UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingeniería Informática

Universitat Politècnica de València

Desarrollo multiplataforma de aplicaciones de control y comunicación para robots móviles

Proyecto Final de Carrera

Ingeniero en informática

Autor: Vicent Mayans Roca

Director: Ángel Valera Fernández

10 de septiembre de 2012

Resumen

En este proyecto se desarrollan diversos algoritmos de control y comunicación para robots móviles de la plataforma LEGO Mindstorms NXT, utilizándose para ello las plataformas software de programación nxtOSEK, en un entorno Windows, y LeJOS, en un entorno Linux. Con el objetivo final de dotar de autonomía al robot en un terreno obstaculizado se estudian distintos tipos de comunicación, haciendo especial hincapié en Bluetooth y USB, y estrategias de generación y resolución de trayectorias por parte de robots móviles, escogiendo el algoritmo de persecución pura. Estos conocimientos son desarrollados en las tres aplicaciones finales: generación y seguimiento de trayectorias fijas, generación y seguimiento de trayectorias dinámicas recibidas por USB y navegación autónoma con visión artificial a través del sensor Kinect de Xbox.

Palabras clave: robótica móvil, persecución pura, comunicación USB, Bluetooth, navegación autónoma.

Tabla de contenidos

1.- Introducción

1.1. Introducción y justificación

La búsqueda incesante del hombre por satisfacer su necesidad de comunicación ha sido el impulso que ha logrado la instauración de instrumentos y tecnologías cada día más poderosos y veloces en el proceso comunicativo.

En las últimas décadas la aparición de la comunicación inalámbrica entre dispositivos se ha impuesto con el uso de teléfonos móviles, ordenadores portátiles, PDA's y en general en cualquier dispositivo electrónico. Han aparecido diversos tipos de comunicación como Wifi, Bluetooth o infrarrojos, que aunque hasta el momento no ofrecen prestaciones tan altas como la comunicación por cable, no cabe duda que algún día lo harán.

Al mismo tiempo la tecnología mecánica y electrónica también ha evolucionado y, dejando a un lado los estereotipos de los robots como máquinas que acaban descontrolándose y tomando iniciativa propia, poco a poco se han ido implantando en la sociedad como herramientas que facilitan tareas o resuelven problemas.

Esta tecnología ha llegado incluso al campo de la educación, donde se pueden encontrar juguetes que realmente son auténticos robots en miniatura y que ofrecen tal versatilidad que se hace uso de ellos para llevar a cabo investigaciones sobre problemas que podrían abordar más tarde robots más grandes.

La automatización para que estos robots realicen tareas en diferentes entornos con los que tienen que interactuar mediante la integración de visión artificial y sensores varios, el reconocimiento de formas, la comunicación entre robots y entre robot y usuario o la planificación de trayectorias, aún supone un reto en algunos aspectos y ocupa una gran parte de la línea de investigación dentro de la robótica.

Este proyecto engloba el uso de las nuevas tecnologías mecánicas y electrónicas, como los robots de Lego Mindstorms NXT, con el uso de últimas tecnologías en comunicación, como el Bluetooh y USB, y con el estudio de la automatización de la generación y el seguimiento de trayectorias de vehículos móviles.

1.2. Objetivos

El principal objetivo de este proyecto es tratar de llegar a conocer en profundidad todas las opciones que ofrece el robot Lego Mindstorms NXT con los firmwares nxtOsek y LejOS y tratar de implementar en base a ese conocimiento diferentes aplicaciones que más tarde se puedan extrapolar a otros robots o escenarios.

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

- Estudio del firmware nxtOsek para la plataforma Lego Mindstorm NXT y de la librería de programación ECRobot en el entorno Simulink.

- Estudio del firmware LejOS para la plataforma Lego Mindstorm NXT.
- Estudio de la comunicación vía Bluetooth y USB entre robots NXT y el PC.
- Estudio de algoritmos de generación de trayectorias.
- Estudio de las estrategias de seguimiento de trayectorias.
- Estudio de la visión artificial desde el sensor Kinect de Xbox.
- Desarrollo de aplicaciones que hagan uso de los conocimientos adquiridos.

2.- Desarrollo teórico

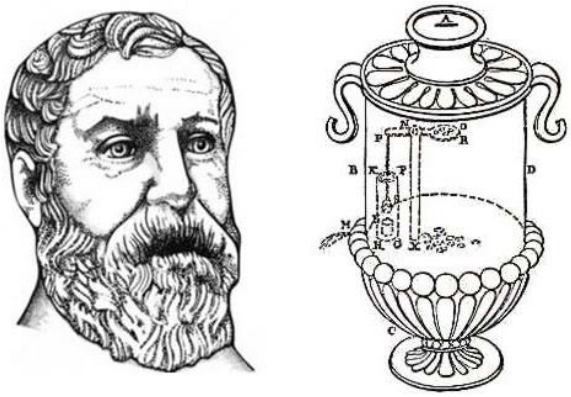
2.1. Introducción a la robótica

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.



Herón de Alejandría junto uno de sus inventos, un dispensador de agua sagrada operado por monedas.

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

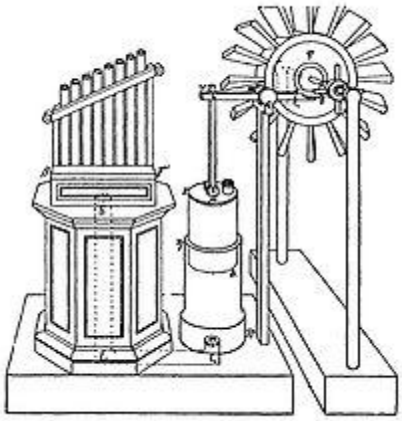


Figura 3: Hydraulis de Ctesibios

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.

Figura 4: 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.

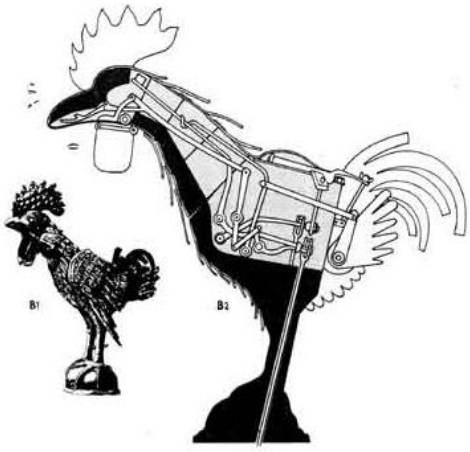


Figura 5: El Gallo de Estrasburgo

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.

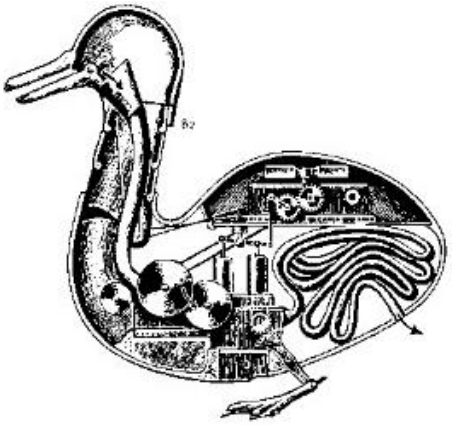


Figura 6: Pato con aparato digestivo

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.

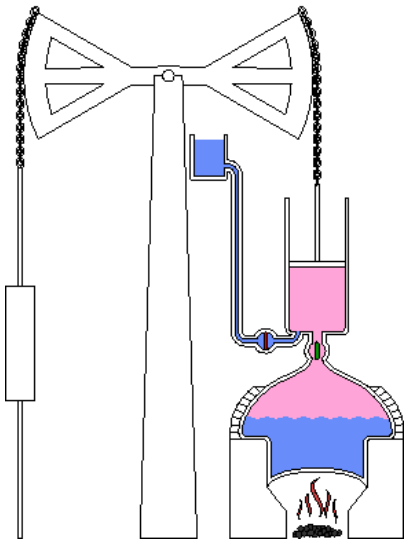


Figura 7: Máquina de vapor atmosférica.

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.

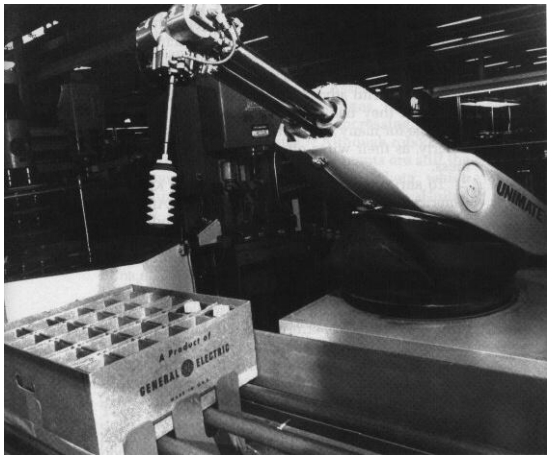


Figura 8: Unimate

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.

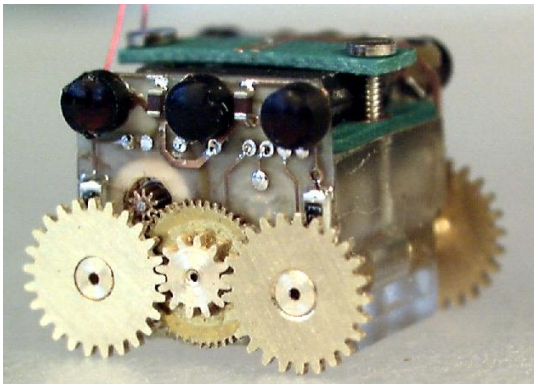


Figura 13: Robot Smoovy

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.

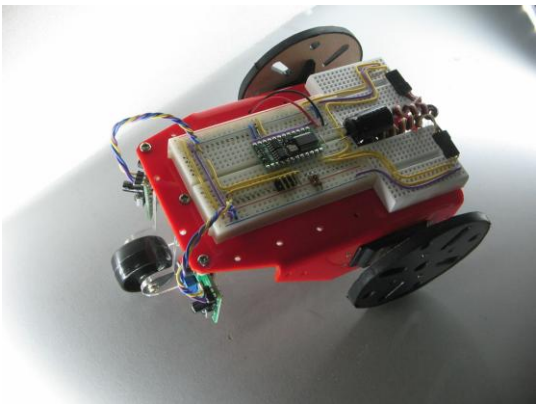


Figura 16: Robot rodante con las ruedas en configuración triciclo

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

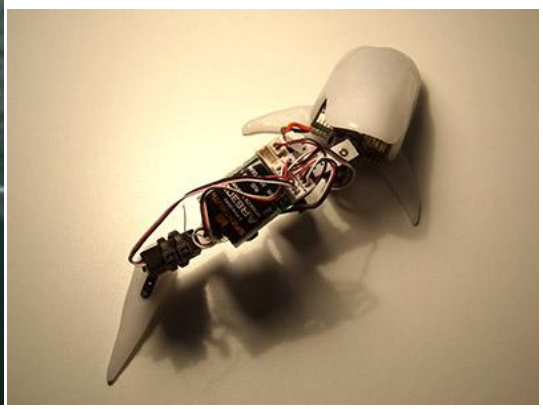


Figura 22: Robot ballena

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.

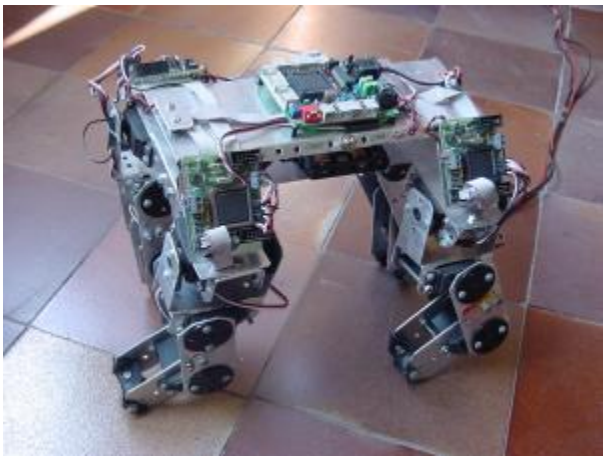


Figura 26: Estructura básica fabricada en aluminio para un robot cuadrúpedo.

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.

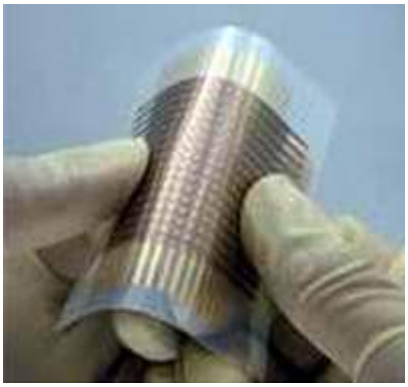


Figura 30: Sensor táctil flexible

• **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 reluctancia 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.

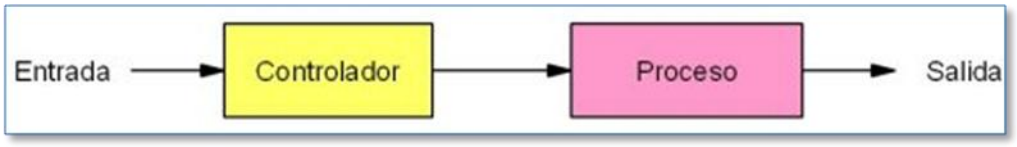


Figura 36: Esquema de un controlador en bucle abierto

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

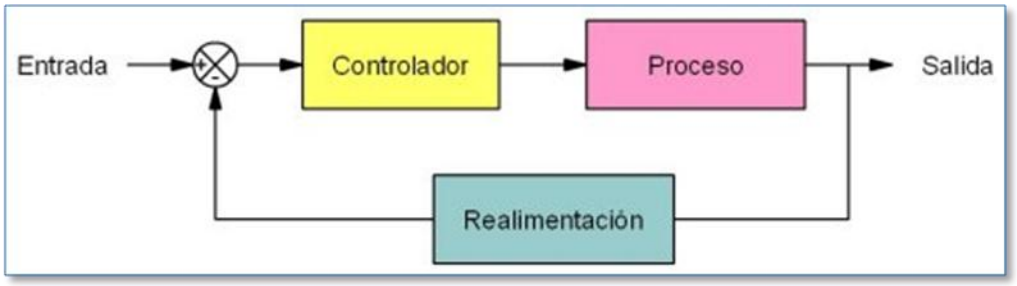


Figura 37: Esquema de un controlador en bucle cerrado

- **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:

- **Baterías:** sin duda la forma más comúnmente utilizada como fuente de alimentación en todo tipo de robots. Son baratas, fiables y se pueden encontrar en cientos de formatos, voltajes y dimensiones.

Figura 38: Diferentes tipos de baterías de uso doméstico

Desde la pila de botón más pequeña y ligera para alimentar un micro robot podemos ir a pesadas pero potentes baterías de plomo usadas en vehículos pesados que necesitan un aporte importante de energía.

Otro 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.

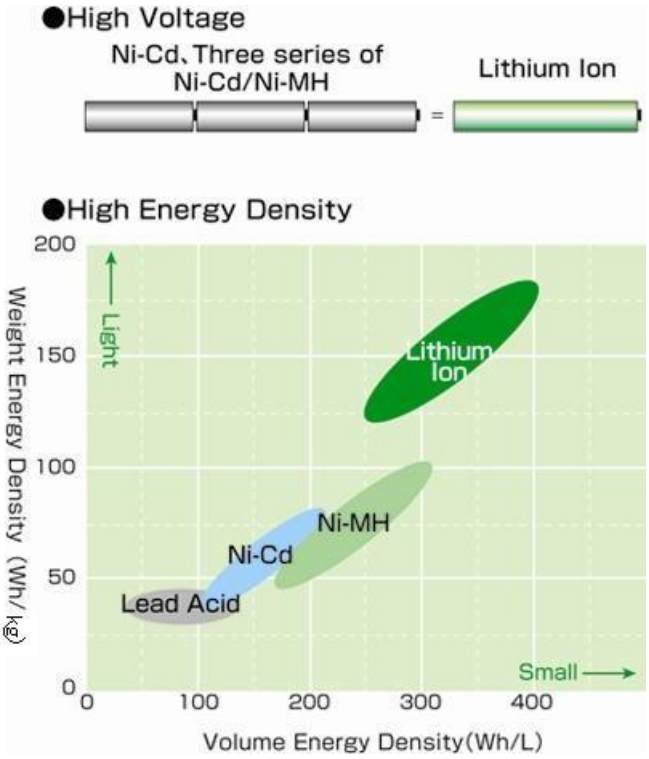


Figura 41: Gráfico comparativo del rendimiento de diferentes tipos de baterías

- **Bombonas de aire:** usadas tanto para el movimiento de actuadores neumáticos como para la propulsión de motores de aire. Este tipo de motores tiene una larga historia, pese a que su difusión es reducida. Su principio es equivalente al de los motores de explosión, con la salvedad que en lugar de producir la combustión de gasolina y oxígeno para generar los gases que impulsan los pistones, se usa directamente aire almacenado en un depósito a alta presión.

Figura 42: Diferentes tipos de bombonas de aire comprimido

- **Baterías inerciales:** este curioso mecanismo tampoco tiene una difusión amplia en el terreno de los robots móviles, pero si tiene aplicación e entornos donde la alimentación es crítica, como centros de proceso de datos. Fue probada en vehículos durante sus albores, concretamente en autobuses urbanos de Zúrich durante los años 50 y algunas locomotoras de tren durante los 70, pero no se ha extendido su uso de forma masiva.

El principio de funcionamiento es sencillo: se acelera un rotor (habitualmente mediante electricidad) confiriéndole así una elevada energía rotacional, típicamente de varias decenas de miles de rpm. Posteriormente se decelera poco a poco, recuperando así la energía eléctrica de nuevo.

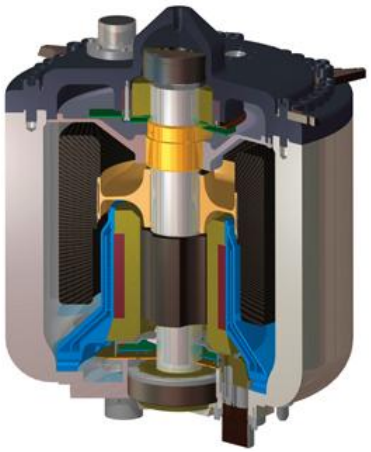


Figura 43: Sección de una batería inercial

La clave del aprovechamiento energético, que oscila entre el 90% y el 98%, consiste en reducir al máximo el rozamiento. Esto se consigue mediante usando un rotor de carbono (que presenta menos resistencia que el acero) estabilizado magnéticamente en un entorno controlado, generalmente un contenedor sellado tras realizarle el vacío.

- **Células de hidrogeno:** Otra tecnología emergente que poco a poco está demostrando su validez son las baterías formadas a base de células que usan el hidrógeno como combustible para producir electricidad.

Su funcionamiento se basa en una reacción química al igual que en las baterías tradicionales pero a diferencia de ellas, aquí no se producen cambios de estado en los electrodos o en los reactantes. Simplemente se consumen los reactantes y la reacción se produce prácticamente de forma indefinida mientras haya suficiente combustible para sustentar el proceso.

Figura 44: Célula de hidrógeno

Las principales ventajas son su elevado rendimiento así como su larga vida, que va asociada a la práctica ausencia de mantenimiento que necesitan. En contraposición el

combustible a día de hoy todavía es relativamente caro y difícil de conseguir aunque esta situación mejorará cuando haya una mayor demanda de estas células.

