**UNIVERSIDAD PONTIFICIA COMILLAS**

Escuela Técnica Superior de Ingeniería (ICAI)

Ingeniero Técnico Informático



Estudio Teórico Práctico de la Robótica al Servicio de la Sociedad

Titulación: Ingeniería Técnica Informática en Gestión y Sistemas.

Director: José Miguel Ordax Cassá.

Alumno: Miguel Alfonso Ruiz Nogués.

Índice de la memoria

Parte I Memoria 2

Capítulo 1 Introducción 3

1.1 Motivación del proyecto 4

1.2 Objetivos 4

1.3 Metodología / Solución desarrollada 6

1.4 Recursos 9

1.4.1 Técnicas 9

1.4.1.1 Beavior Programming 9

1.4.1.1.1 Behavior Programming en leJOS 11

1.4.2 Hardware 13

1.4.2.1 Lego NXT 13

1.4.2.2 Notebook Asus 1005HA 15

1.4.2.3 Bluetooth Headphones 16

1.4.3 Software 16

1.4.3.1 Ubuntu 10.04 16

1.4.3.2 Java: 17

1.4.3.3 leJOS: 19

1.4.3.4 Eclipse: 20

1.4.3.5 SubVersiones 22

1.4.3.6 Text To Speech 22

1.4.3.7 Sphinx 4 22

1.4.3.8 PHP 22

Parte II Robótica 23

Capítulo 1 Introducción a la Robótica 24

1.1 Introducción 24

1.2 Breve Historia de la Robótica 24

1.3 Clasificación de Robots 28

1.3.1 Clasificación según su Generación: 29

1.3.1.1 Primera Generación: Robots Play-Back. 29

1.3.1.2 Segunda Generación: Robots Controlados por Sensores. 32

1.3.1.3 Tercera Generación: Robots Controlados por Visión. 36

1.3.1.4 Cuarta Generación: Robots Controlados Adaptablemente. 40

1.3.1.5 Quinta Generación: Robots con Inteligencia Artificial. 44

Parte III El Problema 45

Capítulo 1 Introducción 46

Capítulo 2 Solución 47

2.1 La plataforma del Robot 47

2.1.1 Partes del Robot: 47

2.1.1.1 Ladrillo Inteligente: 47

2.1.1.2 Ultrasonic Sensor: 49

2.1.1.3 Color Sensor: 51

2.1.1.4 Motores: 52

2.2 La plataforma en el centro comercial 53

2.3 La plataforma Software 54

Capítulo 3 Valoración Final 55

3.1 Conclusiones 55

3.2 Aspectos Negativos 55

3.3 Trabajos Futuros 55

Bibliografía 56

Parte IV Anexos 57

Anexo A: Lista de Problemas 58

1.1 Problema 1: Instalar Java en Ubuntu 58

1.2 Problema 2: SAM-BA mode 58

1.3 Problema 3: Conexión USB 59

***Índice de figuras***

[Ilustración 1: Fases del Proyecto 7](#_Toc291495760)

[Ilustración2: Planificación Temporal de las Actividades del Proyecto 9](#_Toc291495761)

[Ilustración 3: LEGO RXC. 11](#_Toc291495762)

***Índice de tablas***

[Tabla 1: Resumen del Hardware del NXT 49](#_Toc291605884)

[Tabla 2: Métodos para la clase Motor 55](#_Toc291605885)

1. Memoria

# Introducción

La Organización Nacional de Ciegos de España ( ONCE ) tiene un total de 69.934 de afiliados a 30 de junio de 2010. El 80% de los afiliados padecen deficiencia visual, es decir, son personas que mantienen un resto visual que, con las ayudas ópticas y electrónicas correspondientes, les resulta funcional o útil para su vida diaria. El 20% restante de los afiliados, son personas que no ven nada en absoluto o solo perciben luz.

“El perro guía supone, por la seguridad y autonomía que proporciona, una ayuda inestimable en el desplazamiento de las personas con ceguera que optan por él como auxiliar de movilidad.” (ONCE). Según los datos ofrecidos públicamente por la ONCE un mínimo de 13.986 personas en España están necesitadas de un perro guía, y tan solo 1000 de ellos disponen de uno.

Hoy en día, en España, para adquirir un perro guía es necesario disponer de 10.000 euros aproximadamente, o apuntarse a una lista de entrega de la ONCE, con un tiempo medio de espera entre 5 y 7 años.

Este proyecto, pretende proponer una solución complementaria a los perros guía, que ayude a la movilidad de personas con discapacidad visual en entornos cerrados como puede ser un supermercado, un cine, un aeropuerto, etc.

Según las definiciones aportadas anteriormente se puede afirmar que una máquina programada capaz de guiar a una persona con ceguera en un entorno cerrado entra en la definición de robot.

## Motivación del proyecto

(A la vista de los trabajos ya hechos se puede saber cuál es la motivación del proyecto) (Aquí hay que vender el proyecto y responder a la pregunta ¿Por qué alguien puede querer comprar este proyecto?) (Si fueseis comerciales, aquí es donde hay que convencer al público)

## Objetivos

El proyecto persigue dos objetivos paralelos claramente diferenciados:

1. **Fomentar la robótica al servicio de la sociedad:**

Entre los logros de la robótica se encuentra, la conquista de otros planeas, la sustitución de brazos y piernas, desactivación de bombas. Esto demuestra que la robótica brinda un sinfín de posibilidades.

Según la IFR ( International Federation of Robotics) se prevé un incremento en las ventas de robots domésticos del 16,4 % entre 2010 y 2013.

El incremento de las ventas demuestra un aumento del interés en la robótica de carácter doméstico, pero son muy pocas las universidades y centros escolares que tienen un departamento de robótica.

La forma de colaborar con la expansión de la robótica es mediante la publicación de vídeos y contenido en internet, aumentando así los resultados en Google. Otra forma de demostrar que la robótica está al alcance de todos en términos económicos, es utilizando el pack de robótica para estudiantes desarrollado por LEGO Group y el MIT.

1. **Crear un prototipo de robot para guiar a personas ciegas a través de un supermercado.**

Las personas con déficit visual son tan solo la punta del iceberg de las personas que tiene problemas de movilidad en diferentes entornos. El robot planteado, está enfocado en personas con déficit visual, pero pretende poder ser extrapolado a todas las personas con problemas de movilidad.

## Metodología / Solución desarrollada

El proyecto sigue un modelo de ingeniería del software denominado Modelo en Cascada. Este modelo está marcado por que la finalización de una fase marca el comienzo de la siguiente, tomando como datos de entrada los datos de la fase anterior.

