

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.

Juan Antonio Breña Moral

Alumno: Miguel Alfonso Ruiz Nogués.

AGRADECIMIENTOS

I wish to express my sincere gratitude to professor Rich Piking and Vic Grout, for directing the project.

I would also like to special thanks to Juan Antonio Breña Moral, leJOS developer who helped me changing the scope for my project, in order to use the latest technology.

Last but not least I wish to avail myself of this opportunity, express a sense of gratitude and love to my friends and my beloved parents for their manual support, strength, help and for everything.

Wrexham

30th of March of 2011

ESTUDIO TEÓRICO PRÁCTICO DE LA ROBÓTICA AL SERVICIO DE LA SOCIEDAD

Autor: Miguel Alfonso Ruiz Nogués⁽¹⁾,

Directores: José Miguel Ordax Cassá⁽²⁾, Juan Antonio Breña Moral⁽³⁾

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas.

Resumen

La robótica es una combinación de ciencia y tecnología en vías de desarrollo que realiza diferentes tareas para ayudar al ser humano. Éste proyecto pretende *construir un prototipo robot capaz de ayudar y guiar a personas con discapacidad visual o móvil en diferentes entornos* y a un bajo coste para que sirva de ejemplo para centros escolares y universidades.

Palabras Claves: Robótica, Java, leJOS, Behavior Programming,FreeTTS, Sphnix4.

1. 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

¹ Estudiante ITIGyITIS de la universidad de ICAI. E-mail: mrnogues89@gmail.com

² Profesor de ICAI. E-mail: ordax@es.ibm.com

³ Equipo de Investigación de leJOS .Web: <http://www.juanantonio.info/index.php>. E-mail: bren@juanantonio.info

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 propone una solución robótica que permita a personas con discapacidad visual realizar una compra en un supermercado.

2. Objetivos del Proyecto:

El proyecto persigue dos objetivos paralelos y claramente diferenciados:

Por un lado persigue fomentar la robótica al servicio de la sociedad en Universidades y Centros Escolares mediante la utilización de software libre y hardware de bajo coste, también con la publicación de resultados, vídeos e imágenes en internet.

Por otro lado el proyecto desarrolla un prototipo de robot capaz de ayudar a personas con discapacidad visual a realizar la compra en un supermercado. La persona se comunica con el robot mediante voz indicándole que productos desea adquirir.

3. Método de Resolución:

El robot se desplaza sobre una superficie, que simula ser un centro comercial, dibujada con líneas negras que indican el camino del robot, y marcas que indican la situación de los productos.

El robot consta de 3 sensores de luz, dos de los cuales se utilizan para el camino (la línea de guiado), el tercero se utiliza para identificar las marcas de área de producto, de esta forma el robot conoce su posición. Para desplazarse el robot utiliza dos motores capaces de generar una velocidad de 120 rpm.

La comunicación entre el robot y la persona, se realiza a través de unos auriculares-micrófono con conexión Bluetooth. De ésta forma, el individuo comunica el producto que desea obtener mediante el micrófono, el robot realiza una consulta a la base de datos situada en internet (programada en PHP) para obtener la posición del producto, y posteriormente se desplazará a la posición. La conversión texto-voz se realiza mediante un programa Text to Speech (Sphinx 4).

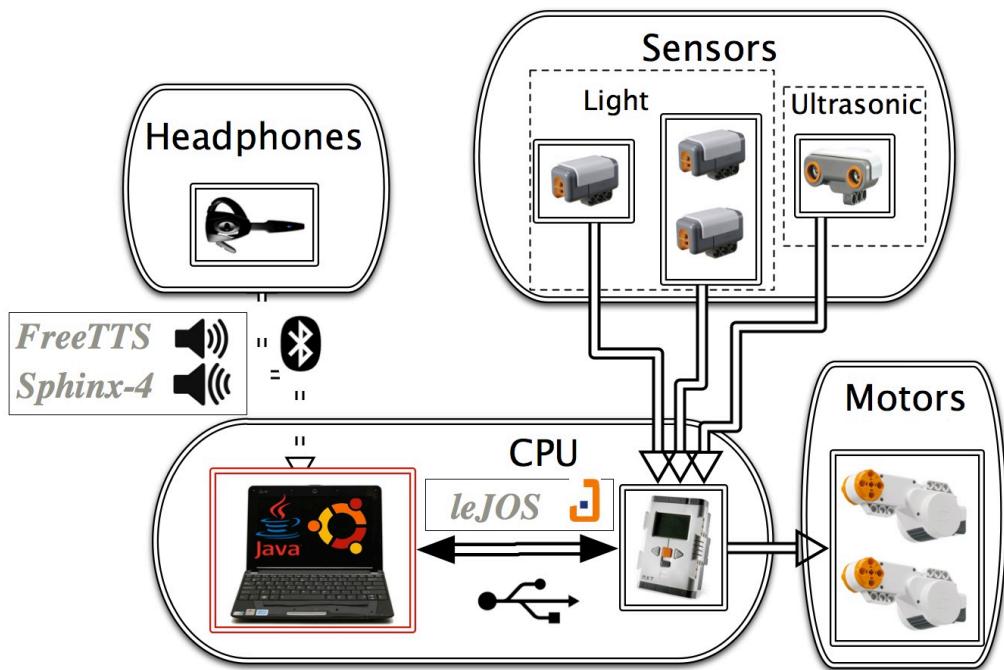
El CPU del robot es una comunicación entre un Brivk NXT y un netbook. El verdadero núcleo es el netbook, que es quien procesa la comunicación por voz, y quien a través de conexión USB indica al brick recibe las lecturas de los sensores y controla los motores.

4. Resultados:

Los resultados obtenidos se pueden dividir en dos, según los objetivos:

1) Prototipo:

Se ha conseguido desarrollar un prototipo de robot programado en Java, que con tecnología en estado-de-arte de reconocimiento y emisión de voz (Sphinx 4, FreeTTS) es capaz de comunicarse con una persona para obtener el producto desarrollado, y desplazarse a su posición. En la siguiente imagen se puede observar la configuración hardware y el uso de las tecnologías utilizadas:



2) Económico:

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

Como se puede observar en la tabla, el presupuesto total conseguido es de 560 dólares, este precio es un logro a la hora de demostrar que la robótica está al alcance de todos.

Gracias a la utilización de software de código abierto, y gracias a que el prototipo creado no es para uso comercial, se ha evitado la necesidad de pagar por

las licencias de los programas software. Gracias a esto último, se tiene la posibilidad de disponer del software necesario tanto en casa como en un laboratorio, centro Universitario, o centro Escolar.

5. Conclusiones:

6. Referencias:

- [1] A. Autor, B. Autor, *Título de Libro*, Editorial, Lugar de Edición, (AÑO).
- [2] H. Autor, N. Autor, *Abreviatura Revista*, Vol.(Número) (AÑO) primera página.
- [3] J. Autor, *Actas de Nombre del Congreso*, Ref. Comunicación, Lugar, (AÑO).
- [4] Identificación de Norma, *Título de la Norma*, (AÑO).



Índice de la memoria

Índice de la memoria

| | |
|--|-----------|
| Parte I : Memoria | 1 |
| Capítulo 1 Introducción | 2 |
| 1.1 Motivación del proyecto | 3 |
| 1.2 Objetivos..... | 4 |
| 1.3 Metodología / Solución desarrollada | 6 |
| 1.4 Recursos..... | 10 |
| 1.4.1 Técnicas | 10 |
| 1.4.1.1 Beavior Programming | 10 |
| 1.4.1.1.1 Behavior Programming en leJOS | 12 |
| 1.4.2 Hardware..... | 14 |
| 1.4.2.1 Lego NXT | 14 |
| 1.4.2.2 Notebook Asus EEE 1005HA..... | 18 |
| 1.4.2.3 Bluetooth Headphones..... | 18 |
| 1.4.3 Software | 19 |
| 1.4.3.1 Ubuntu 10.04..... | 19 |
| 1.4.3.2 Java:..... | 20 |
| 1.4.3.3 leJOS: | 21 |
| 1.4.3.4 Eclipse:..... | 23 |
| 1.4.3.5 SubVersiones..... | 25 |
| 1.4.3.6 Sphinx 4 | 25 |



Índice de la memoria

| | |
|---|-----------|
| 1.4.3.7 FreeTTS | 31 |
| Parte II Robótica..... | 34 |
| Capítulo 1 Introducción a la Robótica..... | 35 |
| 1.1 Introducción | 35 |
| 1.2 Breve Historia de la Robótica | 35 |
| 1.3 Clasificación de Robots | 41 |
| Parte III El Problema | 42 |
| Capítulo 1 Introducción | 43 |
| 1.1 Planteamiento: | 43 |
| 1.2 TortuleBot..... | 44 |
| 1.2.1 Software: | 45 |
| 1.2.2 Hardware..... | 46 |
| 1.2.3 Conclusión..... | 47 |
| Capítulo 2 Solución..... | 48 |
| 2.1 La plataforma del Robot..... | 48 |
| 2.1.1 Netbook | 49 |
| 2.1.2 Ladrillo Inteligente: | 50 |
| 2.1.3 Ultrasonic Sensor: | 53 |
| 2.1.4 Color Sensor: | 55 |
| 2.1.5 Motores: | 58 |
| 2.1.6 Chasis | 60 |
| 2.2 La plataforma en el centro comercial..... | 62 |
| 2.3 La plataforma Software | 63 |



Índice de la memoria

| | |
|--|-----------|
| 2.3.1 Diagrama UML | 63 |
| 2.3.2 División de CPU: | 65 |
| 2.3.3 LineFollower | 66 |
| 2.3.4 Navegación | 68 |
| 2.3.4.1 Representación del entorno | 69 |
| 2.3.4.2 ¿Dónde estoy? – Posicionamiento | 73 |
| 2.3.4.3 ¿Dónde quiero ir? – Objetivo | 74 |
| 2.3.4.4 ¿Cómo voy? – Generación de camino..... | 74 |
| 2.4 Estudio Económico..... | 76 |
| Capítulo 3 Valoración Final | 79 |
| 3.1 Conclusiones..... | 79 |
| 3.1.1 Conclusiones tecnologicas..... | 79 |
| 3.1.1 Objetivos | 80 |
| 3.2 Aspectos Negativos..... | 81 |
| 3.3 Trabajos Futuros..... | 82 |
| 3.3.1 Montecarlo..... | 82 |
| 3.3.2 ROS (Robot Operating System)..... | 84 |
| 3.3.3 Microsoft Kinect..... | 86 |
| Capítulo 4 Bibliografía | 89 |
| Parte IV Anexos | 91 |
| Anexo A: Lista de Problemas | 92 |
| 1.1 Problema 1: Instalar Java en Ubuntu | 92 |
| 1.2 Problema 2: SAM-BA mode..... | 92 |
| 1.3 Problema 3: Conexión USB..... | 93 |



UNIVERSIDAD PONTIFICIA COMILLAS

Escuela Técnica Superior de Ingeniería

Índice de la memoria

| | |
|--|------------|
| 1.4 Problema 4: Sensor de Luz | 93 |
| 1.5 Problema 5: Estructura del Robot | 93 |
| 1.6 Problema 6: Sensor de Marcas de Producto I..... | 94 |
| 1.7 Problema 7: Sensor de Marcas de Producto II..... | 95 |
| 1.8 Problema 8: Sensor de Marcas de Producto III..... | 97 |
| Capítulo 2 Anexo B: Pruebas | 99 |
| Anexo C: Archivo De Configuración Sphinx 4 | 100 |
| Anexo D: Publicación del Proyecto..... | 103 |



Índice de figuras

Índice de figuras

| | |
|---|----|
| Ilustración 1: Fases del Proyecto | 6 |
| Ilustración 3: Behavior Programming | 12 |
| Ilustración 4: LEGO RXC. | 15 |
| Ilustración 5: TortleBot vision. http://www.youtube.com/watch?v=MOEjL8JDvd0 | 45 |
| Ilustración 6: Hardware TurtleBot. [9]..... | 46 |
| Ilustración 7: Ladrillo Inteligente del NXT..... | 50 |
| Ilustración 8: Sensor de Ultrasonido. | 53 |
| Ilustración 9: Funcionamiento del Sensor de Ultrasonido..... | 53 |
| Ilustración 10 : Sensor Óptico..... | 55 |
| Ilustración 11: Calibración de Light Sensor..... | 57 |
| Ilustración 13: Material para la Plataforma del Supermercado | 63 |
| Ilustración 14: Comparación Algoritmo PID | 67 |



UNIVERSIDAD PONTIFICIA COMILLAS

Escuela Técnica Superior de Ingeniería

Índice de figuras

Índice de tablas

| | |
|---|----|
| Tabla 2: Métodos para la clase Motor..... | 59 |
|---|----|



UNIVERSIDAD PONTIFICIA COMILLAS

Escuela Técnica Superior de Ingeniería

Parte I: MEMORIA



Capítulo 1 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.

