD el número de encoders que ha contado hasta el momento que ha dejado de aplicar voltage a los motores. También es normal que tan sólo una rueda de justo 360 y la otra varíe ya que esa variación se corresponde a la diferencia de tiempo entre la lectura de cada rueda.

Figura 83. Lego realizando el increment test

En el apartado de *Encoders por vuelta* se puede modificar el valor de los encoders por vuelta y se actualizará en el robot mediante Bluetooth para llevar a cabo la prueba pero, en general, es un test en el que pocas veces habrá que modificar ese valor porque tal caso querría decir que los encoders vienen defectuosos de fábrica o se han estropeado por algún motivo.

Para volver al menú principal se pulsa en *Atras* y se continúa con el siguiente test, *Ratio Test*. La interfaz es la siguiente:

Figura 84. Ratio Test

En este test se calibra el ratio, es decir, la relación entre la distancia entre ruedas y el diámetro de las mismas. El objetivo es encontrar el ratio que consiga que al pulsar iniciar, el robot gire el número de vueltas que se le indique sobre sí mismo. Lo recomendable es empezar con una vuelta y luego ir ajustando hasta conseguir que de tres vueltas y su posición final sea idéntica a la inicial.

Figura 85. Lego realizando el ratio test.

El parámetro inicial del ratio se corresponde con una medida manual y visual del diámetro y la distancia entre ruedas. Se debe ajustar a la configuración o montaje que se haya hecho con el robot.

Una ver esté claro el valor del ratio, en la configuración de los parámetros del robot para el cálculo de su odometría, habrá que modificar la distancia entre las ruedas como el *Ratio*diámetroderueda.*

El siguiente test, *Diameter test*, calibra el diámetro de cada rueda del Lego haciendo que recorra la trayectoria definida por un cuadrado. La interfaz en el computador es la siguiente:

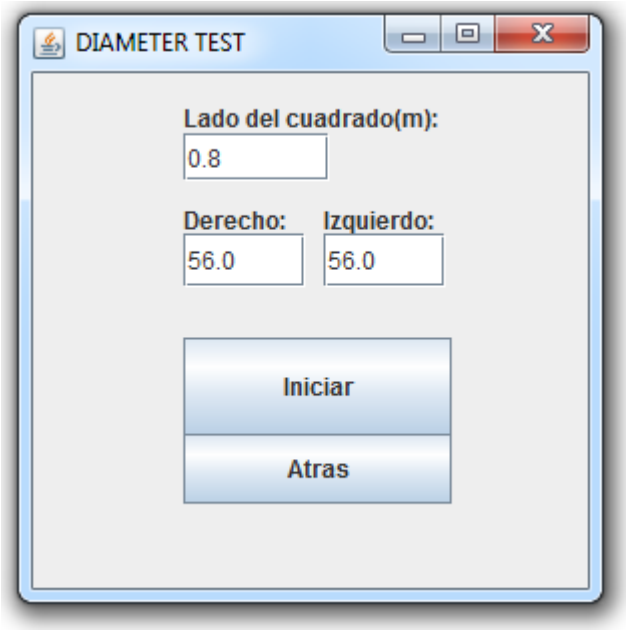


Figura 86.Diameter test

Se especifican las dimensiones del cuadrado, así como el diámetro de cada rueda que según las especificaciones es de 56mm. No obstante, el Lego al no ser unas ruedas duras y rígidas como las del e-puck, éste valor no suele ser correcto. Al pulsar en iniciar, el robot debe hacer el cuadrado perfecto, si el cuadrado se abre significa que habrá que aumentar en un factor delta la rueda interior a la trayectoria del cuadrado y disminuir el mismo valor delta en la rueda exterior. Si, por el contrario, se cierra demasiado, habrá que disminuir la rueda interior y aumentar la exterior.

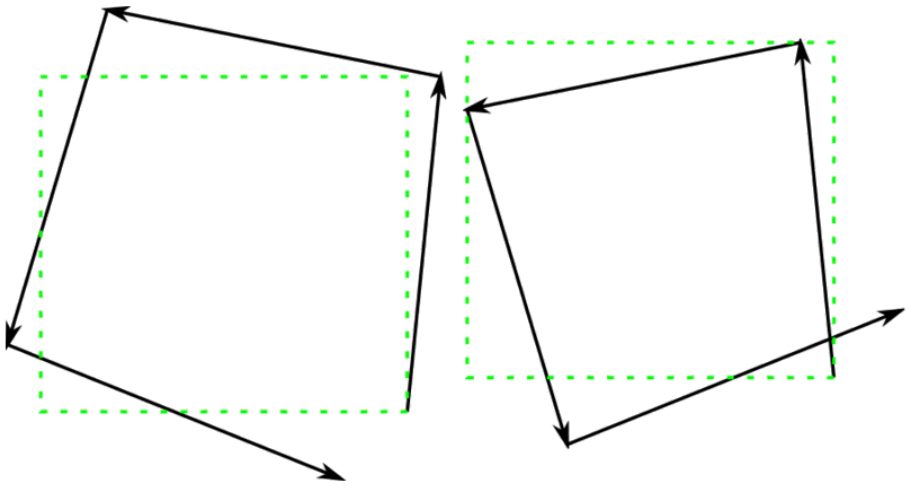


Figura 87. Cuadrado muy abierto y cuadrado muy cerrado.

Cuando haga el recorrido correctamente, habrá que guardar el valor del diámetro de cada rueda.

Por último, el test de escala o *Scaling factor* se encarga de calibrar el factor de escala entre la distancia real recorrida y la que el robot piensa que ha recorrido siguiendo la relación:

$$ScalingFactor = (real\ distance)/(parameter\ distance)$$

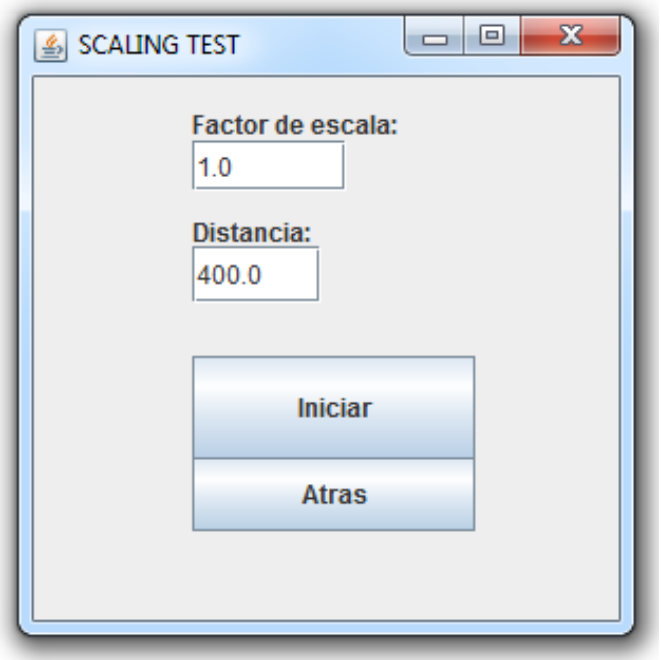


Figura 88. Scaling test

Por defecto el factor de escala tiene valor uno, lo cual quiere decir que es totalmente fiel a la distancia que recorre. Esa distancia se determina también desde la interfaz y luego se verifica con algún instrumento de medida. Es también uno de los test más complicados de conseguir.

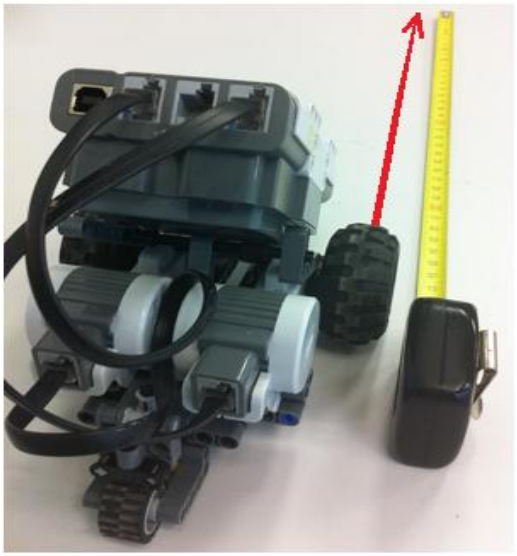


Figura 89. Lego NXT realizando el Scaling test

Tanto Ratio Test como Diameter Test, son los test más decisivos para una correcta calibración. También hay que tener en cuenta que los test se deben empezar con valores más o menos fiables y distancias cortas y poco a poco aumentar las distancias para hacer que resalte la acumulación de errores y así se redefinan los parámetros.

La clave para una buena calibración es tener vista y saber en qué medida ir variando los parámetros del robot para que se obtengan los resultados esperados.

3.3.2.3 Soluciones a los errores no sistemáticos de la odometría

Se ha descrito un método para evitar o para disminuir en gran medida la acumulación de errores sistemáticos en el cálculo de la odometría, pero ¿Cómo abordar el problema de los errores no sistemáticos que ocurren en tiempo real? Patinaje de ruedas por suelos resbaladizos o sobre-aceleración, desniveles o fuerzas externas que interactúan con el robot.

Una opción es instalar sensores en el robot como acelerómetros, giroscopios e incluso una brújula, y mediante un filtro de Kalman estimar la posición ya no sólo

partiendo únicamente de los valores de los encoders sino a partir también de los valores de todos los sensores. De esta manera, se consigue ampliar bastante el tiempo de precisión del cálculo de la odometría, no obstante pasados varios minutos, el algoritmo comienza a acumular errores y de nuevo la odometría resulta imperfecta.

Es entonces cuando surge la idea de hacer uso de un sistema de visión a modo de vigilancia y localización. Unas cámaras cenitales que gobiernen el escenario por donde se mueven los robots y que, en caso de que las covarianzas del cálculo de la odometría sean muy altas, la cámara ofrezca la posición real a los robots y éstos reinicien sus cálculos de odometría.

Para hacer uso de este método, es necesario reconocer los robots desde las cámaras y decidir qué metodología utilizar para las consultas de su localización. Todo esto se recoge en el siguiente apartado.

3.3.3 Sistemas de visión

Con el propósito de cubrir el mayor área de suelo posible del laboratorio, se instalaron dos cámaras cenitales modelo Logitech Webcam Pro 9000, ya descritas en el apartado 3.3.1.1.

Se instalaron en el techo de manera que sus campos de visión se solapasen unos metros para poder tener la opción de usar estereovisión en un futuro. Una vez instaladas en su sitio, el siguiente paso fue calibrarlas.

3.3.3.1 Calibración

Al tratarse de dos cámaras que comparten cierta área en común, lo deseable era que compartiesen el mismo sistema de coordenadas global también.