- **Biomasa:** Una fuente de energía muy interesante y que aun se encuentra por explotar. Pequeñas cantidades de basura orgánica permitirían alimentar durante horas un dispositivo dotado de esta tecnología. A parte del ahorro económico que supone, tiene especial interés en zonas sin acceso directo a fuentes de energía o en momentos de conflicto militar donde frecuentemente escasean los suministros energéticos.

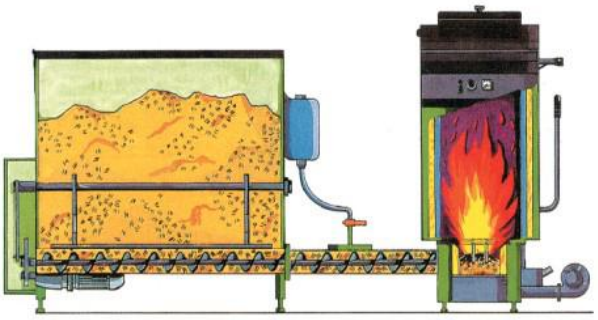


Figura 45: Caldera de biomasa.

- **Células nucleares:** Se basan en las emisiones de un isótopo radioactivo para obtener energía. Hay diversas técnicas para realizar la conversión de las radiaciones en energía eléctrica aprovechable, pero el resultado es el mismo. Poseen una larga vida útil pero su peligrosidad desaconseja su uso.

- **Motor de combustión:** El más conocido entre los sistemas de propulsión. Se usan motores muy similares a los de los vehículos, sobre todo los que llevan las motocicletas. Como combustible una mezcla de aceite y gasolina u otros elementos volátiles como el etanol. Se pueden utilizar tanto para obtener energía mecánica como energía eléctrica si añadimos un generador.

2.3.2.6. Comunicaciones

Dado que este es uno de los aspectos sobre los que versan las experiencias realizadas en este proyecto, se le dedicara el apartado siguiente a este tema, de forma que se traten en profundidad las características más relevantes:

2.3.2.6.1. USB

El Universal Serial Bus (bus universal en serie) o Conductor Universal en Serie (CUS), abreviado comúnmente USB, es un puerto de comunicación de periféricos a un computador. Fue creado en 1996 con la finalidad de eliminar la necesidad de adquirir tarjetas separadas para poner en los puertos de bus ISA o PCI y mejorar las capacidades plug-and-play permitiendo a esos dispositivos ser conectados o desconectados sin necesidad de reiniciar el sistema. Sin embargo, en aplicaciones donde se necesita ancho de banda para grandes transferencias de datos los buses PCI o PCIe salen ganando e igual sucede si la aplicación requiere robustez industrial.

Se pueden clasificar en cuatro tipos según su velocidad de transferencia de datos:

- Baja velocidad (1.0): Tasa de transferencia de hasta 1,5 Mbps (192KB/s). Utilizado en su mayor parte por dispositivos de interfaz humana como los teclados, ratones e incluso artículos del hogar.

- Velocidad completa (1.1): Tasa de transferencia de hasta 12Mbps (1,5MB/s), según el estándar, pero se dice en fuentes independientes que habría que realizar nuevamente las mediciones.

- Alta velocidad (2.0): Tasa de transferencia de hasta 480 Mbps (60MB/s) pero por lo general de hasta 125Mbps (16MB/s). Está presente casi en el 99% de los ordenadores actuales. El cable USB 2.0 dispone de cuatro líneas, una para datos, una para corriente y otra de toma de tierra.

- Super alta velocidad (3.0): Actualmente se encuentra en fase experimental y tiene una tasa de transferencia de hasta 4.8 Gbps (600MB/s). Esta especificación será diez veces más veloz que la anterior 2.0 y prevé su lanzamiento a corto plazo. Se han incluido cinco conectores extra, desechando el conector de fibra óptica propuesto inicialmente y será compatible con los estándares anteriores.

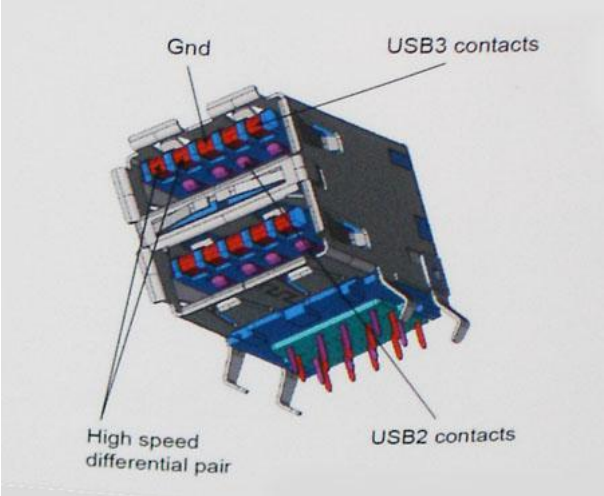


Figura 46: Prototipo de USB 3.0

2.4. Comunicaciones inalámbricas

2.4.1. 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

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.

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.

En una comunicación Bluetooth se pueden alcanzar tasas de transmisión de datos de 720 kbps (1 Mbps de capacidad bruta) con un rango óptimo de 10 metros (como se ha comentado antes 100 metros con repetidores).

2.4.3. Otras alternativas

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.

ZigBee vs. Bluetooth

ZigBee es muy similar al Bluetooth pero con algunas diferencias:

- Una red ZigBee puede constar de un máximo de 65535 nodos distribuidos en subredes de 255 nodos, frente a los 8 máximos de una subred (Piconet) Bluetooth.

- Menor consumo eléctrico que el de Bluetooth. En términos exactos, ZigBee tiene un consumo de 30mA transmitiendo y de 3uA en reposo, frente a los 40mA transmitiendo y 0.2mA en reposo que tiene el Bluetooth. Este menor consumo se debe a que el sistema ZigBee se queda la mayor parte del tiempo dormido, mientras que en una comunicación Bluetooth esto no se puede dar, y siempre se está transmitiendo y/o recibiendo.

- Tiene un velocidad de hasta 250 kbps, mientras que en Bluetooth es de hasta 1 Mbps.

- Debido a las velocidades de cada uno, uno es más apropiado que el otro para ciertas cosas. Por ejemplo, mientras que el Bluetooth se usa para aplicaciones como los teléfonos móviles y la informática casera, la velocidad del ZigBee se hace

insuficiente para estas tareas, desviándolo a usos tales como la Domótica, los productos dependientes de la batería, los sensores médicos, y en artículos de juguetería, en los cuales la transferencia de datos es menor.

- Existe una versión que integra el sistema de radiofrecuencias característico de Bluetooth junto a una interfaz de transmisión de datos vía infrarrojos desarrollado por IBM mediante un protocolo ADSI y MDSI.

Tipos de dispositivos

Se definen tres tipos distintos de dispositivo ZigBee según su papel en la red:

- **Coordinador ZigBee (ZigBee Coordinator, ZC):** El tipo de dispositivo más completo. Debe existir uno por red. Sus funciones son las de encargarse de controlar la red y los caminos que deben seguir los dispositivos para conectarse entre ellos.

- **Router ZigBee (ZigBee Router, ZR):** Interconecta dispositivos separados en la topología de la red, además de ofrecer un nivel de aplicación para la ejecución de código de usuario.

- **Dispositivo final (ZigBee End Device, ZED):** Posee la funcionalidad necesaria para comunicarse con su nodo padre (el coordinador o un router), pero no puede transmitir información destinada a otros dispositivos. De esta forma, este tipo de nodo puede estar dormido la mayor parte del tiempo, aumentando la vida media de sus baterías. Un ZED tiene requerimientos mínimos de memoria y es por tanto significativamente más barato.

Como ejemplo de aplicación en Domótica, en una habitación de la casa tendríamos diversos Dispositivos Finales (como un interruptor y una lámpara) y una red de interconexión realizada con Routers ZigBee y gobernada por el Coordinador.

Funcionalidad

Basándose en su funcionalidad, puede plantearse una segunda clasificación:

- Dispositivo de funcionalidad completa (FFD): También conocidos como nodo activo. Es capaz de recibir mensajes en formato 802.15.4. Gracias a la memoria adicional y a la capacidad de computar, puede funcionar como Coordinador o Router ZigBee, o puede ser usado en dispositivos de red que actúen de interface con los usuarios.

- Dispositivo de funcionalidad reducida (RFD): También conocido como nodo pasivo. Tiene capacidad y funcionalidad limitadas (especificada en el estándar) con el objetivo de conseguir un bajo coste y una gran simplicidad. Básicamente, son los sensores/actuadores de la red.

- Un nodo ZigBee (tanto activo como pasivo) reduce su consumo gracias a que puede permanecer dormido la mayor parte del tiempo (incluso muchos días seguidos). Cuando se requiere su uso, el nodo ZigBee es capaz de despertar en un

tiempo ínfimo, para volverse a dormir cuando deje de ser requerido. Un nodo cualquiera despierta en aproximadamente 15 ms. Además de este tiempo, se muestran otras medidas de tiempo de funciones comunes:

- Nueva enumeración de los nodos esclavo (por parte del coordinador): aproximadamente 30 ms.

- Acceso al canal entre un nodo activo y uno pasivo: aproximadamente 15 ms.

Protocolos

Los protocolos se basan en investigaciones recientes sobre algoritmos de red (ad hoc on-demand distance vector, vector de distancias bajo demanda; neuRFon) para la construcción de redes ad-hoc de baja velocidad. La mayoría de redes grandes están pensadas para formar un clúster de clústeres. También puede estructurarse en forma de malla o como un solo clúster. Los perfiles actuales de los protocolos soportan redes que utilicen o no facilidades de balizado.

Las redes sin balizas (aquéllas cuyo grado de balizado es 15) acceden al canal por medio de CSMA/CA. Los routers suelen estar activos todo el tiempo, por lo que requieren una alimentación estable en general. Esto, a cambio, permite redes heterogéneas en las que algunos dispositivos pueden estar transmitiendo todo el tiempo, mientras que otros sólo transmiten ante la presencia de estímulos externos. El ejemplo típico es un interruptor inalámbrico: un nodo en la lámpara puede estar recibiendo continuamente ya que está conectado a la red; por el contrario, un interruptor a pilas estaría dormido hasta que el mecanismo se activa. En una red así la lámpara sería un router o coordinador, y el interruptor un dispositivo final.

Si la red utiliza balizas, los routers las generan periódicamente para confirmar su presencia a otros nodos. Los nodos pueden desactivarse entre las recepciones de balizas reduciendo su ciclo de servicio (duty cycle). Los intervalos de balizado pueden ir desde 5,36 ms a 15,36 ms * 214 = 251,65824 segundos a 250 kbps; de 24 ms a 24 ms * 214 = 393,216 segundos a 40 kbps; y de 48 ms a 48 ms * 214 = 786,432 segundos a 20 kbps. Sin embargo, los periodos largos con ciclos de servicio cortos necesitan que una temporización precisa, lo que puede ir en contra del principio de bajo coste.

En general, los protocolos ZigBee minimizan el tiempo de actividad de la radio para evitar el uso de energía. En las redes con balizas los nodos sólo necesitan estar despiertos mientras se transmiten las balizas (además de cuando se les asigna tiempo para transmitir). Si no hay balizas, el consumo es asimétrico repartido en dispositivos permanentemente activos y otros que sólo no están esporádicamente.

Los dispositivos ZigBee deben respetar el estándar de WPAN de baja tasa de transmisión IEEE 802.15.4-2003. Éste define los niveles más bajos: el nivel físico (PHY) y el control de acceso al medio (MAC, parte del nivel de enlace de datos, DLL). El estándar trabaja sobre las bandas ISM de uso no regulado detalladas más arriba. Se definen hasta 16 canales en el rango de 2,4 GHz, cada uno de ellos con un ancho de banda de 5 MHz. La frecuencia central de cada canal puede calcularse como: FC = (2405 + 5*(k-11)) MHz, con k = 11, 12, ..., 26.

Las radios utilizan un espectro de dispersión de secuencia directa. Se utiliza BPSK en los dos rangos menores de frecuencia, así como un QPSK ortogonal que transmite dos bits por símbolo en la banda de 2,4 GHz. Ésta permite tasas de transmisión en el aire de hasta 250 kbps, mientras que las bandas inferiores se han ampliado con la última revisión a esta tasa desde los 40 kbps de la primera versión. Los rangos de transmisión oscilan entre los 10 y 75 metros, aunque depende bastante del entorno. La potencia de salida de las radios suele ser de 0 dBm (1 mW).

Si bien en general se utiliza CSMA/CA para evitar colisiones en la transmisión, hay algunas excepciones a su uso: por una parte, las tramas siguen una temporización fija que debe ser respetada; por otra, las confirmaciones de envíos tampoco siguen esta disciplina; por último, si se asignan slots de tiempo garantizados para una transmisión tampoco es posible que exista contención.

Hardware y software

El software se ha diseñado para ejecutarse en procesadores y microcontroladores de bajo coste, con un diseño de radio muy optimizado para lograr bajos costes con altos volúmenes de producción. Utiliza circuitos digitales siempre que es posible y evita los componentes analógicos.

Si bien el hardware es sencillo, el proceso de certificación de un dispositivo conlleva una validación completa de los requerimientos del nivel físico. Esta revisión intensiva tiene múltiples ventajas, ya que todas las radios fabricadas a partir de una misma máscara de semiconductor gozarán de las mismas características de radiofrecuencia. Por otro lado, un nivel físico mal controlado podría perjudicar no sólo al propio dispositivo, sino al consumo de energía de otros dispositivos en la red. Otros estándares pueden compensar ciertos problemas, mientras que ZigBee trabaja en márgenes muy estrechos de consumo y ancho de banda. Por ello, según el 802.15.4, las radios pasan validaciones ISO 17025. La mayoría de fabricantes planea integrar la radio y el microcontrolador en un único chip.

Topologías de red

ZigBee permite tres topologías de red:

- Topología en estrella: el coordinador se sitúa en el centro.
- Topología en árbol: el coordinador será la raíz del árbol.
- Topología de malla: al menos uno de los nodos tendrá más de dos conexiones.

La topología más interesante (y una de las causas por las que parece que puede triunfar ZigBee) es la topología de malla. Ésta permite que si, en un momento dado, un nodo del camino falla y se cae, pueda seguir la comunicación entre todos los demás nodos debido a que se rehacen todos los caminos. La gestión de los caminos es tarea del coordinador.

Estrategias de conexión de los dispositivos en una red Zigbee

Las redes ZigBee han sido diseñadas para conservar la potencia en los nodos 'esclavos'. De esta forma se consigue el bajo consumo de potencia. La estrategia consiste en que, durante mucho tiempo, un dispositivo "esclavo" está en modo "dormido", de tal forma que solo se "despierta" por una fracción de segundo para confirmar que está "vivo" en la red de dispositivos de la que forma parte. Esta transición del modo "dormido" al modo "despierto" (modo en el que realmente transmite), dura unos 15ms, y la enumeración de "esclavos" dura alrededor de 30ms, como ya se ha comentado anteriormente.

En las redes Zigbee, se pueden usar dos tipos de entornos o sistemas:

- **Con balizas**

Es un mecanismo de control del consumo de potencia en la red. Permite a todos los dispositivos saber cuándo pueden transmitir. En este modelo, los dos caminos de la red tienen un distribuidor que se encarga de controlar el canal y dirigir las transmisiones. Las balizas que dan nombre a este tipo de entorno, se usan para poder sincronizar todos los dispositivos que conforman la red, identificando la red domótica, y describiendo la estructura de la "supertrama". Los intervalos de las balizas son asignados por el coordinador de red y pueden variar desde los 15ms hasta los 4 minutos.

Este modo es más recomendable cuando el coordinador de red trabaja con una batería. Los dispositivos que conforman la red, escuchan a dicho coordinador durante el "balizamiento" (envío de mensajes a todos los dispositivos, entre 0,015 y 252 segundos). Un dispositivo que quiera intervenir, lo primero que tendrá que hacer es registrarse para el coordinador, y es entonces cuando mira si hay mensajes para él. En el caso de que no haya mensajes, este dispositivo vuelve a "dormir", y se despierta de acuerdo a un horario que ha establecido previamente el coordinador. En cuanto el coordinador termina el "balizamiento", vuelve a "dormirse".

- **Sin balizas**

Se usa el acceso múltiple al sistema Zigbee en una red punto a punto cercano. En este tipo, cada dispositivo es autónomo, pudiendo iniciar una conversación, en la cual los otros pueden interferir. A veces, puede ocurrir que el dispositivo destino puede no oír la petición, o que el canal esté ocupado.

Este sistema se usa típicamente en los sistemas de seguridad, en los cuales sus dispositivos (sensores, detectores de movimiento o de rotura de cristales), duermen prácticamente todo el tiempo (el 99,999%). Para que se les tenga en cuenta, estos elementos se "despiertan" de forma regular para anunciar que siguen en la red. Cuando se produce un evento (en nuestro sistema será cuando se detecta algo), el sensor "despierta" instantáneamente y transmite la alarma correspondiente. Es en ese momento cuando el coordinador de red, recibe el mensaje enviado por el sensor, y activa la alarma correspondiente. En este caso, el coordinador de red se alimenta de la red principal durante todo el tiempo.

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 hemos visto, 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.

2.5. LEGO NXT

De todos es sabido el largo camino que lleva recorrido la firma Lego en lo que se conoce como ‘construcciones’, es decir, juguetes dirigidos a un público generalmente infantil o adolescente. Esta andadura empezó en el año 1934 en Dinamarca y con el tiempo fue cobrando popularidad entre padres, hijos y sobretodo educadores que veían en estos juguetes una forma muy interesante de divertir y a la vez potenciar la creatividad de sus hijos y alumnos.

La oferta de juguetes va desde los pocos meses de edad hasta la adolescencia e incluso un poco más allá, contando con temáticas urbanas, submarinas e incluso espaciales. Entre todas estas líneas cabe destacar una que desde el principio ha contado ha despertado la curiosidad de determinado sector de público, en el que podían englobarse ingenieros y apasionados de la técnica. Hablamos de la denominada línea *Technic*. La principal característica de esta gama de juguetes y la que sin duda la ha hecho tan famosa, es el peculiar enfoque de construcción que ha concebido Lego con ella.

Este enfoque se aparta del juguete tradicional y experimenta con elementos propios de la tecnología actual, tanto en mecánica como en electrónica. Así, entre las piezas de estos 'kits' se encuentran engranajes, ejes, motores eléctricos, cables y hasta elementos tan concretos como levas y pistones, juntas cardan y amortiguadores.

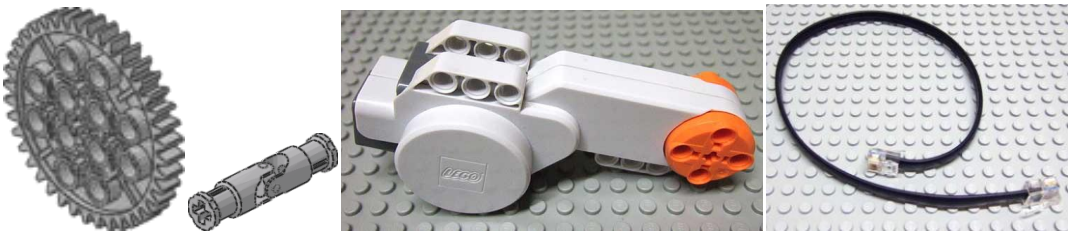


Figura 47: De izquierda a derecha, engranaje, junta cardan, motor y cable de LEGO

Con estos elementos se emula a la perfección la realidad y se permite construir representaciones a escala de vehículos y sistemas con mecanismos a imagen y semejanza de sus modelos reales, teniendo incluso un funcionamiento análogo o lo más aproximado posible, a como lo hacen dichos dispositivos en la realidad.