1.1 MOTIVACIÓN DEL PROYECTO

Este proyecto surgió como solución al problema que tiene las personas con discapacidad visual y de movilidad. Antes de decantarme por una solución robótica, analice el problema y establecí posibles soluciones, como la creación de una herramienta para páginas web que ayude su adaptación a personas con discapacidad visual.

La solución robótica fue escogida por motivos tecnológicos, personales y económicos:

- Tecnológicos: La robótica es una combinación de ciencia y tecnología en vías de desarrollo que aporta soluciones para ayudar al ser humano. Parte de los logros de la robótica



son; la conquista de otros planetas, la sustitución de brazos en personas.

- Económicos: El mayor gasto de una empresa suele ser en muchas ocasiones el coste humano. Esto significa que crear una aplicación que requiera el menor número de empleados posible, reduciría gastos, provocando beneficio.
- Personal: La robótica no es una asignatura que es estudiada en la carrera, es una asignatura que se aprende por interés propio. Un buen punto de partida para adquirir conocimientos y aplicarlos es el proyecto fin de carrera.

Durante los años de universidad, he aprendido una serie de herramientas y técnicas relacionadas con la informática, que me permiten entender los conceptos de la robótica para poder así desarrollar un prototipo de robot, e introducirme en mundo de la robótica que espero que sea en donde trabaje el día de mañana.

1.2 *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.

- 2) 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.

1.3 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.

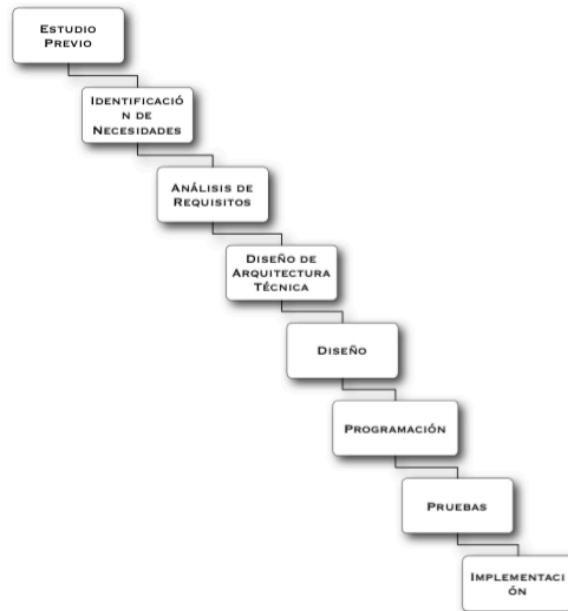


Ilustración 1: Fases del Proyecto

I. 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 el prototipo, y poder indicar las técnicas y tecnologías más apropiadas para su implementación real.

Identificación de Necesidades:

En esta fase se pretende comprender el entorno global del problema que queremos solucionar, para poder así elaborar una lista de requisitos generales del robot.

II. Análisis de Requisitos:

El objetivo de esta fase es alcanzar un conocimiento suficiente del robot, definiendo necesidades, problemas y requisitos. Se estudiará a TurtleBOT, un robot en estado de arte que utiliza Microsoft Kinect como único sensor.

III. Diseño de Arquitectura Técnica:

Tras el estudio de las tecnologías existentes para el desarrollo de robots se pretende escoger la más adecuada para la implantación.

IV. Diseño:

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



UNIVERSIDAD PONTIFICIA COMILLAS

Escuela Técnica Superior de Ingeniería (ICAI)

V. Programación:

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

VI. Pruebas:

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

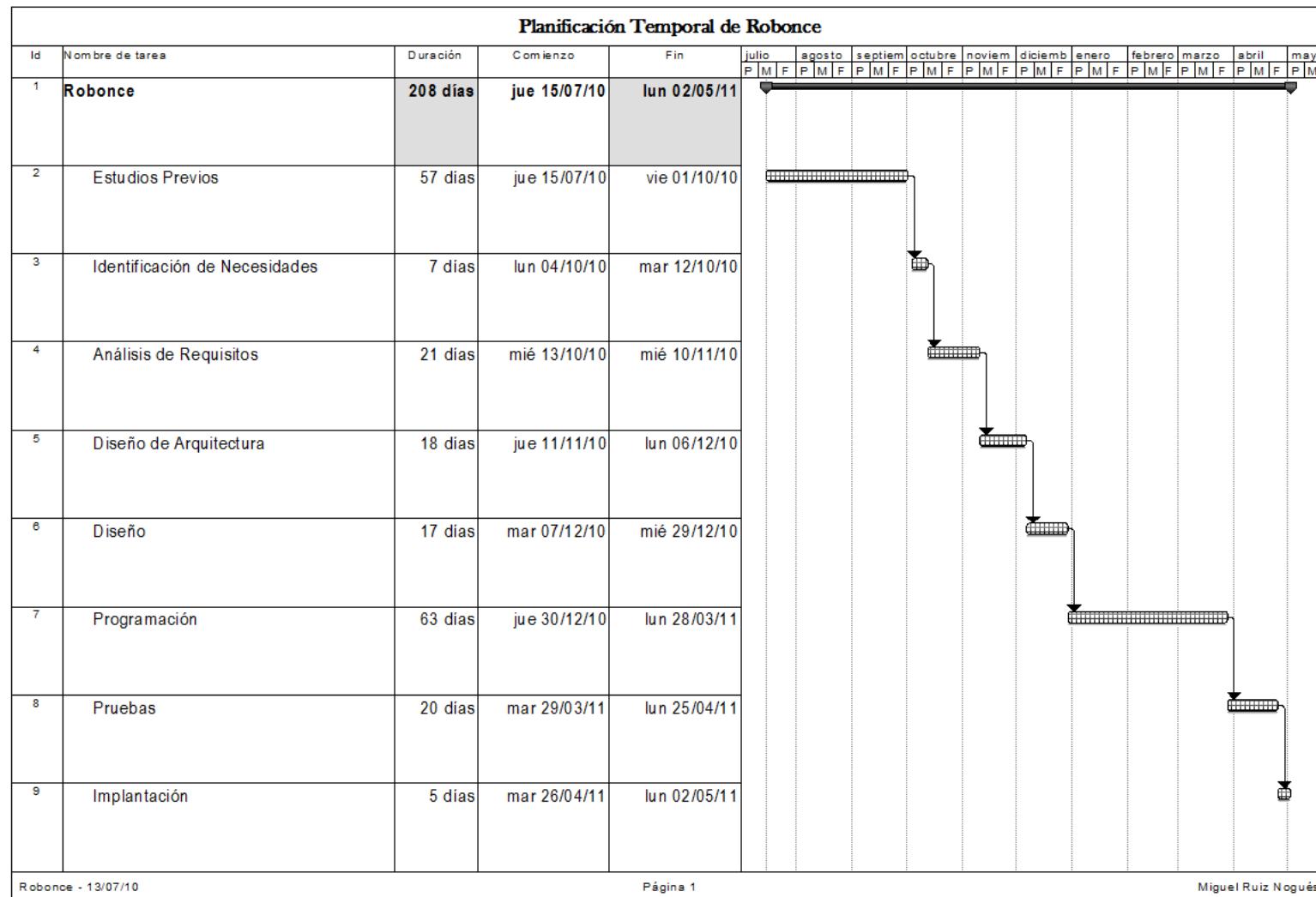
VII. Implementación:

Se recreara un supermercado en miniatura para la demostración.



UNIVERSIDAD PONTIFICIA COMILLAS

Escuela Técnica Superior de Ingeniería (ICAI)





1.4 RECURSOS

1.4.1 TÉCNICAS

1.4.1.1 Beahavior 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 *Behavior 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 consigue como resultado de que los *behavior* consigan sus fines de forma independiente. Si se construye correctamente, el robot parece pensar y tomar decisiones sobre su entorno para completar la tarea.



Behavior Programming

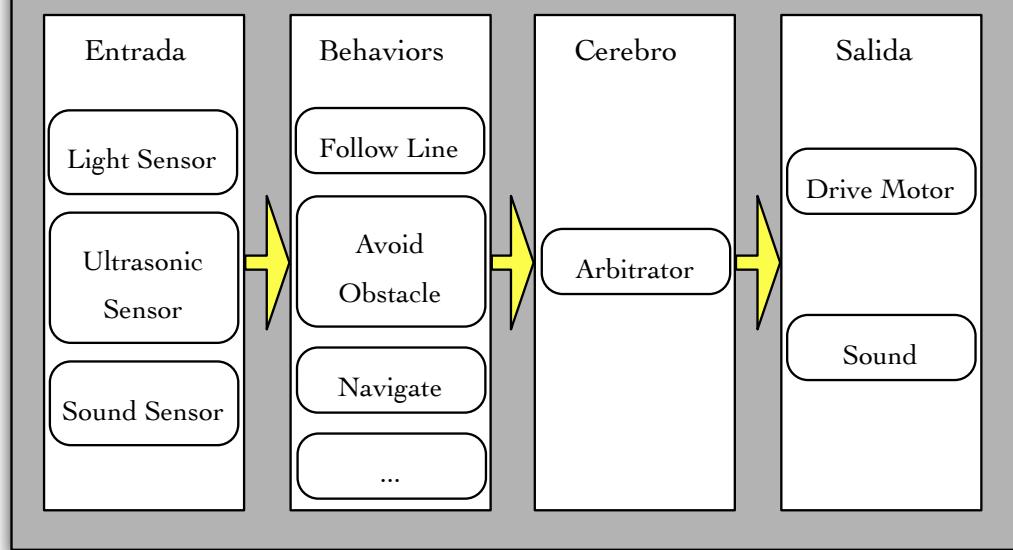


Ilustración 2: Behavior Programming

1.4.1.1.1 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 ()

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

Método: void suppress ()

3. 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 llamará al método `supress ()` de aquel comportamiento cuyo método `action ()` esté en ejecución, terminado la ejecución de éste inmediatamente.

1.4.2 HARDWARE

1.4.2.1 *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: LEGO RXC.). 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 3: LEGO RXC.

Esta primera generación de Mindstorms 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 sido escogido para este proyecto por la relación posibilidades-precio que ofrece, pero existen otras alternativas en el mercado:

- **Arduino** es una plataforma electrónica de prototipos de código abierto basada en hardware flexible, software y fácil utilización. Está pensada para cualquier persona con intención de crear objetos o entornos interactivos. Arduino cuenta con un IDE para



desarrollar software, un lenguaje de programación de alto nivel parecido a C, buen precio. El gran inconveniente de Arduino es que está pensado para electrónica, más que para programación.

- **Sun Spot** es una plataforma desarrollada por Sun Microsystems (Oracle) diseñada para la comunicación vía internet, y de diferentes aparatos electrónicos. El lenguaje de programación de la plataforma es Java. El inconveniente principal de la plataforma, es que no ha sido diseñada para robótica, sino para la interacción de diferentes aparatos electrónicos.

Las desventajas que se han encontrado a la hora de hacer el prototipo con Lego NXT son principalmente 2:

1. La plataforma creada por lego es débil, y tiene problemas para sostener el peso del netbook.
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 características de procesamiento se convirtió en el procesador principal.



1.4.2.2 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 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.

1.4.2.3 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.

En una primera instancia se escogió para la comunicación oral el sensor de sonido de lego, pero luego se cambió a los headphones para acercar el micrófono a la boca del individuo lo máximo posible, esto evita la lectura de ruidos innecesarios.

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



1.4.3 SOFTWARE

1.4.3.1 Ubuntu 10.04

Ubuntu 10.04 también conocido como Lucid Lynx, es una versión LTS (Long Term Support) lo que supone que desde su lanzamiento el 29 de Abril de 2010 hasta pasados tres años estará bajo soporte técnico.

Los requisitos mínimos de instalación son:

- Procesador 1GHz x86.
- Memoria RAM: 512 mb.
- Disco duro: 5 GB.

Linux fue escogido entre otros sistemas operativos por tres grandes motivos; De cara a los objetivos, Linux es un sistema operativo de código abierto, que no supone un gasto. Por otro lado, la conexión entre el netbook y el NXT Brick, se realiza con pc.comms, y a la hora de dar permisos sobre el USB en Windows plantea problemas. Por último Linux es un sistema operativo estudiado en la universidad, que además es utilizado por muchas de las grandes empresas de hoy en día como Google, HP, Dell, etc.



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

1.4.3.2 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 lenguajes 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.



1.4.3.3 *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.

leJOS es el encargado de la comunicación entre el brick NXT y el netbook, mediante el paquete pc.comms que se estudiará más adelante.