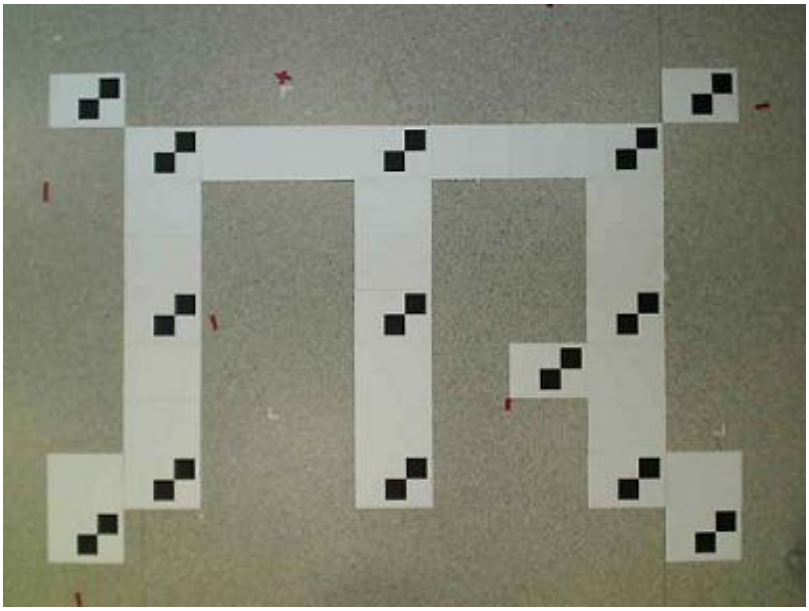


Figura 90. Primera plantilla de calibración.

Para ello se intentó montar una plantilla a partir de folios blancos con cuadrados negros imprimidos de dimensiones conocidas como la que se muestra en la figura 90, con 28 puntos, pero la imperfección de las impresiones sumadas a la imprecisión de situar los folios a mano uno junto a otro, desembocó en la idea de imprimir una plantilla en tres únicos paneles de papel de dimensiones 5x0.9m, para cubrir los casi 14 metros cuadrados de escenario.

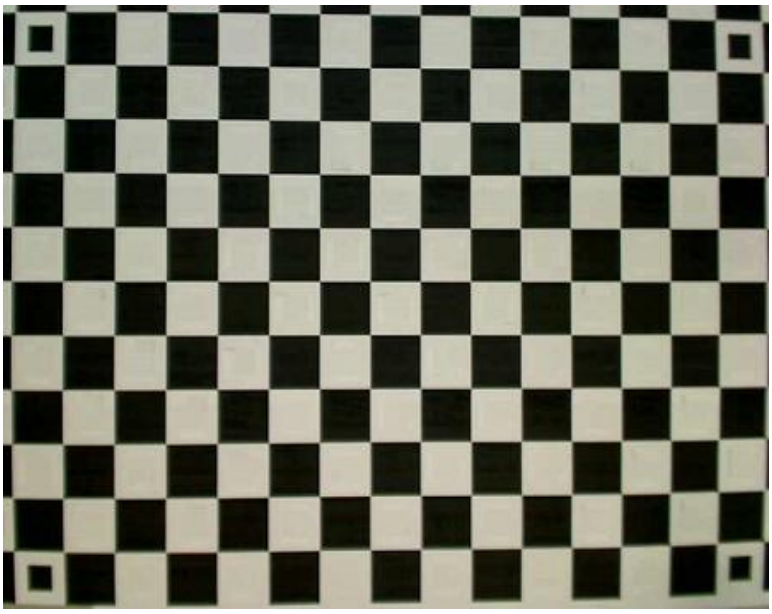


Figura 91. Plantilla de calibración definitiva.

De esta manera las dos cámaras compartían también la misma plantilla creada tan sólo uniendo los tres paneles y reduciendo errores de separación entre los cuadrados.

El método de calibrado escogido fue el de mínimos cuadrados. Las capturas se procesaron con la librería Image Processing Toolbox de Matlab, para discretizar los cuadrados y calcular los valores en píxeles de los centroides. Las medidas reales venían determinadas por el fichero de Autocad que definió la plantilla, no obstante se volvió a medir manualmente con un metro para corregir algunas imperfecciones de la impresión.

Figura 92. Plantilla de calibración.

Con el fin de realizar un correcto calibrado, se llevó a cabo un estudio sobre la relación entre la fidelidad de la estimación de la matriz de calibración y el número de puntos utilizados para su cálculo.

Las siguientes figuras representan el estudio de la precisión de las matrices de calibración de la cámara 1 y la cámara 2 calculadas a partir de 81 puntos (estrellas y líneas rojas) y calculadas a partir de 61 puntos (círculos y líneas azules).

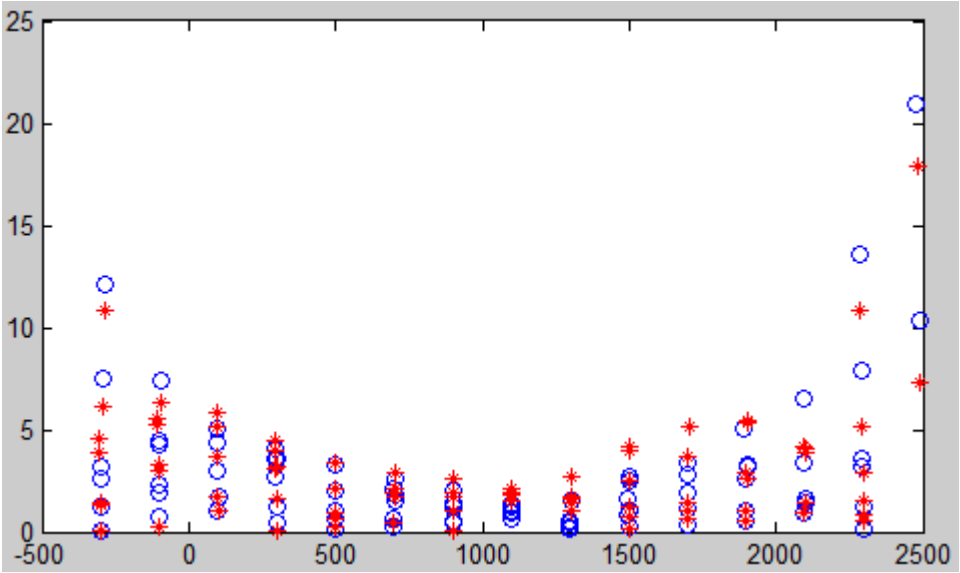


Figura 93. Error de la cámara 1

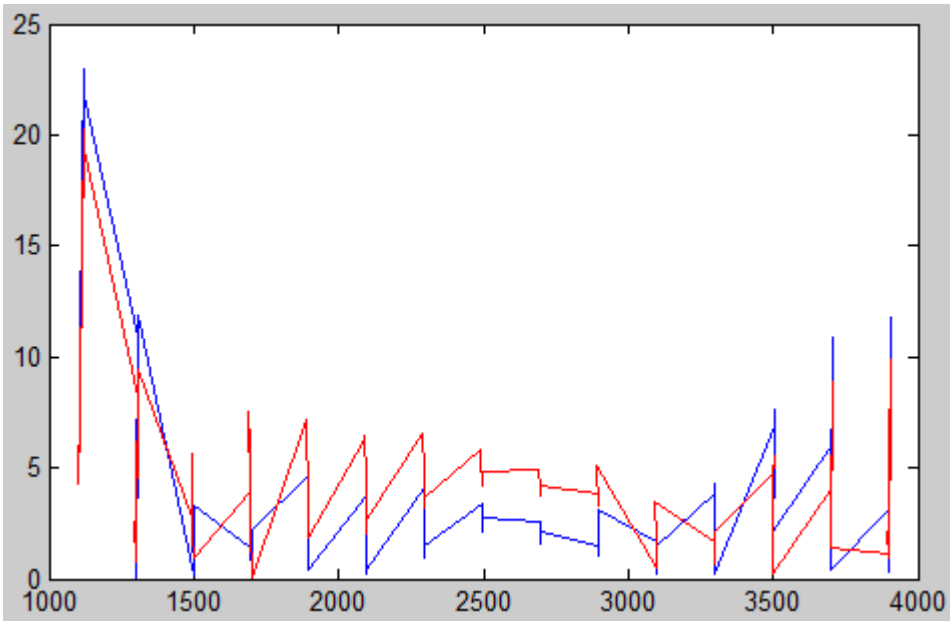


Figura 94. Error cámara 2.

Como se puede observar, en general la estimación es más fiel a la realidad cuanto más cerca del centro de la cámara se sitúan los puntos. A medida que los puntos se alejan del centro, el error aumenta exponencialmente. Esto es debido sin duda a la deformación de la lente de la cámara que distorsiona las esquinas y los extremos de la imagen.

Comparando la diferencia de calibración según el número de puntos utilizados, cabe destacar que a pesar de que con menor número de puntos (color azul), en el

centro la estimación de la posición es más precisa, en los extremos, conforme se alejan del centro, la pendiente exponencial del error es mayor llegando a dar una cota máxima de error de 20.94mm en la cámara 1 y de 23.07mm en la cámara 2. Mientras que por otro lado, con mayor número de puntos (color rojo) la estimación en el centro no es tan precisa pero en los extremos el aumento del error es algo más relajado, ofreciendo una cota de error máxima de 17.9mm en la cámara 1 y 20.33mm en la cámara 2. Por lo tanto, finalmente se optó por el uso de las matrices de calibración calculadas a partir de 81 puntos.

Como ya se ha comentado, la razón de que la calibración sea más exacta en el centro que en los extremos es debido a la deformación de la lente. Se propone y se plantea para un futuro, mejorar la precisión de las matrices de calibración de las cámaras mediante un método alternativo que tenga en cuenta la deformación angular de la cámara.

3.3.3.2 Detección de señuelos y posiciones

Una vez están calibradas las cámaras, se plantea la manera de reconocer varios robots mediante distintos señuelos que se les instalen encima.

Para la gestión de la visión se opta por hacer uso de la librería JavaCV descargable libremente desde la dirección http://code.google.com/p/javacv/ y donde además se incluyen ejemplos y manuales. Esta librería es un amplio wrapper o traductor que ofrece prácticamente todas las opciones disponibles en la librería OpenCV para el lenguaje de programación C. Para su uso tan sólo hay que descomprimir los archivos e incluir en el Java Build Path del proyecto las librerías *javacpp.jar*, *javacv-windows-x86_64.jar*, *javacv-windows-x86.jar* y *javacv.jar*.

En cuanto al tipo de señuelos, surgen dos formas: triángulos isósceles y círculos.

3.3.3.2.1 Triángulos

Figura 95. Robot Lego con señuelo.

Los señuelos en forma de triángulos isósceles ofrecen un posicionamiento preciso a partir del cálculo de su centroide, además de que el cálculo de la orientación es sencillo de hallar mediante el cálculo de la mayor distancia desde el centroide del señuelo al punto más alejado que forma parte del perímetro. Para reconocer los distintos robots se utilizan distintos colores y se también se varía el tamaño de los triángulos.

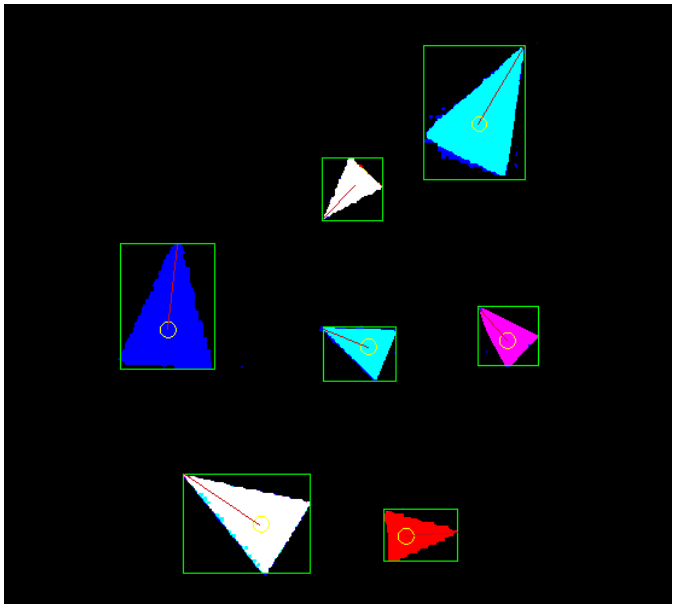


Figura 96. Discretización de triángulos con centroides y orientación.