Figura 48: Diferentes modelos de la línea LEGO Technic

Sin embargo, los tiempos en que los juguetes tradicionales, y en especial las construcciones, reinaban entre sus potenciales consumidores están tocando a su fin de la mano de la revolución tecnológica. De esta forma han sido prácticamente desbancados por dispositivos con silicio en sus entrañas, como consolas, ordenadores y

demás parafernalia electrónica. LEGO por lo tanto, siguiendo el ejemplo de sus clientes, no se ha conformado con ésta aproximación a la tecnología moderna de finales de siglo, en su día revolucionaria, sino que como mandan los tiempos ha decidido ir más allá.

2.5.1. Ladrillo NXT

Tradicionalmente la línea Technic de LEGO ha hecho sus incursiones en el campo de la tecnología electrónica, añadiendo a su oferta de luces, motores, baterías y mandos con hilos que pese al atractivo que ofrecían en si día, han quedado atrás frente a los elementos de última tecnología que a día de hoy copan el mercado de la electrónica de consumo. Por ello, LEGO ha tomado el testigo tecnológico incorporando lo que ha venido a llamarse como '*bricks*' o '*ladrillos*' inteligentes a varias de sus líneas de productos.

Figura 49: De izquierda a derecha versiones RCX y NXT del ladrillo de control

Estos 'ladrillos', proveen de una unidad de control para robots o sistemas electrónicos avanzados que permite implementar escenarios hasta el momento impensables en un juego de construcciones, incluso los de la línea Technic de la propia marca. Su corazón está formado por un microcontrolador o procesador, que en sus últimas versiones ha alcanzado grados de potencia sorprendentes si pensamos en ellos como en un juguete.

Por ello, cada vez más personas se han interesado en estos sistemas, no sólo para la docencia o el ocio, sino incluso para la investigación. Prácticamente todas las universidades a nivel mundial poseen ya kits de LEGO con estos *bricks* inteligentes y se usan en las ramas de la enseñanza y investigación como las relacionadas con la robótica o la mecatrónica.

El modelo usado en la realización de las experiencias que comprende este trabajo es el último disponible en el mercado, el Mindstorms NXT. Este está formado por un kit que incluye: el *brick* NXT, múltiples sensores (luz/color, ultrasonidos, contacto, sonido) motores y cableado eléctrico, y el set de construcción habitual de la línea Technic, formado por miles de piezas de construcción, engranajes, ejes correas y poleas.

A continuación una lista de las características electrónicas del *brick* NXT, extraídas del manual de LEGO:

Procesador principal: Atmel® 32-bit ARM® processor, AT91SAM7S256
- 256 KB FLASH
- 64 KB RAM
- 48 MHz

Co-procesador: Atmel® 8-bit AVR processor, ATmega48
- 4 KB FLASH
- 512 Byte RAM
- 8 MHz

Unidad de comunicaciones Bluetooth CSR BlueCoreTM 4 v2.0 + EDR System
- Soporte de Serial Port Profile (SPP)
- 47 KByte RAM Internos
- 8 MBit FLASH Externos
- 26 MHz

Comunicación vía USB 2.0 (12 Mbit/s)

4 puertos de entrada con interfaz de 6 hilos y soporte para conexiones AD y DA
- 1 puerto de alta velocidad, IEC 61158 Tipo 4/EN 50170 compatible
- todos los puertos de entrada cuentan con soporte del bus **I²C**

3 puertos de salida con interfaz de 6 hilos y soporte para lectura desde encoders

Conectores de 6 hilos estándar industrial, RJ12 con ajuste a derechas

Display grafico LCD de 100 x 64 píxel (blanco y negro)
- Área de visión de 26 x 40.6 mm

Altavoz de salida con resolución de 8-bit
- Soporta tasas de muestreo de 2 a 16 KHz

4 botones de goma para interacción con el usuario

Alimentación: 6 pilas de tipo AA
- Se recomienda usar pilas alcalinas
- También se encuentra disponible una batería de Ion de Litio de 1400 mAH
- Cargador opcional disponible para la batería anterior

2.5.2. Motores

La gama de motores de que consta la línea LEGO Technic es bastante elevada, contando entre sus filas con aproximadamente 13 modelos diferentes. Estos motores van desde los pequeños micromotores hasta la joya de la corona de esta familia que no es otro que el motor incluido en el set del NXT. Este motor posee unas características muy buenas en cuanto a respuesta y rendimiento y cuenta asimismo con un juego de engranajes reductores y un *encoder* rotacional en su interior.

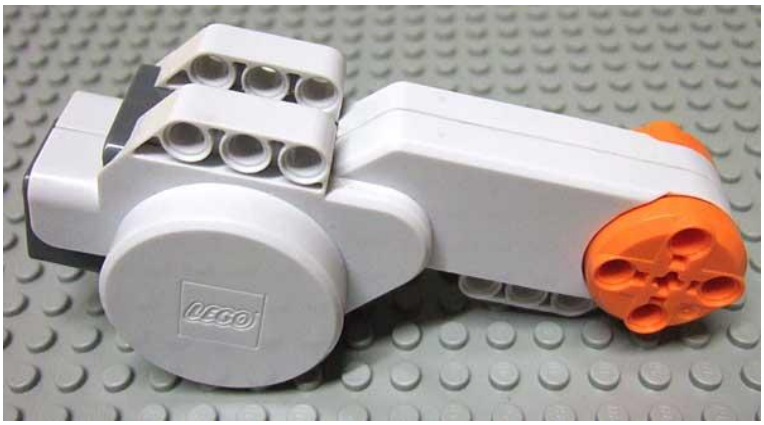


Figura 50: Motor de LEGO incluido en el set NXT

El juego de engranajes reductores garantiza un par de fuerza elevado para el motor, lo que se traduce en la capacidad de realizar un esfuerzo mayor sin la necesidad de elementos externos como era necesario en modelos anteriores, y más importante aún, impidiendo el sobreesfuerzo del motor y la consecuente reducción de su vida útil.

EL *encoder* o tacómetro, por su parte, permite conocer el número de vueltas que ha dado el eje del motor con una precisión de 1 grado. Es decir, si el motor realiza un giro completo el *encoder* verá incrementada su cuenta en 360 unidades. Esto es cierto sea cual sea el sentido de movimiento del motor, incrementándose en un sentido y decrementándose al girar en sentido contrario.

2.5.3. Sensores

En este apartado se realiza un repaso por el conjunto de sensores disponibles para el LEGO Mindstorms NXT, tanto los de marca propia de LEGO como aquellos ensamblados y distribuidos por terceros pero que son plenamente compatibles con la plataforma NXT.

2.5.3.1. Fabricados por LEGO

Son cuatro los sensores incluidos en el kit de LEGO Mindstorms NXT:

- **Bumper:** detecta la presión realizada sobre la punta del sensor. Útil para detectar colisiones, presencia de objetos o como mecanismo final de carrera.

- **Sonido:** consta de un pequeño micrófono capaz de captar sonidos.

- **Luz/Color:** este sensor cumple una doble función, es capaz de detectar la presencia de luz en un entorno y a su vez es capaz de identificar un pequeño rango de colores de los objetos que 've'.

- **Ultrasonidos:** permite detectar la presencia de objetos a una cierta distancia. Útil en las mismas situaciones que el bumper pero sin llegar al contacto.

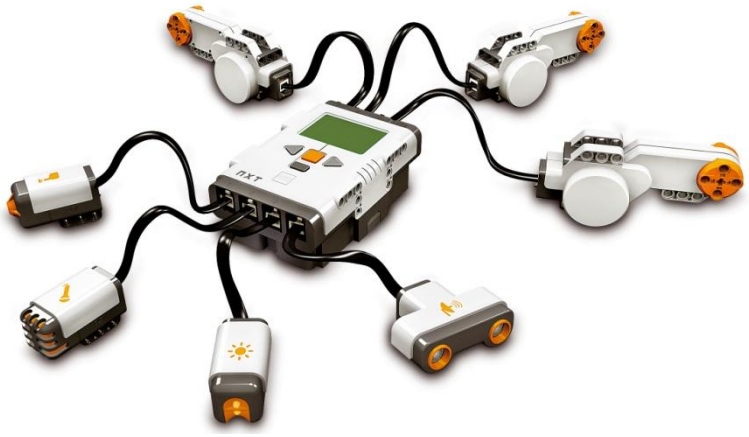


Figura 51: Ladrillo NXT con todos sus sensores y motores conectados

2.5.3.2. Fabricados por terceros

Fabricantes como Hi-Technic comercializan sensores compatibles con el NXT, de entre los cuales destacamos los siguientes:

- **Brújula:** permite conocer la orientación del robot respecto al norte magnético.
- **Acelerómetro:** detecta aceleración en tres ejes y giro en uno.
- **Seguidor de infrarrojos:** es capaz de detectar el haz de un emisor en un ángulo de 135º, indicando así mismo la dirección desde la que proviene el haz de luz.

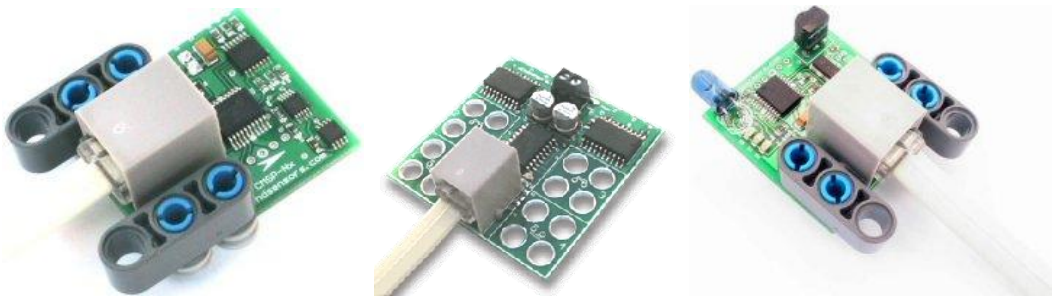


Figura 52: (De derecha a izquierda) Brújula, acelerómetro y seguidor de infrarrojos.

2.5.3.3. Personalizados

Es posible crear sensores propios y personalizados compatibles con el bus I^2C.

2.5.4. Comunicaciones

El LEGO MINDSTORMS NXT ha supuesto una mejora muy importante respecto a su predecesor RCX en el apartado de comunicaciones. Este cambio se debe a que se ha

desechado el rudimentario interfaz de comunicaciones infrarrojo y se ha adoptado el nuevo estándar Bluetooth que abre un nuevo mundo de posibilidades.

Los aspectos más relevantes de sistema adoptado se tratan a continuación.

2.5.4.1. Capas del sistema de comunicación

La siguiente figura muestra las principales capas de la pila de comunicación entre el dispositivo empotrado y el software que se encuentra en el PC. Como se puede comprobar, se puede establecer una comunicación full dúplex entre el PC y el NXT usando o no comunicaciones cableadas. Destaca la capacidad del NXT de usar el protocolo SPP que permite comunicarse vía Bluetooth como si de un puerto serie se tratase.

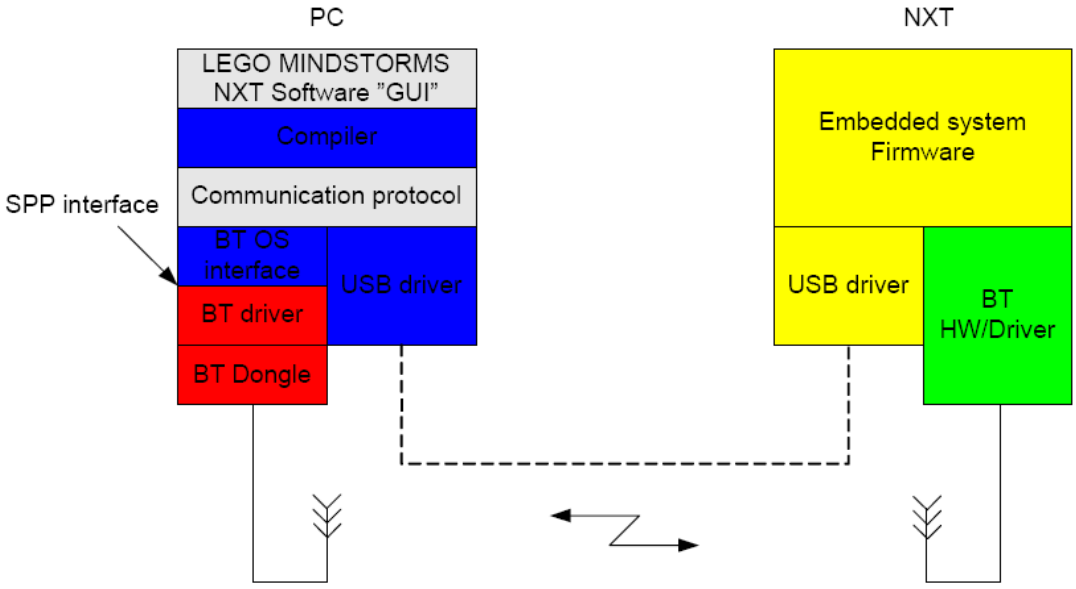


Figura 53: Diagrama de bloques de la comunicación PC-NXT

El firmware que lleva el NXT permite acceder al protocolo de comunicación con el NXT escribiendo o leyendo información en "crudo" o utilizando el protocolo establecido que posteriormente se explicará. La desventaja que se tiene si no se utiliza el protocolo es que es imposible comprobar la información enviada o recibida en el buffer, por lo que se requiere un cierto control a bajo nivel de las tramas enviadas.

Estudios realizados durante el desarrollo han demostrado que dicha comunicación tiene ciertas desventajas cuando se comienza a transmitir ráfagas de información, ya sea porque son demasiado largas y el tiempo entre la recepción de un paquete a otro es demasiado pequeño o por el retardo que se genera cuando se cambia de modo recepción de información a modo de transmisión de información.

Para evitar en la medida de lo posible de la longitud de los paquetes a enviar una solución que se ha implementado es añadir al paquete que se envía el tamaño que ocupa el paquete a enviar al comienzo del paquete.

Para mejorar el retardo que es causado por el cambio de modo de comunicación se ha añadido la posibilidad de enviar comandos directos sin necesitad de reconocimiento, es decir, no se asegura que lo que se envíe al NXT sea procesado correctamente a causa de cualquier tipo de error. Con eso se evita el cambio de modo de transmisión y los 30 ms que antes afectaban, ahora no se producen.

El formato de los paquetes que se envían en ambas direcciones tiene la forma que muestra la figura.

Length, LSB	Length, MSB	Command Type	Command	Byte 5	Byte 6	Etc.

Figura 54: Formato de trama en NXT

Se pueden distinguir, como se observa en la figura, cuatro partes que proporcionan información cuando una trama llega a su destino:

- **Bytes 0 y 1:** Indican el tamaño que tiene el paquete enviado, en dicho tamaño no se contemplan esos dos bytes de información.

- **Byte3:** Tipo de comando. Existen dos tipos de comandos, los comandos directos y los comandos de sistema. Estos a su vez pueden ser con contestación de estado o sin contestación

0x00 → Comando directo con respuesta.
0x80 → Comando directo sin respuesta.
0x01 → Comando de sistema con respuesta.
0x81 → Comando de sistema sin respuesta.
0x02 → Indica que el paquete recibido es la respuesta a otro paquete.

- **Byte 4:** Comando específico. Dentro de los dos tipos posibles hay comandos que tienen un código específico para realizar diferentes funciones dentro del ladrillo NXT.

- **Bytes 5 a N:** Se utiliza para información adicional del comando específico, cada comando tiene un formato que difiere del resto.

En lo que concierne a los paquetes de respuesta el formato que siguen es el que se muestra a continuación:

- **Byte 2** → Siempre vale lo mismo y es 0x02 para indicar respuesta.

- **Byte 3** → Estado de la respuesta. Este valor puede variar según estos valores.

0x00 → Éxito
0x81 → Sin manejadores
0x82→ Sin espacio
0x83→ No hay más ficheros
0x84→ Se esperaba final de fichero
0x85→ Final de fichero
0x86→ No es un fichero lineal
0x87→ Fichero no encontrado

0x88→ Todos los manejadores están cerrados
0x89→ Sin espacio lineal
0x8A → Error desconocido
0x8B → El fichero está ocupado
0x8C → No hay buffers de escritura
0x8D → No se puede añadir
0x8E → El fichero está lleno
0x8F → El fichero existe
0x90 → Módulo no encontrado
0x91 → Fuera del límite
0x92 → Nombre incorrecto de fichero
0x93 → Manejador ilegal

2.5.4.2. Comandos dentro del protocolo de comunicación

En el siguiente apartado se describirá la arquitectura del protocolo de comunicación del NXT. Esta capa está situada debajo de los puertos de comunicación tanto del de USB como del de Bluetooth ya que será la que maneje la información que será intermediada dentro de las capas de comunicación USB y Bluetooth.

En cada una de las capas que forman la pila del protocolo se comprueban los errores y la configuración de los paquetes a enviar tanto para USB como para Bluetooth.

Dejando a un lado las características físicas del protocolo se va a abordar los posibles ficheros que la nueva comunicación del NXT es capaz de manejar. Se pueden agrupar los tipos de ficheros en:

- **Descargas**
  - De firmware.
  - De programas definidos por el usuario.
  - De programas generados por una aplicación externa (pe. RobotC)
  - Programas de prueba (ficheros pre-programados).
  - Sonidos y gráficos.
- **Subidas**
  - De programas.
  - De gráficos y sonidos.
  - De información de las salidas de los programas (logs).
- **Borrados de información en el brick**
  - Sonidos, gráficos.

- De programas definidos por el usuario.
- Programas de prueba (ficheros pre-programados).
- De programas generados por una aplicación externa (pe. RobotC).
- Ficheros de salida de los programas (logs).

- **Comandos de comunicación con el NXT**
  - Obtener la lista de ficheros dentro del NXT.
  - Comandos dirigidos a la máquina virtual (VM).
  - Mensajes de comunicación con los buzones (buffers donde se almacena la información de salida y entrada de NXT).
  - Datos directos desde el puerto de alta velocidad.

2.5.5. Software

El éxito de la programación de los bricks del NXT ha dado lugar a la implementación de diferentes lenguajes para desarrollar aplicaciones que controlen los sensores, los motores, las comunicaciones... en definitiva, a un gran abanico de posibilidades según sea el lenguaje preferido para el programador. Tanto es así que al poder acceder al código del firmware al ser de libre distribución se puede adaptar a diferentes Sistemas Operativos. De hecho, existe una versión para Linux que posibilita la programación de estos robots móviles sin tener que abandonar su querido entorno de trabajo habitual.

En los siguientes puntos se van a citar una serie de lenguajes para compararlos entre sí y ver para qué funciones un lenguaje ofrece más facilidades y rapidez que otros.