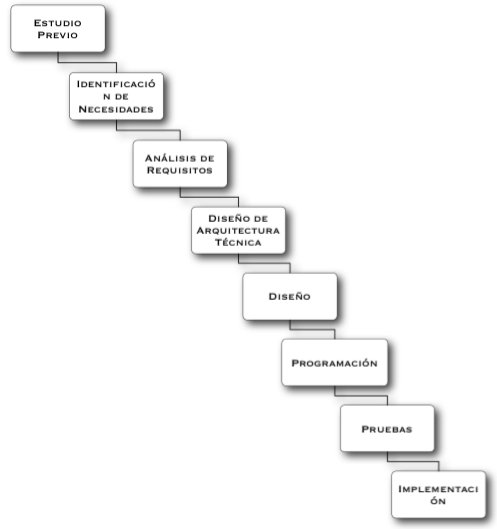


Ilustración 1: Fases del Proyecto

1. Estudios Previos:

Para adquirir las bases necesaria para la realización de este proyecto, se ha decidido realizar un estudio teórico de la robótica. Con este estudio se pretenden adquirir las bases necesarias para desarrollar al completo un robot.

1. Identificación de Necesidades:

En esta fase se pretende comprender el entorno global del problema que queremos soluciona. A continuación se elaborará el Documento de Conceptos del Sistema, donde se recogen los objetivos esperados.

1. Análisis de Requisitos:

El objetivo de esta fase es alcanzar un conocimiento suficiente del robot, definiendo necesidades, problemas y requisitos. Se estudiará a “maggie”, un robot desarrollado por la Universidad Carlos III de Madrid que fue desarrollado con el propósito de ayudar a personas ciegas.

1. Diseño de Arquitectura Técnica:

Tras el estudio de las tecnología existente s para el desarrollo de robots se pretende escoger la más adecuada para la implantación.

1. Diseño:

Consiste en obtener los últimos detalles necesarios para la programación del robot.

1. Programación:

En esta fase se generará el código necesario para que el robot tenga las funciones definidas en las fases anteriores.

1. Pruebas:

Durante la programación se realizaran pruebas unitarias, pero una vez que el código esté terminado, se probará que el robot tiene la funcionalidad requerida.

1. Implementación:

Se recreara un supermercado en miniatura para la demostración.

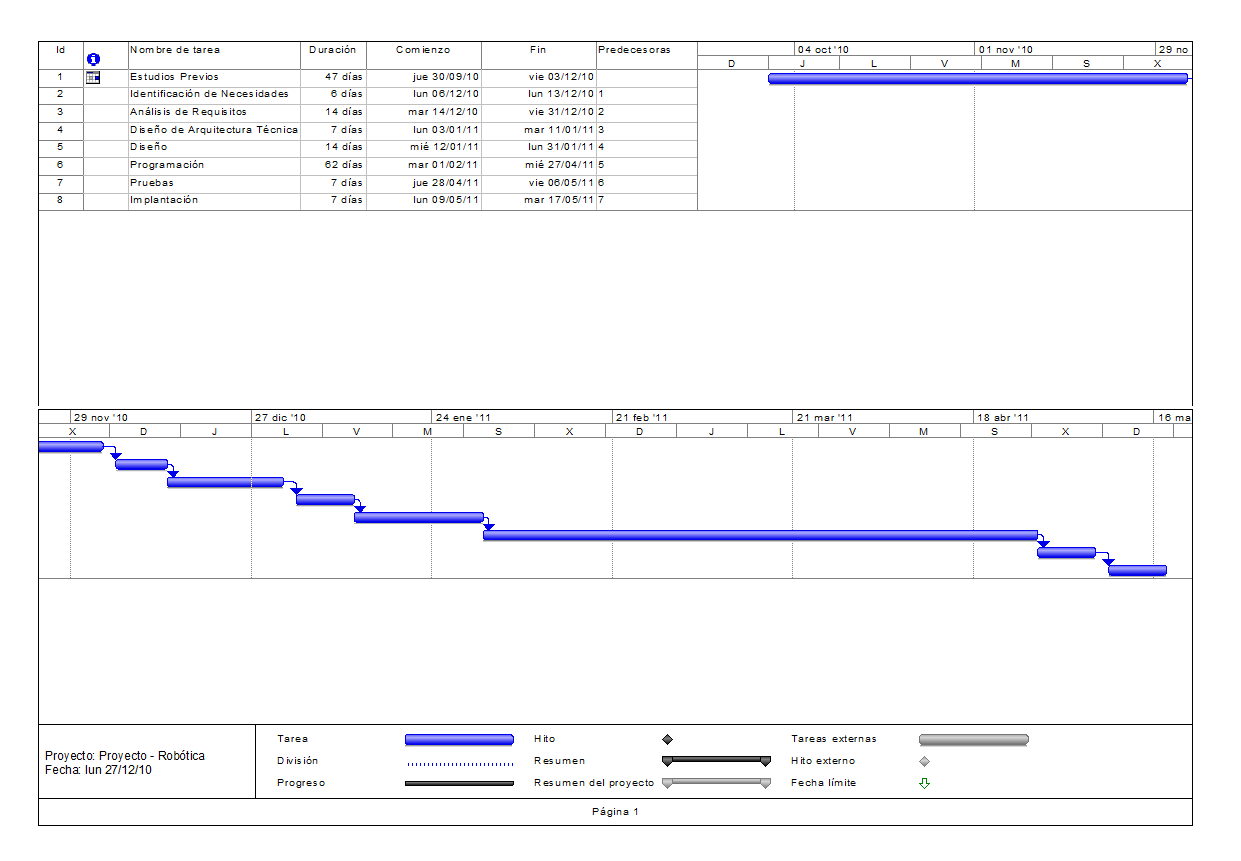


Ilustración2: Planificación Temporal de las Actividades del Proyecto

## Recursos

### Técnicas

#### Beavior Programming

Cuando se empieza a programar un robot por primera vez, normalmente la mentalidad es la misma que la de la programación estructura. Pero aplicar este tipo de programación en robots, acaba convirtiendo el código en un laberinto difícil de modificar. La forma de programación *Beahavior Programming* requiere un poco más de planificación antes de ponerse a codificar, pero el resultado es notablemente más agradecido en cuanto a comprensión, y modificación se refiere.

El concepto de *Beahavior Programming* es proporcionar al robot un conjunto de comportamientos independientes a su objetivo, pero que en conjunto permitan al robot llevar a cabo sus tareas.

es una forma de programación en paralelo donde los *behaviors* (comportamientos) tienen sus propias metas, y compiten por el control del robot. Esto proporciona una plataforma de oportunidades que en última instancia llevan a cabo la tarea principal.

En la programación estructurada, los programas realizan una tarea específica. La programación de robots es más compleja, puesto que requiere añadir cada nueva situación a la toma de decisiones. Esto hace que la programación se vuelva complicada y propensa a errores.

La *Beahavior Programming* emplea un sistema jerárquico de conductas escrito específicamente para realizar una acción basada en un conjunto de factores desencadenantes. Lo bueno de este sistema, es que la complejidad reside en añadir comportamientos sin necesidad de cambiar los existentes. Esta forma de programación considera cada *behavior* de una forma independiente para determinar el control del robot. A simple vista parece que este tipo de programación trae el caos, pero en última instancia resulta ser muy flexible y robusta. El fin último se consigue a consta de conseguir fines más pequeños, es decir, el fin último se cosnigue como resultado de que los *behavoir* consigan sus fines de forma independiente. Si se construye correctamente, el robot parece pensar y tomar decisiones sobre su entorno para completar la tarea.