No obstante, cuando el número de robots es elevado, el reconocimiento de los distintos colores ya no resulta tan sencillo y sobre todo, cuando se trata con robots como el Robotino, de grandes dimensiones, la discretización entre el señuelo y el resto del robot se complica. Además también existe el problema de la deformación de la lente en los extremos del campo de visión, lo que en ocasiones puede provocar que los triángulos isósceles no ofrezcan una correcta orientación al deformarse por la cámara.

Es por todo esto por lo que se plantea otro tipo de señuelo que no sea dependiente de la variedad de colores de los señuelos disponibles, y que además ofrezca una buena respuesta para el cálculo de la orientación frente a las deformaciones que puede sufrir.

3.3.3.2.2 Círculos

La idea de usar círculos surge para intentar solventar las dificultades con los triángulos comentadas en el apartado anterior. La plantilla de estos señuelos circulares responde a la que se muestra en la siguiente figura.

Figura 97. Diseño del señuelo circular.

Todos los círculos son en blanco y negro puesto que con este formato, cada robot se distingue por el número de circunferencias concéntricas definidas en su señuelo y también por la línea que dicta su orientación. En las siguientes figuras se muestra sobre la imagen real capturada las correspondientes detecciones de los distintos robots y su discretización.

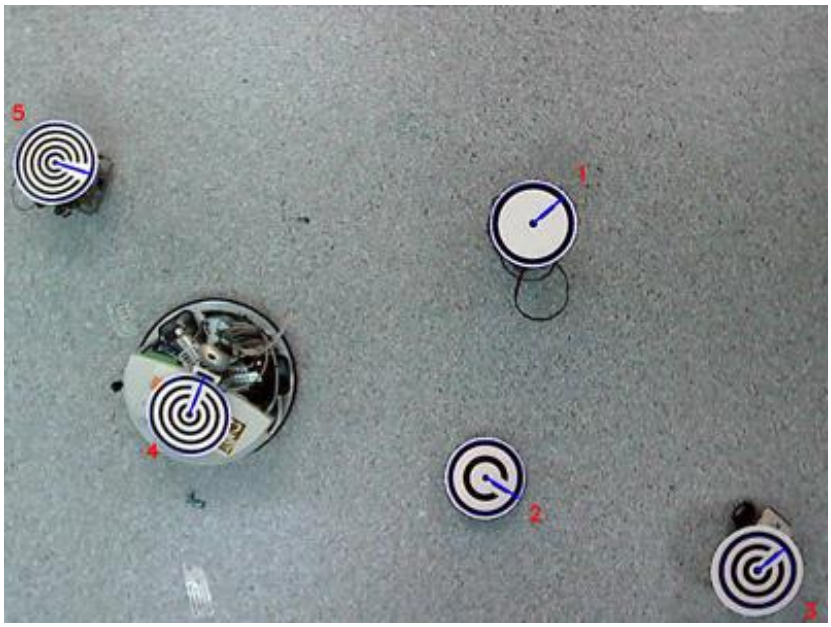


Figura 98. Imagen real con información de detección superpuesta.

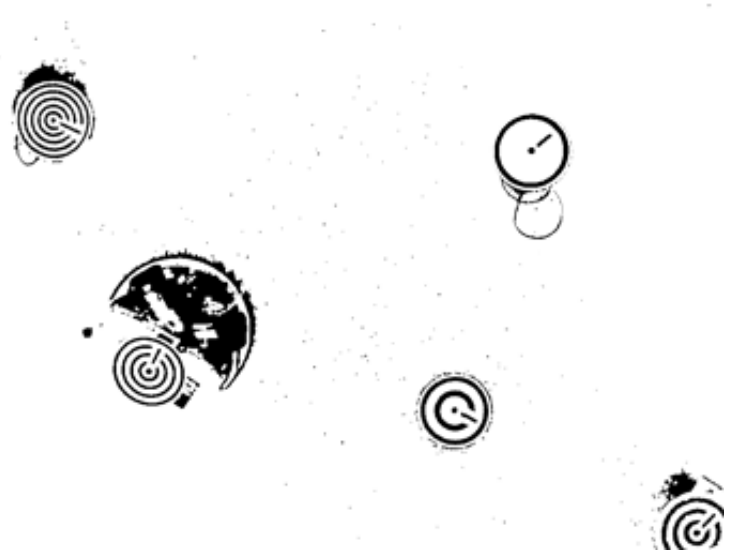


Figura 99. Resultado de la discretización.

El análisis de la imagen se realiza en varios pasos. Primero se umbraliza la imagen de manera que queden tan sólo los negros más oscuros y después se hace uso de una función de detección de círculos que ofrece la librería. Luego tan sólo queda calcular los centros de los círculos y clasificarlos para saber si son concéntricos y según su número diferenciar entre el resto. El segmento desde el centro se detecta estimando su longitud y posición, que debe estar cerca a la distancia del radio de la circunferencia más externa y así definir la orientación.

Este tipo de señuelos tienen la pega de que el número de círculos concéntricos que puede incluir un señuelo es dependiente de su tamaño. Si se desea aumentar en gran medida el número de robots a reconocer, se debe aumentar el tamaño de los señuelos dado que hace falta que los círculos tengan un grosor mínimo para que sea detectados por la cámara y reconocidos por la función de la librería.

3.3.3.3 Gestión de la cámara

La idea principal del uso de la cámara es la de corregir o confirmar la posición de un robot o bien perdido o bien con unos valores de covarianza muy elevados, es decir, un alto grado de incertidumbre respecto a su posición. Mediante la consulta de su posición a la cámara, el control puede reiniciar su odometría y de este modo anular los errores acumulados desde que comenzó su control. Hay que tener en cuenta que si el robot continúa moviéndose mientras se realiza la consulta, la posición que reciba desde la cámara, no será su posición actual correcta, si no la posición correcta donde tendría que haber estado cuando se capturó la imagen desde la cámara.

Bajo estas premisas surgen dos estrategias para gestionar la actualización de la posición de un robot: las consultas periódicas y la consulta mediante evento.

3.3.3.3.1 Recepción periódica de posiciones

Como bien su propio nombre indica, la estrategia de recepciones periódicas consiste en que cada cierto intervalo de tiempo el robot espera recibir su posición determinada por la cámara.

Desde el lado del procesamiento de la cámara en el PC, siempre existirán dos procesos, uno que captura, procesa la imagen y actualiza los valores y otro encargado de enviar la posición de los robots. Unificar los dos procesos en uno sólo, aumentaría el tiempo de comunicación dado que al tiempo de recepción y envío de datos habría que sumarle el de la captura y procesamiento de la imagen, lo cual no es muy adecuado ya que mientras tanto la odometría del robot continúa trabajando acumulando errores.

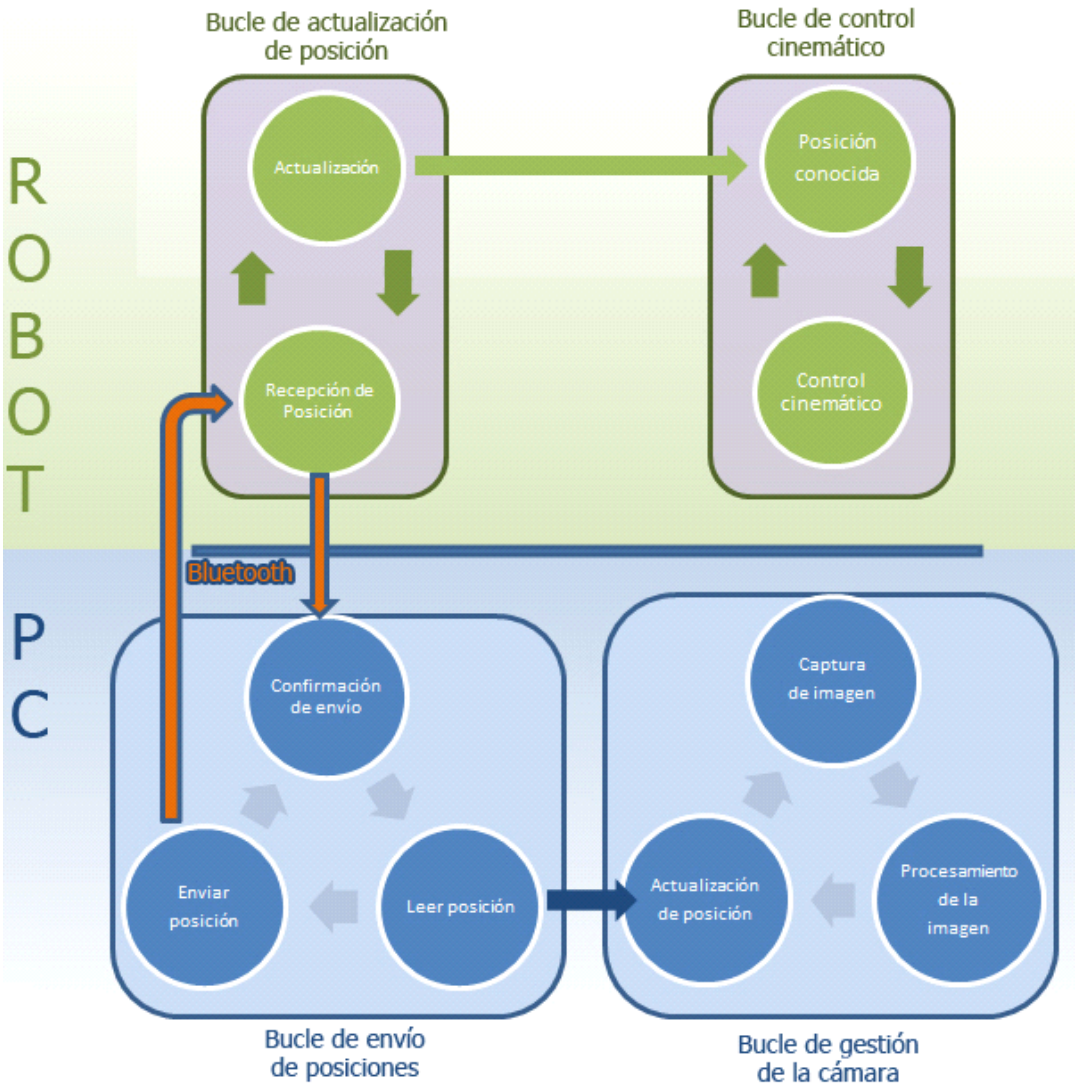


Figura 100. Esquema de procesos de la recepción periódica

En cuanto al robot, el bucle de control cinemático de cualquier robot tiene un periodo de muestreo bastante rápido (normalmente no superior a los 50ms) y el tiempo medio de recepción de la información de la cámara puede superar los 50ms. Luego también es necesario tener dos procesos independientes, uno que gestione el control cinemático y otro que gestione la actualización de la posición mediante recepciones periódicas desde la cámara. Es por eso, que este método, a pesar de llevar un seguimiento más fiable y más ceñido al posicionamiento real, cuanto más breve es el periodo de actualización de la cámara, también consume más recursos del robot y por ejemplo causa problemas en sistema de tiempo real o en sistemas con prioridades fijas donde al algoritmo de control siempre se le otorga mayor prioridad que al de consulta.