La librería clases.jar contiene el API de leJOS, que a su vez se encuentra en la página oficial de leJOS [7] .

leJOS proporciona cinco soluciones básicas en las que se basa el proyecto:

- JVM para el NXT Brick: necesaria para la programación en Java del Brick.
- API para NXT Brick: necesaria para la programación. Una plataforma sin API es una plataforma sin documentación.
- Paquete PC.comms: necesario para la comunicación en tiempo real del netbook con el Brick a través de USB.
- Por último, lejos dispone de unas herramientas como; NXJflashg que permite instalar el firmware de lejos a través de un entorno visual, NXJbrowse permite gestionar los archivos del



NXT y aporta información sobre la conexión, NXJMonitor permite probar los sensores y actuadores.

- Se distribuye libremente.

A rasgos generales, leJOS es una plataforma idónea para introducirse en el mundo de la robótica, aportando conceptos básicos de robótica tradicional y de inteligencia artificial, que utiliza un lenguaje de programación extendido mundialmente. La página oficial [7] aporta tutoriales de iniciación y avanzados, junto con documentación detallada de todos los paquetes y clases, el foro lo revisan castamente los desarrolladores, que con amabilidad responden a dudas, y solucionan problemas.

RobotC es un entorno de pago muy popular entre los programadores de NXT. Es el más común en los campeonatos, pero fue descartado por utilizar un lenguaje de alto nivel (C), y por romper el objetivo económico.

NXT-G es el IDE original de Lego, está basado en NI Labview, y es el que viene incorporado en los NXT originales. Es un lenguaje sencillo, y muy visual, pero no es práctico para este proyecto. Además éste lenguaje solo se utiliza para programar robots lego.

BricxCC es un entorno de libre distribución, que se puede considerar junto con leJOS, el más completo de todos, permitiendo el



manejo del Brick desde conexión USB y bluetooth. Igual que RobotC, éste entorno se programa en un lenguaje de algo nivel.

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 89.

1.4.3.4 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`.

2) 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.

1.4.3.5 SubVersiones

SubVersiones nos permite llevar un seguimiento de los cambios en los ficheros de código fuente del proyecto.

Para el proyecto se ha utilizado RapidSVN por ser de libre distribución, y por ser un interfaz gráfico.

1.4.3.6 Sphinx 4

El reconocimiento de voz es una mezcla realmente compleja entre matemáticas, física, informática y lingüística. Para crear un sistema de reconocimiento de voz son necesarios años de investigación.

Para los desarrolladores que simplemente quieren utilizar un reconocimiento de voz ya creado, existe Sphinx 4, que es un sistema en estado-de-arte de reconocimiento de voz programado totalmente en Java. Fue creado en la universidad de Carnegie Mellon, con la colaboración de Sun Microsystems, Mitsubishi Electric Research Labs, HP y aportaciones de la Universidad de California en Santa Cruz y el MIT.



Sphinx 4 es una evolución de Sphinx 3 pero mucho más flexible, lo que le ha convertido en una herramienta mucho más útil.

Para la instalación de Sphinx 4 es necesario tener instalado Java, puesto que está basado en este, pero aparte es necesario instalar Ant 1.6.0 (o posterior) para la compilación de proyectos de gran envergadura.

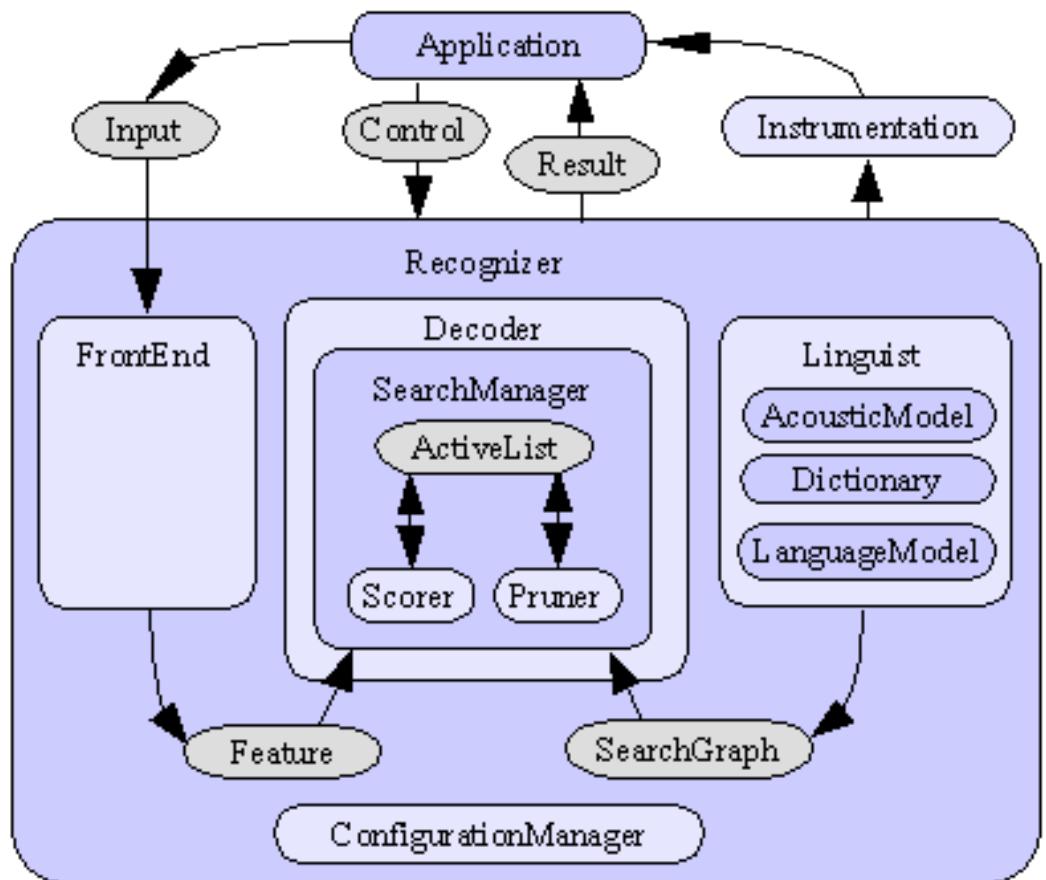
Al igual que FreeTTS existen diferentes paquetes para descargar con diferentes posibilidades:

- Bin: el paquete bin se utiliza para crear aplicaciones que utilicen reconocimiento de voz, sin la posibilidad de cambiar los archivos fuente de Sphinx.
- Src: este paquete contiene los archivos fuente para la modificación del código fuente de Sphinx.

*Para la realización del prototipo utilizamos el paquete Bin, puesto que solo se requiere la utilización de reconocimiento de voz, sin necesidad de cambiar el código fuente.

Una vez descargado el paquete, es necesario compilar el proyecto, pero antes es importante figurar la variable JSAPI, esto es simplemente descomprimir la librería jsapi.jar en el directorio lib. Una vez realizado esto, es hora de compilar el proyecto con el comando ant; se agruparán las clases de Sphinx-4 en la carpeta "bld", las librerías





Cuando el reconocedor se inicia, se construye el FrontEnd (la que genera las características de la voz), el Decoder, y el Linguist (que genera el gráfico de la búsqueda) de acuerdo con la configuración especificada por el usuario. Estos componentes, a su vez construyen sus propias subcomponentes. Por ejemplo, el Linguist construye el AcousticModel, el Dictionary, y el LanguageModel. Se utiliza la información contenida en los tres para construir un gráfico de búsqueda para ajustar las palabras. El decodificador construye el SearchManger, que a su vez construye el Scorer, el Pruner, y el ActiveList.



La mayoría de estos componentes representa interfaces. El SearchManager, Linguist, AcousticModel, Dictionary, LanguageModel, ArchiveList, Scorer, Pruner, y el SearchGraph son interfaces de Java. No puede haber diferentes implementaciones de estas interfaces. Por ejemplo, si existen dos implementaciones diferentes SearchManager, entonces, ¿cómo sabe el sistema cual utilizar?: El usuario lo especifica a través del archivo de configuración (Archivo XML del que se ha hablado anteriormente).

Spinhx-4 implementa un algoritmo de paso de testigo, esto significa, que para que no existan muchos caminos de búsqueda, cada vez que la búsqueda llega al siguiente estado en el gráfico, se crea un "token" de puntos al anterior y al siguiente. La ActiveList mantiene un seguimiento de todos los caminos de búsqueda activos a través del gráfico de búsqueda mediante el almacenamiento de la última muestra de cada ruta.

Cuando la aplicación solicita al Recognizer para realizar el reconocimiento, el SearchManger le pedirá al anotador anote cada token de la ActiveList contra el siguiente vector obtenido por el FrontEnd. Esto le da un valor a cada una de las rutas activas. El Pruner entonces "corta" los token. Cada camino que sobrevive, se extiende a los siguientes estados creando un token nuevo. El proceso se repite hasta que no se puedan



obtener más vectores del FrontEnd para anotar. Esto generalmente significa que no existen datos de entrada. Es en este momento cuando nos fijamos en todos los caminos que han llegado al estado final, y se devuelve la ruta cuyo Score (valor) es más alto.

Una vez entendido y configurado los ficheros de Sphinx 4, queda explicar su funcionamiento en java. Como se comentó con anterioridad, Java interactúa con Sphinx 4 a través del Recognizer, es más, el resultado escuchado es el valor devuelto del Recognizer. Los pasos para crear el método listen son:

- 1) Indicar mediante URL la ubicación del fichero XML de configuración al ConfigurationManager, este lee el fichero para poder aplicar las configuraciones al recognizer y micrófono.

```
cm = new ConfigurationManager  
(Navegar.class.getResource("voice.config.xml"));
```

- 2) A continuación se inicializa el Recognizer con las configuraciones antes leidas.

```
recognizer=(Recognizer)cm.lookup("recognizer");  
recognizer.allocate();
```

- 3) El siguiente paso es inicializar el micrófono para la escucha.

```
Microphone microphone = (Microphone)  
cm.lookup("microphone");
```



```
if (!microphone.startRecording()) {  
    System.out.println("Cannot start  
microphone.");  
    recognizer.deallocate();  
    System.exit(1);  
}
```

- 4) Por último solo queda escuchar, y ajustar la escucha a la palabra pre-definida mas próxima.

```
Result result = recognizer.recognize();  
  
if (result != null) {  
    resultText =  
result.getBestFinalResultNoFiller();  
}else {  
    System.out.println("I can't hear what  
you said.\n");  
}
```

1.4.3.7 FreeTTS

“Un sintetizador de voz escrito completamente en el lenguaje de programación Java”([16] FreeTTS está basado en Flite, que es un sistema sintetizador de voz a tiempo real desarrollado en Carnegie Mellon Univeristy, el cual, a su vez deriva de Festival, que a su vez deriva del proyecto Festival de la universidad de Edimburgo y el proyecto FestVox de la propia universidad de Carnegie Mellon.

FreeTTS fue desarrollado en los laboratorios de Sun Microsystems por el “Speech Integration Group”.

FreeTTS dispone de 4 soluciones utilizadas en el proyecto:



- Soporte para JSAPI (Java Speech API permite incorporar tecnología de voz en interfaces de usuario).
- Documentación API desarrollada (Imprescindible para su programación).
- Tutoriales y guías oficiales de nivel avanzado.

Como se comentó en la primera línea, FreeTTS está escrito totalmente en Java por lo que requiere tenerlo instalado. (Proceso que ya se realizó con anterioridad).

Existen tres paquetes de descarga dependiendo de lo que se vaya a realizar:

- Bin: Paquete para el desarrollo de aplicaciones sin realizar modificaciones a los archivos fuente. Incluye los archivos jar (las librerías), documentación y programas de prueba.
- Src: Paquete para realizar cambios en el propio sistema, es decir, cambiar los archivos fuente. Incluye archivos fuente para su modificación, documentación, y programas de prueba.
- Tst: Paquete de control para los cambios realizados en los archivos fuente. Contiene utilidades para probar y analizar errores, y bugs.



*El paquete seleccionado para la realización del prototipo ha sido el Bin, puesto que solo requerimos de un método say () para dictar la lista de productos.

Una vez se ha descargado el paquete bin, tan solo quedaría compilarlo utilizando ant como se explicó para leJOS y Sphinx. Tras su compilación, quedaría importar las librerías a nuestro proyecto, y ya se podría utilizar.

La utilización de FreeTT en el proyecto se basa en la creación de un método say, que recibe como parámetro un String y lo emite, para ello es necesario:

- 1) Inicializar FreeTTS en el sistema indicándole el tipo de voz a utilizar:

```
System.setProperty("freetts.voices",
    "com.sun.speech.freetts.en.us.cmu_us_kal.KevoiceDirectory");
```

- 2) Crear el Synthesizer que para crear una conexión con el audio, es decir, indicando el idioma, el modo y la forma de emisión de la voz entre otros.