Ilustración 3: Behavior Programming

##### Behavior Programming en leJOS

A la hora de realizar una programación basada en comportamientos en lejos, disponemos de una interfaz y una clase. La interfaz **Behavior** es muy genérica, puesto que los distintos *behaviors* que se pueden programar son muy variados.

Todas las clases e interfaces para la programación de comportamientos se encuentra en el paquete lejos.subsumption.

**lejos.robotics.subsumption.Behavior:**

Según el api de leJOS el interfaz de Behavior representa un objeto que contiene un comportamiento específico que pertenece a un robot. Cada comportamiento debe definir tres cosas:

1. Las circunstancias por las cuales éste comportamiento toma el control del robot.

Método: void action ()

1. La acción que realiza el comportamiento cuando tiene el control del robot.

Método: void supress ()

1. Las acciones que realiza el comportamiento cuando no tiene el control del robot.

Método: void takeControl ()

**lejos.robotics.subsumption.Arbitrator:**

Cuando un objeto se crea una instancia Arbitrator, se construye pasándole como parámetro una matriz de objetos Behavior. Una vez creado el objeto, se llama al método strat () que comienza a “arbitrar”, es decir, decidir qué Behavior deben estar activas. El objeto Arbitrator llama al método TakeControl () en cada objeto Behavior, empezando por el objeto con el número índice más alto en la matriz. El objeto Arbitrator realiza pasadas sobre los objetos Behavior, hasta encontrar alguno que quiera tomar el control, es entonces cuando se ejecuta el método action () del objeto una única vez. Si dos objetos Behavior luchan por el control del robot, solo el de mayor índice lo obtendrá.

En resumen, ésta clase tiene tres responsabilidades:

1. Determinar el comportamiento de mayor prioridad intentando tomar el control del robot.
2. Sustituir el comportamiento activo si su prioridad es inferior al de otro solicitando el control.

Cabe destacar que solo se llamara al método supress () de aquel comportamiento cuyo método action () este en ejecución, terminado la ejecución de éste inmediatamente.

### Hardware

#### Lego NXT

Lego Mindstorms NXT es el resultado de un acuerdo entre el LEGO y el Instituto Tecnológico de Massachusetts (MIT) en 1986. LEGO acordó financiar las investigaciones del departamento de epistemología y del departamento de aprendizaje, a cambio, LEGO obtendría nuevas ideas para sus productos.

De las investigaciones realizadas en el MIT nació el *Programmable Brick* que LEGO transformó en lo que sería la primera generación de Mindstorms, el RCX (Ilustración 3). Esta versión fue lanzada en septiembre de 1998 con un precio aproximado de 200 dólares. Los resultados superaron todas las expectativas; vendieron más 80.000 unidades en tres meses y se hicieron hueco en el mercado internacional de robótica.



Ilustración : LEGO RXC.

Esta primera generación de Mindstroms tenía un ladrillo menos inteligente que el NXT (CPU Hitachi H8/3292 a 16MHz, 16 KB de ROM y 32 KB de RAM), tres entradas de sensores y tres salidas para los motores, y un puerto de infrarrojos. La apariencia del RCX es mucho más cercana a los tradicionales bloques de Lego, de hecho, tanto los sensores como el bloque programable conservan la forma de los ladrillos característica de LEGO.

En 2004, debido a los malos resultados de LEGO en el año anterior, con pérdidas de unos 188 millones de euros, se difundió el rumor de que iban a abandonar la línea Mindstorms y regresar a su mercado tradicional. Sin embargo, en enero de 2006, LEGO anunció el NXT, que comenzó a venderse en junio de ese año.

La línea de productos Mindstorms no siempre ha sido bien acogida por los fans de LEGO, que lo acusó de perder el rumbo, o por los miembros de este grupo de la epistemología del MIT, ya que algunas decisiones comerciales contradicen los resultados de las investigaciones conjuntas de LEGO y el MIT. En cualquier caso, hoy LEGO Mindstorms tiene un hueco considerable entre los entusiastas de la robótica de todas las edades y en todo el mundo.

Lego NXT ha sigo escogido para este proyecto por la relación posibilidades-precio que ofrece. Es la plataforma ideal para realizar prototipos de robots.

Las denventajas que se han encontrado a la hora de hacer el prototipo son 2:

1. La plataforma creada por lego es debil, y tiene problemas para sostener el peso del robot.
2. El brick no tiene la potencia de procesamiento necesaria para Sphinx 4, por lo que se procedió a instalar un netbook que por sus caracterísitcas de procesamiento se convirtió en el procesador principal.

#### Notebook Asus EEE 1005HA

El netbook es el verdadero centro de procesamiento del robot, encargándose de la comunicación por voz, y ejecutando el código.

El robot es la combinación entre el notebook y el NXT brick. El notebook actúa como centro de procesamiento de señal que el robot necesita de procesamiento empleando Sphinx 4.

El programa se ejecuta en el ordenador y se comunica a con el brick a través de conexión USB. Cuando el robot llega a una determinada posición, se conecta a internet a través del notebook y realiza una consulta a la base de datos y a continuación transfiere los datos a la persona a través del headphone.

#### Bluetooth Headphones

Los headphones son usados por la persona con déficit visual para comunicarse con el robot. Éstos se comunican con el notebook mediante Bluetooth.

La comunicación Bluetooth y USB en lejos, se realiza con el paquete lejos.pc.comm.

### Software

#### Ubuntu 10.04

Ubuntu es una distribución de Linux de código abierto. Su elección para éste proyecto se debe a dos razones; La primera es para cumplir con el objetivo de acercar la robótica a colegios y universidades, demostrando que no es necesario un gran esfuerzo económico. La segunda es por utilizar un sistema operativo estudiado en la universidad, y que es la base de algunas empresas como Google, IBM, HP o Carrefour entre otras.

Para instalar Ubuntu se ha utilizado Unetbootin (<http://unetbootin.sourceforge.net/>), éste programa permite instalar Ubuntu desde USB de forma gratuita.

#### Java:

En 1999, un equipo de ingenieros de Sun Microsystems conocido como el “Green Team” tuvo la idea de unificar los dispositivos digitales para el consumo y los ordenadores. El desarrollo de la idea terminó en la creación de un lenguaje para la programación de microsistemas.

En 1995 el equipo liderado por James Gosling presentó Java con una nueva orientación hacia internet.

Java se ha escogido para este proyecto, entre otros lenguaje de programación como C/C++ y ensamblador, por ser el lenguaje de programación orientado a objetos más utilizado.

Para instalar java en Ubuntu es necesario escribir tres líneas en el terminal:

1.sudo add-apt-repository "deb http://archive.canonical.com/

lucid partner"

2.sudo aptitude update

3.sudo aptitude install sun-java6-jdk

La primera línea añade el repositorio desde donde vamos a descargar java al fichero source.list, que es donde se encuentra la lista de repositorios, a continuación actualizamos y por último se instala java 6.