2.5.5.1. nxtOSEK

nxtOSEK es un sistema operativo en tiempo real y de código abierto para NXT. Esta plataforma ofrece:

- Entorno de programación ANSI C / C++, usando el conjunto de herramientas GCC; con API (Interfaz de Programación de Aplicaciones) en C y C++ para los sensores de NXT oficiales y fabricados por terceros, así como para los servomotores.
- TOPPERS/ATK: kernel que ofrece características multitarea y en tiempo real.
- Ejecución rápida y menos consumo de memoria. nxtOSEK se ejecuta de manera nativa en el ARM7 y por sí sólo consume únicamente 10Kbytes

Existe soporte oficial para Windows, y no oficial para Linux. MacOSX no está soportado. Algunas de sus ventajas son su ligereza, el soporte para I2C y para RCX, y la posibilidad de usar archivos WAV para audio y BMP para imágenes, eliminando las limitaciones que existen en otros lenguajes. Tiene soporte para coma flotante. Permite conexión USB con el PC. También bluetooth con el PC y entre ladrillos, sin embargo, esto último está limitado a una conexión simple entre 2 NXT (no podemos conectar más de un ladrillo esclavo al ladrillo maestro).

2.5.5.2. Embedded Coder Robot NXT

Embedded Coder Robot NXT es un entorno de desarrollo para NXT basado en MATLAB y Simulink. Ofrece capacidad de programar el control del ladrillo y un entorno gráfico de realidad virtual 3D donde simular y visualizar el comportamiento del modelo. Esta programación en realidad es un modelado a partir de los bloques proporcionados por Simulink (operaciones matemáticas, control de flujo del código…) y los de la propia librería de ECRobot que permiten acceder a las funcionalidades de NXT. Estos modelos son traducidos por Real-Time Workshop Embedded Coder a código C, usando las funciones de nxtOSEK para los bloques de ECRobot. El objetivo principal de este entorno es pues la facilitación de la programación en nxtOSEK gracias a lo visual e intuitivo que resulta.

2.5.5.3. LeJOS

LeJOS es un firmware disponible libremente para instalar y reemplazar el original de los ladrillos de Lego. Existen dos variantes, LeJOS NXJ para el ladrillo NXT y LeJOS RCX para el ladrillo RCX. Este software incluye la máquina virtual Java, por lo que permite a los robots de Lego Mindstorms que se programen en el lenguaje Java y además que se pueda utilizar tanto en Windows como en Linux junto a cualquier software compilador de Java como Eclipse o NetBeans.

Aunque es un software prácticamente en desarrollo, poco a poco se le han ido añadiendo interesantes funcionalidades como navegación, analizadores de visión y otros servicios.

2.5.5.4. Otras alternativas

- **BricxCC:** Bricx Command Center es un conocido IDE que soporta programación del RCX con NQC, C, C++, Pascal, Forth, y Java utilizando brickOS, pbForth y LeJOS. Con BricxCC se pueden desarrollar programas en NBC y NXC. Tanto NBC como NXC utilizan el firmware estándar del NXT. Este software está disponible en código abierto. John Hansen ofrece diversas utilidades, que poco a poco va incluyendo en BricxCC.

- **NBC (Next Byte Codes)**: NBC es un lenguaje dirigido a programadores con una síntesis de lenguaje ensamblador. Se puede utilizar como editor BricxCC. Sorosy.com ofrece un depurador para la programación con NBC.

- **NXC:** NXC es un lenguaje de alto nivel similar a C. Utiliza el firmware original de LEGO y está disponible para Windows, Mac OSX y Linux (ia32). Ha sido desarrollado por John Hansen. Hay disponible una guía del programador y un tutorial en ingles (http://bricxcc.sourceforge.net/nbc) y se puede utilizar como editor Bricxcc.

- **RobotC:** Es un lenguaje muy parecido a C que hace uso de un firmware más rápido que el original. Incluye una interfaz de programación con distintas funciones que nos ayudan en el desarrollo, como un depurador. Dentro de la aplicación se pueden encontrar numerosos ejemplos escritos en dicho lenguaje para probarlos directamente con el robot para comprobar su funcionamiento.

- **NXT-G:** Lego NXT-G está diseñado para trabajar con el firmware original de Lego. Este software está basado en Labview, un lenguaje por bloques que es intuitivo, rápido y sencillo de programar. Por defecto este lenguaje lleva programadas rutinas que hacen que aún sea más rápida la programación. El problema que tiene este lenguaje es que está destinado para alumnos de enseñanza obligatoria con lo cual el aprovechamiento que puede proporcionar esta herramienta no es muy elevado.

La lista de lenguajes que se ha expuesto es algo más extensa pero no era la finalidad de este apartado comentar todas las posibilidades. Sin embargo se va a proceder a agrupar todas las características comunes a todos los lenguajes para hacer una comparativa mucho más visual y rápida. En las siguientes tablas no se recogen todas las características que tiene cada uno de los lenguajes pero puede ser suficiente para hacerse una idea sobre que lenguaje que mejor se adapte a sus necesidades.

Características	NXT-G Retail	NXT-G Educational	RoboLab 2.9	NBC	NXC	Robot C	NI LabVIEW Toolkit	leJOS NXJ	pbLua	LEJOS OSEK
Tipo de lenguaje	Gráfico	Gráfico	Gráfico	Ensamblador	Como C	C	Gráfico	Java	Lua	ANSI C
Firmware	Estándar	Standard	Estándar(#1)	Estándar	Estándar	Estándar (#1)	Estándar	Modificado	Modificado	Modificado
IDE (¿incluido?)	Si	Si	Si	Si	Si	Si	No (#6)	plugins para Eclipse y NetBeans	No (#7)	Eclipse CDT(GCC+ ATMEL SAM-BA)
Windows	Si	Si	Si	Si	Si	Si	Si	Si	Si (#7)	Si
Mac OSX	Si	Si	Si	Si	Si	Aún no	Si	Si	Si (#7)	No
Linux	No	No	No	Si	Si	No	No	Si	Si (#7)	No (puede)
Eventos	No	No	Si	No	No	Si	No	Eventos estándar de java		Si (OSEK RTOS)
Multitarea	Si	Si	Si	Si	Si	Si	Si	Si		Si (OSEK RTOS)
Bluetooth Brick hacia PC	Si	Si	No	Si	Si	Si	Si	Si	Si	Si
Bluetooth Brick hacia Brick	Si	Si	No	Si	Si	Si	Si	Si	Si	Aún no
Bluetooth Brick hacia otro dispositivo	No	No	No	No	No	Si	No	Si	Si	No
I2C Support	(#5)	(#5)	Si	Si	Si	Si	Si	Si	Si	Si (EEUU sólo)
File System	Si	Si	Si	Si	Si	Si	Si	Si	Aún no	Sin planear
Floating Point	No	No	Si	No	No	Si	¿No?	Si	(#8)	Si
Datalog	No	No	Si	No	No	Si	¿No?	Si	No	Aún no

Figura 55: Comparativa de diferentes lenguajes de desarrollo para NXT

Software	Tipo de Lenguaje	Tipo de Control	Requiere Firmware del NXT	Tipo de Conexión	Fuente de Conexión	Windows	Mac OSX	Linux	Lectura de Sensores	Websites
LEGO NXT Mobile Application	Simple RC	Control Remoto	Estándar (#2)	Bluetooth	Teléfono o PDA	-	-	-	No	LEGO
FunkNXT	Simple RC	Control Remoto	Estándar	Bluetooth	Teléfono	-	-	-	Si	FunkNXT
BT RC	NXT-G	NXT a NXT remotamente	Programa ejecutándose en el NXT	Bluetooth	Otro NXT	-	-	-	Programable por el usuario	BTRC
Simple BT Remote	Simple RC	Control Remoto	Estándar	Bluetooth	Escritorio	Si	No	No	Si	Simple Windows RC
RobotC	Simple RC	Control Remoto	Estándar (#1)	USB/BT	Escritorio	Si	Aún no	No	Si	Robot C Web Site
BricxCC	Simple RC	Control Remoto	Estándar	USB/BT	Escritorio	Si		No	Si	BricxCC Web Site
OnBrick PDA	Gráfico	RC Programable	Estándar	Bluetooth	PDA	-	-	-	Si	OnBrick
OnBrick PC	Gráfico	RC Programable	Estándar	Bluetooth	Escritorio	Si	No	No	Si	OnBrick
NXT Director	Simple RC	Control Remoto Modificable	Estándar	Bluetooth	Palm PDA	-	-	-	¿No?	Director
RoboDNA	Simple RC	Control Remoto	Estándar	Bluetooth	Escritorio	Si			Si	RoboDNA
MS Robotics Studio	.NET	Programa de usuario ejecutándose en el PC	Estándar	Bluetooth	Escritorio	Si			Si	(version no commercial gratis) Download site o Microsoft Site
NI LabVIEW Toolkit	Gráfico (LabVIEW G)	Programa de usuario ejecutándose en el PC	Estándar	USB/BT	Escritorio	Si	Si		Si	LabVIEW toolkit Site
RoboLab	Gráfico	Programa de usuario ejecutándose en el PC	Estándar	USB	Escritorio	Si	Si		Si	Robolab
iCommand	Java	Programa de usuario ejecutándose en el PC	Estándar	Bluetooth	Escritorio o PDA	Si		Si	Si	iCommand
LEGO::NXT	Perl	Programa de usuario ejecutándose en el PC	Estándar	USB/BT	Escritorio	Si	Si	Si	Si	Perl
nxt-Ruby	Ruby	Programa de usuario ejecutándose en el PC	Estándar	Bluetooth	Escritorio	Si	Si	Si	Si	Ruby
NXT#	C#	Programa de usuario ejecutándose en el PC	Estándar	Bluetooth	Escritorio	Si			¿Si?	NXT#
Mindsqualls	C#	Programa de usuario ejecutándose en el PC	Estándar	Bluetooth	Escritorio	Si			Si	Mindsqualls
NXT Python	Python	Programa de usuario ejecutándose en el PC	Estándar	USB/BT	Escritorio	Si			¿Si?	Python
My Robot Me	¿Gráfico?	Programa de usuario ejecutándose en el PC	Estándar	USB/BT	Escritorio	Si	No	No	Si	Robot Me

Notas

(1) RobotC usa firmware estándar que proviene del software de LEGO.
(2) Aplicaciones de LEGO para móviles pueden enviar mensajes a los programas que están ejecución en un NXT.

Figura 56: Comparativa de las distintas funcionalidades.

2.6. Entorno de desarrollo de software

Para el trabajo llevado a cabo se estudiaron las posibles opciones disponibles para desarrollar el software. Las dos principales se detallan a continuación:

2.6.1. MATLAB

MATLAB (MATrix LABoratory) es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware.

Las funcionalidades de Matlab se agrupan en más de 35 cajas de herramientas y paquetes de bloques (para Simulink), clasificadas en las siguientes categorías:

MATLAB (Cajas de herramientas)	**Simulink**
Matemáticas y Optimización	Modelado de punto fijo
Estadística y Análisis de datos	Modelado basado en eventos
Diseño de sistemas de control y análisis	Modelado físico
Procesado de señal y comunicaciones	Gráficos de simulación
Procesado de imagen	Diseño de sistemas de control y análisis
Pruebas y medidas	Procesado de señal y comunicaciones
Biología computacional	Generación de código
Modelado y análisis financiero	Prototipos de control rápido y SW/HW HIL

Figura 57: Lenguajes utilizados con Eclipse.

Es un software muy usado en universidades y centros de investigación y desarrollo. En 2004, se estimaba que MATLAB era empleado por más de un millón de personas en ámbitos académicos y empresariales.

2.6.1.1. Simulink

Simulink es un entorno de programación visual, que funciona sobre MATLAB.

Es un entorno de programación de más alto nivel de abstracción que el lenguaje interpretado MATLAB (archivos con extensión .m). Simulink genera archivos con extensión .mdl (de "model"). Viene a ser una herramienta de simulación de modelos o sistemas, con cierto grado de abstracción de los fenómenos físicos involucrados en los mismos. Se hace hincapié en el análisis de sucesos, a través de la concepción de sistemas (cajas negras que realizan alguna operación).

2.6.2. Eclipse

Conocido por cualquier programador de lenguaje Java, aunque también utilizado para el desarrollo en otros lenguajes de programación como C/C++, PHP o Python, Eclipse 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. A continuación se muestra una tabla resumen de la situación actual del software.

Estado actual de Eclipse	
Líneas de código fuente	2,063,083
Esfuerzo estimado de desarrollo(persona-año / persona-mes)	604.33 / 7,251.93
Estimación de tiempo (años-meses)	6.11 / 73.27
Estimación del nº de desarrolladores en paralelo	98.98
Estimación de coste	$ 81,636,459

Figura 58: Estado actual de Eclipse.

Un punto muy importante a notar son los diversos lenguajes de programación utilizados en el desarrollo del proyecto, de acuerdo al análisis realizado usando SLOCCount, el lenguaje más utilizado es Java, seguido de ANSI C.

Lenguajes de programación utilizados en Eclipse 3.2.1

Lenguaje	Líneas de código	%
Java	1,911,693	92.66 %
ANSI C	133,263	6.46%
C++	10,082	0.49%
JSP	3,613	0.18%
sh	2,066	0.10%
perl	1,468	0.07%
php	896	0.04%
sed	2	0.00%

Figura 59: Lenguajes utilizados con Eclipse.

Actualmente Eclipse continúa en desarrollo y su última versión es Juno, la número 4.2, que salió en junio de 2012.

3.- Desarrollo práctico

Una vez expuestas las bases teóricas necesarias para el desarrollo de nuestros objetivos, se continúa con la descripción de la parte práctica del proyecto. Esta parte está dividida en dos. En la primera utilizaremos el entorno ECRobot en Windows con el firmware nxtOSEK y en la segunda utilizaremos LeJOS en Linux Ubuntu.

3.1. Prámbulos y configuraciones

3.1.1. Montaje del robot NXT

Siguiendo el guión de las finalidades de este proyecto, el montaje del robot NXT no puede ser de cualquier manera. Se necesita que cumpla unas necesidades de movilidad, de tracción y de dirección. De las opciones planteadas en la parte teórica, se opta por la forma de triciclo: dos ruedas motrices que controlan tracción y dirección al mismo tiempo con un mecanismo diferencial y una rueda trasera más pequeña y libre.

3.1.2. ECRobot en Windows

Como ya se ha comentado en el apartado teórico, la librería ECRobot se utiliza en el entorno Simulink de MATLAB. MATLAB es un software de pago, por lo que tendremos que adquirir una licencia individual o de estudiante en la web del propietario http://www.mathworks.es/products/matlab, donde también se puede conseguir una versión de prueba por 30 días. Se requiere una versión de 32 bits a partir de la R2007.a. En este caso se ha usado la versión R2009.b del producto. Una vez conseguido e instalado podemos pasar a la configuración del entorno.

3.1.2.1. Instalación del entorno ECRobot

Para que este proceso no se nos haga engorroso y tengamos que ir paso a paso, existe el proyecto ECRobotInstaller que se puede descargar desde http://www.mathworks.com/matlabcentral/fileexchange/25207. Es básicamente una recopilación de scripts para MATLAB que nos descargar e instalan automáticamente tanto las dependencias como el entorno y el firmware del NXT. En el mismo archivo descargado tenemos las instrucciones para realizarlo correctamente.

3.1.2.2. Instalación del driver USB de Lego

El propio ECRobotInstaller nos descarga e instala el driver Lego para Windows, pero, además, también es interesante descargar e instalar desde http://mindstorms.lego.com/en-us/support/files/default.aspx#Driver el driver Fantom de Lego Mindstorms NXT, gracias al que podremos comunicarnos con el robot desde nuestros propios programas escritos en C.

3.1.3. LeJOS en Linux Ubuntu

Antes de empezar hay que señalar que en todo momento, tanto para el desarrollo como la ejecución de las aplicaciones, se ha trabajado en una distribución Ubuntu 11.04 (Natty).

En cuanto a LeJOS, en su página oficial tenemos varios tutoriales en http://lejos.sourceforge.net/nxt/nxj/tutorial/index.htm para familiarizarnos y conocer más a fondo la librería. Además, en la sección "Getting Started > Linux" del mismo tenemos explicado perfectamente como instalar y preparar nuestro entorno, aunque a continuación se van a destacar los puntos esenciales y más importantes.

3.1.3.1. Instalación del Java Development Kit (openJDK)

Puesto que se va a trabajar con el lenguaje Java se necesita instalar la Máquina Virtual Java, con sus entornos de desarrollo y ejecución. En este proyecto se ha utilizado la versión 6 de openJDK, que es fácilmente instalable desde consola mediante el comando "*sudo apt-get install openjdk-6-jdk*".

3.1.3.2. Instalación de libUSB

También se necesita tener instaladas las librerías apropiadas para la comunicación NXT-PC. En este caso ha de ser la versión 0.1 de libUSB junto con su paquete de desarrollo. El comando para instalarlo todo es "*sudo apt-get install libusb-0.1-4 libusb-dev*". Una vez instalado hay que dotar al usuario de permisos de lectura y escritura sobre la conexión USB, para lo que hay que crear una regla udev creando el fichero */etc/udev/rules.d/70-lego.rules* con el siguiente contenido:

```
# Lego NXT brick in normal mode

SUBSYSTEM=="usb", DRIVER=="usb", ATTRS{idVendor}=="0694", ATTRS{idProduct}=="0002", GROUP="lego", MODE="0660"

# Lego NXT brick in firmware update mode (Atmel SAM-BA mode)

SUBSYSTEM=="usb", DRIVER=="usb", ATTRS{idVendor}=="03eb", ATTRS{idProduct}=="6124", GROUP="lego", MODE="0660"
```

Posteriormente con las llamadas desde consola "*sudo groupadd lego*" y "*sudo gpasswd -a <usuario> lego*" se crea el grupo con los permisos y se añade al usuario al mismo.

3.1.3.3. Instalación de LeJOS NXJ

A continuación se pasa ya a instalar tanto el software de LeJOS como el firmware en el NXT. Para ello se debe descargar el archivo comprimido desde http://lejos.sourceforge.net/nxj-downloads.php (se recomienda la versión 0.9.0 utilizada en este proyecto).

El software incluye una biblioteca de clases de Java (Classes.jar) que implementan la LEJOS NXJ Application Programming Interface (API), herramientas de PC para actualizar el firmware del robot, cargar los programas, depurarlos, un API de PC para escribir programas de PC que se comunican con los programas de LEJOS NXJ a través de Bluetooth o USB y otras muchas funciones y programas de ejemplo.

Una vez descargado el archivo se descomprime en la ubicación deseada y posteriormente se edita el archivo *~/.bashrc* para añadir la variable *NXJ_HOME* con la ubicación de la instalación (e.g. "*~/lejos_nxj*") y agregar *$NXJ_HOME/bin* a la variable *PATH*. Hecho esto, se accede desde consola a la carpeta *$NXJ_HOME/build* y se ejecuta el comando "*ant*", con lo que tendremos la instalación completada.

Para instalar el firmware al NXT, se conecta el robot al ordenador por USB y se llama desde consola al comando nxjflash.

Para realizar las aplicaciones y pruebas se ha utilizado el entorno de trabajo de Eclipse, por su popularidad entre programadores en Java y por la familiaridad personal con la que se contaba anteriormente.

3.1.3.4. Configuración del entorno de desarrollo (Eclipse)

Desde http://www.eclipse.org/downloads/ se descarga la última versión de ECLIPSE IDE para desarrolladores Java y se descomprime en el directorio deseado del disco duro.

Aunque en la página de leJOS también existe un tutorial en esta dirección http://lejos.sourceforge.net/nxt/nxj/tutorial/Preliminaries/UsingEclipse.htm para la configuración de la librería en Eclipse y con la explicación extensa de todo lo que podemos aprovechar en este entorno, se destacan los puntos esenciales para la instalación del plug-in de leJOS:

Se hace clic en "Help > Install New Software..."