- 3) Preparar el Synthesizer para emitir.

- 4) Emitir:

```
synthesizer.speakPlainText(frase, null);
synthesizer.waitEngineState(Synthesizer.QUEUE_EMPTY);
```



UNIVERSIDAD PONTIFICIA COMILLAS

Escuela Técnica Superior de Ingeniería (ICAI)

Robótica

Parte II ROBÓTICA



Capítulo 1 INTRODUCCIÓN A LA ROBÓTICA

1.1 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]

1.2 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 Vinci en 1495. El “robot” que Leonardo diseño tenía la capacidad de



Robótica

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 Čapek 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 Engelberger fundan la primera empresa de la historia dedicada exclusivamente a la fabricación de robots.

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



Robótica

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 minusválidas.

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”.



Robótica

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.

1975 – Víctor Scheinman desarrolla el Universal Manipulation Arm (Puma), cuyo uso es para robots industriales.

1977 – ASEA, una empresa de robots europea, ofrece dos tamaños de robot según su potencia eléctrica. Ambos robots utilizan un controlador de microprocesador para la programación y operación.

1979 – Se establece el Instituto de Robótica en la universidad de Carnegie Mellon. Posteriormente desarrollarán Sphinx4 y FreeTTS que serán usados en el proyecto.

1981 – Takeo Kanade construye el brazo de accionamiento directo. Es el primero en tener los motores instalados directamente en las



Robótica

articulaciones del brazo. Este desarrollo hace que sea más rápido y más preciso que los anteriores brazos.

1982 – Fanuc de Japón y General Motors crean una empresa conjunta: GM Fanuc, dedicada a la construcción de robots para Norte América.

1984 – Joseph Engelberger inicia la “Transición Tobótica” para desarrollar robots de servicio.

1986 – Lego Group y el Instituto de Tecnología de Massachusetts (MIT) firman un contrato por el cual, Lego financia las investigaciones de MIT a cambio de obtener ideas para sus productos.

- Honda comienza un programa de investigación robótica que comienza con la premisa de que el robot "deben coexistir y cooperar con los seres humanos, haciendo lo que una persona no puede hacer y por el cultivo de una nueva dimensión de la movilidad para beneficiar en última instancia, la sociedad".

1989 – Se presenta Genghis, un robot creado en el MIT capaz de caminar.

- El Dr. Seymour Papert se convierte en el profesor de Investigación del Aprendizaje de LEGO.



UNIVERSIDAD PONTIFICIA COMILLAS

Escuela Técnica Superior de Ingeniería (ICAI)

Robótica

1993 – Seiko Epson desarrolla un robot llamado Monsieur, el robot más pequeño del mundo, certificado por el Libro Guinness de los Récords.

1995 – Se crea Intuitive Surgical formada por Fred Moll, Younge Rob y John Freud. Con objetivo de diseñar y comercializar sistemas robóticos quirúrgicos.

1998 – LEGO lanza su primera invención robótica SystemTM 1.0. Lego renombra el producto a MINDSTORMS.

- Breazeal Cynthia comienza en el MIT a trabajar en el robot Kismet, que pueden imitar los estados emocionales de un bebé.

1999 – Lego lanza “The Robotics Discovery Set”.

2000 – Honda estrena un nuevo robot humanoide ASIMO.

- LEGO lanza el MINDSTORMS Robotics Invention System2.0.

2001 – LEGO lanza MINDSTORMS Ultimate Builder's Set.



1.3 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.



UNIVERSIDAD PONTIFICIA COMILLAS

Escuela Técnica Superior de Ingeniería (ICAI)

El Problema

Parte III EL PROBLEMA



Capítulo 1 INTRODUCCIÓN

1.1 PLANTEAMIENTO:

Hoy en día según la ONCE existen alrededor de 14.000 personas en España que crecen de la dependencia necesaria y suficiente para realizar una compra en un supermercado. Dentro de esta cifra solo se encuentran personas cuya visión les impide distinguir un producto de otro. A esta cifra cabría añadir toda aquella persona que por su condición no se puede valer por sí misma para realizar una compra en un supermercado.

Existen diversas soluciones actualmente, pero todas ellas implican la intervención de otra persona o empleado para; o bien seleccionar los productos e introducirlos en la cesta, o bien si ese proceso estuviera automatizado se requeriría a una persona que transporte los productos al domicilio de la persona.

Una de las soluciones actuales reside en Internet. Para las personas sin déficit visual existe la posibilidad de realizar la compra por internet, corriendo un riesgo menor, por ejemplo, comprar aguacates con la intención de cenar esta noche con ellos, y que vengan maduros por no haber podido ir en persona a seleccionarlos. Pero para las personas con



El Problema

déficit visual, la página web debe estar adaptada para personas ciegas o disponer de una línea de braille, que es un objeto que transforma la página web a lenguaje braille (económicamente al alcance de unos pocos).

Gracias a la solución robótica las personas con déficit visual se ven especialmente agradecidas, puesto que la primera solución práctica que no engloba la acción de otra persona u empleado para la compra de productos en un supermercado. Esto se debe a la interacción directa entre la persona y el robot, quien dotado con comunicación oral guía a la persona por el supermercado.

1.2 *TORTULEBOT*

TortuleBot es un robot de código abierto creado por willowgarage **¡Error! No se encuentra el origen de la referencia.** Gracias al sensor 3D que lleva incorporado, el robot es capaz de navegar evitando obstáculos, realizar un plano de una casa, reconocer personas en movimiento.



El Problema

1.2.1 SOFTWARE:

TurtleBot está creado bajo la idea de código abierto. El SDK (Software Development Kit) del robot, está basado en el sistema operativo para robots ROS (Robot Operating System [11]. Dentro del SDK de TurtleBot están integrados los driver del hardware, también cuenta con herramientas de desarrollo, que permiten al robot navegar autónomamente entre otras.

Las librerías utilizadas para el manejo de la visión 3D son OpenCV [12] y PCL [13] a parte de las que ofrece ROS.

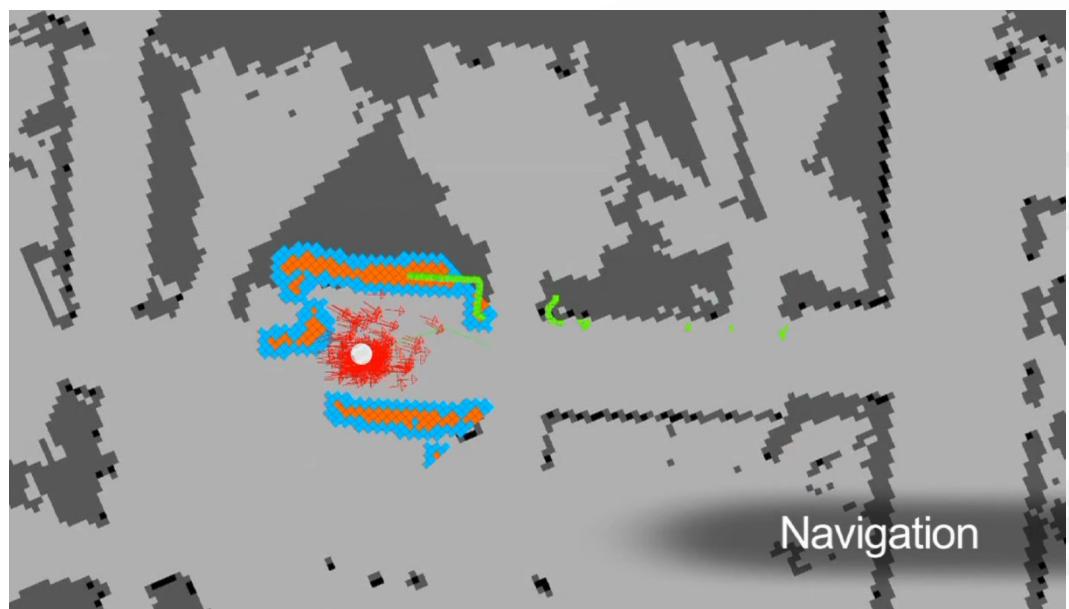


Ilustración 4: TortleBot vision. <http://www.youtube.com/watch?v=MOEjL8JDvd0>



El Problema

1.2.2 HARDWARE

TurtleBot se compone de 3 partes principales:

- A. iRobot Create: Es la base del robot, crea
- B. Microsoft Kinect: cámara y visión 3D.
- C. Netbook: El netbook utilizado oficialmente es el ASUS
1215N, pero cualquiera con la capacidad de procesar datos
de imágenes 3D es apto.



Ilustración 5: Hardware TurtleBot. [10]



1.2.3 CONCLUSIÓN

TurtleBot y éste proyecto son muy parecidos, es más, uno de los objetivos principales de este proyecto es similar: “Acercar la robótica a la sociedad a un precio asequible”. El precio del kit completo de TurtleBot es de 1200 dólares, mientras que el precio del robot de este proyecto ronda los 600 como veremos más adelante.

La diferencia de precio se debe a la diferencia de hardware, puesto que el software en ambos proyectos es de libre distribución. TurtleBot utiliza una estructura pensada y diseñada pieza a pieza para el robot, mientras que en este proyecto se crea una a parte de las piezas de LEGO. En cuanto a los sensores, TurtleBot utiliza Microsoft Kinect, cuyo precio ronda los 150 dólares, mientras que éste proyecto utiliza los sensores proporcionados por lego, que son suficientes para la creación de un prototipo, pero insuficientes para la implantación real.

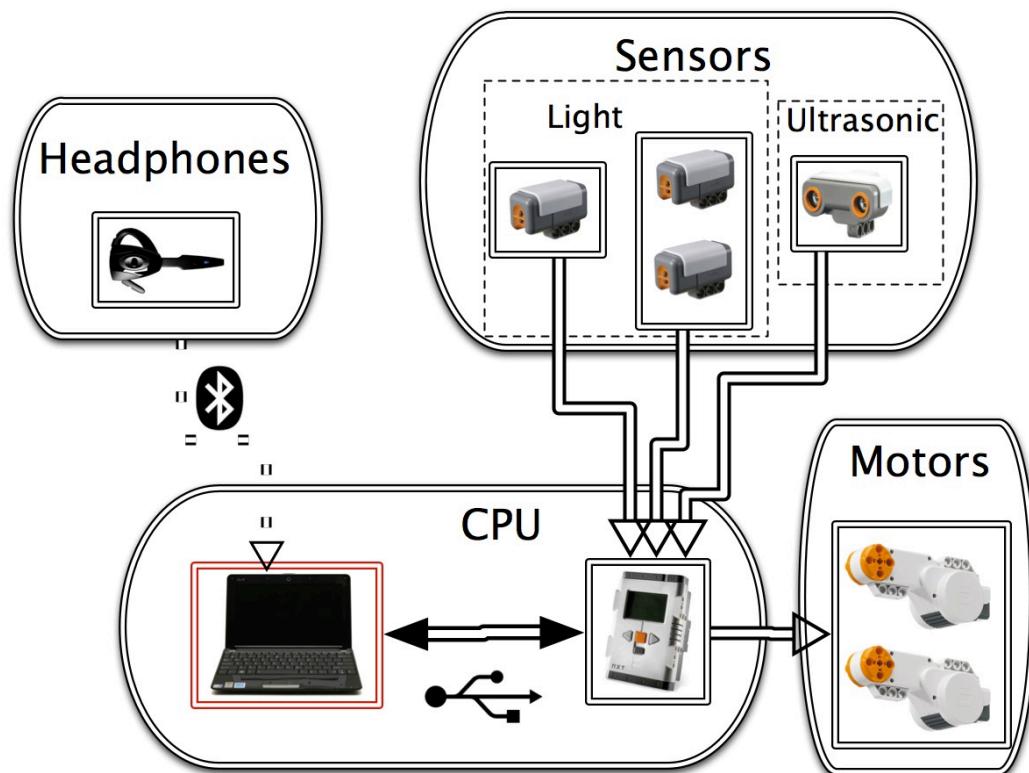
TurtleBot utiliza la tecnología en estado de arte, es decir, la tecnología de programación de robots más puntera que existe, mientras que la tecnología utilizada para este proyecto va un paso por detrás.



Capítulo 2 SOLUCIÓN

2.1 LA PLATAFORMA DEL ROBOT

La siguiente imagen, representa la estructura hardware, y su vinculación con el CPU.





El Problema

2.1.1 NETBOOK

El netbook Asus 1005HA es el netbook elegido por su relación calidad-precio. Con precio aproximado de 250 euros, el asus 1005HA nos ofrece las siguientes características:

- Procesador: Atom N280 a 1.66 GHz.
- Autonomía de hasta 10.5 horas.
- 2 GB de memoria RAM.
- 3 puertos USB.