#### leJOS:

Lejos es un sistema operativo pequeño basado en Java que ha sido adaptado a Lego Mindstorms. Éste consiste en reemplazar el software de lego NXT por el de una JVM ( Java Virtual Machine). leJOS también es de código abierto.

La librería clases.jar contiene el API de leJOS, que a su vez se encuentra en la página oficial de leJOS [6].

leJOS se escogió para este proyecto entre RobotC, NXC, NBC, NQC y RoboLAB, por programarse en Java.

Para instalar leJOS en Ubuntu se realizan los siguientes pasos:

1. Ubicación del software leJOS en ruta.
2. Compilación del proyecto.
3. Habilitar permisos USB.
4. Configuración de variables.

Para más detalle consultar la guía de instalación de Juan Antonio Breña Moral [8].

#### Eclipse:

El Proyecto Eclipse fue creado originalmente por IBM en noviembre de 2001 y apoyado por un consorcio de proveedores de software. Hoy en día, la comunidad Eclipse se compone de individuos y organizaciones de una sección transversal de la industria del software.

En general, la Fundación Eclipse ofrece cuatro servicios a la comunidad:

1) Infraestructura de TI,

2) gestión de la PI,

3) Proceso de Desarrollo, y

4) Desarrollo de Ecosistemas.

En nuestro proyecto, Eclipse es el entorno de programación escogido por su compatibilidad con leJOS, por ser de código abierto y por ser el de mayor contenido en internet en cuanto a relación con leJOS se refiere.

Para instalar eclipse en Ubuntu se puede hacer de diferentes maneras, la más sencilla es escribir sudo apt-get install eclipse en el terminal.

Una vez instalado leJOS, y eclipse en Ubuntu, tan solo quedan dos pasos para poder utilizar eclipse para compilar y descargar en el brick los programas:

1. Importar las librerías:

Para los programas que se realizan para que ser ejecutados desde el brick, es necesario importar el archivo clases.jar al proyecto. Mientras que para los programas que se van a ejecutar en el notebook, y que se van a comunicar a través de USB o Bluetooth con el brick, es necesario importar bluecove-2.1.0.jar, bluecove-gpl-2-1-0.jar ( para las comunicaciones por bluetooth), y pccomm.jar.

1. Crear los botones para Compilar y Descargar:

El último paso es crear los botones para que no sea necesario compilar y descargar el programa a través de comandos. El botón Compilar hace referencia al comando nxjc, mientras que el botón de Descargar hace referencia al comando nxj.

#### SubVersiones

#### Text To Speech

#### Sphinx 4

#### PHP

1. Robótica

# Introducción a la Robótica

## Introducción

Un robot puede ser visto en diferentes niveles de sofisticación dependiendo de la perspectiva del espectador. Por un lado un técnico puede ver un robot como una colección de componentes mecánicos y electrónicos, pero por otro lado, un programador simplemente lo ve como una máquina para ser programada.

La Real Academia Española (RAE) define el robot como: “Máquina o ingenio electrónico programable, capaz de manipular objetos y realizar operaciones antes reservadas solo a las personas”. [[2](#_Bibliografía)]

## Breve Historia de la Robótica

1495 - En 1950 se encontraron unos bocetos de un guerrero mecánico con forma humana que fueron diseñados por Leonardo Da Vincien 1495. El “robot” que Leonardo diseño tenía la capacidad de imitar algunos movimientos humanos como sentarse, mover los brazos, mover el cuello, y corregir automáticamente la mandíbula.

1921- La palabra “robot” viene de la palabra croata “robota” que en checo significa “Trabajo forzado” “Exclavitud”. Esta palabra fue utilizada por primera vez para referirse a humanos mecánicos por Karel Capek en su obra Rossum`s Universal Robots (R.U.R).

1942 - Isaac Asimov escribe las tres leyes de la robótica:

“ 1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.

2. A robot must obey any orders given to it by human beings, except where such orders would conflict with the First Law.

3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law”

1954 - George Devol, pide la patente para el primer robot programado.

1956 - George Devol y Joseph Engelberge fundan la primera empresa de la historia dedicada exclusivamente a la fabricación de robot.

- Edmund C. Berkeley crea Squee “ el robot ardilla” que con cuatro sensores, era capaz de localizar una luz y recoger una pelota.

1959 - John McCarthy y Marvin Minsky ponen en marcha un laboratorio de inteligencia artificial en el Instituto de Tecnología de Massachusetts (MIT).

1961 - Heinrich Ernst desarrolla el “MH-1”, un brazo mecánico controlado manualmente, en el MIT.

- General Motors instala el primer robot creado por Unimation en su planta de fabricación de New Jersey, USA.

1963 - Rancho Los Amigos Hospital (California) diseña el primer brazo mecánico controlado por ordenador. Fue diseñado para ayudar a personas minusbalidas.

1966 - Joseph Weizenbaum crea en el MIT el primer programa con inteligencia artificial, fue bautizado con el nombre de Eliza.

- El Instituto de Investigación de Stanford, crea el primer robot capaz de reaccionar sobre su entorno.

1967 - Richard Greenblatt escribe un programa capaz de jugar al ajedrez.

Japon importa su primer robot.

1968 - Kawasaki patenta el robot hidráulico diseñado por Unimation y empieza la producción en Japón.

1969 – Victor Scheinman, profesor de Ingeniería mecánica de la Universidad de Stanford, crea el “Stanford Arm”.

1970 – Shakey, un robot creado en Menlo Park USA, es presentado como el primer robot móvil controlado por inteligencia artificial capaz de ver y evitar obstáculos.

- Stanfor University produce “Stanford Cart”, un sigue líneas que puede ser controlado desde un ordenador usando radio-comunicaciones.

1973 – El departamento de IA de la Universidad de Edimburgo presenta Freddy II, un robot capaz de montar objetos de forma automática a partir de un montón de piezas.

* Cincinnati Milacron Corporation lanza el primer robot controlado por una minicomputadora.

## Clasificación de Robots

Existen diversas formas de clasificación de robots para determinar su potencial en función del software, su físico, sus funciones, y en especial su sofisticación en función de la interacción con los sensores. Los cinco tipos de clasificación de robots objeto de este proyecto son:

1. según su arquitectura.
2. según su generación.
3. según su nivel de inteligencia.
4. según su nivel de control.
5. Según su nivel de programación.

### Clasificación según su Generación:

#### Primera Generación: Robots Play-Back.

Los robots de primera generación empezaron en un mundo construido por y para seres humanos, por lo que éstos tendía a parecerse a humanos. El tamaño, la forma, la fuerza, debían ser lo más semejante posible al de los humanos para que la movilidad por dicho mundo resultara más exitosa. El robot de generación, debía ser eficiente en terreno plano, puesto que la mayoría de tareas suceden aquí, pero también deben ser seguro en escaleras y en terreno irregular para que no quedarse atrapados en una “isla”. También debía ser capaz de manipular objetos cotidianos, y encontrarlos.. Los componentes de esta máquina existen en laboratorios de todo el mundo, y sugerir directrices para un diseño práctico de esta década.