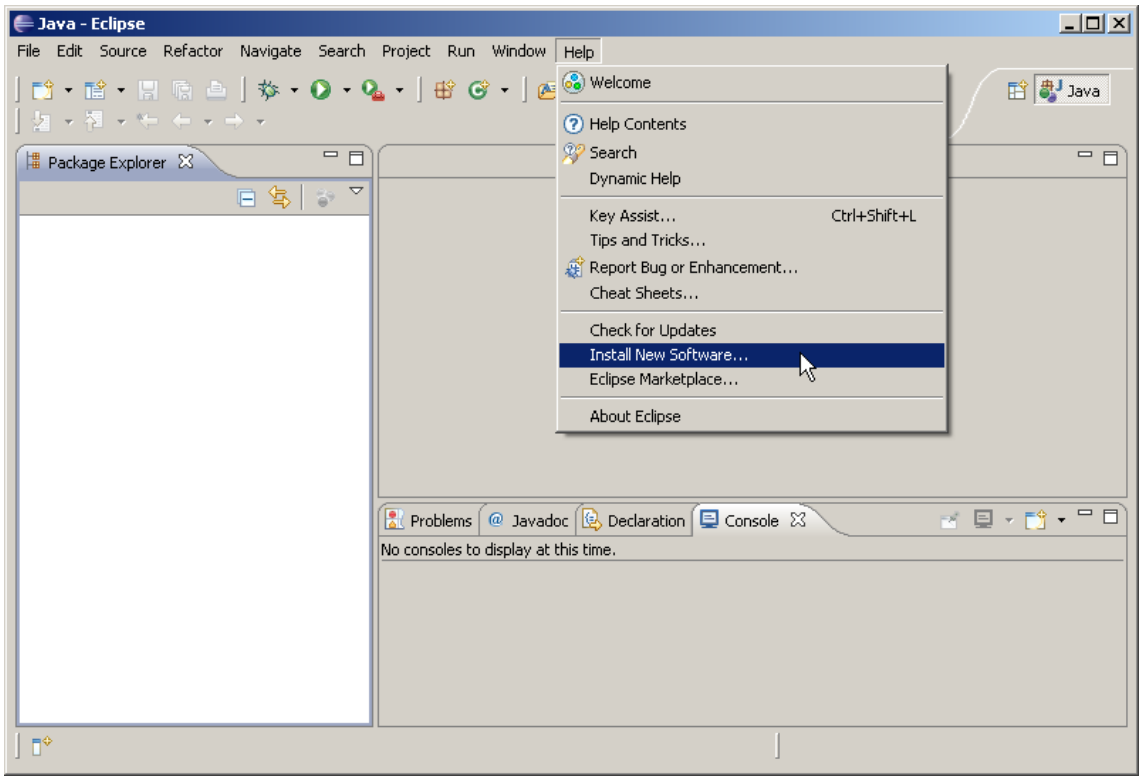


Figura 60: Configuración de Eclipse 1

En la nueva ventana se introduce en el campo "Work with:" la dirección http://lejos.sourceforge.net/tools/eclipse/plugin/nxj/ y se pulsa enter.

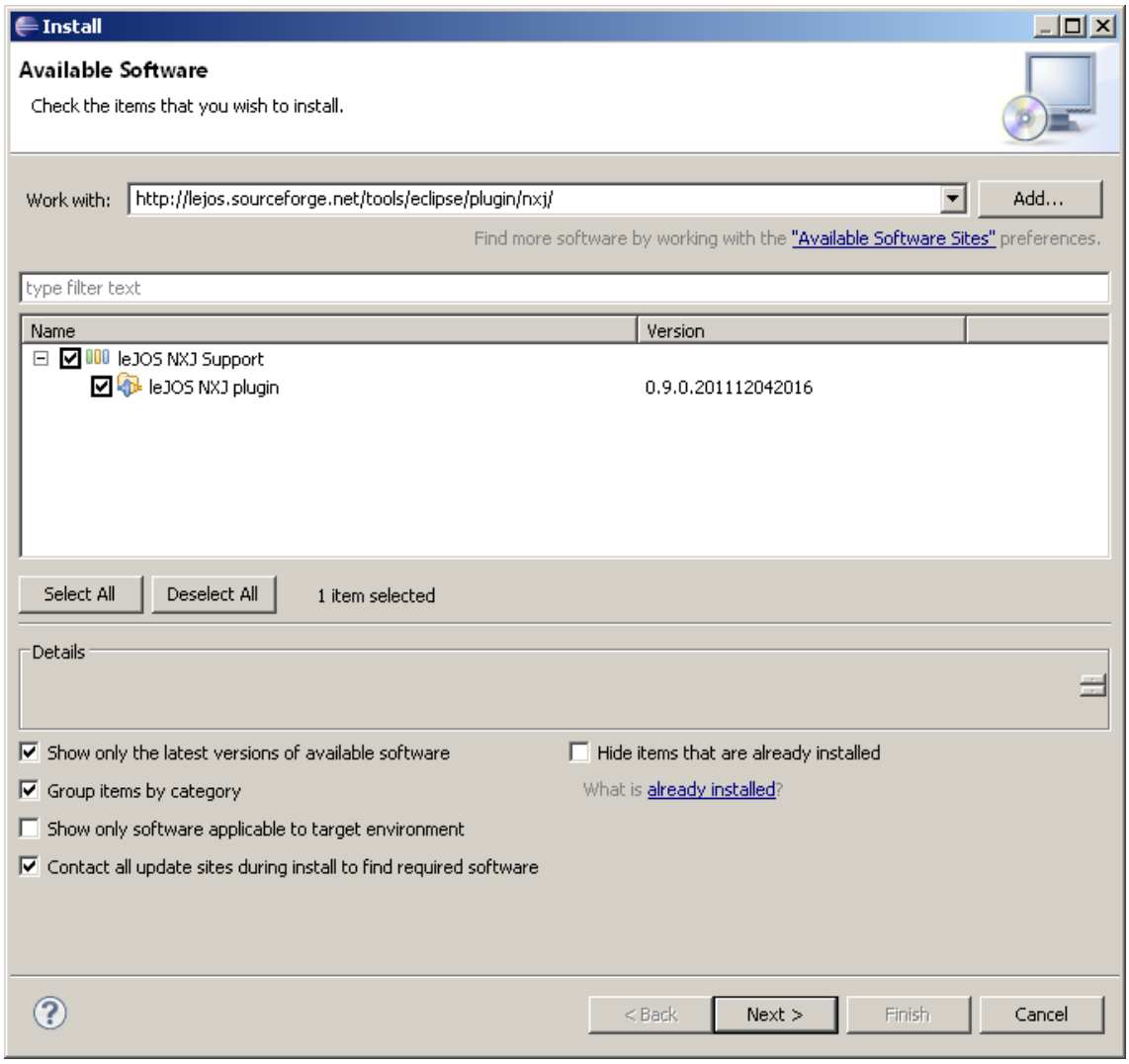


Figura 61: Configuración de Eclipse 2

Con la casilla del plug-in seleccionada se pulsa Next y se empieza la instalación. Una vez finalizada se procede a reiniciar Eclipse, que ya tendrá instalado el plug-in deseado. A continuación se puede revisar y modificar las preferencias del mismo en "Window > Preferences > LeJOS NXJ".

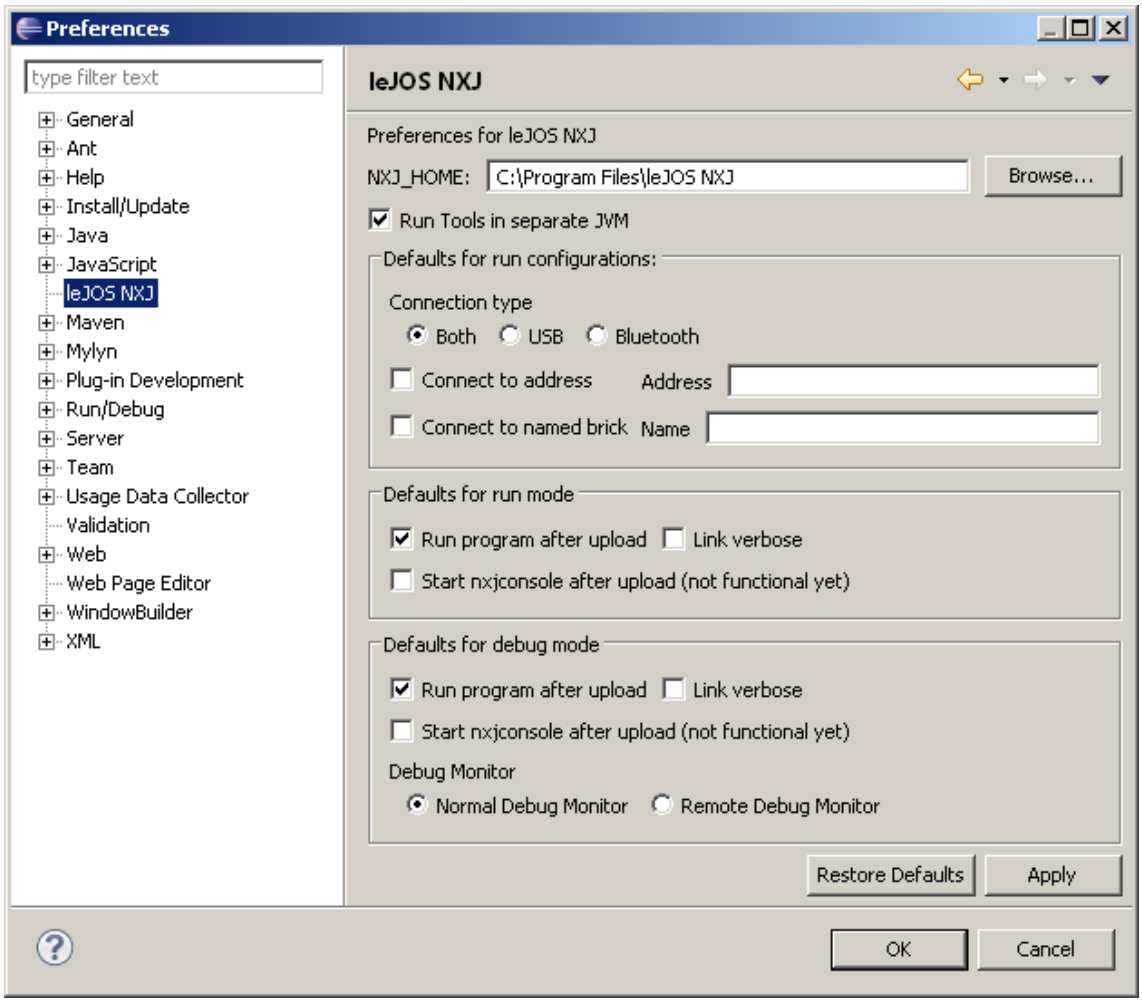


Figura 62: Configuración de Eclipse 3

Hay que asegurarse de que en el campo NXJ_HOME esté indicada la ruta donde se instaló el software de LeJOS. Con todo instalado y configurado ya se puede proceder a crear proyectos LeJOS tanto para PC como NXT, seleccionando la opción deseada en el menú "File > New > Project..." así como ejecutarlos (teniendo conectado el robot vía USB en el caso de los proyectos para NXT) desde la opción Run As....

3.1.3.5. Instalación de ROS

Para la ejecución de la última aplicación del proyecto es necesaria la instalación de ROS y algunos paquetes y pilas que se señalan a continuación.

En la página oficial de ROS http://www.ros.org/wiki/diamondback/Installation/Ubuntu se encuentra un tutorial para la instalación completa de ROS. Se ha de añadir el repositorio a la lista con el comando "*sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu natty main" > /etc/apt/sources.list.d/ros-latest.list'*", se instalan las llaves "*wget http://packages.ros.org/ros.key -O - | sudo apt-key add -*" y se actualiza la lista de repositorios "*sudo apt-get update*". Se necesita una instalación completa que se

consigue con el comando "*sudo apt-get install ros-diamondback-desktop-full*" y se finaliza añadiendo las variables globales al entorno del sistema con la orden "echo "source /opt/ros/diamondback/setup.bash" >> ~/.bashrc".

Con la base de ROS instalada es el momento de instalar el paquete que se utilizará, rgbdslam. En la página http://www.ros.org/wiki/rgbdslam se encuentra el tutorial de instalación y uso del mismo. En primer lugar hay que crear un directorio personal para los paquetes con las llamadas "*mkdir -p ~/ros*", "*echo 'export ROS_PACKAGE_PATH=~/ros:$ROS_PACKAGE_PATH' >> ~/.bashrc*" y "*source ~/.bashrc*". Entonces en este directorio se descargan los dos paquetes necesarios con las órdenes "*svn co http://alufr-ros-pkg.googlecode.com/svn/trunk/rgbdslam_freiburg/rgbdslam*" y "*svn co https://code.ros.org/svn/ros-pkg/stacks/vslam/trunk/g2o ~/ros/g2o*", se instalan las dependencias "*sudo apt-get install libglew1.5-dev libdevil-dev libsuitesparse-dev*" y se construyen los paquetes "*rosmake --rosdep-install rgbdslam*". Antes de poder ejecutar rgbdslam se necesita el paquete openni_camera, que se instala mediante la llamada "*sudo apt-get install ros-diamondback-openni-kinect*" seguida de "*rosmake openni_camera*".

En este momento ya se tiene todo el entorno preparado para el desarrollo y la ejecución de las aplicaciones.

3.2. Estrategias del control cinemático

Para el control de trayectorias existen dos métodos geométricos diferentes que resultan interesantes para el trabajo a realizar: el control o seguimiento de trayectorias mediante la estrategia del punto descentralizado y el algoritmo de persecución pura.

3.2.1. 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 $\boldsymbol{e}$ del eje de tracción del robot.

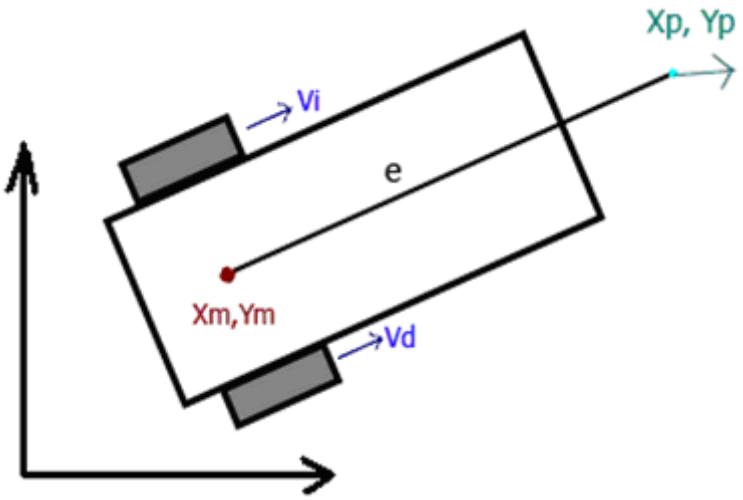


Figura 63: Punto descentralizado

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

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} \dot{x}_m + e\cos(\theta) \\ \dot{y}_m + e\sin(\theta) \end{bmatrix}$$

Se calcula la derivada (velocidad) de ese punto:

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} \dot{x}_m - e\cos(\theta)\dot{\theta} \\ \dot{y}_m + e\sin(\theta)\dot{\theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -e\sin(\theta) \\ 0 & 1 & e\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{x}_m \\ \dot{y}_m \\ \dot{\theta} \end{bmatrix}$$

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:

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

Para el desarrollo del control cinemático de los NXT 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.

3.2.2. Persecución Pura.

A diferencia de la estrategia del punto descentralizado, para aplicar el algoritmo de persecución pura es necesario conocer la trayectoria previamente.

Dada una posición (X_R,Y_R) y un punto objetivo (Xob,Yob) se puede analizar el recorrido como sigue:

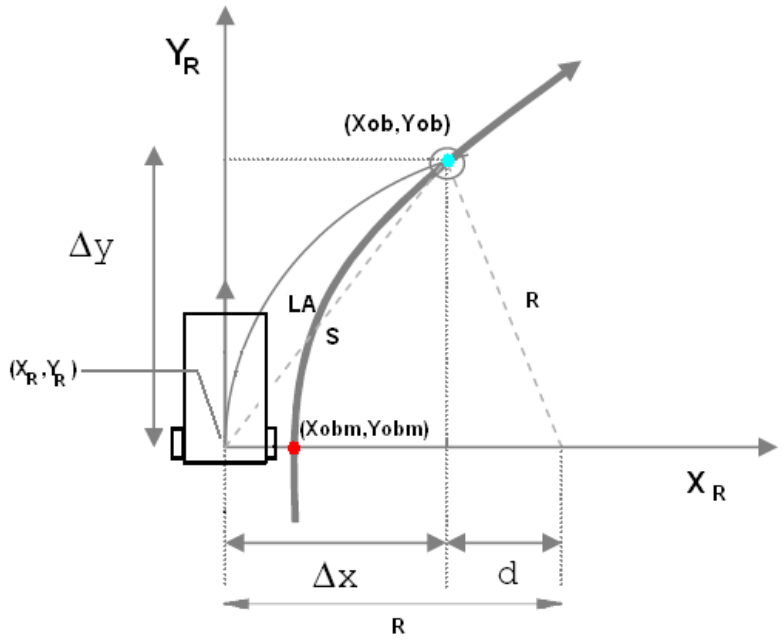


Figura 64: Persecución pura

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 estará definida en función de las coordenadas actuales y del punto objetivo:

$$LA = \sqrt{(x_{ob} - x_R)^2 + (y_{ob} - y_R)^2}$$
$$\Delta x = (x_{ob} - x_R)\cos\theta + (y_{ob} - y_R)\sin\theta$$

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

$$v_d = V + \frac{b}{2}\omega = V + \frac{b}{2}\frac{V}{r} = V\left(1 + \frac{b}{2}\frac{1}{r}\right)$$
$$v_d = V\left(1 + \frac{b}{2}\gamma\right)$$

$$v_i = V - \frac{b}{2}\omega = V - \frac{b}{2}\frac{V}{R} = V\left(1 - \frac{b}{2}\frac{1}{R}\right)$$
$$v_i = V\left(1 - \frac{b}{2}\gamma\right)$$

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.

3.3. Desarrollo en ECRobot

Se va a proceder ya a la explicación de las aplicaciones realizadas en este proyecto, empezando por lo desarrollado con la librería ECRobot. Pero antes de pasar al detalle de las aplicaciones se van a puntualizar los bloques más importantes de la librería utilizados y se va a estudiar la conexión vía Bluetooth utilizada en este entorno.

3.3.1. Descripción de bloques

Para conseguir un prefecto control del robot en tiempo real se necesitará trabajar a bajo nivel sobre sus sensores y actuadores. En este caso, ECRobot ofrece unos bloques que permiten enviar directamente la potencia deseada a los motores, así como leer la posición de sus encoders.

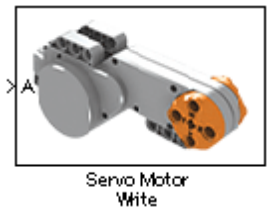


Figura 65: Bloque servo entrada

Este bloque está asociado a la función de nxtOSEK *ecrobot_set_motor_speed(U8 port_id, S8 speed)*, la cual envía la potencia *speed* (con valores entre -100 y 100)al motor conectado al puerto *port_id*.