3.3.3.3.2 Consultas mediante eventos

Con el fin de no saturar el procesamiento en el robot y unificar los dos procesos en uno sólo consumiendo un menor número de recursos, se ha implementado la gestión de la cámara por consultas mediante eventos. De esta manera, sólo cuando el robot considera que su incertidumbre de posicionamiento es muy alta y desea actualizar su posición real, entonces manda una consulta a la cámara y ésta le envía la información con su posición. Puede ser el mismo algoritmo de control quien, si determina que necesita actualizar su posición, genera un hilo nuevo que enviará una consulta a la cámara y que una vez consiga la respuesta, actualizará los valores en el bucle de control principal y luego morirá.

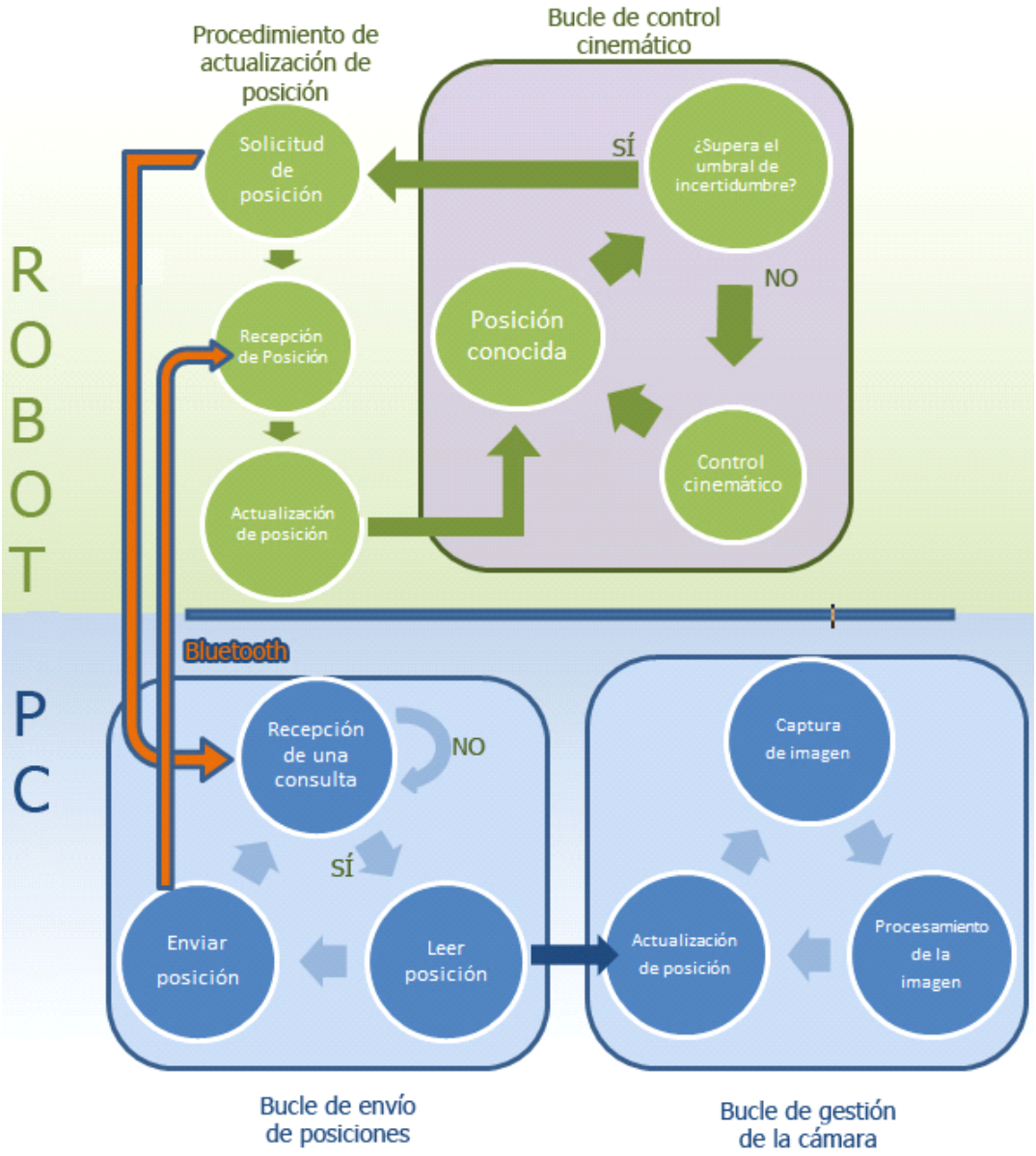


Figura 101. Esquema de procesos de las consultas mediante eventos.

Con esta estrategia el tráfico de las comunicaciones es mucho menor y se optimiza el procesamiento que debe realizar el robot al suprimir el bucle de comunicaciones de la estrategia de gestión periódica.

En cuanto a la gestión en el computador, la única diferencia con la estrategia anterior es que el proceso que antes enviaba posiciones periódicamente, ahora espera a recibir una solicitud de consulta para enviar la posición del robot.

3.3.4 Estrategias del control cinemático de robots móviles

Se han estudiado e implementado las dos principales estrategias de control cinemático para robots móviles: el control de trayectorias mediante la estrategia del punto descentralizado y el seguimiento de caminos con algoritmo de persecución pura.

3.3.4.1 Control de trayectoria.(Punto descentralizado)

El control de posición por punto descentralizado se establece a partir de la posición y velocidad de un punto que está separado una distancia ***e*** del eje de tracción del robot.

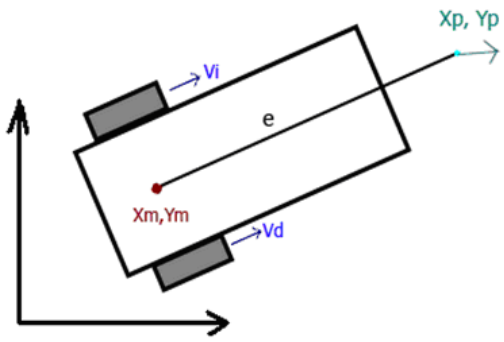


Figura 102. Punto descentralizado.

De este modo las coordenadas del punto descentralizado vienen definidas por:

$$\begin{bmatrix} \dot{x_p} \\ \dot{y_p} \end{bmatrix} = \begin{bmatrix} \dot{x_p} + e\cos\theta \\ \dot{y_p} + e\cos\theta \end{bmatrix}$$

Se calcula la derivada (velocidad) de ese punto:

$$\begin{bmatrix} \dot{x_p} \\ \dot{y_p} \end{bmatrix} = \begin{bmatrix} \dot{x_m} - e\cos(\theta)\dot{\theta} \\ \dot{y_m} + e\cos(\theta)\dot{\theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -e\sin(\theta) \\ 0 & 1 & e\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{x_m} \\ \dot{y_m} \\ \dot{\theta} \end{bmatrix}$$

Para conocer la posición del robot en todo momento se parte de la ecuación cinemática directa diferencial:

$$\begin{bmatrix} \dot{x_m} \\ \dot{y_m} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 1/2 \\ -1/2b & 1/2b \end{bmatrix} \begin{bmatrix} v_i \\ v_d \end{bmatrix}$$

Donde sustituyendo queda:

$$\begin{bmatrix} \dot{x_p} \\ \dot{y_p} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -e\sin(\theta) \\ 0 & 1 & e\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{x_m} \\ \dot{y_m} \\ \dot{\theta} \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 & -e\sin(\theta) \\ 0 & 1 & e\cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 1/2 \\ -1/2b & 1/2b \end{bmatrix} \begin{bmatrix} v_i \\ v_d \end{bmatrix}$$

$$\begin{bmatrix} \dot{x_p} \\ \dot{y_p} \end{bmatrix} = \frac{1}{2b} \begin{bmatrix} b\cos(\theta) + e\sin(\theta) & b\cos(\theta) - e\sin(\theta) \\ b\sin(\theta) - e\cos(\theta) & b\sin(\theta) + e\cos(\theta) \end{bmatrix} \begin{bmatrix} v_i \\ v_d \end{bmatrix}$$

Las velocidades de las ruedas se pueden obtener despejando de la ecuación anterior:

$$\begin{bmatrix} v_i \\ v_d \end{bmatrix} = \frac{1}{e} \begin{bmatrix} e\cos(\theta) + b\sin(\theta) & e\sin(\theta) - b\cos(\theta) \\ e\cos(\theta) - b\sin(\theta) & e\sin(\theta) + b\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{x_p} \\ \dot{y_p} \end{bmatrix}$$

Finalmente, para el desarrollo del control cinemático de los robots se ha utilizado la siguiente ecuación:

$$\begin{bmatrix} \dot{x_p} \\ \dot{y_p} \end{bmatrix} = \begin{bmatrix} k_{v_x} \dot{x_{ref}} \\ k_{v_y} \dot{y}_{ref} \end{bmatrix} + \begin{bmatrix} k_{p_x} & 0 \\ 0 & k_{p_y} \end{bmatrix} \begin{bmatrix} x_{ref} - (x_m + e\cos\theta) \\ y_m - (y_m + e\sin\theta) \end{bmatrix}$$

Entre las ventajas de la estrategia del punto descentralizado cabe destacar que no es necesario conocer la trayectoria completa para aplicarla sino que al mismo tiempo que se calcula la velocidad necesaria para que el robot alcance la siguiente posición, se calcula el siguiente punto en función de la posición actual donde se encuentra el robot en ese momento. Lo cual ofrece precisión y una continuidad uniforme en el movimiento.

3.3.4.2 Seguimiento de camino. (Persecución pura)

A diferencia de la estrategia del punto descentralizado, para aplicar el algoritmo de persecución pura normalmente es necesario conocer la trayectoria previamente o al menos varios puntos futuros por donde se desea pasar.

Dada una posición (X_R,Y_R) y un punto objetivo (X_{ob},Y_{ob}) se puede analizar el recorrido como sigue:

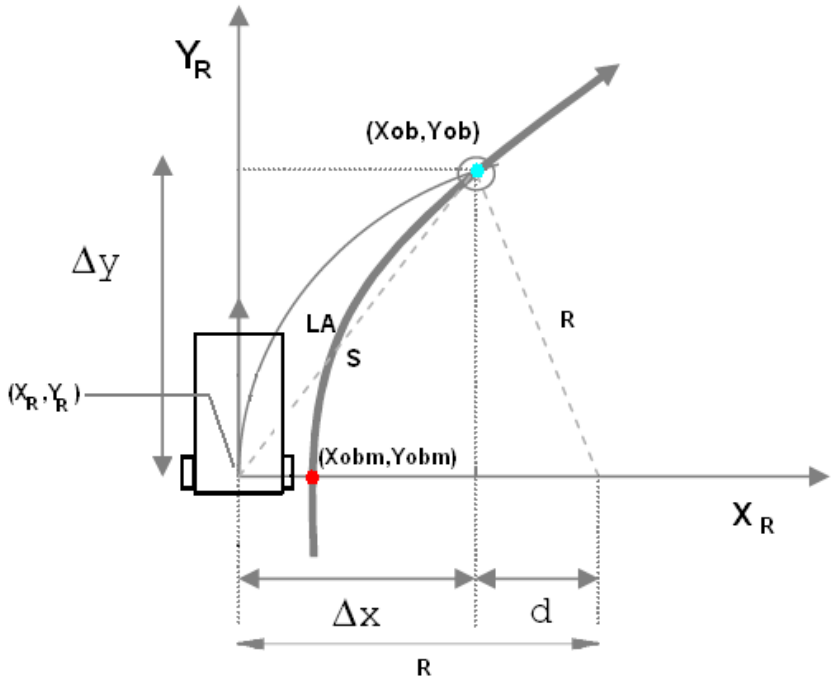


Figura 103. Persecución pura.