Éstos robots podían moverse por un entorno de forma controlada, sabiendo en todo momento donde se encuentran con un procesamiento de 1.000 MIPS (Millones de instrucciones por segundo). Cuando el robot no se encentra en movimiento, no hay energía suficiente para construir un mapa de espacio de trabajo para localizar objetos en particular, y planificar y controlar los movimientos del brazo. Cuando no está ocupada con sus funciones robóticas único, el robot debe compartir con los ordenadores personales de su tiempo la capacidad de comunicarse a través de redes inalámbricas, para generar e interpretar oraciones habladas y para generar y leer texto impreso. Programas para aplicaciones específicas - muchas obtenidos a través de redes de alta velocidad - se organizan los conceptos básicos para realizar tareas útiles.

Los robots universales encontrarán sus primeros usos en las fábricas, almacenes y oficinas, donde será más versátil que la vieja generación de robots que reemplazan. Debido a su amplitud de aplicación, su número debe crecer rápidamente, y sus costos disminuyen. Con el tiempo llegarán a ser lo suficientemente barato para algunos hogares, se extiende la utilidad de las computadoras personales de algunas tareas en el mundo de los datos para muchos en el mundo físico. Tal vez un programa de limpieza de la casa se incluye con cada robot, como los programas de procesamiento de texto fueron enviados con los primeros ordenadores personales.

Al igual que con las computadoras, algunas aplicaciones de los robots sorprender a sus fabricantes. programas robot puede ser desarrollado para hacer el trabajo mecánico de luz (montaje de otros robots, por ejemplo), entregar los inventarios almacenados, preparar las comidas gourmet específica, afinar ciertos tipos de coche, alfombras gancho de modelado, el césped de malezas, las carreras, jugar, organizar la tierra , piedra y ladrillo o esculpir. Algunas tareas se necesitan accesorios especializados de hardware como las herramientas y sensores químicos. Cada aplicación requiere su propio software original, muy complejo para los estándares actuales del programa de computadora. Los programas contienen los módulos para reconocer, comprender, manipular, transportar y montaje de elementos particulares - módulos desarrollados a través de programas de aprendizaje en supercomputadoras (con cerca de 100.000 MIPS). Con el tiempo, una creciente biblioteca de módulos subtarea puede facilitar la construcción de nuevos programas.

Un robot de primera generación tienen el poder del cerebro de un reptil, pero la mayoría de los programas de aplicación será tan dificultades para cumplir con sus funciones principales que van a dotar al robot con la personalidad de una lavadora.

#### Segunda Generación: Robots Controlados por Sensores.

Robots de primera generación serán esclavos rígidos a los programas inflexible, implacable en la búsqueda de sus tareas - o repetir sus errores. Sus programas contendrá los resultados del aprendizaje congelados hace en grandes ordenadores bajo la supervisión humana. A excepción de episodios especializados como la grabación de una nueva ruta de limpieza o la ubicación de los objetos de trabajo, que será incapaz de aprender nuevas habilidades o la adaptación a las circunstancias imprevistas - aunque modestas modificaciones de la conducta se requiere una nueva programación, probablemente de los proveedores de software original.

Robots de segunda generación, con treinta veces la potencia de procesamiento, será más flexible, ya que pueden hacer algo de a bordo de aprendizaje. La idea fundamental en el aprendizaje adaptativo es "cerrar el círculo" en el comportamiento: evaluar el efecto de cada acción en un contexto dado para mejorar el proceso que generó la acción. En la técnica más simple, una alternativa de comportamiento que tiene éxito es más probable que sea invocado en circunstancias similares, mientras que una alternativa que no se vuelve menos probable. Más rápido estadística enfoques de aprendizaje como las redes neuronales en repetidas ocasiones ajustar los parámetros de control de la conducta-a impulsar respuestas reales más cerca de un ideal. Programas para los robots de segunda generación se utilizan muchas técnicas como el aprendizaje, la creación de nuevas habilidades - y las trampas nuevas.

Si un robot de primera generación que trabajan en su cocina se mete en problemas - por ejemplo, en su defecto para completar un paso clave porque una parte del espacio de trabajo es torpemente pequeño - tiene la opción de abandonar la tarea, el cambio de su entorno, o de alguna manera la obtención de alteración de software que realiza el paso problemática de una manera diferente. Un robot de segunda generación se realizan una serie de falsos comienzos, pero lo más probable será encontrar su propia solución, adaptarse a su hogar en miles de formas más sutiles, y poco a poco mejorar su rendimiento. Si bien la personalidad de un robot de primera generación está determinada exclusivamente por la secuencia de operaciones en el programa de aplicación que se ejecuta en el momento, el personaje de un robot de segunda generación es más un producto de la suite de programas de acondicionamiento que alberga. El sistema de acondicionamiento puede, en tiempo, censurar un programa de aplicación completa, si diera resultados negativos.

Robots de segunda generación de 2010 se han ordenadores de a bordo tan poderoso como las supercomputadoras que aprendió de máquinas de primera generación en 2000. Pero en 2010, supercomputadoras será proporcionalmente más potente (alrededor de 3 millones de MIPS), y se juegan un papel de antecedentes para la segunda generación. Los numerosos programas individuales de un conjunto acondicionado - cada respuesta a un estímulo específico - interactúan entre sí y con los programas de control del robot y el medio ambiente de manera que será demasiado enredado para anticipar con precisión. Sería posible evaluar suites particular por ponerlas en práctica en los robots - la prueba de fuego en todo caso - pero eso sería una manera lenta y peligrosa para tamizar un gran número de candidatos en bruto - algunos sin duda se comportan de forma inesperada que podrían dañar el robot, o incluso poner en peligro los probadores.

Más rápido y más seguro proyecciones iniciales que se podría hacer en simulaciones con superordenadores fábrica de robots en acción. Para ser de valor, simulaciones tendría que ser buenos modelos, predecir con exactitud las cosas tales como la probabilidad de que un apretón dado puede levantar un objeto determinado, o que un módulo de visión puede encontrar algo que se da en el desorden en particular. Simulación del mundo cotidiano en detalle físico completo seguirá siendo más allá de la capacidad de un ordenador en 2010, pero debería ser posible a la aproximación de los resultados, generalizando los datos recopilados de robots reales: fundamentalmente para aprender de la experiencia de trabajo de los robots reales todos los días cómo se comportan las cosas. Un gran esfuerzo de recopilación sistemática bajo la supervisión humana probablemente será necesario que no haya demasiadas lagunas y distorsiones. Un simulador adecuado que contienen al menos miles de modelos aprendidos para diversas interacciones básicas (los llamamos primitivas interacción), en lo que equivale a una versión robótica de la física del sentido común.