Estas características se alejan mucho a las del Brick NXT, dándonos la posibilidad del tratamiento por voz. Como cabe de esperar, el netbook es el núcleo del CPU del robot y es quien se comunica a través de pc.comms (como veremos más adelante) con el NXT Brick para recibir las lecturas de los sensores y enviar las órdenes a los motores.



El Problema

Para comunicarse con el usuario, utiliza un micrófono-auricular conectado vía bluetooth. El reconocimiento de voz lo hace a través de Sphinx 4, y la emisión de voz la realiza a través de FreeTTS.

2.1.2 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 6: Ladrillo Inteligente del NXT.



Ilustración 6: Ladrillo Inteligente del NXT



UNIVERSIDAD PONTIFICIA COMILLAS

Escuela Técnica Superior de Ingeniería (ICAI)

El Problema

La “Tabla 1: Características NXT” está tomada de la página oficial de LEGO [3] 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.



El Problema

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 líneas, 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: Características NXT



El Problema

2.1.3 ULTRASONIC SENSOR:



Ilustración 7: 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 (viaja a unos 35cm por milisecondo). A diferencia de los otros sensores éste da el resultado en unidades de medida

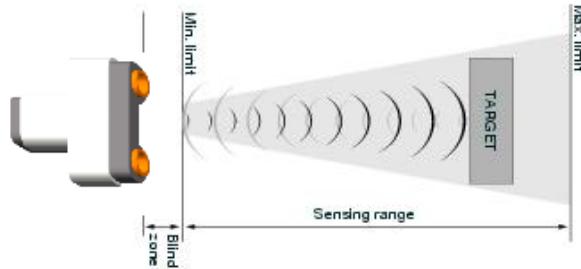


Ilustración 8: Funcionamiento del Sensor de Ultrasonido.



El Problema

El alcance máximo del sensor es de unos 170 cm aproximadamente. Si no se detecta eco de vuelta, el valor de retorno es 255.

El código está tomado de la página web de leJOS [4], 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.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 Problema

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 que guía al robot.

2.1.4 COLOR SENSOR:

Ilustración 9 : Sensor Óptico.

El sensor de luz mide la cantidad de luz que llega a una célula foteléctrica (a una resistencia). El sensor contiene un LED que emite una luz que se refleja y es captada por la resistencia, que es un fototransistor. La resistencia es baja con luz y alta con oscuridad . De acuerdo a [5], 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], abarca de 0 a 1000 lux.



El Problema

Es necesario calibrar el sensor para obtener los valores esperados.

El ejemplo siguiente proviene de la página oficial de leJOS [4] 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.drawString(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.



El Problema

Cómo el sensor de luz depende de la iluminación del lugar en el que se encuentre, es necesario calibrarlo antes de utilizarlo. La función de calibración consiste en tomar las mediciones de los máximos y mínimos rangos de blanco y negro, y guardarlos en una variable, es decir, tomar las mediciones del color blanco y negro en las condiciones de iluminación del lugar, como se aprecia a continuación:

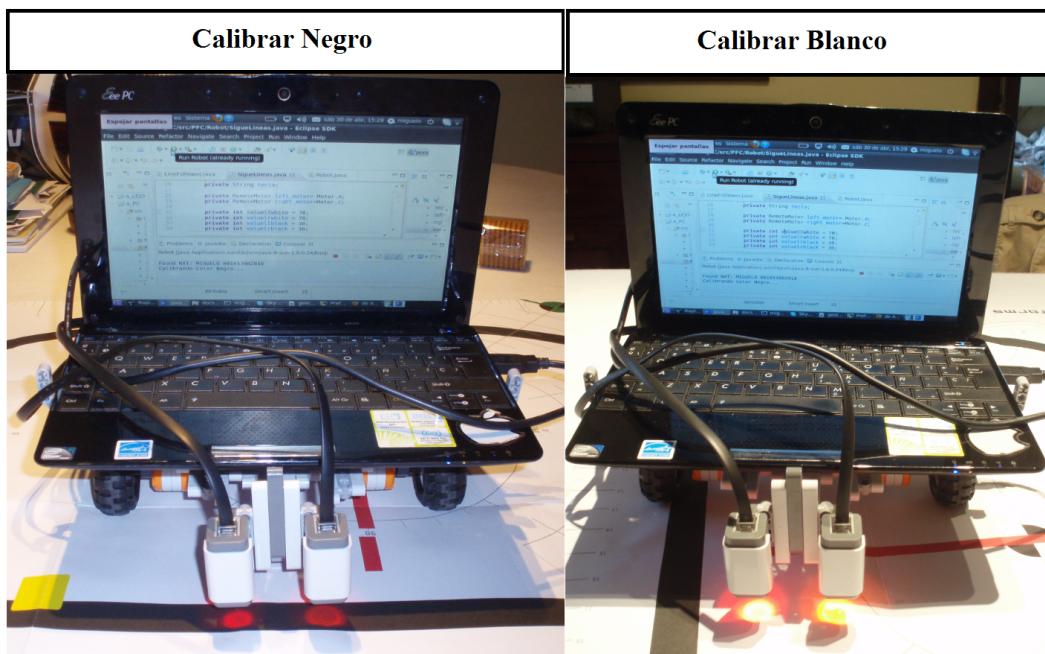


Ilustración 10: Calibración de Light Sensor

Para guardar el valor obtenido su utilizan los métodos `calibrateLow()` para el negro, y `calibrateHigh()` para el blanco. A la hora de leer las mediciones, consideramos que una medición es negra cuando es +- 10 putos del valor obtenido al calibrar.



El Problema

2.1.5 MOTORES:

La locomoción de un robot, es el modo en que éste se desplaza. Existen diferentes maneras de desplazamiento en robótica; piernas, ruedas brazos, aletas... De los diferentes modos de locomoción, el uso de las ruedas es el más estable, sencillo, y robusto para desplazarse en un terreno plano.

Los motores o actuadores de un robot son los que le dan movilidad. Existen diferentes tipos de actuadores; neumáticos, hidráulicos, motores eléctricos de corriente continua, motores servo... Los dos primeros son actuadores de gran fuerza y potencia, que se utilizan para maquinaria pesada, los motores servo se utilizan para movimientos con un grado de giro concreto, es decir, los servos generalmente giran 180 grados, porque



El Problema

no son útiles para una rueda. Para el proyecto se utilizan los motores eléctricos de Lego, cuyas características se verán a continuación.

De acuerdo a las pruebas realizadas en [6], 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], 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



El Problema

2.1.6 CHASIS

Para la construcción del chasis o estructura externa del robot, se escogió las que vienen por defecto con el paquete de Lego. Antes de tomar esta decisión se intentó crear una estructura con una caja de plástico. El cable a lego se decidió, porque con el paso del tiempo, los encajes de los motores con la caja holgaban.

La estructura del robot ha pasado por diferentes fases desde su inicio. Cuando comenzó el proyecto, se construyó una estructura parecida a la de un coche, con cuatro ruedas fijas, pero cuando se probó la estructura ocurrieron dos problemas:

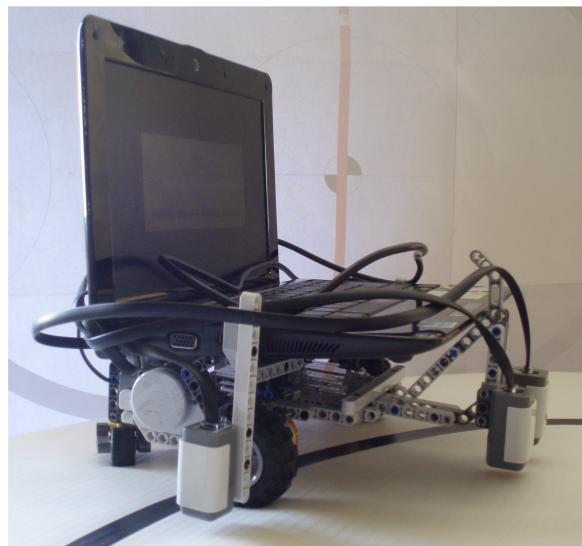
- 1) Las ruedas construidas soportaban el peso del notebook con facilidad, pero a la hora de girar, el robot tenía que arrastrar lateralmente las ruedas traseras, lo que suponía que de vez en cuando las ruedas traseras derraparan.
- 2) La distancia de separación de las ruedas motrices es demasiado corta, y produce inestabilidad.



El Problema



La segunda generación corregía las ruedas traseras, por unas de rotación. Las mismas que se utilizó para el primer linefollower. Y la distancia entre las ruedas se casi duplicaba. Con esto se consiguió más estabilidad, y evitar problemas a la hora de girar. La segunda generación plantaba plantea un problema que no se puede resolver con las piezas de lego: (Parte IV1.5).





El Problema

2.2 LA PLATAFORMA EN EL CENTRO COMERCIAL

La plataforma del centro comercial pretende simular la navegación a través de marcas en el suelo. Cada marca corresponde a un producto.

Las marcas de producto están grabadas en memoria y el robot conoce donde se encuentra cada producto.

Para desarrollar la plataforma para el centro comercial se utilizan 3 elementos:

- Cartulina de Colores.
- Mantel de color blanco.
- Cinta aislante de color negro.

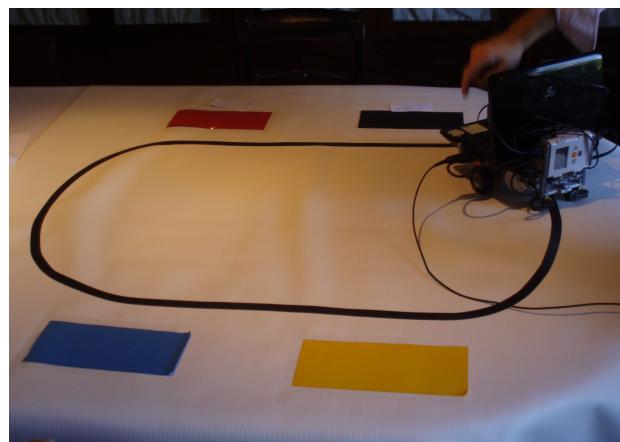




El Problema

Ilustración 11: Material para la Plataforma del Supermercado

Hasta las pruebas de sistema, las marcas de producto eran 4 colores claramente diferenciados, pero por los continuos problemas del sensor de productos (Parte IV1.8) se modificó a marcas de producto negras con un punto de partida del robot.

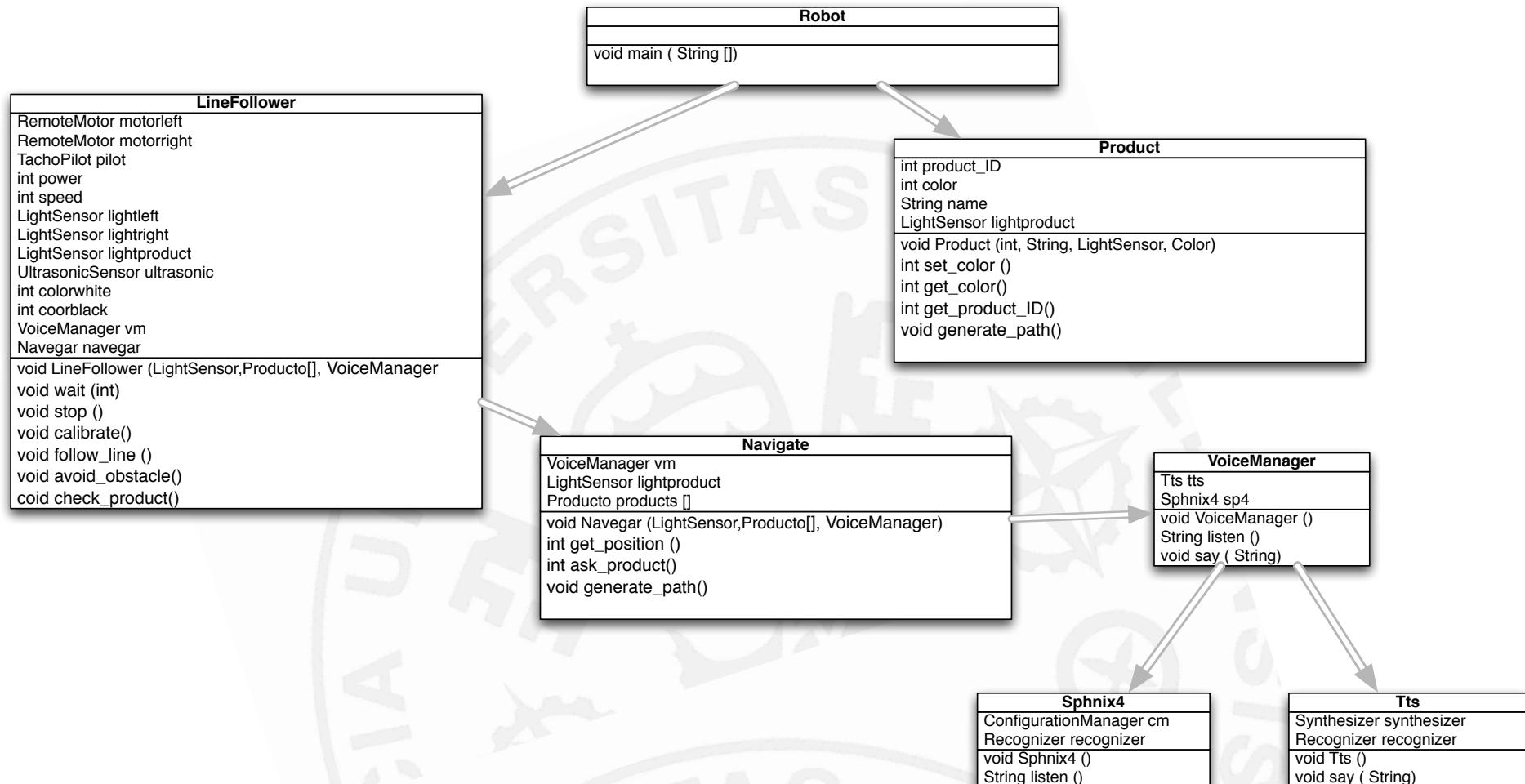


2.3 LA PLATAFORMA SOFTWARE

2.3.1 DIAGRAMA UML



El Problema





El Problema

2.3.2 DIVISIÓN DE CPU:

El CPU del robot se encuentra dividido entre el netbook y el NXT brick. Esto es necesario puesto que el NXT Brick no dispone de la capacidad de procesamiento necesaria para ejecutar el TTS y procesar la voz.