Para una velocidad V constante, en cada período de control se calcula el punto (X_{obm},Y_{obm}) del camino que esté más próximo al robot (X_R,Y_R) y se elige el punto objetivo (X_{ob},Y_{ob}) situado a una distancia fija S.

Partiendo de la ley de los catetos donde:

$$\Delta x^2 + \Delta y^2 = LA^2$$

Del esquema se puede deducir:

$$\Delta x + d = R \rightarrow d = R - \Delta x$$

Y sustituyendo y desarrollando se puede calcular la curvatura (γ) como:

$$(R - \Delta x)^2 + \Delta y^2 = R^2$$

$$R^2 - 2R\Delta x + \Delta x^2 + \Delta y^2 = R^2$$

$$2R\Delta x = LA^2 \rightarrow R = \frac{LA^2}{2\Delta x}$$

$$\gamma = \frac{1}{R} = \frac{2\Delta x}{LA^2}$$

La distancia LA (Look At) estará definida en función de las coordenadas actuales y del punto objetivo:

$$LA = \sqrt{(x_{ob} - x_R)^2 + (y_{ob} - y_R)^2}$$

$$\Delta x = (x_{ob} - x_R)\cos\theta + (y_{ob} - y_R)\sin\theta$$

Dada una velocidad de referencia (v_{ref}) la velocidad lineal y angular se corresponderá a:

$$v = v_{ref}$$

$$w = v_{ref}\gamma$$

$$\gamma = \frac{2(x_{ob} - x_R)\cos\theta + (y_{ob} - y_R)\sin\theta}{(x_{ob} - x_R)^2 + (y_{ob} - y_R)^2}$$

Y, por último, se calculan las velocidades de cada rueda como sigue:

$$v_d = v + w\gamma$$

$$v_i = v - w\gamma$$

De esta manera se aplican las respectivas velocidades a los motores para que el robot siga la trayectoria que se propone.

El hecho de deber conocer y si no, en peor caso, tener que calcular los puntos del camino que debe recorrer previamente el robot a la ejecución de las acciones de control cinemático, aumenta el coste y con ello el tiempo de la ejecución. Sin embargo ofrece la ventaja de poder asegurar que el robot pase por puntos clave y que siga el camino de una forma más estricta, además de que es una estrategia de recorrido de trayectoria que no depende del tiempo.

Si por ejemplo el robot encuentra un obstáculo mediante el control por punto descentralizado, el robot se aleja de la trayectoria para evitar la colisión y durante todo ese periodo de tiempo, el algoritmo continúa generando nuevos puntos para la trayectoria de manera que cuando el robot se libere del peligro de colisión, el punto objetivo de la trayectoria habrá avanzado dejando puntos intermedios sin contemplar y el robot recortará la trayectoria para intentar alcanzar el punto objetivo actual. Eso no ocurre en la persecución pura ya que en el momento en el que ya no exista peligro

de colisión, el algoritmo de control buscará el punto de la trayectoria más cercano a la posición actual del robot y ese se marcará como objetivo. Esta diferencia también es apreciable en el siguiente ejemplo, donde el robot intenta realizar una trayectoria circular comenzando en la coordenada -50,-45 y con una orientación de 0 grados.

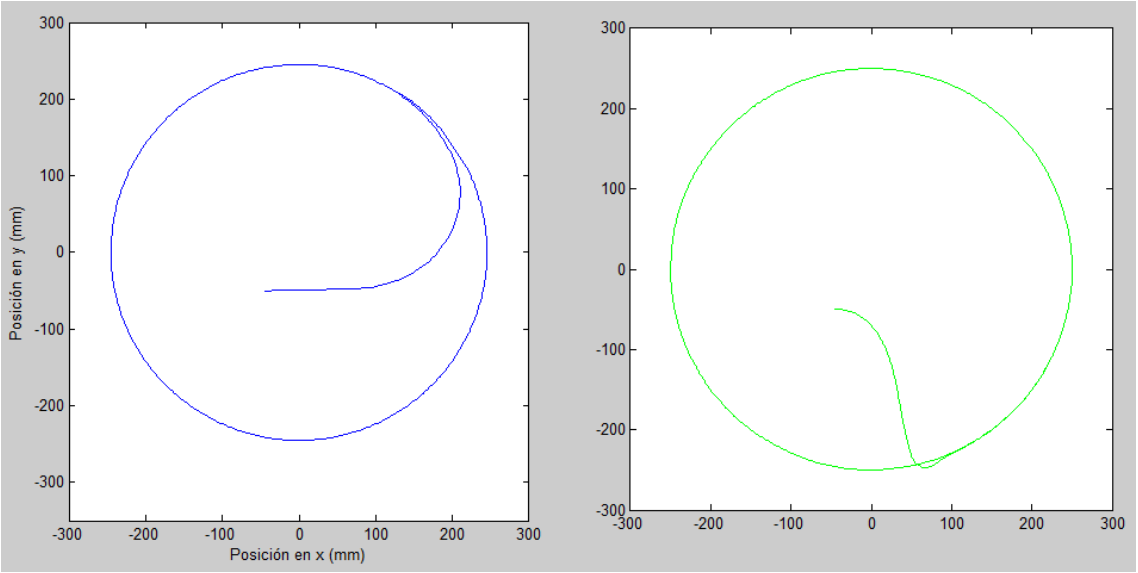


Figura 104. Persecución pura (izq) y punto descentralizado (der).

Cuando el robot intenta alcanzar la trayectoria, con el control por persecución pura se dirige hacia el punto más cercano a su posición actual teniendo en cuenta su orientación y una vez alcanza ese punto se mantiene fiel a la trayectoria. En cambio, con el control por punto descentralizado, aunque la trayectoria comienza a generarse en el punto 0, 250, durante el tiempo que tarda el robot en acercarse a la trayectoria, el propio algoritmo cambia los puntos objetivos haciendo que el robot vaya recortando hasta alcanzar la trayectoria en la coordenada 250, 0.

Dependiendo del tipo de aplicación y de trayectoria, habrá que valorar los puntos fuertes y las deficiencias de los algoritmos de control y elegir el que más se adapte a sus necesidades.

3.4 Aplicaciones desarrolladas

3.4.1 Detección de posiciones mediante el AR Drone

El sistema de posicionamiento mediante una cámara cenital estática situada a una altura fija resulta sencillo mediante el cálculo de la matriz de calibración de la cámara que ofrece la relación entre píxeles y milímetros como ya se ha comentado anteriormente. Pero, ¿cómo se podría abordar la localización de un señuelo desde un sistema de vigilancia como podría ser el AR Drone que no se mantiene estable y varía tanto en posición como en orientación y altura? La siguiente aplicación pretende dar una solución al problema.

3.4.1.1 Descripción

Figura 105. Posicionamiento con AR Drone.

La aplicación hace uso de la cámara cenital que incorpora el cuadricóptero AR Drone, para localizar, posicionar y detectar la orientación de un robot móvil que se desplaza por el suelo describiendo una trayectoria cualquiera. Para ello se hace uso de un sistema de reconocimiento de balizas situado dentro del área de percepción de la cámara.

3.4.1.2 Metodología y desarrollo de la aplicación

Como ya se comentó en la descripción del cuadricóptero, el sistema Linux que incorpora está totalmente cerrado y tan sólo es posible la comunicación y el control del robot mediante el envío y la recepción de paquetes UDP y TCP a través de una aplicación que se ejecuta en un ordenador conectado por Wifi.

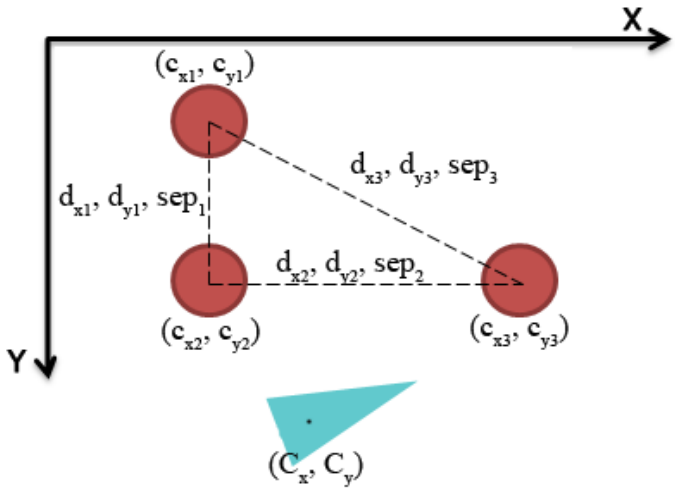
El lenguaje utilizado para el desarrollo de la aplicación ha sido JAVA, y el entorno ECLIPSE. Java ofrece un sencillo tratamiento de las conexiones por puertos UDP y TCP y además la posibilidad de crear una interfaz gráfica en pocos pasos. Como base para el desarrollo de la aplicación del control del AR Drone, se ha hecho uso de la librería JavaDrone disponible en http://code.google.com/p/javadrone/.

El sistema de balizas consta de tres cartulinas en forma de círculo de color rojo y el señuelo que incorpora el robot móvil tiene forma de triángulo isósceles de color azul. Para el procesamiento de la imagen se ha utilizado la librería JavaCV que no es más que un traductor para el acceso a las funciones de la librería OpenCV para C. El tratamiento que se realiza a la imagen es el de filtrado por color para distinguir las balizas del señuelo del robot y del resto de objetos que puedan haber en el escenario. Luego se realiza una umbralización y se obtienen los centroides de los objetos.

Figura 106. Discretización y posicionamiento desde la cámara cenital

El método de posicionamiento mediante tres balizas se basa en la condición de que las balizas deben situarse en una posición o coordenadas globales conocidas y con una forma y separación conocidas. Concretamente formando un triángulo equilátero.