#### Tercera Generación: Robots Controlados por Visión.

robots de adaptación de segunda generación se encuentran puestos de trabajo en todas partes, y puede ser la industria más grande en la tierra. Sin embargo, enseñándoles nuevas habilidades, ya sea por escrito oa través de programas de formación, va a ser muy tedioso. Una tercera generación de robots universales, con ordenadores de a bordo tan poderoso como las supercomputadoras que optimiza los programas de segunda generación, aprenderá mucho más rápido debido a que en gran medida de la prueba y error en la simulación rápida en lugar de físico lento y peligroso. Una vez más, un proceso realizado por supercomputadoras humanos-supervisado en la fábrica en la generación de un robot será mejorado e instalado directamente a bordo de la próxima generación, y una vez más las nuevas oportunidades y nuevos problemas surgen voluntad.

Con un simulador de a bordo, se hace posible que un robot para mantener una cuenta corriente de los acontecimientos reales pasa a su alrededor - para simular su mundo en tiempo real. Hacerlo requiere que casi todo los sentidos robot ser reconocidos por el tipo de objeto que es, por lo que los modelos adecuados de interacción puede llamados a filas. El reconocimiento de objetos arbitrarios de vista es tan difícil como saber cómo van a interactuar: se requieren módulos de formación especial para cada tipo de cosas (los llaman primitivas de la percepción). Algunas primitivas percepción puede estar siendo desarrollado para simuladores de fábrica de segunda generación, para ayudar a automatizar el tedioso trabajo de crear simulaciones de áreas de trabajo del robot, sino un esfuerzo adicional para llenar los vacíos y sistematizar ellos seguramente será necesario para prepararlos para su uso totalmente automático la tercera generación. Percepción primitivas permitirá mapa tridimensional de un robot de una habitación que se transforma en un modelo de trabajo, ya que cada objeto es identificado y vinculado con sus primitivas interacción adecuada.

Una simulación de una actualización permanente de uno mismo y el entorno le da un robot habilidades interesantes. Al ejecutar la simulación de un poco más rápido que en tiempo real, el robot puede ver lo que está a punto de hacer, a tiempo para modificar su intención, si la simulación predice que va a salir mal - una especie de conciencia. En una escala más grande, antes de emprender una nueva tarea, el robot puede simular muchas veces, con sistema de acondicionamiento de comprometidos, aprendiendo de las experiencias simuladas ya que a partir de las físicas. Por lo tanto bien entrenados para la tarea, es probable que tenga éxito la primera vez que lo intentaron físicamente - a diferencia de una máquina de segunda generación, que debe hacer que todos sus errores en la vida real. Cuando se tiene un poco de tiempo, el robot es capaz de reproducir las experiencias anteriores, y tratar de variaciones sobre ellos, tal vez aprender formas de mejorar el rendimiento futuro. Un robot de tercera generación lo suficientemente avanzada, cuya simulación se extiende a otros agentes - los robots y la gente - se podrá observar una tarea realizada por otra persona, y formular un programa para hacer la tarea en sí: podría imitar.

A pesar de que será capaz de adaptar, imitar y crear programas sencillos de sus propios robots, de tercera generación aún se basará en los programas de alimentación externa para hacer trabajos complicados. Desde sus funciones motoras y perceptivas será muy sofisticados, y grandes sus recuerdos y habilidades potenciales, será posible escribir maravillosamente los programas de control elaborados por ellos, el cumplimiento de trabajos grandes, con matices dentro de matices. Será cada vez más difícil para los programadores humanos para realizar un seguimiento de los muchos detalles e interacciones. Afortunadamente, la tarea puede ser muy automatizado. Shakey, el primer robot móvil controlado por ordenador, desarrollado en el SRI a finales de 1960, había en su corazón un programa de razonamiento llamado STRIPS (Stanford Research Institute Problem Solver) que expresa la situación del robot y las capacidades como oraciones de la lógica simbólica, y resuelto para la secuencia de acciones que permitió alcanzar un resultado de la solicitada como prueba de un teorema matemático. En 1969, en equipos con un 0,1 MIPS, ni los teoremas ni el procesamiento sensorial que siempre que su entrada podría manejar la complejidad de las situaciones realistas y Shakey se limitaba a las maniobras en torno a unas pocas cuadras. Sin embargo, la idea era buena: dada una descripción correcta del estado inicial y deseada del mundo, y suficiente tiempo y espacio para trabajar, un demostrador de teoremas se encuentra una solución absolutamente correcta, de lo que se requiere la generalidad, la sutileza y astucia, si existe en absoluto. En el momento de la generación de robots universales en tercer lugar, los superordenadores proporcionará 100 millones de MIPS, y (gracias al progreso continuo en la parte superior hacia abajo industria de la Inteligencia Artificial) existen programas que serán capaces de razonar TIRAS-como con la riqueza del mundo real. Así que los superordenadores de fábrica en el 2025 aceptará objetivos compleja (encontrar una secuencia de acciones del robot que monta el robot se describe en la base de datos de diseño siguientes), y compilarlos a través de demostradores de teoremas en los programas de control maravillosamente compleja para los robots de tercera generación, que, a su vez adaptarlos a sus circunstancias concretas.

#### Cuarta Generación: Robots Controlados Adaptablemente.

En las décadas mientras que los "de abajo hacia arriba" la evolución de los robots es lenta la transferencia de las facultades perceptivas y motoras del ser humano en una máquina, la industria de la Inteligencia Artificial convencional será el perfeccionamiento de la mecanización del razonamiento. Dado que los programas de hoy en día ya coinciden los seres humanos en algunas áreas, los de 40 años a partir de ahora, se ejecutan en equipos de un millón de veces más rápido que hoy en día, debe ser bastante sobrehumana. Los programas de hoy razonamiento trabajo de pequeñas cantidades de información inequívoca preparado por los seres humanos - los datos de los sensores del robot, como las cámaras es demasiado voluminoso y demasiado ruidoso para que los usen. Sin embargo, un simulador de robot de buena voluntad contener perfectamente organizados y etiquetados en las descripciones del robot y su mundo, listos para responder a las preguntas de un programa de razonamiento pidiendo, por ejemplo, si un cuchillo sobre un mostrador, o si el robot es la celebración de una taza, o incluso si un ser humano está enojado