El netbook es el verdadero núcleo de procesamiento, en éste es donde se ejecuta el programa, es el encargado de la comunicación con la persona procesando la voz, es el encargado de almacenar la base de datos de productos, es quien recibe del NXT brick las lecturas de los sensores, y quien se encarga de enviar las órdenes al brick para mover los motores.

Las clases necesarias para establecer la conexión entre el netbook y el brick a través de USB se encuentran en el paquete **lejos.pc.comm**.

La conexión es lo primero que se establece al iniciar el programa que gestiona al robot.

```
NXTConnector conn = new NXTConnector();  
  
if (!conn.connectTo("MIGUELO", NXTComm.LCP)) {  
    System.err.println("Fallo en la Conexión");  
    System.exit(1);
```



El Problema

{}

```
NXTCommand.getSingleton().setNXTComm(conn.getNXTComm());
```

2.3.3 LINEFOLLOWER

Las complejidad de programar un sigue líneas es muy variada. Por un lado, es posible programarlo con tan solo 10 líneas de código y un sensor, como es el caso del programa original. Por otro lado, se hacen competiciones para ver que robot es capaz de recorrer un circuito a mayor velocidad, utilizando varios sensores.

El proyecto comenzó utilizando un sensor de luz para desarrollar la clase LineFollower. El programa utiliza dos estados para seguir la línea, el robot está constantemente girando hacia la derecha o hacia la izquierda, dependiendo de la posición de la línea, e independientemente de si se encuentra en una línea recta o una curva. Éste método es sencillo, pero los tiempos de recorrido son demasiado largos.

Por temas de velocidad y fiabilidad, se procedió a la instalación de un sensor de luz adicional, brindándonos la posibilidad de aplicar un algoritmo PID (Proporcional Integral Derivativo).

El algoritmo PID consiste en un bucle de control recursivo, que hace que la trayectoria del robot sea más fluida. Éste algoritmo requiere



El Problema

de tres parámetros previos; el proporcional (indica cuánto se ha alejado el robot de la línea, y en qué dirección), el integral (indica el error acumulado sobre el tiempo, es decir, si el robot ha estado en la línea, o no, durante los últimos instantes), y el derivativo (indica la velocidad a la que el robot oscila hacia derecha e izquierda de la línea). Estos valores hacen que el algoritmo PID sea complicado de aplicar, y requiera de tiempo y conocimientos estadísticos. Es apropiado aplicarlo para cosas en las que la velocidad, o la batería del robot son importantes.

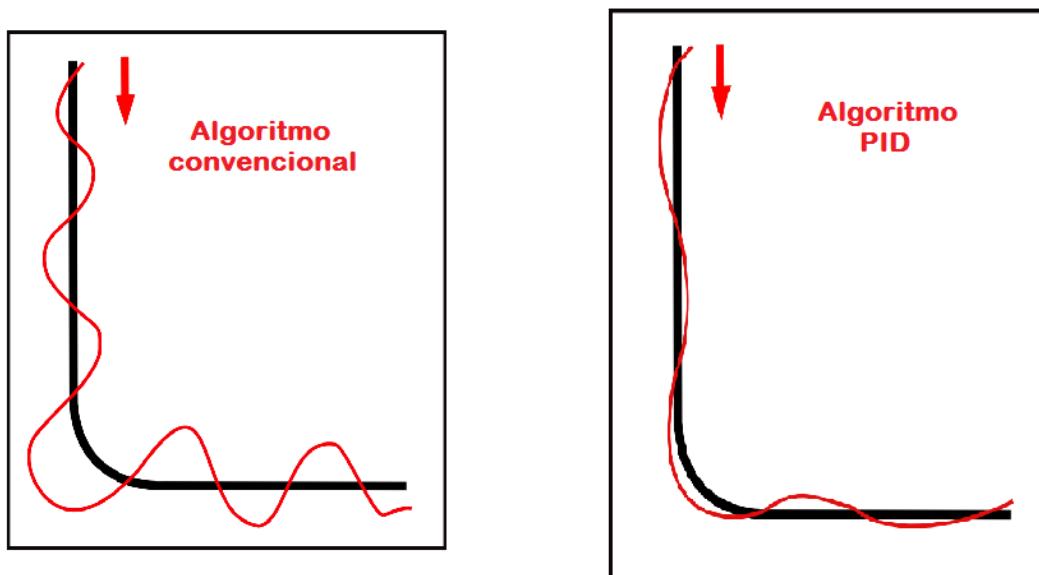


Ilustración 12: Comparación Algoritmo PID



El Problema

La implementación de un algoritmo PID en un robot lego NXT, fue el proyecto de Kevin McLeod (University of British Columbia). El informe se distribuye gratuitamente en su web personal. [14]

Para la implementación del prototipo se ha escogido un LineFollower de 3 estados, y dos sensores. El algoritmo establece en qué estado se encuentra dependiendo de las lecturas de los sensores de luz, es decir:

1. Si los dos sensores ven blanco, avanza.
2. Si el sensor izquierdo ve blanco, giro a la izquierda.
3. Si el sensor derecho ve negro, giro a la derecha.
4. Un cuarto estado es cuando no es ninguno de los anteriores, donde el programa lo que hace es avanzar un poco.

2.3.4 NAVEGACIÓN

La navegación de un robot móvil es la ciencia o arte de atravesar un entorno para alcanzar un destino sin chocar con ningún obstáculo, se puede resumir en responder a tres preguntas básicas, que son:

- ¿Dónde estoy?
- ¿Dónde quiero ir?



El Problema

- ¿Cómo voy?

Para resolver estas preguntas existen multitud de técnicas, que dependen de varios factores como el mapeado del entorno, es decir, si el robot conoce o no conoce el entorno donde se mueve y cómo lo conoce.

Existen dos formas de navegación claramente diferenciadas, que son; navegar conociendo el entorno, ó navegar sin conocer el entorno. En la primera, el robot conoce el entorno donde se mueve, y conoce la posición de los objetos fijos. En la segunda el robot ha de estar dotado de una gran capacidad para de detección de objetos que se refleja en los sensores, por que no conoce ni objetos fijos, ni móviles y tiene que interactuar con ellos para no chocar, en este caso la representación del entorno se hace de forma autónoma a medida que el robot va conociendo el entorno.

En la creación del prototipo, y en la creación robot, es necesario el conocimiento del entorno para el posicionamiento de los productos.

2.3.4.1 Representación del entorno

Existen dos tipos de representación de entorno:

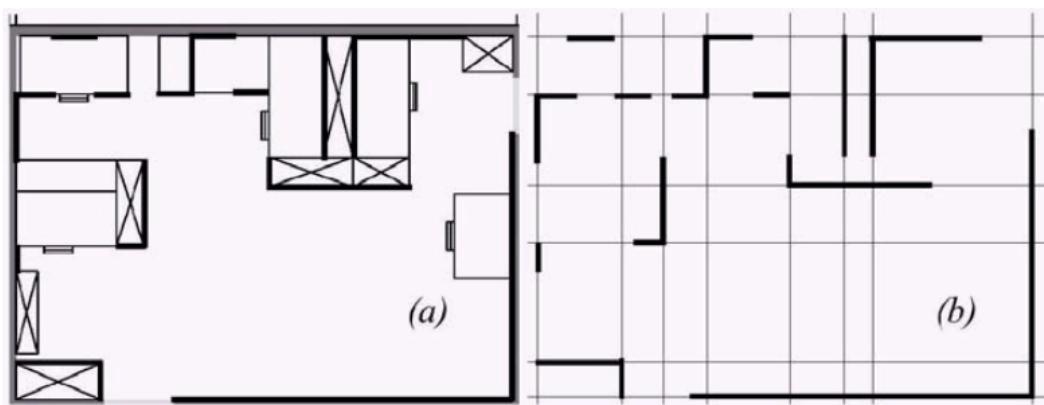
- Continuo: Representación de gran precisión que requiere una gran carga computacional.



El Problema

- Discreto: Representación simplificada del entorno.

La representación continua está basada en la arquitectura del edificio, representando al detalle los objetos fijos. Existen diferentes técnicas de representación en forma continua, por ejemplo:



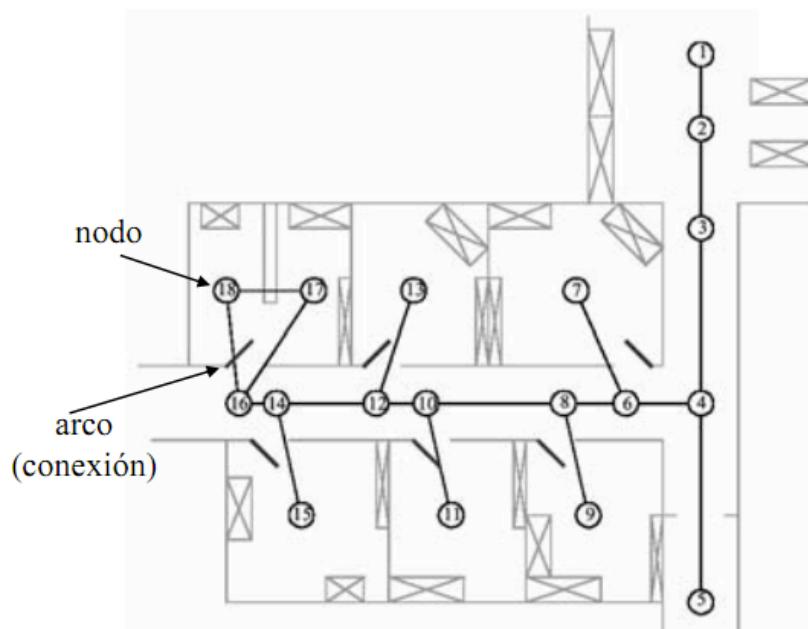
Para el prototipo y para la implementación del robot, se ha escogido una representación discreta, puesto que si el robot navega siguiendo una línea, que ha sido puesta exclusivamente para la movilidad del robot, cuando el sensor de ultrasonidos detecte un obstáculo que evita que el robot siga su línea, se sabe que es un objeto móvil y que es el objeto el que se quitará de la trayectoria.