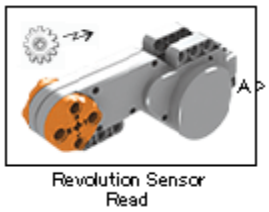


Figura 66: Bloque servo salida

Este bloque está asociado a la función de nxtOSEK *ecrobot_get_motor_rev(U8 port_id)*, la cual devuelve el valor en grados del encoder del motor conectado al puerto *port_id*.

3.3.2. Estudio de la conexión vía Bluetooth

Para tareas de prueba y depuración se propone descubrir y analizar la comunicación vía Bluetooth de los robots NXT mediante ECRobot. Principalmente se pretende recibir en el PC las coordenadas a las que se dirige el robot durante la persecución pura.

3.3.2.1. Instalación y configuración del dispositivo Bluetooth

Para poder comunicarse por Bluetooth con el NXT, habrá que instalar un dispositivo que cumpla dicha función. Algunos ordenadores de hoy en día ya incorporan un dispositivo Bluetooth integrado pero no todos son compatibles con los NXT, es recomendable instalar el original de Lego para evitar estos problemas.

Figura 67: Bluetooth Lego

Sin necesidad de drivers, este Bluetooth se instala automáticamente al conectarlo a un puerto USB. Se controlará mediante el bloque proporcionado por ECRobot, pero antes es imprescindible que se realice el enlace entre los dispositivos que vayamos a utilizar. Para realizar este emparejamiento se utiliza la aplicación BlueSoleil.

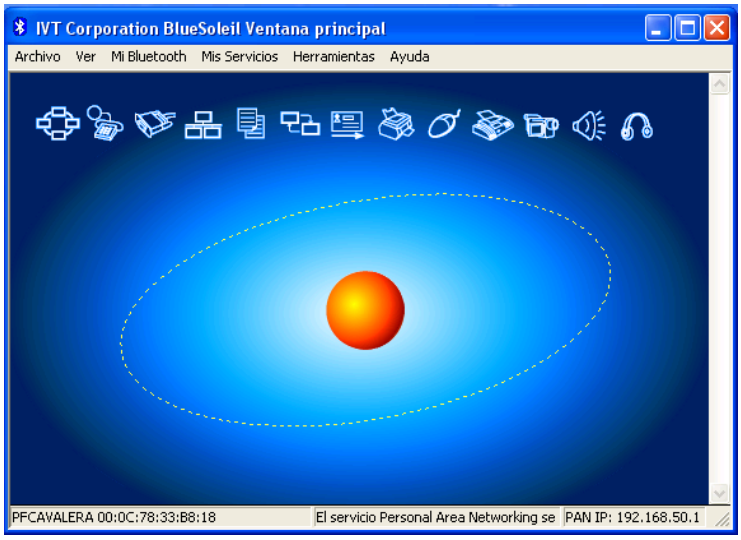


Figura 68: BlueSoleil

El primer paso es establecer la contraseña por defecto. En el menú “Mi Bluetooth > Seguridad” se introduce el valor “1234” en el campo Contraseña del Bluetooth. Hecho esto se descubren los dispositivos cercanos pulsando la esfera naranja.

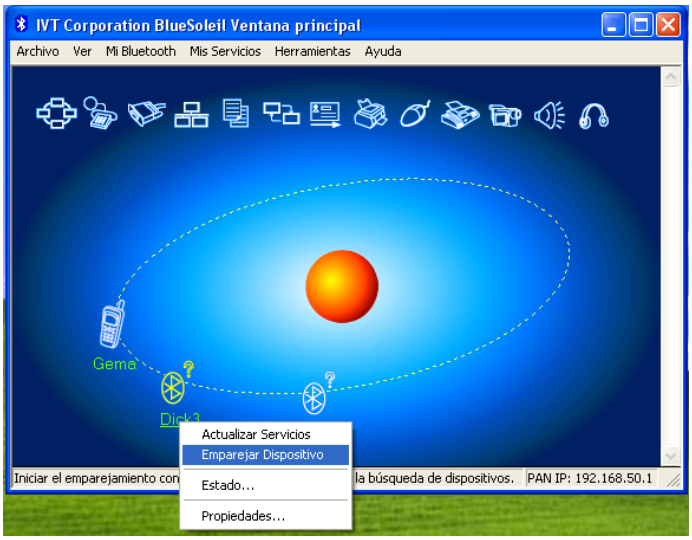


Figura 69: BlueSoleil

Cuando aparece el robot de interés se pincha con el botón derecho sobre el mismo y se elige "Emparejar Dispositivo". Una vez emparejado se elige "Actualizar Servicios" y se ejecuta la aplicación en el robot, aguantando en la pantalla de espera hasta que aparece el símbolo [BT].

Figura 70: Pantalla nxtOSEK con Bluetooth

Ahora se vuelve a pulsar con el botón derecho sobre el dispositivo en la pantalla del BlueSoleil, eligiendo la opción "Conectar > Servicio Puerto Serie de Bluetooth" y completando así la conexión entre el PC y el robot. Entonces se pueden ejecutar consecutivamente y en este orden las aplicaciones de PC y del NXT.

3.3.3. Ballbot

Esta primera aplicación se ha utilizado para entrar en contacto con el entorno ECRobot y el sistema nxtOSEK, tomando directamente el proyecto realizado y compartido por Yorihisa Yamamoto, por lo que no se va a detallar en profundidad el funcionamiento del mismo, sino que se van a remarcar los aspectos más básicos. El

proyecto con su explicación al detalle puede ser descargado desde http://www.mathworks.com/matlabcentral/fileexchange/23931.

3.3.3.1. Descripción de la aplicación

Un ballbot es un robot móvil que sigue el mismo principio que un péndulo invertido de dos ejes para conseguir autobalancearse y mantener el equilibrio sobre una pelota.

Esta aplicación básicamente controla la inclinación del robot en sus dos ejes, gracias a dos giróscopos y dos motores, uno correspondiente a cada eje a balancear. Los giróscopos, calibrados al inicio de la ejecución, miden la velocidad de rotación del robot en cada eje, información a partir de la cual se calculan las potencias a aplicar a cada motor.

Adicionalmente, el robot también es capaz de detectar obstáculos frente a él, gracias al sensor de distancia de LEGO, desplazándose posteriormente hacia atrás para evitar topar con ellos.

3.3.3.2. Montaje del robot

Un aspecto importantísimo en el desarrollo de este tipo de robot es el de su montaje. Además de ser esta una configuración muy específica y especial, hay que tener muy en cuenta la altura y anchura del montaje final del robot, puesto que cuanto más bajo esté el centro de gravedad menos problemas de estabilidad se encontrarán.

Este aspecto ha sido ampliamente estudiado y óptimamente resuelto por el desarrollador de la aplicación, que ofrece en el archivo de descarga de la aplicación un manual de construcción con las piezas necesarias y pasos a seguir. Dos de los componentes más importantes en el montaje, además de los motores, son el giróscopo de HiTechnic y el sensor ultrasónico de LEGO.

El aspecto final del robot es el siguiente.

Figura 71: Montaje Ballbot

3.3.3.3. Desarrollo

Para el desarrollo de la aplicación Ballbot se han utilizado, además de los motores ya detallados anteriormente, varios bloques específicos de la librería ECRobot.

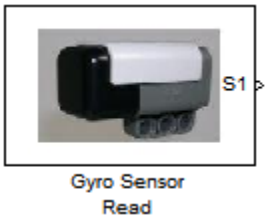


Figura 72: Bloque del giróscopo en ECRobot

El giróscopo mide la velocidad de rotación del eje en el que se encuentre, devolviendo su bloque de ECRobot un valor crudo entre 0 y 1023.

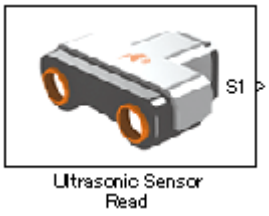


Figura 73: Bloque del sensor ultrasónico en ECRobot

El sensor ultrasónico mide la distancia hasta un obstáculo detectado, devolviendo su bloque el valor de la distancia en centímetros entre 0 y 255 (-1 en caso de error).

El proyecto consta de varios modelos y aplicaciones, la mayoría orientados a la simulación del Ballbot en el PC, pero la aplicación que nos interesa y que funcionará en nuestro robot es nxt_ballbot_controller.mdl. Ésta está dividida en 4 tareas:

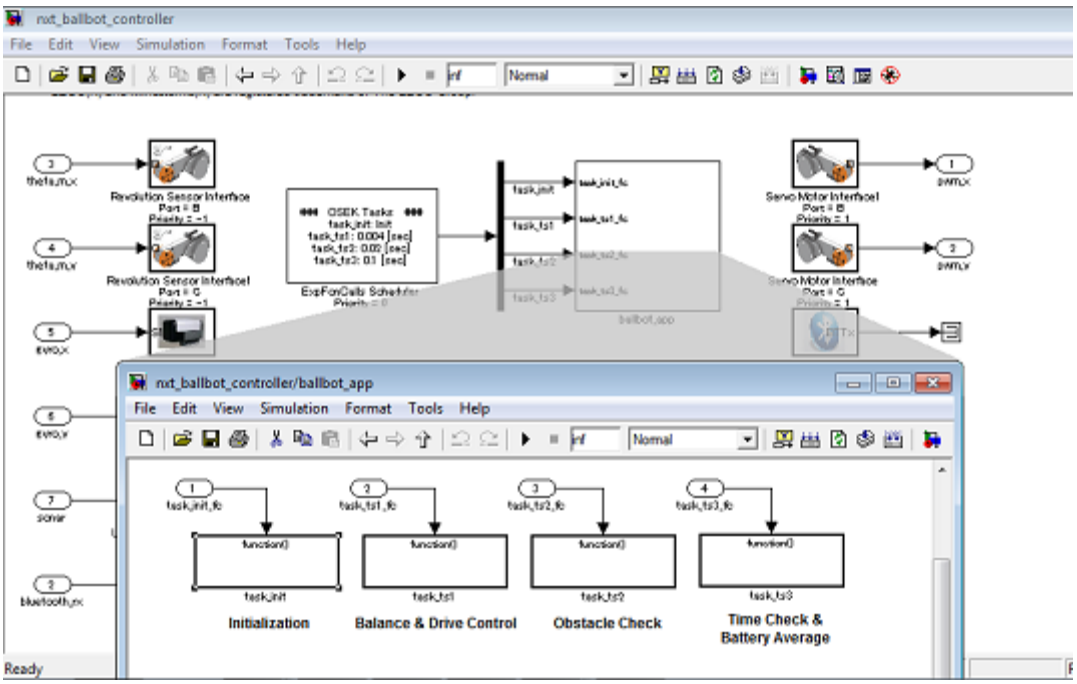


Figura 74: Ventana de la estructura de la aplicación

La primera es la tarea de inicialización que solo se lleva a cabo al principio de la ejecución. En ésta se inicializan las variables globales.

La segunda es la tarea que lleva el grueso de la aplicación, la de control y equilibrado del robot, así como la calibración de los giróscopos al inicio de la ejecución. Al ser esta la más importante tendrá una frecuencia de ejecución de 4ms.

La tercera tarea se encarga de detectar obstáculos con el sensor ultrasónico. En caso de detección se avisa a la tarea principal a través de una variable global para que ésta actúe sobre los motores de manera apropiada. Es ejecutada cada 20ms.

La cuarta y última tarea comprueba el estado de la batería para controlar el voltaje a aplicar a los motores, así como si el robot está fuera de control para parar la ejecución. También avisa al usuario mediante un pitido del fin del calibrado de los giróscopos, suponiendo esto el inicio del bucle de control. Esta tarea se ejecuta cada 100ms.

3.3.3.4. Ejecución

El primer paso, una vez construido el modelo y antes de poderlo ejecutar en el robot, es el de la generación del código y el correspondiente ejecutable y su posterior descarga en el NXT. Para ello contamos en la ventana principal del modelo con los botones mostrados

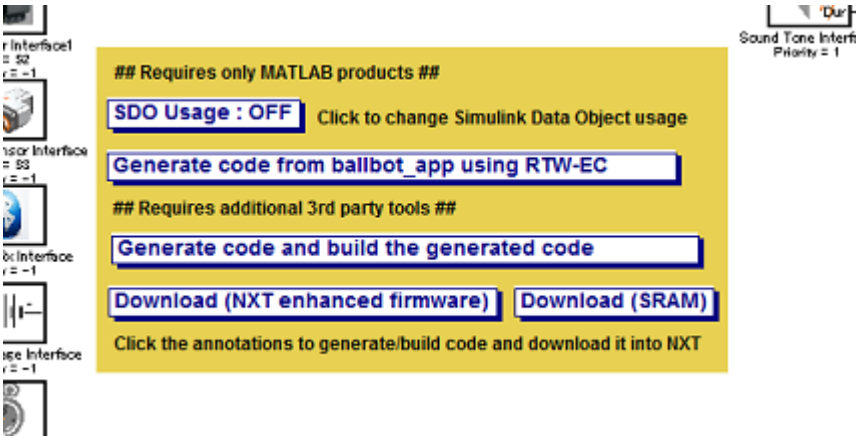


Figura 75: Ventana de generación del ejecutable

Primero se pulsa "Generate code and build the generated code". Una vez construido (se indicará en la ventana principal de MATLAB) se conecta mediante USB al PC el NXT encendido y se pulsa "Download (NXT enhanced firmware)". Entonces se ejecuta la aplicación en el robot, aguantando el robot en posición vertical sobre la pelota hasta que emite un pitido avisando del inicio del movimiento. Desde este momento el robot realiza su tarea de autobalanceado sin necesidad de interacción del usuario.

3.3.3.5. Problemas encontrados y soluciones aplicadas

Durante la utilización de esta aplicación se encontró un problema a destacar. Desde un principio no se contaba con las ruedas especificadas en el manual de construcción. El robot fue construido con unas ruedas con un diámetro ligeramente mayor, por lo que la fricción de las mismas con la pelota no era la deseada, consiguiendo así muy poca estabilidad, incluso sin poder iniciar la ejecución en algunas ocasiones. Una vez conseguidas las ruedas correctas se mejoró el funcionamiento hasta hacerlo ideal. Por esto se recomienda encarecidamente la utilización de las ruedas LEGO con referencia 44309.

3.3.4. Seguimiento de trayectorias fijas mediante el algoritmo de persecución pura

3.3.4.1. Descripción de la aplicación

Habiéndose familiarizado con el entorno ECRobot, sus bloques propios y los más básicos de Simulink, se pasa a desarrollar desde cero una aplicación en este entorno, en concreto, una aplicación de seguimiento de trayectorias fijas mediante el algoritmo de persecución pura.

Esta aplicación es capaz de generar una trayectoria (circular, cuadrada, espiral cuadrada, ocho cuadrado o flor) para posteriormente seguirla mediante el algoritmo de

persecución pura en bucle, ya que éste sigue la trayectoria estrictamente y se pueden conseguir resultados bastante satisfactorios.

Además, en cada iteración del algoritmo se envía mediante Bluetooth a un PC la referencia de la trayectoria generada y la coordenada real que sigue el robot para comprobar el resultado.

3.3.4.2. Metodología y desarrollo

La aplicación está formada por dos tareas.

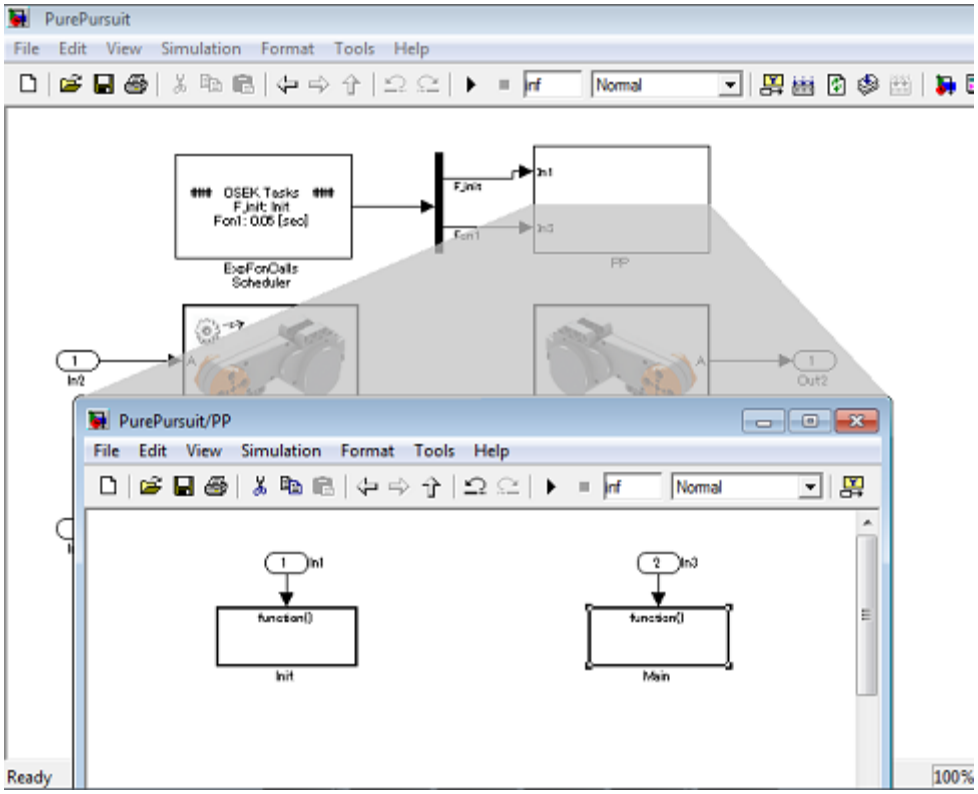


Figura 75: Ventana principal PurePursuit

La primera de ellas es la de inicialización, que solo será ejecutada al iniciar la aplicación. Aquí se inicializarán las variables y, además, se generará la trayectoria escogida. En este entorno de programación esta trayectoria no puede ser escogida en tiempo de ejecución, así que se deben de tener varias versiones de la aplicación, cada una con una trayectoria distinta, o se debe de cambiar la trayectoria y recompilar cada vez la aplicación.

La segunda tarea, con una frecuencia de ejecución de 50ms, es la principal, en la que se sigue la trayectoria anteriormente inicializada mediante el algoritmo de persecución pura.