Los robots universales de cuarta generación tienen computadoras al mismo tiempo lo suficientemente potente para simular el mundo, y la razón por la simulación. Al igual que los superordenadores de fábrica de la tercera generación, los robots de cuarta generación será capaz de diseñar programas de robot ultra sofisticada, para otros robots o para sí mismos. Porque de otro regalo de la industria de la Inteligencia Artificial, que también será capaz de entender los lenguajes naturales. Mientras que el entendedores idioma original probablemente usará una base de datos verbales de sentido común similar a la que está desarrollando el proyecto Cyc, donde se define el significado de palabras en referencia a otras palabras, en un robot de cuarta generación y las declaraciones de algunos conceptos se entender más profundamente, a través de la acción del simulador. Cuando alguien le dice al robot de "el agua se está ejecutando en la bañera", el robot puede actualizar su simulación del mundo que incluye el flujo en la bañera no visto, en una extrapolación simulada indicaría un exceso no deseado después, y así motivar a que el robot vaya a cerrar el grifo. Una representación puramente verbal puede lograr lo mismo si se incluyen las declaraciones tales como "Una bañera de llenado se desbordará si su agua no se cierra", pero un número modesto de los principios generales en un simulador, que interactúan en las combinaciones, puede proporcionar el equivalente información de un número indefinido de oraciones. Del mismo modo un programa de razonamiento, hacer inferencias acerca de las cosas físicas, que se podría mejorar mediante un simulador: inferencias candidato sería rechazada si no en una simulación paralela de un caso típico, y, por el contrario, las coincidencias persistentes en la simulación podría sugerir que las declaraciones se puede demostrar - el robot se visualiza como se escuchaba, hablaba y motivado. Una versión modesta pero muy exitosa de este enfoque fue utilizado en uno de los primeros programas de inteligencia artificial, un demostrador de teoremas de geometría por Herbert Gelernter en 1959. A partir de los postulados y reglas de inferencia de Euclides "Elementos", resultado del programa Gelernter de algunos de los teoremas, utilizando algebraica "esquemas" para eliminar las direcciones falsas en las pruebas. Antes de intentar probar dos triángulos congruentes en una cierta construcción, por ejemplo, el programa generará un ejemplo de la construcción, usando números al azar para las cantidades no especificadas, y medir los triángulos resultantes. Si no eran lo suficientemente similares - dentro de la precisión de la aritmética - el programa de abandono que el enfoque y trató de otra cosa.

la comprensión del lenguaje simulador-aumentada y razonamiento puede ser tan eficaz en los robots que se apruebe para su uso en los programas de ordenador normal, "tierra" en el mundo físico a través de las experiencias de los robots que atentos los simuladores. Con el tiempo la distinción entre los controladores de robot y razonadores sin cuerpo disminuirá, y los programas de razonamiento a veces se vinculará a los órganos del robot de interactuar físicamente con el mundo, y la mente del robot a veces se jubilarán en los equipos grandes, para hacer algunas intensa pensar fuera de línea.

Un robot de cuarta generación será capaz de aceptar las declaraciones de propósito de los seres humanos, y "compilar" en programas detallados que realizar la tarea. Con una base de datos sobre el mundo en general, los estados pueden llegar a ser bastante generales - cosas como "ganarse la vida", "hacer más robots" o ". Hacer un robot más inteligente" De hecho, los robots de cuarta generación tendrá la competencia general de los seres humanos, y se parecen a nosotros en algunos aspectos, pero en otros, como si nada se ha visto el mundo antes. A medida que el diseño de sus propios sucesores, el mundo será cada vez extraño.

#### Quinta Generación: Robots con Inteligencia Artificial.

Solo este apartado es objeto de una tesis

1. El Problema

# Introducción

# Solución

## La plataforma del Robot

### Partes del Robot:

#### Ladrillo Inteligente:

El Ladrillo Inteligente es el elemento fundamental del robot. Éste incluye la CPU, el sistema de comunicación, los puertos de entrada/ salida, y el interfaz de usuario. Todo en un bloque sólido de plástico de pequeño tamaño y peso, como se muestra en la Ilustración 4: Ladrillo Inteligente del NXT.



Ilustración 4: Ladrillo Inteligente del NXT

La *Tabla 1* está tomada de la página oficial de LEGO [[3](#_Bibliografía)] y en ella se aprecia que lego NXT no es solo un juguete, sino que su procesador similar a un INTEL 486, sumado a los puertos de entrada/salida y las diferentes formas de comunicación, hacen del NXT que el NXT tenga un potencial muy elevado.

|  |  |
| --- | --- |
| LEGO Mindstorm NXT Intelligent Brick | |
| Procesador Principal | Atmel 32-bit ARM (AT91SAM7S256) |
| 256 KB de Memoria Flash |
| 65 KB de Memoria Ram |
| Reloj a 48 MHz |
| Segundo Procesador | Atmel 8-bit AVR (ATmega48) |
| 4 KB de Memoria Flash |
| 512 B de Memoria Ram |
| Reloj a 8 MHz |
| Procesador para la Comunicación Bluetooth | CSR BlueCore 4 v2.0 (System EDR) |
| Serial Port Profile (SPP) |
| 47 KB de Memoria Ram Interna |
| 8 MB de Memoria Flash Externa |
| Reloj a 26 MHz |
| USB 2.0 | Puerto a 12 Mbits/s |
| Puertos de Entrada | 4, por cable de 6 l íneas, con soporte digital y analógico siendo uno de ellos de alta velocidad (IEC 61158 Tipo 4/EN 50170) |
| Puertos de Salida | 3, por cable de 6 líneas. |
| Soporte para entrada desde encoders |
| Display | LCD monocromo |
| Resolución: 100x64 (26x40.6 mm) |
| Altavoz | Canal de 8 bit |
| Ancho de banda: de 2 a 16 Khz |
| Interfaz de usuario | 4 botones |
| Alimentación | 6 pilas AA |

Tabla 1: Resumen del Hardware del NXT

#### Ultrasonic Sensor:



Ilustración 5: Sensor de Ultrasonido.

El sensor de ultrasonido ayuda a al robot NXT a medir distancias y a identificar los objetos.

El sensor mide la distancia a un objeto mediante el cálculo del tiempo que tarda una onda de sonido en golpear el objeto y volver. A diferencia de los otros sensores éste da el resultado en unidades de medida.

El alcance máximo del sensor es de unos 170 cm aproximadamente.

Si no hay eco fue detectado, el valor devuelto es de 255

El código está tomado de la página web de leJOS [[4](#_Bibliografía)], y servirá de ejemplo para ver cómo funciona el sensor programado en java.

/\*

Sample of Ultrasonic Java Testing Program.

\*/

import lejos.nxt.\*;

public class SonicTest {

public static void main(String[] args) throws Exception {

UltrasonicSensor sonic = new UltrasonicSensor(SensorPort.S1);

while (!Button.ESCAPE.isPressed()) {

LCD.clear();

LCD.drawString(sonic.getVersion(), 0, 0);

LCD.drawString(sonic.getProductID(), 0, 1);

LCD.drawString(sonic.getSensorType(), 0, 2);

LCD.drawInt(sonic.getDistance(), 0, 3);

}

}

}

Para utilizar el sensor es necesario crear una instancia de UltrasonicSensor, e indicar en que puerto se encuentra conectado. Para calcular la distancia a cualquier objeto se utiliza el método getDistance() que devuelve la distancia en centímetros.

El sensor de ultrasonidos es un elemento secundario en el robot, su función es evitar la colisión con obstáculos, para evitar el desplazamiento inesperado, y en consecuente, la pérdida de la línea (nodo) que guía al robot.

#### Color Sensor:



Ilustración 7 : Sensor Óptico.

El sensor óptico incluye un fototransistor para medir la iluminación con un LED que puede ser activado dependiendo de si la medida se toma en un lugar oscuro o iluminado. De acuerdo a [[5](#_Bibliografía)], el rango de medición del sensor es más ancho que el rango de visión del ojo, esto provoca que en la práctica el sensor obtenga resultados inesperados. El rango de visión, según [[5](#_Bibliografía)], abarca de 0 a 1000 lux.