Dentro de las representaciones discretas la más común es la representación topológica, que consiste en la representación de las características relevantes del entorno. La representación se realiza mediante Nodos (representan área de interés que en nuestro caso son las marcas de productos) y Arcos (representan la unión de los nodos que



El Problema

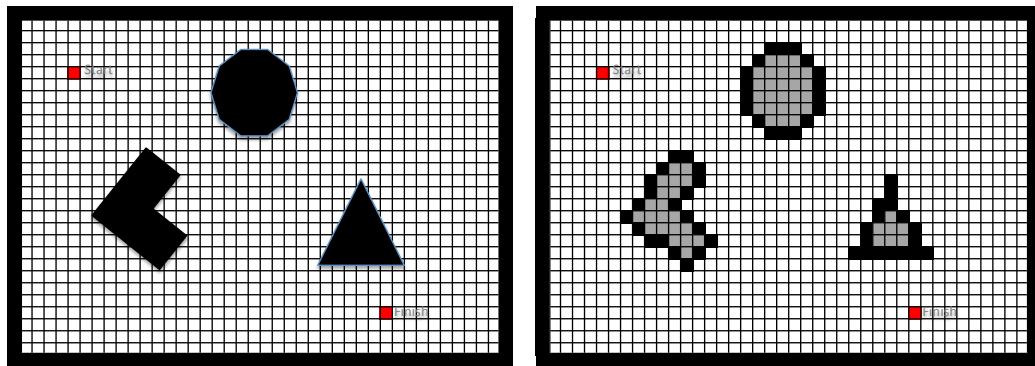
en nuestro caso es la línea negra que sigue el robot), en definitiva, la representación topológica se realiza mediante grafos.



Existen otro tipo de representaciones discretas como puede ser la representación por descomposición en celdas, o en polígonos cuyas funciones son representar los espacios vacíos por los que el robot puede navegar.



El Problema



El tipo de representación escogido es el topológico. De forma intuitiva se asocian los nodos y arcos, con los productos y su unión entre ellos por medio de líneas. Pero existe un factor más por el cual la representación topológica es la más adecuada; al haber dotado la superficie del supermercado con líneas negras para el guiado del robot evitamos la necesidad de que el robot tenga conciencia de objetos fijos , como estanterías , con los que pueda chocar, puesto que las líneas se presupone que no atraviesan estanterías. Este factor hace que solo sea necesario representar la posición de los productos y el modo de obtener la trayectoria más adecuada para acceder a ellos.

Una vez entendida la representación del entorno se puede pasar a responder las preguntas antes mencionadas:



El Problema

2.3.4.2 ¿Dónde estoy? – Posicionamiento

La localización o posicionamiento es el proceso por el cual el robot sabe su situación en dentro del entorno en el que se desplaza. La localización se puede clasificar de local y global. En el posicionamiento local se posiciona al robot dependiendo de su situación inicial, mientras que en la localización global se utiliza tecnología de redes para su posicionamiento.

El posicionamiento global es mucho más sofisticado y complejo que el posicionamiento local, en consecuencia, es más complejo. Para el prototipo se seleccionó posicionamiento local como premisa para Esmeralda 01, es decir, se podría haber escogido posicionamiento a través de wifi (a través de GPS no tiene sentido por la poca cobertura que obtendría el robot en el recinto cerrado), pero se escogió posicionamiento local para aprender y asentar bien los conocimientos de cara a su utilización con visión artificial en Esmeralda 01.

La localización local de un robot se puede resolver con diversas técnicas que se pueden clasificar en; determinísticas (basadas en marcas en la navegación) y probabilísticas.

La localización del robot se realiza de una forma determinística mediante el método `get_position` de la clase Navegación, utilizando las marcas de producto como marcas de navegación para el posicionamiento.



El Problema

En el prototipo se han identifican dos estados; estoy en una marca, no estoy en una marca. De esta forma como el robot sabe la colocación de los productos, cada vez que se topa con una marca de producto, añade un al contador de productos, para así saber que la siguiente marca que se encuentre corresponde al siguiente producto. El método `get_position` devuelve el identificador del producto donde se encuentra, o -1 si se encuentra entre medias de productos.

2.3.4.3 ¿Dónde quiero ir? – Objetivo

El prototipo para saber donde quiere ir, llama al método menú, que mediante la comunicación oral preguntará al individuo que producto desea adquirir. Una vez entendido el producto, se dispondrá a desplazarse.

2.3.4.4 ¿Cómo voy? – Generación de camino

La generación de trayectoria de un robot es un proceso que depende de innumerables factores, y tiene diferentes soluciones; para ir de un punto A a un punto B, se pueden escoger el camino más corto, el más seguro, dependiendo de las necesidades del usuario.

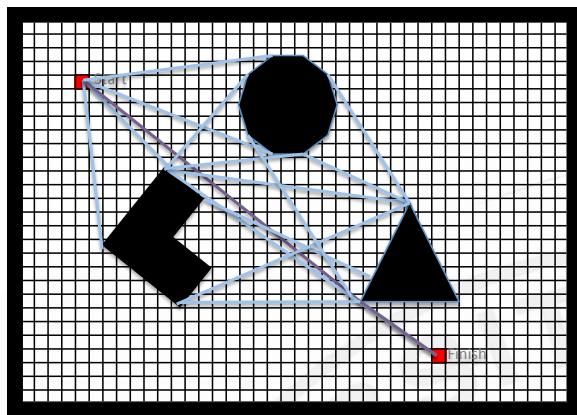
La puesta en escena de nuestro prototipo es un supermercado, donde el usuario está interesado en realizar la compra lo más rápido posible, por lo que el camino que estamos buscando es el camino más corto.



El Problema

Una vez definidas las necesidades del usuario hay que observar las diferentes opciones que tiene el robot de cara la generación de la trayectoria más corta(que coincide con ser las más rápida porque la velocidad es constante en todos los puntos).

Dependiendo de la representación del entorno escogida, la generación de trayectoria se hace dependiendo de los huecos libres y la visión del robot, o se hace siguiendo una línea con los algoritmos matemáticos de generación de caminos de grafos.



Gracias a las líneas de guiado con las que cuenta el terreno donde se mueve el robot, el problema de la generación de trayectoria se resume a un problema matemático que se puede resolver mediante los algoritmos matemáticos de grafos.



El Problema

En el prototipo se implantó Dijkxtra, pero como el entorno donde se mueve el robot, es lineal, se decidió sustituir Dijkxtra por una navegación lineal haciendo una pasada por los cuatro productos. No se contempla la opción de no encontrar la marca de producto, puesto que antes de iniciar la navegación, el robot se asegura de que el producto pedido se encuentra en stock.

```
DIJKSTRA (Grafo G, nodo_fuente s)
para u ∈ V[G] hacer
    distancia[u] = INFINITO
    padre[u] = NULL
    distancia[s] = 0
    Encolar (cola, grafo)
    mientras que cola no es vacía hacer
        u = extraer_minimo(cola)
        para v ∈ adyacencia[u] hacer
            si distancia[v] > distancia[u] + peso (u, v)
            hacer
                distancia[v] = distancia[u] + peso (u, v)
                padre[v] = u
                Actualizar(cola,distancia,v)
```

2.4 ESTUDIO ECONÓMICO

| Hardware | Cantidad | Precio |
|-----------------------|----------|--------|
| NXT Brick | 1 | 158.95 |
| NXT Ultrasonic Sensor | 1 | 32.95 |



El Problema

| | | |
|----------------------|--------|------------------|
| NXT Color Sensors | 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 ⁴ |

Software

| | |
|--------------|----------------------|
| Ubuntu 10.04 | Gratis ⁵ |
| Java | Gratis ⁶ |
| leJOS | Gratis ⁷ |
| Eclipse | Gratis ⁸ |
| SVN | Gratis ⁹ |
| FreeTTS | Gratis ¹⁰ |
| Sphinx 4 | Gratis ¹¹ |

Al utilizar software de código abierto, evitamos tener que comprar licencias a un alto precio. Esto mismo nos ofrece la posibilidad de

⁴ Existen descuentos para estudiantes, centros escolares y universidades.

⁵ <http://www.ubuntu.com/>

⁶ <http://www.java.com/es/download/>

⁷ <http://lejos.sourceforge.net/>

⁸ <http://www.eclipse.org/>

⁹ <http://rapidsvn.tigris.org/> - Linux.

¹⁰ <http://freetts.sourceforge.net/docs/index.php>

¹¹ <http://cmusphinx.sourceforge.net/sphinx4/>



UNIVERSIDAD PONTIFICIA COMILLAS

Escuela Técnica Superior de Ingeniería (ICAI)

El Problema

disponer del software necesario para desarrollar el proyecto tanto en casa como en la Universidad, Centro Escolar, o donde corresponda.



Capítulo 3 VALORACIÓN FINAL

3.1 CONCLUSIONES

Robonce, que es el nombre del prototipo, ha sido un éxito en diversos aspectos:

3.1.1 CONCLUSIONES TECNOLOGICAS

Robonce es el primer robot publicado que combina tecnología leJOS con comunicación a través de reconocimiento de voz, esto ha provocado que también se haya convertido en el primer robot que utiliza leJOS y un netbook integrado como centro de procesamiento.

La necesidad de integrar el notebook en el robot se debe a la impotencia del Brick NXT para procesar el reconocimiento de voz, y la necesidad transmitir datos entre el Brick NXT y el notebook alta velocidad, quedándose corta la velocidad de transmisión a través de comunicación Bluetooth.



El Problema

En este mismo terreno de innovación tecnológica, cabe destacar que la tecnología utilizada para la emisión y reconocimiento de voz, son tecnologías en estado-de-arte (Sphinx4 y FreeTTS).

3.1.1 OBJETIVOS

De cara al cumplimiento de los objetivos iniciales, se puede afirmar que robonce ha superado con creces los objetivos iniciales.

El primer objetivo establecido fue alcanzar una solución al problema de implantación, que se ha conseguido resolver mediante el diseño e implantación de un robot con las cualidades de comunicación y desplazamiento requeridas.

En segundo lugar, partiendo de los datos de previsión de ventas publicados por la IFR (International Federation of Robotics), se estableció como objetivo fomentar la robótica al servicio de la sociedad. Existen dos hechos que demuestran el cumplimiento de este objetivo:

- Por un lado, gracias a la búsqueda de información realizada y a las publicaciones en pequeños foros y páginas web, el proyecto atrajo la atención de uno de los desarrolladores de leJOS, quien tras unas reuniones accedió a codirigir el proyecto junto a José Miguel Ordax cassá.



El Problema

- Por otro lado, el proyecto ha servido como punto de partida para que el departamentito de robótica del instituto Antonio Machado, inicie el desarrollo de Esmeralda01, un robot que cuya tecnología es 100% innovadora. Véase trabajos futuros.

3.2 ASPECTOS NEGATIVOS

A pesar de haber adquirido avanzados conocimientos de robótica para el desarrollo del proyecto, Robonce utiliza un brick cuyas capacidades de proceso fueron insuficientes para gestionar comunicación por voz, por lo que se dispuso a sustituir el CPU a un notebook incorporado en el robot.

Existe una barrera lingüística a la hora de acceder a documentos lingüísticos,

La documentación de robótica se encuentra en un 90% en inglés, lo que plantea una barrera lingüística a la hora de documentarse. Gracias a ello, he adquirido un gran conocimiento de tecnicismos en inglés.

Sustituir el centro de procesamiento del NXT Brick al notebook, hizo necesaria su implantación el robot debilitando la estructura, pero



El Problema

desde otro punto de vista, el procesamiento a través del notebook aumenta notablemente las posibilidades del robot, haciendo posible la implantación de ROS (Robot Operating System) en un futuro.

3.3 TRABAJOS FUTUROS

Robonce es un prototipo desarrollado para adquirir los conocimientos de robótica necesarios para introducirse a investigar en las tecnologías en estado de arte. Esmeralda01 es la evolución de robonce, que utiliza tecnología todavía inexplorada como ROS, y que utiliza un sensor 3D desarrollado por Microsoft con fines de ocio; Kinect.

3.3.1 MONTECARLO

El método de localización de Monte Carlo es un método para determinar la posición del robot en un entorno predefinido mediante el uso de la estadística.

Este método evalúa un gran número de configuraciones hipotéticas que son inicialmente dispersas al azar dentro del espacio de predefinido. Con cada actualización del sensor, la probabilidad de que cada una de las



El Problema

configuraciones hipotéticas sea correcta se actualizada basándose en el método de Bayes.

Según [18] se tiene una población limitada de muestras que representan las posibles posiciones del robot. Las muestras se generan en cada instante a partir de las del instante anterior junto con las mediciones del entorno y las acciones ejecutadas por el robot (desplazamiento). Inicialmente, el conjunto de muestras se distribuye uniformemente de forma aleatoria. Posteriormente, en cada instante de tiempo, se siguen tres pasos recursivamente:

- Predicción: se trasladan las muestras según la acción ejecutada y se estima la nueva posición de las muestras.
- Actualización: se calculan las probabilidades a posteriori de cada muestra, dada la última observación sensorial.
- Remuestreo: se genera un nuevo conjunto de muestras con distribución proporcional a la verosimilitud de las muestras anteriores.



El Problema

3.3.2 ROS (ROBOT OPERATING SYSTEM)

ROS (Robot Operating System) es un sistema operativo usado para desarrollar y controlar sistemas para robots que proporciona las bibliotecas y herramientas su desarrollo. Proporciona abstracción de hardware (la principal razón de su éxito), drivers de dispositivo, bibliotecas, visualizadores, paso de mensajes, gestión de paquetes, y mucho más. ROS se distribuye de forma gratuita, y su código es abierto y libre para su utilización , modificación, e incluso comercialización.