Mediante visión, se calculan los centroides de las balizas $[c_{x1},c_{y1}]$, $[c_{x2},c_{y2}]$,$[c_{x3},c_{y3}]$ y las distancias y separación entre ellas.



$$d_{x1} = c_{x2} - c_{x1}$$

$$d_{y1} = c_{y2} - c_{y1}$$

$$sep_1 = \sqrt{d_{x1}^2 + d_{y1}^2}$$

$$d_{x2} = c_{x3} - c_{x2}$$

$$d_{y2} = c_{y3} - c_{y2}$$

$$sep_2 = \sqrt{d_{x2}^2 + d_{y2}^2}$$

$$d_{x3} = c_{x3} - c_{x1}$$

$$d_{y3} = c_{y3} - c_{y1}$$

$$sep_3 = \sqrt{d_{x3}^2 + d_{y3}^2}$$

Figura 107. Algoritmo de posicionamiento mediante balizas

La correspondencia de las balizas según su posición se realiza a partir de los valores de sus distancias. Una vez reconocidas, el factor de escala viene determinado por:

$$S = \frac{S_x + S_y}{2}$$

Donde $dist\_mm_x$ y $dist\_mm_y$ es la distancia real conocida de los catetos del triángulo. El ángulo de rotación alfa lo define:

$$\alpha = arctan\left(\frac{d_{y2}}{d_{x2}}\right)$$

En caso de que una baliza no coincida con el origen del eje de coordenadas global, habrá que calcular el desplazamiento del origen:

$$T_x = c_{x1} \qquad T_y = c_{y1}$$

Por último, la ecuación que devuelve el valor de las coordenadas en milímetros del objeto detectado en la posición c_x,c_y en píxeles será:

$$\begin{bmatrix} x_{mm} \\ y_{mm} \end{bmatrix} = (\text{int})\left(S \begin{bmatrix} \cos(\alpha) & \sin(\alpha) \\ -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \begin{bmatrix} c_x - T_x \\ c_y - T_y \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \right)$$

3.4.1.3 Sistema de archivos, interfaz y ejecución

Para llevar a cabo el desarrollo de la aplicación se ha partido del código fuente de la librería JavaDrone, descomprimiendo los archivos e importándolos a un nuevo proyecto. A pesar de que para el propósito de la aplicación no es indispensable, para que el proyecto compile sin problemas es necesario incluir en el Java Build Path las librerías *hidapi.jar*, *log4j-1.2.16.jar*, *SteelSeries-3.9.2.jar*, *trident-6.2.jar*. Además de las necesarias para el procesamiento de la imagen con JavaCV, *javacpp.jar*, *javacv-windows-x86_64.jar* y *javacv.jar*.

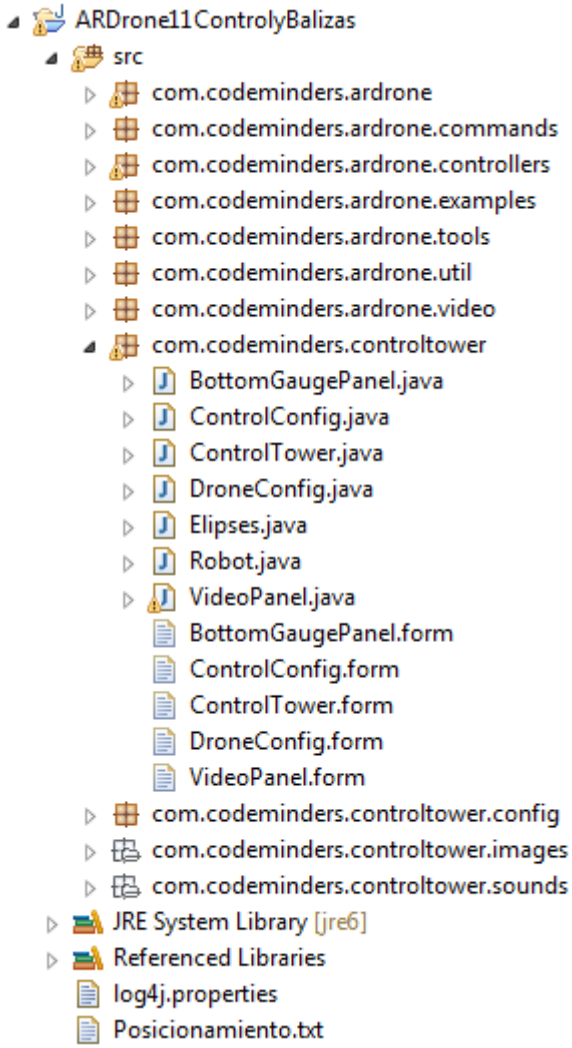


Figura 108. Sistema de archivos del proyecto.

La organización de los paquetes y archivos en el proyecto debe quedar como se muestra en la figura 108. La clase que contiene la función main para que se ejecute el programa es ControlTower.java.

Antes de ejecutar la aplicación, se debe desconectar cualquier dispositivo que pueda estar conectado al AR Drone y luego emparejarlo por Wifi al PC donde se ejecuta la aplicación. Para que se conecte correctamente, en ocasiones es necesario apretar el botón Unpair situado en la parte inferior del cuadricóptero, al lado del sensor de ultrasonidos.

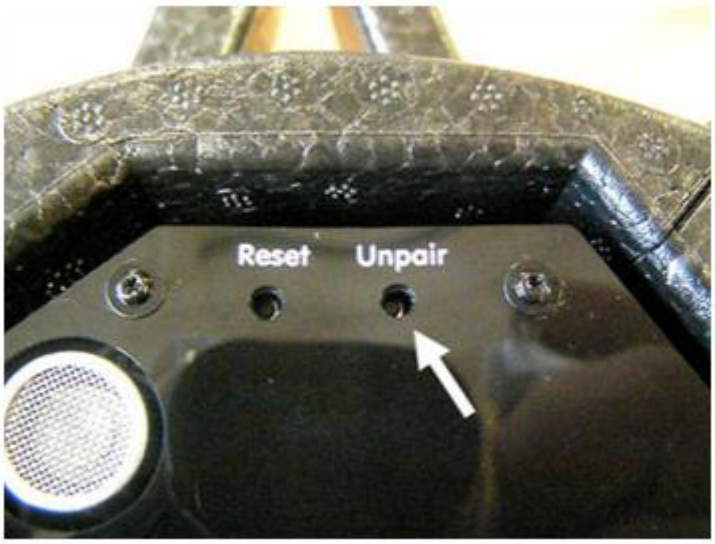


Figura 109. Botón Unpair.

Una vez conectado, al ejecutar la aplicación se mostrará la ventana principal (figura 109). El botón *mapping* sirve para configurar los controles en caso de querer utilizar un mando de la PS3 para controlar el robot, el de *instruments* oculta o muestra los indicadores gráficos de altitud, brújula, estado de la batería, etc. El botón de *tuning* configura la altitud máxima que se le permite al robot, el ángulo y las velocidades máximas. Mientras no se recibe la conexión con la cámara, se muestra el mensaje “No video connection”.

Para controlar el AR Drone se maneja desde el teclado con las teclas:

- *Espacio:* Puesta a punto.
- *Intro:* Despegar/Aterrizar.

- *Flecha arriba:* Ascender.
- *Flecha abajo:* Descender.
- *Flecha izquierda:* Gira hacia la izquierda.
- *Flecha derecha:* Gira hacia la derecha.
- *W:* Avanzar hacia adelante.
- *S:* Ir hacia atrás.
- *A:* Desplazarse hacia la izquierda.
- *D:* Desplazarse hacia la derecha.

Se trata pues de mantener el cuadricóptero de manera que en el área capturada por la cámara cenital se incluyan los tres señuelos y el robot que se desea posicionar. La misma aplicación va guardando los datos de la posición del robot en un fichero para que posteriormente sea posible su análisis o respresentación. En la siguiente figura se muestra una composición de la situación del AR Drone, la captura desde la cámara cenital y la representación de los datos recogidos de un robot móvil que sigue una trayectoria circular.

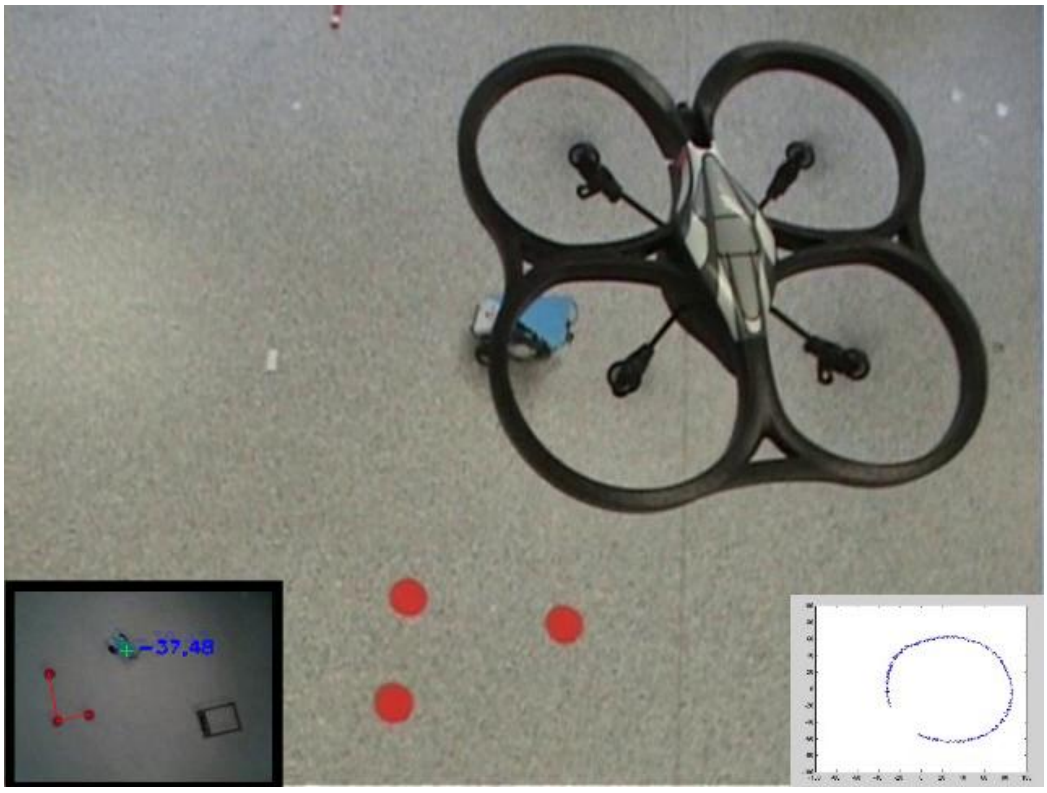


Figura 110. Composición de ejecución y resultados.

Los resultados son muy favorables teniendo en cuenta la inestabilidad de la cámara del AR Drone. La cota de error de los valores oscila en menos de 7 centímetros tanto en X como en Y.

3.4.1.4 Problemas encontrados y soluciones

Para que el algoritmo funcione, obviamente es necesario que la cámara capture las tres balizas y el señuelo del robot móvil dentro de la misma imagen. El principal problema, a parte de la escasa documentación disponible para la programación de una aplicación de control para el AR Drone, es principalmente la resolución y el ángulo de la lente de la cámara. Como ya se comentó en las especificaciones, la cámara tiene una resolución de 176x144 píxeles, lo que no permite realizar un buen procesamiento de la imagen por escasez de información. Además el ángulo de 64 grados de la lente no permite que el área capturada sea muy extensa. Y por otro lado, el sensor de altura y de estabilidad que incorpora el AR Drone lo hacen inestable cuando hay un objeto móvil moviéndose por debajo de él y se debe estar controlando manualmente para mantenerlo en una posición donde no pierda de vista las tres balizas y el señuelo.