El ejemplo siguiente está sacado de la página oficial de leJOS [[4](#_Bibliografía)] y pretende obtener el valor de iluminación obtenido por el sensor:

/\*

Sample of Light Sensor Java Testing Program.

\*/

import lejos.nxt.LCD;

import lejos.nxt.LightSensor;

import lejos.nxt.SensorPort;

public class LightTest {

  public static void main(String[] args) throws Exception {

    LightSensor light = new LightSensor(SensorPort.S1, true);

    while (true) {

      LCD.drawInt(light.readValue(), 1, 0, 0);

    }

  }

}

Para utilizar el sensor es necesario crear una instancia de LightSensor, e indicar en que puerto se encuentra conectado. Para obtener el valor de iluminación es necesario llamar al método readValue().

El sensor de luz es el sensor más importante del proyecto. Es el encargado de guiar al robot sobre el mapa de aristas y nodos que representa el supermercado, y el encargado de diferenciar las zonas de productos.

#### Motores:

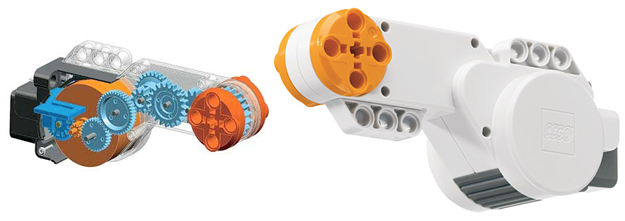


Ilustración 8: Motores del NXT

De acuerdo a las pruebas realizadas en [[6](#_Bibliografía)], el motor del LEGO NXT es capaz de alcanzar una velocidad de 160 rpm sin carga, y hasta 120 rpm con una carga de 11,5 N • cm, se alimentan con 9 V. Es capaz de desarrollar un par máximo de 25 N cm a 60 rpm. No se recomienda exceder de 15 N • cm durante períodos prolongados.

La siguiente tabla está sacada de la página oficial de leJOS [[4](#_Bibliografía)], en ella se observan los diferentes métodos de la clase Motor.

|  |  |
| --- | --- |
| Method name | Notes |
| forward() | Start the motor rotating forward |
| backward() | Start rotating backward |
| changeDirection() | Reverse the direction of rotation |
| stop() | Stop quickly |

Tabla 2: Métodos para la clase Motor

## La plataforma en el centro comercial

## La plataforma Software

### LineFollower

Calibrado: <http://www.google.es/search?sourceid=chrome&ie=UTF-8&q=what+is+Linefollower#hl=es&sa=X&ei=5oC4TYGyLYeahQe5r8CMDw&ved=0CBcQBSgA&q=what+is+Line+Follower&spell=1&bav=on.2,or.&fp=6b4d48afcbf98eaa>

## Estudio Económico

|  |  |  |
| --- | --- | --- |
| **Hardware** | **Cantidad** | **Precio** |
| NXT Brick | 1 | 158.95 |
| NXT Ultrasonic Sensor | 1 | 32.95 |
| NXT Color Senosr | 3 | 17.95 |
| NXT Motor | 2 | 24.95 |
| Notebook Asus 1005HA | 1 | 250 aprox |
| Bluetooth Headphones | 1 | 10 |
| USB Bluetooth Device | 1 | 5 |
|  | TOTAL: | 560[[1]](#footnote-1) |
| **Software** |  |  |
| Ubuntu 10.04 | Gratis[[2]](#footnote-2) | |
| Java | Gratis[[3]](#footnote-3) | |
| leJOS | Gratis[[4]](#footnote-4) | |
| Eclipse | Gratis[[5]](#footnote-5) | |
|  |  | |
| SVN | Gratis[[6]](#footnote-6) | |
| Sphinx 4 | Gratis[[7]](#footnote-7) | |
| PHP | Gratis | |

Al utilizar software de código abierto, evitamos tener que comprar licencias a un alto precio. Esto mismo nos ofrece la posibilidad de disponer del software necesario para desarrollar el proyecto tanto en casa como en la Universidad, Centro Escolar, o donde corresponda.

# Valoración Final

## Conclusiones

## Aspectos Negativos

## Trabajos Futuros

# Bibliografía

1. Española, R. A. (2010). *Diccionario de la lengua española.* Madrid: RAE.*NXT® motor internals.* (Noviembre de 2008). Recuperado el 5 de Diciembre de 2010, de http://www.philohome.com/nxtmotor/nxtmotor.htm
2. Asimov, I. (1950). *I, Robot.* Colección Diamante.
3. Española, R. A. (2010). *Diccionario de la lengua española.* Madrid: RAE.
4. Group, T. L. (s.f.). *LEGO Mindstorm*. Recuperado el 15 de Septiembre de 2010, de http://mindstorms.lego.com/en-us/Default.aspx
5. Hurbain, M. G. (2007). *Extreme NXT.* Apress.
6. leJOS. (2 de Septiembre de 2009). *Java for LEGO Mindstorms.* Obtenido de Java for LEGO Mindstorms: http://lejos.sourceforge.net/
7. Microsystems, L. &. (s.f.). *Lejos, Java for LEGO*. Recuperado el 4 de Octubre de 2010, de http://lejos.sourceforge.net/
8. Moral, J. A. (2010). *Instalación del proyecto LeJOS.* Universidad de Alclá, Madrid.
9. Anexos

# Anexo A: Lista de Problemas

## Problema 1: Instalar Java en Ubuntu

**Problema:** E:Línea 54 mal formada en lista de fuentes /etc/apt/sources.list

**Solución:**

1. Abrir /etc/apt/source.list
2. Borrar los repositorios del final.
3. Actualizar.
4. Continuar la instalación.

## Problema 2: SAM-BA mode

**Problema** : “An error occurred: Failed to open device in SAM-BA mode”

**Solución**:

1. sudo modprobe cdc\_acm
2. Resetear Shell
3. sudo rmmod cdc\_acm
4. sudo -s
5. echo "" >> /etc/modprobe.d/blacklist.conf
6. echo "blacklist cdc\_adm" >> /etc/modprobe.d/blacklist.conf

## Problema 3: Conexión USB

**Problema:** Driver USB no instalado para lejos

**Solución:**

## Problema 4: Sensor de Luz

**Problema:** Incompatibilidad del sensor 64892 (ColorLightSensor).

**Solución:** Sustituir los sensores 64892 por 55969 (Light Sensor).

1. Existen descuentos para estudiantes, centros escolares y universidades. [↑](#footnote-ref-1)
2. <http://www.ubuntu.com/> [↑](#footnote-ref-2)
3. <http://www.java.com/es/download/> [↑](#footnote-ref-3)
4. <http://lejos.sourceforge.net/> [↑](#footnote-ref-4)
5. <http://www.eclipse.org/> [↑](#footnote-ref-5)
6. <http://rapidsvn.tigris.org/> - Linux. [↑](#footnote-ref-6)
7. <http://cmusphinx.sourceforge.net/sphinx4/> [↑](#footnote-ref-7)