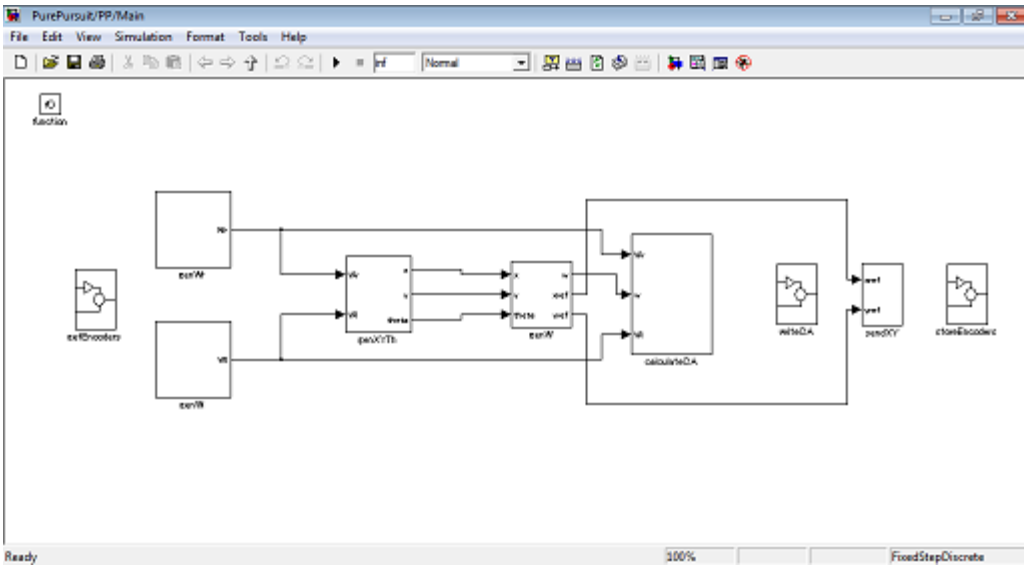


Figura 76: Tarea principal de PurePursuit

Esta tarea calcula, a partir de los valores actuales y anteriores de los encoders de los motores, la velocidad que lleva el robot y, por tanto, la posición (x,y) y rotación (theta) en la que se encuentra. Con esta coordenada calculada se pasa a buscar el siguiente punto en la trayectoria a seguir más cercano a ella y respetando una distancia mínima o *lookahead*. Esta búsqueda se hace en una ventana de puntos entre el actual (último punto al que se ha viajado) y un número variable a elección según las necesidades de la trayectoria en cuestión. Una vez encontrado el punto de referencia al que se viajará entra en juego el algoritmo de persecución pura, el cual, como se ha comentado en el apartado 3.2.2, calcula las velocidades que han de llevar las ruedas y, con ello, las potencias a aplicarles.

Por último, se envían las referencias y coordenadas reales recorridas mediante Bluetooth al PC. Para ello se utiliza el bloque correspondiente de ECRobot.

Figura 77: Bloque de envío por Bluetooth

Este bloque envía un vector de 32 enteros positivos entre 0 y 255. Por esto hay que destripar los datos a enviar. Cada coordenada será dividida en signo (1 = positivo, 2 = negativo), valor dividido entre 255 y resto del valor entre 255. La trama resultante sería la siguiente.

X	Y	Ref X	Ref Y	0
3 enteros	3 enteros	3 enteros	3 enteros	20 enteros

Figura 78: Trama de envío Bluetooth

Siendo cada coordenada al detalle:

Valor / 255	Valor % 255	Signo
1 entero	1 entero	1 entero

Figura 79: Coordenada al detalle

Para recibir vía Bluetooth en el PC se tiene en ejecución la aplicación en lenguaje M de MATLAB que crea una conexión con el puerto COM concreto con el que se ha establecido conexión con el NXT. Lee los paquetes que recibe iteración a iteración reconstruyendo los valores según el método contando anteriormente y acaba mostrando por pantalla los gráficos resultantes.

3.3.4.3. Ejecución

Antes de nada, en cualquier ventana del modelo se ha de acceder a "Simulation > Configuration Parameters... > Real-Time Workshop > Interface" y marcar la opción floating point number para activar el uso de números en coma flotante.

De la misma manera que la anterior aplicación, ésta ofrece en la ventana principal del modelo dos botones para compilar y descargar el ejecutable al NXT.

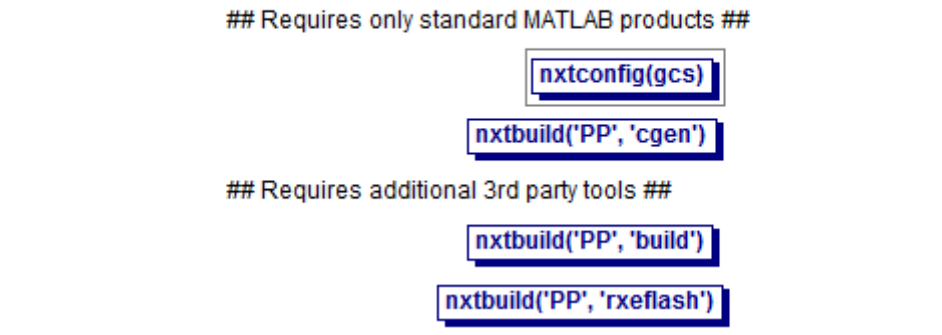


Figura 80: Ventana compilación y descarga PurePursuit

Primero se pulsa nxtbuild('PP', 'build') para generar el código y el ejecutable. Una vez hecho, se conecta el NXT encendido al PC mediante USB y se pulsa en nxtbuild('PP', 'rxeflash') para descargar el ejecutable. Antes de ejecutarlo hay que

seguir los pasos del apartado 3.3.2 para establecer la conexión vía Bluetooth y ejecutar el programa de recepción en el PC (RecibePP.m) que se mantendrá a la espera de los envíos del robot. Para empezar la ejecución de la aplicación hay que tener el NXT en el suelo libre y con espacio. Se muestra un ejemplo de ejecución de la trayectoria cuadrada (referencia marcada con cruces, trayectoria seguida con una línea recta).

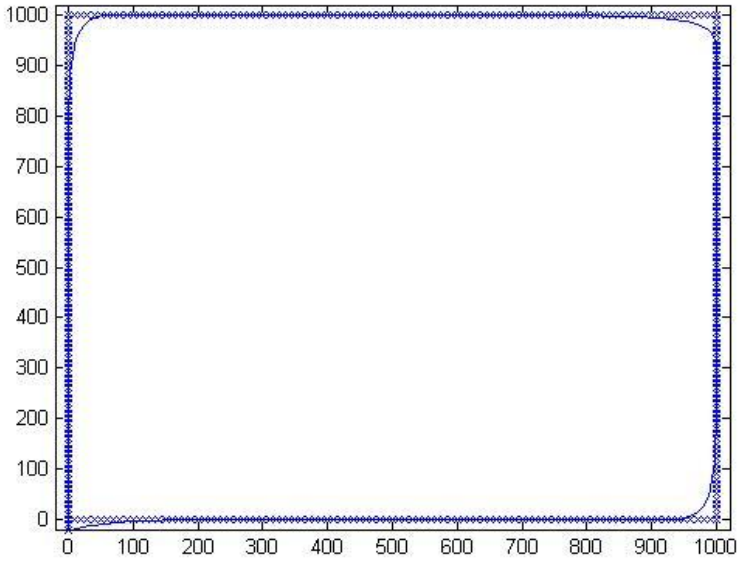


Figura 81: Resultado de la ejecución

3.3.4.4. Problemas encontrados y soluciones aplicadas

Durante el desarrollo de esta aplicación se encontró básicamente un problema relacionado con el tamaño de la ventana en la búsqueda del siguiente punto al que viajar, que impedía que el robot realizara la trayectoria deseada al completo. Si la trayectoria tiene puntos donde confluye varias veces (por ejemplo, el centro de una trayectoria tipo flor, por donde la misma pasa 8 veces distintas) y se utiliza un ventana demasiado grande (la mitad del número de puntos de la trayectoria en un principio) el robot entra en conflicto al llegar a esta zona, entrando en un bucle infinito. Para solucionarlo se optó por una ventana de tamaño 1/8 el número de puntos de la trayectoria, con el que se consiguió un resultado óptimo en todas las pruebas de todos los tipos de trayectoria.

3.4. Desarrollo en LeJOS

3.4.1. Estudio de la conexión vía USB

Como objetivo final del proyecto se propone alcanzar incrementalmente una solución que comunique un PC con el NXT vía USB para que el robot reciba los

movimientos a realizar gracias al mapeado del terreno y la evitación de obstáculos realizadas a partir del sensor Kinect de Xbox. Se escoge este tipo de conexión por delante de Bluetooth por la fiabilidad que da durante la comunicación.

Para el trabajo y desarrollo con la Kinect se necesita además un entorno Linux. Tras varios intentos resulta imposible trabajar en este sistema operativo con los lenguajes ECRobot y nxtOSEK, principalmente por la incompatibilidad con el sistema operativo del driver Fantom necesario para establecer una comunicación USB con este lenguaje. Por ello se decide continuar el trabajo en LeJOS, el cual en su API facilita mucho este tipo de comunicación, la que además es totalmente compatible con la librería libUSB de Linux

Como único prerrequisito para que funcione la comunicación hay que haber seguido los pasos del apartado 3.1.3.2. A partir de aquí la conexión y reconocimiento del NXT por parte del PC es son automáticos conectándolos con el cable USB proporcionado por LEGO.

Figura 82: Cable USB de Lego

3.4.2. Estudio del movimiento del robot NXT

En las aplicaciones que se van a realizar sobre el robot NXT se necesita un control total sobre el mismo, con una respuesta rápida ante cambios de cualquier tipo, por lo que es muy importante conseguir un movimiento preciso y manejable en tiempo real. LeJOS ofrece una extensa API con la que trabajar directamente sobre los motores.

3.4.2.1. Descripción de funciones de alto nivel

Algunas de estas funciones existentes en las librerías de LeJOS son: *void forward()* para mover el motor hacia adelante, *void backward()* para mover el motor hacia detrás y *void stop()* para parar el motor.

Además, existen otras clases como *DifferentialPilot* o *Navigator*, con sus correspondientes métodos para actuar sobre los motores, que permiten tratar al robot como un objeto y moverlo a puntos concretos o seguir trayectorias. Pero la precisión de estas clases no se acerca en nada a la deseada, por lo que se decide tratar los motores a bajo nivel para desarrollar un control más exhaustivo sobre el NXT.

3.4.2.2. Gestión de la trayectoria a bajo nivel

Para gestionar las trayectorias que seguirá el robot hay que actuar directamente sobre cada motor, aplicándole la acción de control deseada, así como conocer el valor de su encoder para controlar la posición y la velocidad. Para ello se ofrece en la clase *NXTMotor* funciones *void resetTachoCount()*, para poner a 0 el valor del encoder, *void getTachoCount()*, para obtener el valor del encoder en grados y *void setPower()*, para dotar al motor de la acción de control entre 0 y 100. Esta última función ha de ir seguida de las anteriormente comentadas *forward()* o *backward()* para indicarle el sentido.

3.4.3. Gestión del tiempo de muestreo del bucle de control

Para que el cálculo de la velocidad y posición del robot a partir de la lectura de los encoders de los motores y posterior cálculo de la acción de control a aplicarles sea exacta, consiguiendo así el movimiento contínuo deseado, se necesita un tiempo de muestreo que se cumpla con máxima precisión en el bucle de control. En el entorno ECRobot y nxtOSEK esto no suponía un problema, ya que al tratarse éste último de un sistema en tiempo real la frecuencia de ejecución de la tarea en cuestión, que ha sido indicada en el desarrollo de la aplicación, es controlada por él mismo. Pero no es el caso de LeJOS, por lo que se ha de desarrollar un método para conseguirlo. Para ello se utiliza la clase *Stopwatch* contenida en la librería de LeJOS. Ésta posee un método *reset()* que inicializará el reloj cada vez que el bucle vuelva a empezar y otro *elapsed()* que medirá el tiempo transcurrido al final del mismo. Si aún no se ha llegado al tiempo de muestreo utilizado se para el hilo principal de la aplicación mediante la llamada *Thread.sleep(milisegundos)* durante el tiempo que falta.

3.4.4. Seguimiento de una trayectoria fija

3.4.4.1. Descripción de la aplicación

Esta primera aplicación no es más que una traducción al entorno LeJOS de la desarrollada en ECRobot explicada en el apartado 3.3.4. Ésta generará y seguirá trayectorias de tipo cuadrado, circular o flor mediante el algoritmo de persecución pura. En este caso se suprime el envío de datos por Bluetooth por considerarlo irrelevante, ya que en este entorno se va a tratar más adelante las conexiones vía USB.

3.4.4.2. Metodología y desarrollo

Este proyecto está formado por dos clases: *PurePursuit* y *Trajectory*.

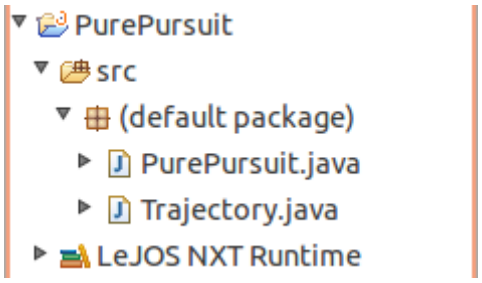


Figura 83: Sistema de archivos del proyecto

La clase *Trajectory* contiene los métodos para la generación de las trayectorias estáticas comentadas en el apartado anterior que puede seguir el robot en cada ejecución de la aplicación. Esta trayectoria se guarda en un array de *WayPoints*, un tipo de datos ofrecido la librería de LeJOS para tratar coordenadas.

La clase *PurePursuit* es la principal. Además de la función principal en la que se desarrolla el grueso de la aplicación, contiene la función *void displayTrajectory()*, mediante la que se muestra por pantalla al inicio de la ejecución un menú para seleccionar la trayectoria a resolver. Posteriormente se genera dicha trayectoria a través de la clase *Trajectory* y se resuelve el movimiento del robot siguiendo el mismo exacto desarrollo comentado en el apartado 3.3.4.2.

3.4.4.3. Ejecución

Para subir y ejecutar la aplicación al NXT hay que, estando encendido, conectarlo al PC vía USB y seleccionar la opción “Run As... > LeJOS NXT Program” que aparece al pulsar con el botón derecho sobre la carpeta del proyecto.

Hecho esto aparecerá en pantalla del robot el menú de selección de la trayectoria que se desea trazar.

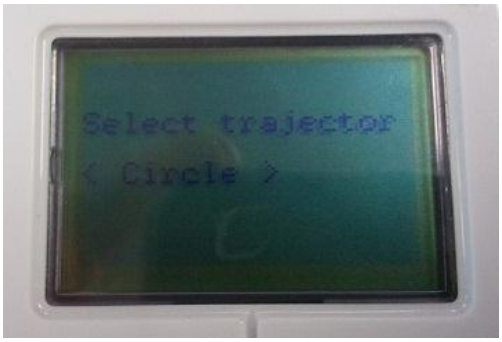


Figura 84: Pantalla de selección de trayectoria

Con los botones grises del NXT se cambian las opciones y, con el robot en el suelo, se selecciona la trayectoria con el botón central. Por pantalla se mostrará un mensaje que indica que el robot ya está en marcha.

3.4.4.4. Problemas encontrados y soluciones aplicadas

Durante el desarrollo y la ejecución de la aplicación en entorno LeJOS, siendo ésta una traducción directa de la versión definitiva de la desarrollada en ECRobot en el apartado 3.3.4, no se ha encontrado ningún problema que impida el funcionamiento normal de la misma. Pero hay que tener en cuenta los problemas y soluciones comentados en la sección 3.3.4.4.

3.4.5. Navegación a coordenadas enviadas desde el PC

3.4.5.1. Descripción de la aplicación

Esta segunda aplicación desarrollada en LeJOS ha sido realizada a partir de la anterior, puesto que usa el mismo algoritmo de navegación. Pero en este caso no se realiza un seguimiento de trayectorias fijas previamente generadas, sino que se navega a coordenadas concretas proporcionadas por el usuario desde un PC vía comunicación USB.

3.4.5.2. Metodología y desarrollo

Esta aplicación tiene una parte en PC en código C y otra en NXT, la cual está formada por dos clases: *PurePursuitUSB* y *Trajectory*.

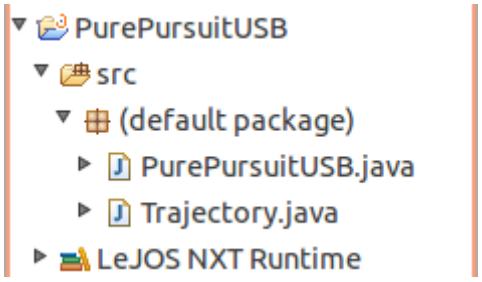


Figura 85: Sistema de archivos del proyecto

La clase *Trajectory* contiene los métodos para generar una trayectoria en línea recta entre dos puntos guardada en un array de *WayPoints* y para borrar el contenido del mismo.

La clase *PurePursuitUSB* esta vez es también la clase principal que desarrolla la parte importante de la aplicación. El desarrollo de la misma es secuencial en un mismo hilo de ejecución. En primer lugar crea la conexión USB mediante el método *waitForConnection()* de la clase *USBConnection*. Se pasan a leer a través de esta conexión las coordenadas a las que se quiere viajar. Estas coordenadas son enviadas desde la aplicación de PC que utiliza la librería *usb.h* para la comunicación, concretamente con la función *usb_bulk_write()*. La función ofrecida por esta librería

para el envío permite enviar un vector de bytes, por lo que para poder enviar coordenadas en coma flotante se tiene que descomponer cada una en su parte entera y decimal, y cada una de estas a su vez en los 4 bytes que está formado un entero. Además se añade en el envío 1 byte que indica si hay que finalizar el programa del robot, el cual es activado cuando finaliza el del PC. La trama utilizada en la comunicación sería la siguiente:

Parte entera X	Parte decimal X	Parte entera Y	Parte decimal Y	Variable de fin de aplicación
4 bytes	4 bytes	4 bytes	4 bytes	1 byte

Figura 86: Trama de envío Bluetooth

De vuelta a la aplicación del NXT, la trama se lee campo a campo desde la clase *DataInputStream* con el método *readInt()*, el cual lee del buffer de la conexión los siguientes 4 bytes ya reconstruidos como un entero. Y por último se lee el byte de fin de programa con el método *read()*. Una vez leídos los cuatro campos de la trama se reconstruyen debidamente las coordenadas. A partir de estas coordenadas destino y de la posición del robot que es conocida en todo momento, se puede generar la trayectoria en línea recta entre ambos puntos. De esto se encarga el método *generateTrajectory()* desarrollado en la clase principal que, antes de llamar al método correspondiente de la clase *Trajectory*, calcula el incremento de cada coordenada y el número de puntos que tendrá la trayectoria.

Llegada a este punto, la aplicación empieza el bucle de persecución pura que se desarrolla de la misma exacta manera que en la aplicación anterior, con unas pequeñas diferencias. La primera es que al inicio del bucle comprueba si hay datos en el buffer de la comunicación, regenerando la trayectoria si es el caso, momento en el cual el robot sigue trazando este nuevo camino. La segunda es que en este caso, y al tratarse la trayectoria de una linea recta que no presenta los conflictos que se encuentran en otro tipo de caminos, en el momento de buscar el siguiente punto al que dirigirse no utiliza una pequeña ventana en la que buscar, sino que busca sin problemas a lo largo de toda la trayectoria desde el punto en el que se encuentra. Y la última diferencia es que en esta versión no se sigue una trayectoria infinita en bucle, por lo que hay que detectar el punto final a partir de la distancia euclídea entre el mismo y el punto actual parando el robot y volviendo a esperar que el PC envíe un nuevo punto o el fin de programa.

3.4.5.3. Ejecución

Antes de empezar la ejecución de la aplicación hay que asegurarse de tener el robot encendido conectado vía USB al PC. Entonces se inicia la aplicación y se espera a que muestre por pantalla el mensaje que nos indica que ha establecido la conexión con el PC.

Figura 87: Mensaje de conexión

Es el momento de iniciar la aplicación del PC desde la consola (si produce una violación de segmento tendrá que ser ejecutada como "sudo").