Si se aumenta la altura del cuadricóptero, el área capturada aumenta pero la resolución de las balizas disminuye y si se disminuye la altura ocurre exactamente a la inversa. Por tanto, las soluciones adoptadas han sido mantener el AR Drone a una altura de 2.5 metros y adecuar el tamaño de las balizas y los señuelos de modo que sean lo más fácilmente posible de capturar dentro del área de la cámara y además que tengan una resolución mínima para poder llevar a cabo correctamente el procesamiento de la imagen.

3.4.2 El rescate

La aplicación de robots de rescate hace uso de un sistema multiagente para combinar y coordinar tres tipos de robots con distintas arquitecturas. Éstos, mediante la ayuda de una cámara cenital, colaboran para llevar a cabo un objetivo: el rescate de uno de ellos. La idea básica es asistir a un robot móvil que ha entrado en estado de emergencia debido, por ejemplo, a problemas con las baterías.

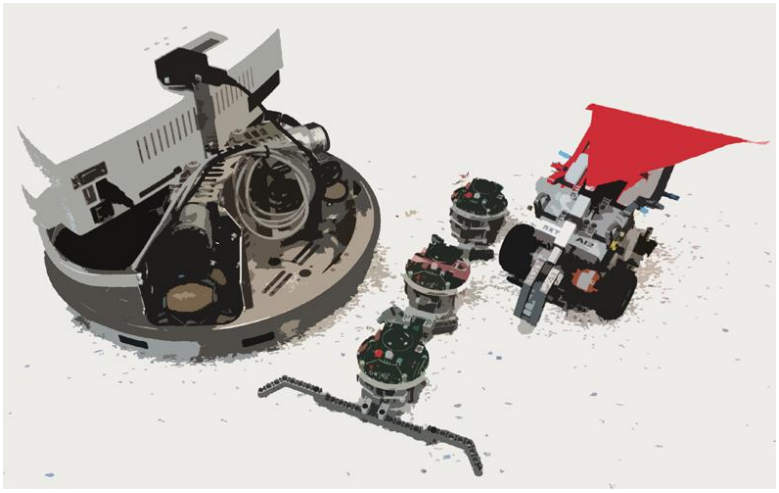


Figura 111. Robots de El rescate

3.4.2.1 Descripción

La aplicación se lleva a cabo en un escenario gobernado por el campo de visión de una cámara cenital. El Robotino con tres robots e-pucks en su remolque se encuentra en la base (o "taller"). Por el resto del escenario se mueve libremente un robot LEGO NXT y en un momento dado éste entra en estado de emergencia, enviando un mensaje de ayuda. En ese momento Robotino recibe la solicitud de ayuda, la acepta y recibe las coordenadas del NXT averiado a partir del procesamiento de la imagen obtenida mediante la cámara. Sale de la base y se sitúa a una distancia relativa al Lego de modo que los e-pucks quedan alineados con el robot averiado. Cuando el Robotino alcanza esa posición, envía la orden a los e-pucks para que bajen del remolque (Figura 112). Una vez queda libre de carga, se sitúa en el lado contrario del NXT y espera a que éste sea cargado en el remolque. En esta situación, los e-pucks reciben la acción de empujar al NXT y cuando entre los tres lo consiguen, (Figura 113) lo cual se detecta también mediante la cámara cenital, éstos retroceden y el Robotino vuelve a la base transportando consigo al NXT en su remolque (Figura 114).

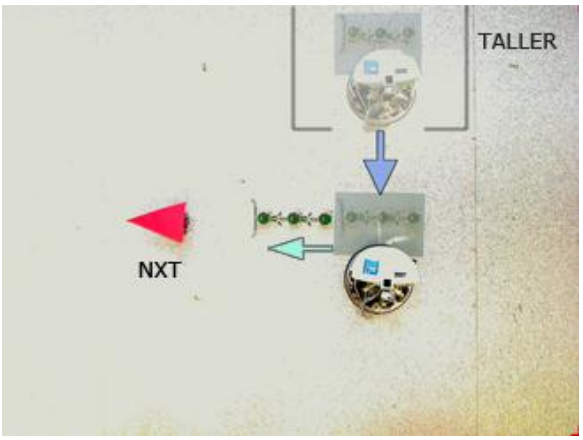


Figura 112. Inicio del rescate.

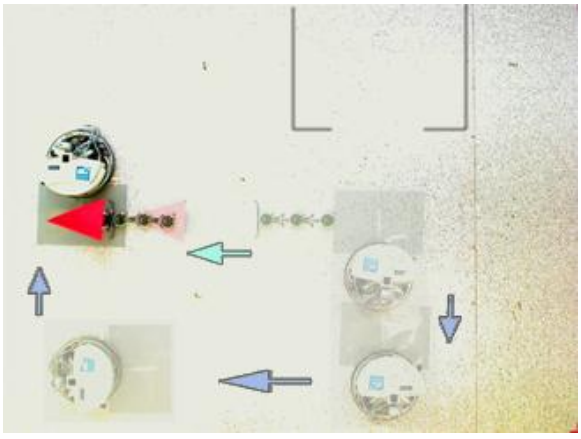


Figura 113. Carga del Lego NXT en el Robotino.

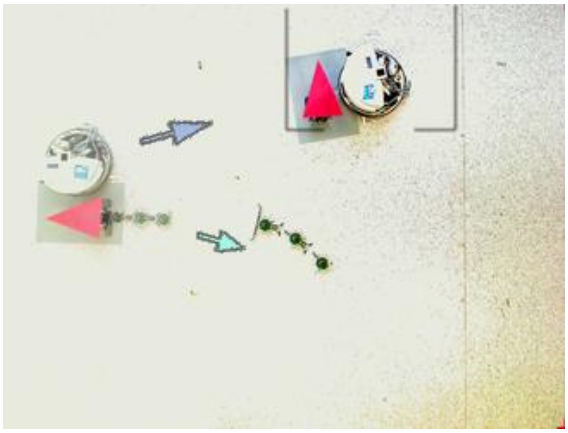


Figura 114. Traslado del LEGO al taller.

3.4.2.2 Metodología y desarrollo de la aplicación

En esta aplicación intervienen: un PC servidor, un Robotino, tres e-pucks, un Lego NXT y una cámara cenital. La arquitectura o el esquema de las comunicaciones se representan en la Figura 115, donde todos los elementos se conectan a un PC servidor a través de distintos protocolos de comunicaciones.

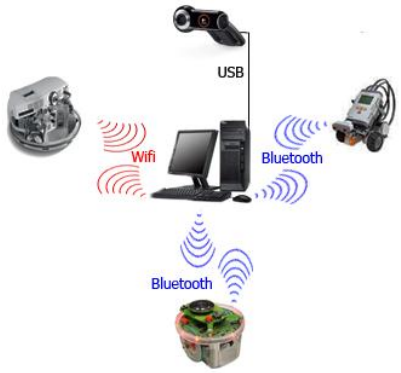


Figura 115. Esquema de comunicaciones.

El Lego NXT y los e-pucks se conectan mediante Bluetooth mientras que el Robotino lo hace a través de Wifi. Independientemente de las arquitecturas de programación de cada robot (Robotino en lenguaje C, NXT en Java y e-pucks en un lenguaje C bajo Erika Enterprise), la aplicación se gestiona desde el PC servidor mediante agentes en el entorno JADE. Los agentes que intervienen son:

- Un robot NXT que solicita el rescate cuando su nivel de baterías es demasiado bajo. Para facilitar el proceso de la detección de su posición y orientación mediante la cámara cenital se le ha colocado en la parte superior del robot un señuelo en forma de triángulo.
- Una cámara cenital.
- Un robot Robotino que para poder cargar y trasladar otros robots, se le ha añadido al chasis una plataforma a modo de remolque.
- Tres e-pucks dotados de unos acoples que facilitan la función de colaborar entre los tres a la hora de empujar algún objeto.

En cuanto al entorno de desarrollo de la aplicación se ha escogido Eclipse y la plataforma JADE, un software libre desarrollado en Java que permite la

implementación de sistemas multi-agente de una forma muy simple, permitiendo la programación de los distintos comportamientos de los agentes. Para la comunicación entre los agentes (el envío y la recepción de mensajes) se ha utilizado el esquema de comunicación directa de modo que los agentes pueden comunicarse entre sí y el sistema queda más distribuido. Cada agente tiene definidos unos estados en los que puede encontrarse y unos comportamientos que debe realizar según vayan ocurriendo sucesos.

Por otro lado, cada robot lleva cargado su propio programa desarrollado específicamente para esta aplicación en el lenguaje que cada uno ofrece. No obstante, la estructura de estos programas responde simplemente a una biblioteca de funciones básicas de movimiento y/o percepción como "gotoposition(x,y)" y que además terminan con envíos de reconocimientos o confirmaciones al agente que se gestiona en el PC.

Para esta aplicación, la gestión de la cámara se ha llevado a cabo mediante la librería de procesamiento de imagen en JavaCV, disponible en http://code.google.com/p/javacv/.

3.4.2.3 Sistema de archivos, interfaz y ejecución

Antes que nada, el PC debe estar conectado por Wi-fi al Robotino y debe haberse hecho el pairing con los tres e-pucks y el Lego NXT de manera que los puertos COM asignados a los e-pucks, sean los mismos que los asignados dentro de la aplicación. De no ser así, se deberán actualizar a los nuevos números de puerto.

A cada robot se le debe cargar su programa correspondiente, a los e-pucks mediante el MPLAB, al NXT mediante el pluging de LeJOS para Eclipse y al Robotino mediante una conexión por SSH por ejemplo.

Cuando ya están todos ejecutándose localmente en los robots, lo primero que inician estos programas es la espera de conexión con el PC. Para cargar el proyecto, el sistema de archivos debe quedar como muestra la siguiente figura.

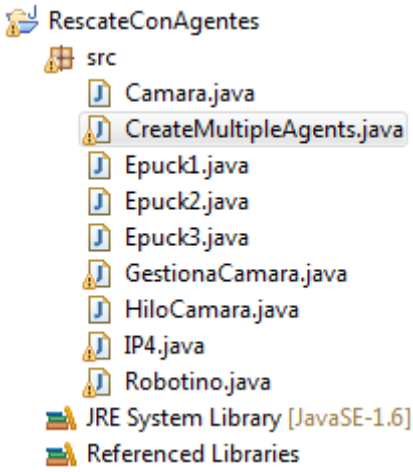


Figura 116. Sistema de archivos de El rescate.

Por supuesto el proyecto debe configurarse como se explicó en la sección 3.2.1 del presente documento e incluir en el Java Build Path las librerías del Lego: *bluecove.jar*, *pccomm.jar*; las de JADE: *commons-codec-1.3.jar*, *http.jar*, *iiop.jar*, *jade.jar*, *jadeTools.jar*, *jcommon-1.0.1.jar*, *jcommon-1.0.10.jar*, *jcommon-1.0.12.jar*, *jcommon-serializer-0.2.0.jar*, *jcommon-xml-1.0.1.jar*; las de JavaCom para la conexión por puerto de comunicaciones: *comm.jar*; y las del procesamiento de imagen: javacpp*.jar, javacv-windows-x86:64.jar, javacv-windows-x86* y *javacv.jar*.

Los archivos java tienen los nombres de los agentes que contienen. La clase que contiene el main a ejecutar es *CreateMultipleAgents.java*. Lo primero que ocurre al ejecutar es la creación de todos los agentes y cada uno su primer comportamiento es intentar realizar la conexión con su respectivo robot. A partir de ahí, cuando todos conecten y ocurra que IP4 solicite ayuda, comenzará la gestión de mensajes entre agentes y los cambios de comportamientos para llevar a cabo el rescate como se ha explicado en el apartado anterior.