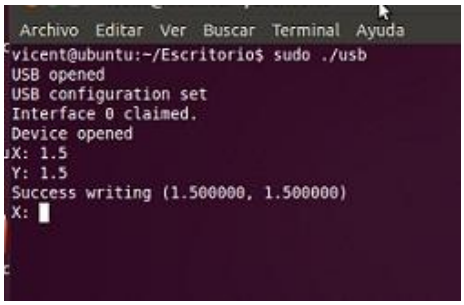


Figura 88: Pantalla de la aplicación de PC

Con el robot ya puesto sobre el terreno se puede empezar a enviar coordenadas (números enteros o formato 1234,5678), incluso mientras el robot está en movimiento, ya que la aplicación, como se ha comentado, tiene la capacidad de regenerar la trayectoria y reconducir al NXT. Para finalizar la aplicación basta con enviar una señal EOF (Ctrl + D en la consola). Durante la ejecución se mostrará un mensaje por pantalla, además de las coordenadas destino y las actuales, actualizándose en tiempo real.

Figura 89: Pantalla durante la ejecución

3.4.5.4. Problemas encontrados y soluciones aplicadas

No han sido pocos los problemas encontrados durante el desarrollo de la aplicación, pero todos ellos han sido satisfactoriamente resueltos.

Primeramente, puesto que la comunicación en la parte de PC está desarrollada con la librería *usb.h*, externa a LeJOS y a LEGO en general, hay que asegurarse de que el modo de conexión indicado en la llamada a *waitForConnection()* del robot es *RAW*, ya que de otra forma no podremos establecer conexión.

Por otro lado, la mayor parte de problemas han venido en el tema de la precisión en el movimiento. En primer lugar, originalmente las trayectorias se hacían punto a punto, lo cual suponía una pérdida de precisión conforme más largo era el camino. Por esto se decidió generar las trayectorias con un número de puntos variable (dependiente de la longitud de las mismas) y un incremento entre ellos razonable para el valor de lookahead dado. Hecho esto también se dobla este valor de lookahead para pretender que la trayectoria se sigue de la manera más recta posible. También se detectaron salidas del camino original en el arranque y frenado, así como cuando el robot tenía que maniobrar para hacer giros de más de 90º, por ser estos movimientos demasiado bruscos. Como solución se utiliza un factor de reducción de la acción de control, que se activa en los primeros y últimos puntos de la trayectoria, suavizando los movimientos comentados. Como último problema en cuanto a la precisión se observó que al llegar al último punto el robot rotaba ligeramente, produciendo esto un error incremental muy molesto. Para solucionarlo se añaden a la trayectoria 10 puntos extras, lo cual cubre en linea recta la distancia lookahead hasta llegar al punto objetivo y consigue que el robot navegue sin desviarse parando en el mismo con la rotación original.

Por último, se detectó lo que podría ser un problema para según qué usos de la aplicación y, sobre todo, para posibles futuras ampliaciones de la misma. Y es que al no haber comunicación bidireccional la aplicación de PC y, en consecuencia, el usuario a menos que tengo constante contacto visual con el robot, no sabe nunca si éste ha llegado a su destino, además de que no se puede asegurar la llegada debido a que el NXT lee constantemente posibles nuevos objetivos. Como solución se decide crear una nueva versión de la aplicación con comunicación bidireccional bloqueante. Ésta tendrá las mismas características que la aplicación original con una importante diferencia a cada lado de la comunicación. Por una lado, en la parte de PC, y tras enviar las coordenadas destino, se esperará mediante la llamada *usb_bulk_read()* a que el robot le avise de su llegada al destino. Esta llamada tiene un tiempo de espera, pasado el cual se avisará al usuario del error en la lectura. Por otro lado, la parte del NXT suprime la lectura de posibles nuevas coordenadas en cada iteración del bucle de persecución pura y, al llegar al destino, envía como aviso un byte mediante el método *write()* seguido de *flush()* de la clase *DataOutputStream*. Con esto se tienen dos aplicaciones, una con comunicación bloqueante y la otra no bloqueante, para que el usuario escoja según sus necesidades.

3.4.6. Navegación con visión artificial desde XBox Kinect

3.4.6.1. Descripción de la aplicación

Así se llega a este última apartado, el que se había establecido como objetivo final para el proyecto y con el que se pretende que el robot pueda navegar autónomamente,

sin que el usuario le de ninguna orden. Para conseguir la autonomía en esta aplicación entra en juego el sensor Kinect de Xbox.

Se plantea un mapa con un punto objetivo que el robot tiene que alcanzar de la manera más eficiente posible evitando los obstáculos presentes en el terreno. Desde la aplicación de PC se trata la información que llega de la cámara y se le envían las órdenes correspondientes a la parte del robot que navegará mediante el algoritmo de persecución pura desarrollado hasta el momento hasta llegar a su destino.

3.4.6.2. Montaje del robot

La configuración utilizada en el robot será la misma, diferencial, ya que la navegación se sigue tratando de igual manera. Pero se ha de encontrar la manera en que el NXT pueda llevar el sensor Kinect como una parte más suya, como sus ojos. Así que se decide construir una estructura sobre el robot que permite llevarlo de la manera deseada. También aparece otro problema, y es que para observar el terreno a recorrer el sensor ha de tener la capacidad de rotar 360º sin desviar al robot de su camino. Para ello se instala un tercer motor dentro de la estructura con la única pero muy importante tarea de rotar la Kinect. El aspecto final del robot es el siguiente.

Figura 90: Aspecto del robot con el sensor Kinect instalado

3.4.6.3. Metodología y desarrollo

Como se intuye del apartado anterior, esta aplicación también tiene una parte en PC y otra en NXT.

La parte de PC está implementada en lenguaje C++ y utiliza paquetes y librerías de ROS para la comunicación y el manejo de la información recibida desde la Kinect. En primer lugar y como preámbulo cabe destacar que se plantea el mapa como una matriz de dos dimensiones de la que deduciremos tanto qué parte del terreno tiene obstáculos que evitar como las posiciones actual y objetivo del robot. Cada casilla de la matriz

representará una porción del terreno real de 40cm x 40cm entre las que el robot irá desplazándose.

En esta parte se llevan a cabo 4 tareas: el mapeado del terreno y el control de la orientación del sensor mediante la técnica SLAM (a partir de la información generada por la pila de ROS *rgbdslam*), la lectura y filtrado de la información recibida de la Kinect para la detección de obstáculos y rellenado del mapa bidimensional, la decisión de la orden a enviar a partir de este mapa y la posición actual del robot (actualizada a partir de la decisión tomada) y la comunicación con el robot mediante la librería *usb.h* para que éste realice las acciones oportunas.

En la comunicación se envían las órdenes codificadas en un número entero y acompañadas de las coordenadas que necesite el robot para resolver la acción asociada a cada una de ellas. Estas coordenadas son descompuestas del mismo modo que el explicado en el apartado 3.4.5.2 de la anterior aplicación. Para facilitar el trabajo y seguir un criterio único las tramas para todas las órdenes tendrán un tamaño de 25 bytes a pesar de que no siempre todos sean necesarios. La trama para las órdenes de desplazamiento, posición, rotación del robot, frenado del motor del sensor y fin de programa sería la siguiente:

Orden	Parte entera X	Parte decimal X	Parte entera Y	Parte decimal Y	Parte entera theta	Parte decimal theta
1 byte	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes

Figura 91: Trama enviada por USB

Evidentemente, no todos los campos son aprovechados por todas las órdenes. Los campos de las coordenadas X e Y son aprovechados por desplazamiento y posición y el de theta por posición y rotación del robot. Las órdenes de frenado del motor del sensor y de fin de programa ignoran todos los campos. Como la rotación del sensor está medida y controlada por la parte del PC, se necesita enviar tanto la theta en la que se encuentra como la destino. Así pues, la trama enviada para esta orden resulta como sigue:

Orden	Vacío	Parte entera theta origen	Parte decimal theta origen	Parte entera theta destino	Parte decimal theta destino
1 byte	8 bytes	4 bytes	4 bytes	4 bytes	4 bytes

Figura 92: Trama enviada por USB

La parte del NXT consta una vez más de dos clases: *PPUSBKinect* y *Trajectory*.

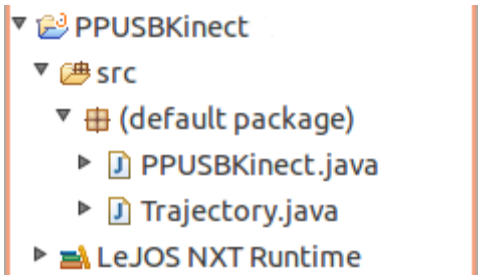


Figura 93: Sistema de archivos de PPUSBKinect

La clase *Trajectory* es la misma que en el proyecto anterior, que genera trayectorias en línea recta entre dos puntos. La clase *PPUSBKinect* se encarga de recibir estas tramas, reconstruir las coordenadas que se le envían y transformar las órdenes en las acciones correspondientes, según la orden enviada, que le llevarán a su objetivo. La función principal consta de un bucle, al inicio del cual se lee la posible orden que haya enviado la parte del PC, momento en el que se descodifica y se actúa debidamente. Se pasa a detallar como actúa el robot ante cada uno de los 6 tipos de orden:

-Posición: Esta orden solo se envía en el inicio de la ejecución. Indica al robot la posición inicial en la que se encuentra para que no haya problemas con los posteriores desplazamientos. Esto se hace porque puede existir el caso en que el NXT no empieza en la casilla (0,0) del mapa. Además, las coordenadas se presentan siempre en valores absolutos, por lo que el robot siempre tiene que estar debidamente localizado.

-Rotación del sensor: Esta acción puede ocurrir al inicio del bucle, cuando el robot está detenido, para posicionar el sensor hacia donde tenga que detectar posibles obstáculos para encontrar el camino libre. Se recibe la rotación actual y la objetivo para poder iniciar la rotación en el sentido correcto con el método *rotate()* de la clase *NXTRegulatedMotor*. En este momento el control es devuelto a la aplicación del PC, que es la que mide constantemente la orientación de la Kinect gracias al método SLAM.

-Frenado del sensor: Esta orden sólo se enviará en el momento en que la parte del PC detecte que sensor ha alcanzado la rotación deseada. Entonces el motor se detendrá con el método *stop()* y se esperará la llegada de una nueva orden.

-Desplazamiento: Esta acción es muy similar a la anterior aplicación del apartado 3.4.5. El robot recibe las coordenadas objetivo y construye de igual manera la trayectoria en línea recta entre éste y el punto en que se encuentra el robot con el método *generateTrajectory()*. Entonces se entra en el bucle de persecución pura que sigue el mismo desarrollo que el explicado en el apartado 3.4.5.2, con la salvedad de que, como en la versión no bloqueante de la anterior aplicación, no lee una posible nueva coordenada a la que viajar, sino que siempre se asegura la llegada al punto objetivo. La parte del PC espera a recibir la confirmación de la llegada.

-Rotación del robot: Antes de desplazarse, y si el robot necesita hacerlo hacia un lateral, se recibe esta orden acompañada del valor de rotación al que se quiere llegar. Para realizar este movimiento se ha creado el método *rotation()*, en el que se ha implementado un sencillo control proporcional de la velocidad de rotación, que finaliza cuando es alcanzada la rotación deseada.

-Fin de programa: Esta es la última orden que recibiría el robot en una ejecución y que indica que se ha llegado al objetivo final o que se ha podido producir algún tipo de error en la parte del PC. Esta orden finalizará la aplicación del robot.

3.4.6.4. Ejecución

Antes de empezar la ejecución se ha de construir el terreno con sus obstáculos y establecer el punto inicial y destino que tienen que ser indicados en la aplicación de PC.

Figura 94: Aspecto inicial del entorno

Con todo colocado en su sitio se conectan el sensor y el robot al PC mediante USB. Entonces se ejecuta el paquete rgbdslam para que esté preparado para el desarrollo del algoritmo de SLAM. Para ello se ejecuta con la llamada "*roslaunch rgbdslam kinect+rgbdslam.launch*" y, cuando se visualiza en la pantalla del PC la imagen que transmite la cámara, se arranca el SLAM pulsando la tecla espacio.

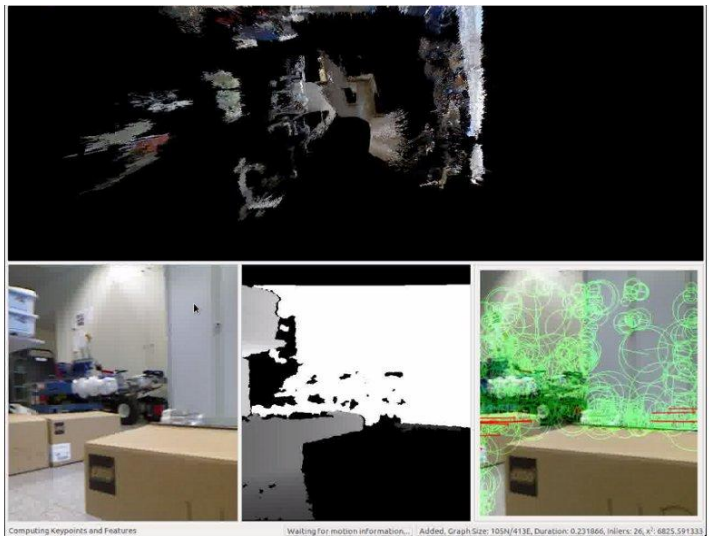


Figura 95: Ventana del rgbdslam con el algoritmo en marcha

En este momento se arranca la aplicación del robot que se pone a la espera de recibir órdenes por parte del PC.

Cuando el robot ya ha establecido conexión y está a la espera es el momento de ejecutar la aplicación del PC desde Eclipse. Entonces el robot ya empezará a moverse autónomamente actuando ante las órdenes enviadas por la aplicación del PC, la cual resuelve el camino que ha de seguir hasta el punto objetivo gracias a la información recibida por el sensor Kinect.

3.4.6.5. Problemas encontrados y soluciones aplicadas

Esta aplicación ha planteado problemas durante su desarrollo, causados principalmente por el peso que ejercen la Kinect y la estructura que lo sostiene sobre el robot.

El primer problema encontrado fue el poco soporte que ofrecía la rueda trasera diferencial con la consecuente deformación del robot por su parte central y excesiva presión sobre sus dos ruedas motoras que dificultan el movimiento. Para solucionarlo se sustituye la pequeña rueda por una pieza especial omnidireccional que aporta mayor fuerza y robustez a la estructura, con lo que también se consigue una mejor distribución del peso.

Figura 96: Pieza trasera omnidireccional

A pesar de este cambio el peso sigue siendo muy grande y esto es algo irreparable a nivel físico. Los movimientos resultan muy lentos a causa de esto, por lo que se decide reducir el factor de reducción de acción de control utilizado para mejorar la precisión, con lo que se consigue mayor fuerza en los motores para arrastrar la estructura sin perder precisión.

En un principio tanto los movimientos frontales como los laterales se realizaban siguiendo directamente la trayectoria creada. Pero debido también al peso de la estructura y a la distancia no muy grande entre casillas los movimientos laterales perdían mucha precisión. Para solucionar esto se decide tratarlos de la manera comentada en el apartado 3.4.6.3, primero rotando el robot mediante un control proporcional y posteriormente realizando el trayecto frontal resultante.

Por última cabe señalar que durante la ejecución y en caso de que se realice un trayecto largo se pueden encontrar problemas con rgbdslam. Al acumular mucha información de la recibida a través de la Kinect se producen problemas de memoria que detienen el programa, haciendo imposible el funcionamiento perfecto de la aplicación. Este es un problema de diseño del programa rgbdslam imposible de solucionar por parte del usuario. Se propone para futuros trabajos la solución de este aspecto.

4.- Conclusiones y futuros proyectos

Llegado el final del proyecto se concluye que se han alcanzado satisfactoriamente los objetivos marcados al inicio. Se ha alcanzado una gran familiaridad con los robots NXT y sus firmwares nxtOSEK, con la librería de bloques ECRobot para Simulink, y LeJOS, con su propia librería para Java. Se han ampliado los conocimientos de los entornos de programación Simulink y Eclipse y de sus herramientas de depuración. Se ha estudiado la gestión de la comunicación Bluetooth y USB que se crea del NXT al PC y viceversa, profundizando en la utilización de las correspondientes librerías y la compatibilidad entre ellas, sus limitaciones, los tipos de datos que permiten enviar y cómo enviar a través de ellos otros tipos de datos de mayor interés. Se han ampliado conocimientos sobre la programación en Java (programación orientada a objetos, creación de clases y sus métodos, gestión del tiempo de ejecución...), en C y en Simulink (utilización de bloques avanzados, gestión de la prioridad y el orden de ejecución...). Se ha trabajado en la gestión y generación de trayectorias, tanto estática como dinámicamente, de objetos móviles y analizado varias estrategias de seguimiento, haciendo gran hincapié en el algoritmo de persecución pura.

En resumen, la tarea de investigación y estudio de los diferentes campos propuestos se ha visto recompensada y reflejada en aplicaciones que cubren con altas expectativas lo estudiado. Las aplicaciones implementadas en ECRobot utilizan comunicación Bluetooth y generan y resuelven trayectorias estáticas, mientras que las desarrolladas para LeJOS utilizan comunicación USB para la generación de trayectorias de manera dinámica. Los problemas surgidos durante el desarrollo del proyecto han motivado a un estudio más exhaustivo de los temas de interés para encontrar las soluciones aplicadas y han aumentado el interés en el ámbito de la robótica móvil para realizar nuevos proyectos en un futuro.

Como ampliación del proyecto realizado se propone el desarrollo de una detección en tiempo real, lo que supondría trabajar con varios hilos de ejecución a la vez. Con esto se conseguiría resolver no sólo un mapa estático previamente creado, sino uno dinámico con la detección de objetos móviles.

También se podría trabajar con varios robots a la vez conectados vía Bluetooth entre ellos. El primero de ellos, el maestro, mapearía el terreno y encontraría el camino más rápido para llegar al objetivo. Esta información la enviaría a los demás robots que podrían llegar directamente con esta información, sin necesidad de sensores ni observación del terreno.

Por último, para futuras investigaciones se propone, por ejemplo, trabajar con otros sensores para realizar el mapeo y detección de obstáculos cuyo uso supusiera menor carga computacional y tuviera mayor rango de detección, como por ejemplo un sensor de infrarrojos. Esto también podría ir unido a la instalación de un dispositivo GPS para la localización del robot. Todo esto permitiría trabajar en espacios mayores al aire libre con mayor precisión.

5.- Bibliografía

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

[2] MATLAB Central http://www.mathworks.com/matlabcentral

[3] nxtOSEK & ECRobot http://lejos-osek.sourceforge.net/

[4] NXT Ballbot: Blog de Tomás Arribas http://nxtballbot.blogspot.com.es/

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

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

[7] LEGO Mindstorms http://mindstorms.lego.com

[8] NXT Wiki http://nxt-wiki.rjmcnamara.com

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

[10] Asociación de robótica educativa Complubot http://complubot.educa.madrid.org/pruebas/lego_nxt_version_educativa/lego_nxt_version_educativa_index.php

[11] Blog de Electribricks http://blog.electricbricks.com

[12] Aula robótica: Blog dedicado a la difusión de la robótica educativa http://aularobotica.blogspot.com.es

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

[14] Herramienta Interactiva para Robótica Móvil. J.L. Guzmán, M. Berenguel, F. Rodríguez, S. Dormido http://www.cinhtia.com.mx/uady/sesionesRM/Articulos/JA2003a-HerramientaInteractivaRoboticaMovil.pdf

[15] Implementation of the Pure Pursuit Path Tracking Algorithm. R. Craig Coulter http://www.ri.cmu.edu/pub_files/pub3/coulter_r_craig_1992_1/coulter_r_craig_1992_1.pdf

[16] libUSB http://www.libusb.org/

[17] ROS Wiki http://www.ros.org/wiki/