3.4.2.4 Problemas encontrados y soluciones

La comunicación entre el PC y los e-pucks no fue trivial. Al principio se esperaba poder gestionar la conexión mediante la misma librería Bluetooth que controla el Lego NXT, pero no fue posible. Como solución se optó por gestionar las conexiones Bluetooth con los e-pucks tratando directamente la conexión a través de los puertos COM asignados a los e-pucks. Aun así el establecimiento de conexión con los e-pucks

sigue siendo el que más tiempo tarda durante toda la aplicación y en ocasiones continúa fallando.

La colaboración de los tres e-pucks empujando a un Lego NXT fue posible gracias a la instalación de unos acoples que guían y encajan entre ellos para facilitar la colaboración en esta acción. Al principio carecían de esta estructura pero por la forma redondeada de estos robots, al empujarse unos con otros perdían la orientación en la mayor parte de ocasiones.

Otro problema encontrado es la inexactitud del cálculo de odometría del Robotino, sobre todo a la hora de realizar giros. No obstante al ser un robot bastante grande y al instalarle un remolque acorde a su tamaño, su inexacto posicionamiento en pocas ocasiones suponía un problema para la correcta ejecución de la aplicación.

Por último, el problema que siempre surge trabajando con robots inalámbricos es que la precisión y el buen funcionamiento están relacionados directamente con el nivel de las baterías. Es por eso, que se debe comprobar que todos los robots que intervienen tengan al menos la mitad de batería disponible para llevar a cabo la aplicación.

4. Conclusiones y futuros proyectos

Todos los objetivos marcados al inicio de esta tesina han sido cumplidos satisfactoriamente. Se ha alcanzado un grado alto de familiaridad con la gestión de sistemas holónicos, en concreto con JADE y con la versatilidad que ofrece. Se han conocido a fondo las características de Eclipse como entorno de programación y como herramienta de depuración de código. Se ha estudiado la gestión de las comunicaciones entre distintos robots, así como sus tiempos de respuesta y sus limitaciones. También se han creado protocolos de comunicación para la confirmación de datos recibidos correctamente. Se han ampliado conocimientos sobre el manejo de varios hilos de ejecución en Java, la herencia de clases, la gestión de variables compartidas,… Se ha investigado sobre las distintas opciones y librerías que se ofrecen para el tratamiento de imágenes en Java, su discretización, reconocimiento de formas, cálculo de centroides, orientaciones, métodos de posicionamiento mediante sistemas de visión inestables, calibración… Se ha trabajado en la gestión y generación de trayectorias de robots móviles, analizando varias estrategias de control, mejorando los algoritmos clásicos… Se han conocido gran variedad de tipos de robots móviles, así como sus características y necesidades de programación. Se ha ampliado conocimientos sobre la odometría y los problemas que puede ocasionar y se ha investigado sobre posibles mejoras aplicables a cualquier tipo de robot. Se ha conocido y estudiado a fondo las capacidades y los problemas del manejo de un robot volador como el AR Drone así como el posicionamiento de robots mediante el método de las tres balizas a través de su cámara cenital. Por último, se ha desarrollado una aplicación basada en un sistema multiagente donde se hace uso de distintos tipos de robots con distintas características que se coordinan entre sí para lograr un objetivo común.

En definitiva, la tarea de investigar y estudiar los diferentes campos que se proponían, se ha visto recompensada y concluida en una aplicación que cubre con altas expectativas lo estudiado. Los problemas y complicaciones surgidos durante la realización de esta tesina, han derivado en soluciones que han motivado aún más la ambición de seguir trabajando e investigando, aún después de haber cumplido con los objetivos.

Para futuras investigaciones, en las que ya se está comenzando a trabajar, se propone unificar la aplicación del rescate con la de localización mediante el AR Drone de tal modo que se sustituya el uso de la cámara cenital para localizar el robot perdido, por la cámara cenital del cuadricóptero. Dado que el método de localización requiere el uso de tres balizas que se sitúen dentro de la escena donde se encuentra el robot perdido, estas balizas podrían ser perfectamente tres e-pucks que entran en modo exploración con el AR Drone. Una vez se localiza el robot perdido, el AR Drone lo posiciona a partir del cálculo de las posiciones de los e-pucks mediante odometría. Así los e-pucks funcionarían como balizas móviles ayudando al AR Drone a localizar y posicionar cualquier objeto.

Otra idea que se plantea, es utilizar los e-pucks como balizas móviles que ayudasen al cuadricóptero a posicionar una o varias marcas naturales del escenario. De este modo, los e-pucks podrían salir del escenario y el AR Drone tendría las referencias necesarias como para seguir posicionando a partir de las marcas naturales.

No hay que olvidar, que el AR Drone también posee una cámara frontal mediante la cual podría posicionarse dentro del sistema de coordenadas global del escenario, a partir del análisis del flujo óptico generado por el movimiento. Esta idea también se está gestando y es uno de los retos para resolver en el futuro. En la siguiente imagen se puede observar un ejemplo.

Figura 117. Flujo óptico a partir del movimiento del AR Drone.

En cuanto al desarrollo de aplicaciones multiagente, en esta tesina el uso de la elección de los comportamientos en JADE se realiza mediante prioridades asignadas a los comportamientos. Otra forma de escoger el comportamiento adecuado, podría ser mediante el uso de lógica difusa, donde cada comportamiento tuviera una ponderación o un determinado valor de coste y el objetivo global de los agentes no fuese sólo cumplir con una meta, si no hacerla cumplir de la forma más eficiente.

5. Bibliografía

Para el desarrollo y documentación del presente trabajo se han consultado las siguientes referencias:

[1] Oracle Java. http://www.oracle.com/technetwork/java/index.html

[2] LeJOS: Java for Lego Mindstorms. http://lejos.sourceforge.net/

[3] Wikipedia. http://es.wikipedia.org/wiki/Wikipedia:Portada

[4] Proyectos prácticos de electrónica y robótica. Jorge Flores Vergaray. http://jorgefloresvergaray.blogspot.com/2009/01/sensores-para-robotica.html

[5] Complubot. De educa Madrid de la universidad de Alcalá de Enares. http://complubot.educa.madrid.org/pruebas/lego_nxt_version_educativa/lego_nxt_version_educativa_index.php

[6] OPENCV: Processing and Java Library. http://www.ubaa.net/shared/processing/opencv/

[7] Christoph Bartneck, PH.D. http://www.bartneck.de/2008/03/04/java-lego-nxt-eclipse-tutorial/#installJava

[8] Industrial Consultant Services. Juan Antonio. http://www.juanantonio.info/jab_cms.php?id=69

[9] Forums nxtasy.org. http://forums.nxtasy.org/lofiversion/index.php/

[10] LEGO Lab, University of Aarhus. http://legolab.daimi.au.dk/

[11] Compile time error messages.http://mindprod.com/jgloss/runerrormessages.html

[12] Bluehack. The Spanish Bluetooth Security Group. http://bluehack.elhacker.net/proyectos/bluesec/bluesec.html

[13] Revista robotiker-tecnalia: Tecnología Bluetooth: características de enlace de radio y establecimiento de la conexión. http://www.robotiker.com/revista/

[14] leJOS Forums, Java for LEGO Mindstorms. http://lejos.sourceforge.net/forum/

[15] Anónimo Colectivo. Taller OpenCV. http://www.anonimocolectivo.org/taller/blog/

[16] Introduction to programming with OpenCV. Gady Adam. Department of Computer Science Illinois Institute of Technology.
http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

[17] Blog walkintothefuture. OpenCV+Java+linux.
http://walkintothefuture.blogspot.com/2009/04/opencv-java-linux.html

[18] Tutorial OpenCV. Raúl Igual. Carlos Medrano.
http://www.scribd.com/doc/19006072/Tutorial-Opencv

[19] Manipulación de imágenes TIFF con Java JAI. Miguel Angel Mena Sevilla.
http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=manipTIFF

[20] Utilizando JAI api para el Tratamiento de imágenes digitales en JAVA. Duvallier.
http://xromsystem.net/?p=1101

[21] Metodo de dilatación de imágenes en Java. Duvallier.
http://xromsystem.net/?p=995

[22] Java Advanced Imaging Binary Builds. https://jai.dev.java.net/binary-builds.html

[23] API Java Advanced Imaging.
http://download.oracle.com/docs/cd/E17802_01/products/products/java-media/jai/forDevelopers/jai-apidocs/index.html

[24] Java source code. JDK Modules. JAI.
http://www.java2s.com/Open-Source/Java-Document/6.0-JDK-Modules/Java-Advanced-Imaging/javax/media/jai/operator/ErodeDescriptor.java.htm

[25] Introducción a OpenCV. Jhon Alvarez Borja.
http://my.opera.com/jalvarezborja/blog/introduccion-a-opencv

[26] Sistemas Robotizados. Universidad de Almería. http://aer.ual.es/docencia_es/sr/

[27] Microbótica: Robot Aided Process. http://www.microbotica.es/wiki/doku.php

[28] Librería AR Drone Java API, JavaDrone. http://code.google.com/p/javadrone/

[29] AR Drone open API platform. https://projects.ardrone.org/

[30] JavaCV: Java interface to OpenCV and more. http://code.google.com/p/javacv/

[31] Running JADE under Eclipse.
http://wrjih.wordpress.com/2008/11/29/running-jade-under-eclipse/

[32] Desarrollo de agentes software. Fernandez, C., Gómez, J., Pavón, J.
www.fdi.ucm.es/profesor/jpavon/doctorado/sma.pdf

[33] Java Agent DEvelopment Framework. http://jade.tilab.com/

[34] Wikispaces: Programación JADE.
http://programacionjade.wikispaces.com/Comunicaci%C3%B3n

[35] Librería de procesamiento de imagen en JAVA. http://processing.org/

[36] RXTX: Librería JAVA para conexiones por puertos COM.
http://rxtx.qbang.org/wiki/index.php/Main_Page

[37] Microchip MPLAB ICD 2. http://www.microchip.com

[38] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano y A. Martinoli. (2009). The e-puck, a robot designed for education in engineering. Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, pp. 59–65.

[39] Erika enterprise. http://erika.tuxfamily.org/

[40] Education and Research Robots from Festo. http://www.festo-didactic.com

[41] Open Management Group. http://www.omg.org/

[42] KSE: Knowledge Sharing Effort public library.
http://www-ksl.stanford.edu/knowledge-sharing/index.html

[43] The Foundation for Intelligent Physical Agents. http://www.fipa.org/

[44] Cyberbotics' Robot Curriculum/Advanced Programming Exercises http://en.wikibooks.org/wiki/Cyberbotics'_Robot_Curriculum/Advanced_Programming_Exercises

[45] Enfoque holónico basado en agentes para el control de organizaciones de robots móviles. Cervera, A., Soriano, A., Gómez, J., Valera, A., Valles, M., Giret, A. Jornadas de Automática 2011, Sevilla.

[46] Application and evaluation of Lego NXT tool for Mobile Robot Control. Valera, A., Vallés, M., Marín, L., Soriano, A., Cervera, A. International Federation of Automatic Control. Congreso Milan, Italia, 2011.