El objetivo principal es permitir a los desarrolladores de software crear robots con más capacidades de una forma rápida y sencilla, utilizando una plataforma común.

ROS comenzó a desarrollarse en el laboratorio de Inteligencia Artificial de Stanford en 2007, a continuación fue Willow Garage quien tomo el relevo, y junto con una veintena de instituciones más, continúan su desarrollo.

Una de las razones por la que ROS está tan liberado reside en que Wollo Garage está fuertemente comprometida con el desarrollo de código abierto y software reutilizable. Es más, Willow Garage tiene publicado en la web code.ros.org dos bibliotecas; una de investigación y otra donde suben los proyectos en los que trabajan en la actualidad.



El Problema

Es posible afirmar que ROS, a pesar de ser muy parecido a otros firmwares de robótica, es el futuro. La diferencia de ROS sobre los diferentes firmwares que se pueden encontrar en el mercado, es su compatibilidad de código y su objetivo de reutilización de código, se puede ver reflejado en un e-mail de Brian Gerkey donde se reflejado que la facilidad de ROS viene en utilizar código ya escrito, en vez de reescribir código. El e-mail compara Player y ROS:

"The answer, as usual, depends. In particular, it depends on what you're trying to do. Player is a great fit for simple, non-articulated mobile platforms. It was designed to provide easy access to sensors and motors on laser-equipped Pioneers. ROS, on the other hand, is designed around complex mobile manipulation platforms, with actuated sensing (tilting lasers, pan/tilt sensor heads, sensors attached to arms). As compared to Player, ROS makes it easier to take advantage of a distributed computing environment, and I would say that the higher-level side of things is more developed in ROS than in Player. Whereas Player offers more hardware drivers, ROS offers more implementations of algorithms. I think that's it's fair to say that ROS is more powerful and flexible than Player, but, as usual, greater power and flexibility come at the cost of greater complexity. While we're working hard to make ROS easy to use, there is still a significant learning curve. Of course, familiarity with Player should help in



El Problema

learning to use ROS, as many of the underlying concepts are similar. As to your specific question regarding OpenCV integration, I think that you'll find quite a bit more ROS code than Player code that uses OpenCV in interesting ways. In the future, you should expect to see even more, as there's significant overlap between the ROS and OpenCV development teams. I should note that ROS leverages a lot of code from the Player project. There are ROS nodes that reuse code from many Player drivers, and both Stage and Gazebo are well-supported and widely used in the ROS community."

ROS actualmente se puede programar en Python y esencialmente en C/C++, pero hoy día 11 de Mayo de 2011 ha salido la implementación nativa de ROS en Java.

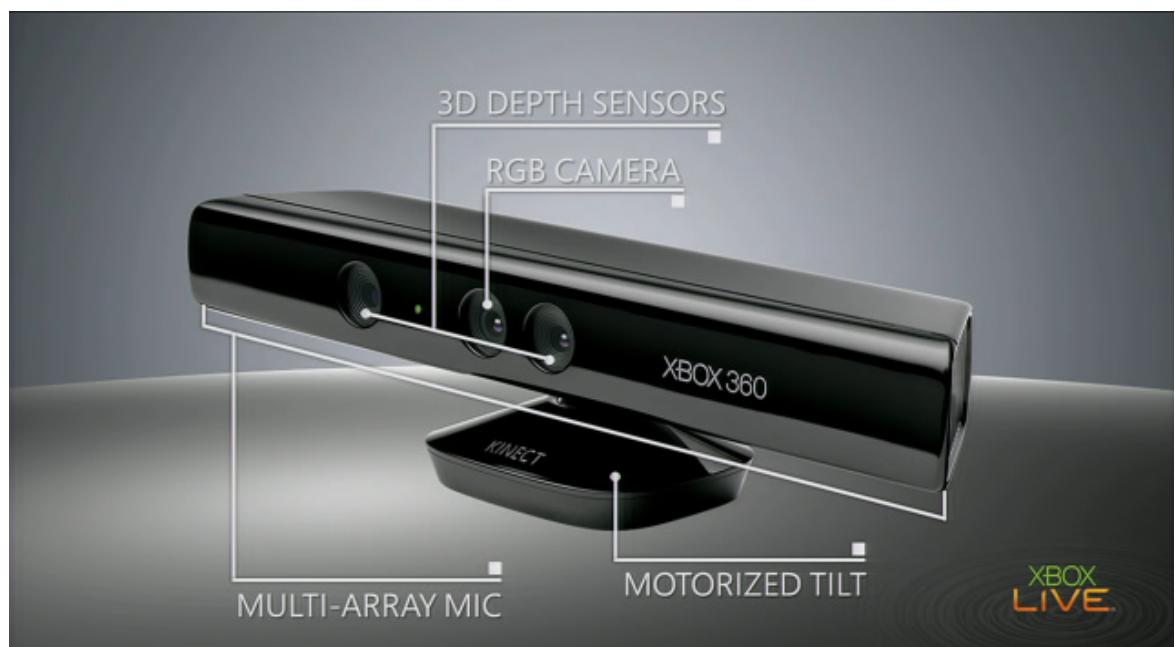
3.3.3 MICROSOFT KINECT

Microsoft Kinect es un sensor de movimiento diseñado por la empresa Microsoft con fines de ocio. Salió a la venta el 14 de Junio de 2010 en los Estados Unidos de América, y desde entonces se ha convertido para la robótica en un sensor de movimiento de bajo coste, sustituyendo caros sensores laser.



El Problema

La gran utilidad de Kinect es que es un todo en uno, es decir, posee todos los sensores necesarios para navegar e interactuar con un usuario:



En la imagen se pueden apreciar los diferentes sensores que posee; Los sensores de posicionamiento de objetos en movimiento en 3D y cámara RGB (a color) hacen posible la visión artificial, y el micrófono multi-vectorial permite el reconocimiento de voz.

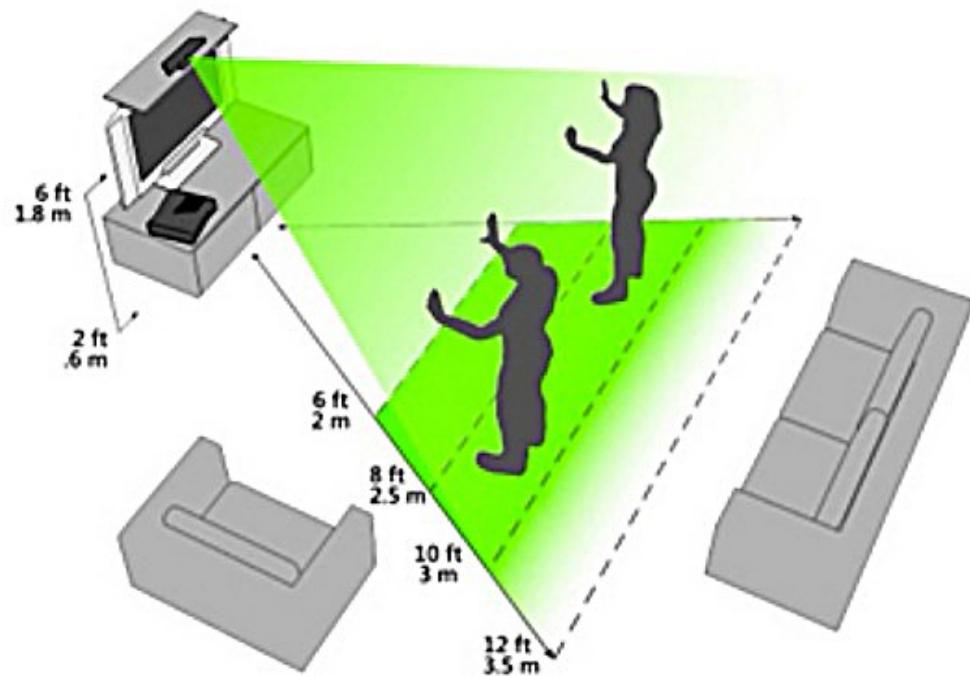
Detalles:

- Microsoft Kinect tiene un campo de visión de 57 grados horizontales y 43 grados verticales, y una profundidad de entre 1,2 y 3,5 metros.
- La resolución de captura es de 320x240.



El Problema

- Es capaz de localizar 20 puntos por persona y hasta un máximo de 6 personas simultáneamente.
- La velocidad de transmisión de datos es de 16 bits y 23 bits a 30 fps para el vídeo, y de 16 bits a 16 kHz para el audio.





Capítulo 4 BIBLIOGRAFÍA

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



El Problema

- [10] Williwigarage. (2010). *TurtleBot*. Obtenido de
<http://www.willowgarage.com/turtlebot>
- [11] ROS. (16 de 03 de 2011). *Robot Operating System*. Obtenido de
<http://www.ros.org/wiki/>
- [12] OpenCV (01/12/2010). Open Source Computer Vision. Obtenido de
de HYPERLINK "<http://opencv.willowgarage.com/wiki/>"
- [13] PLC (01/12/2010). Point Cloud Library. Obtenido de
<http://pointclouds.org/>
- [14] April Robot, Web de Kevin MCL. <http://www.physics.ubc.ca/kevinmcl/projects/lego/APRIL/>
- [15] Sphinx 4 . Carnegie Mellon University. Recuperado el 18 de Febrero de 2011 de <http://cmusphinx.sourceforge.net/sphinx4/>.
- [16] FreeTTS . Carnegie Mellon University. Recuperado el 18 de Febrero de 2011 de <http://freetts.sourceforge.net/docs/index.php>.
- [17] Master en Robótica, Universidad de Alcalá. José María Cañas Plaza – web: http://www.robotica-urjc.es/index.php/Doctorado_-M%C3%A1ster#Teor.C3.ADA
- [18] Sáez, J.M., Escolano, F.: Localización global en mapas 3D basada en filtros de partículas. Marzo 2002.



UNIVERSIDAD PONTIFICIA COMILLAS

Escuela Técnica Superior de Ingeniería (ICAI)

Anexos

Parte IV ANEXOS



ANEXO A: LISTA DE PROBLEMAS

1.1 PROBLEMA 1: INSTALAR JAVA EN UBUNTU

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

Solución:

Abrir `/etc/apt/source.list`

Borrar los repositorios del final.

Actualizar.

Continuar la instalación.

1.2 PROBLEMA 2: SAM-BA MODE

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

Solución:

`sudo modprobe cdc_acm`



Anexos

Resetear Shell

```
sudo rmmod cdc_acm
```

```
sudo -s
```

```
echo "" >> /etc/modprobe.d/blacklist.conf
```

```
echo "blacklist cdc_adm" >> /etc/modprobe.d/blacklist.conf
```

1.3 PROBLEMA 3: CONEXIÓN USB

Problema: Problemas al enlazar con la biblioteca libjlibnxt.so

Solución: Introducir la biblioteca en /usr/lib/.

1.4 PROBLEMA 4: SENSOR DE LUZ

Problema: Incompatibilidad del sensor 64892 (ColorLightSensor).

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

1.5 PROBLEMA 5: ESTRUCTURA DEL ROBOT

Problema: Piezas débiles.



Anexos

Solución: Este problema no tiene una solución buena utilizando piezas del kit de estudiantes de lego. Para mejorar la estabilidad y robustez se cambiaron las ruedas delanteras por unas de más diámetro, ganando altura. Por otro lado, se cambiaron las ruedas traseras por las del LineFollower original, ganando estabilidad.

1.6 PROBLEMA 6: SENSOR DE MARCAS DE PRODUCTO

I

Problema: Las lecturas de los diferentes colores para identificar los productos son muy parecidas, y el lector se confunde con facilidad.

Solución: Para encontrar una solución al problema, realice un experimento que consistía en realizar medidas con un sensor y mismas condiciones de luminosidad para encontrar los colores con mayor diferencia numérica

- Negro 49
- Marrón 56-57
- Azul 58-59
- Verde 61



Anexos

- Rojo 64-65
- Rosa 65-66
- Azul Claro 67
- Amarillo 67
- Orange 69-70
- Blanco 70

Como cabía de esperar, los colores más diferentes entre sí fueron los colores primarios, pero el supermercado está diseñado para 4 productos, por lo que se procedió a añadir un tercer color; se escogió el negro, puesto que es el más diferente al fondo.

1.7 PROBLEMA 7: SENSOR DE MARCAS DE PRODUCTO

II

Problema: El sensor de productos sigue fallando a pesar del estudio de los colores realizado.

Solución: Este problema NO tiene una solución que no sea cambiar el sensor, pero por problemas de presupuesto, y de versión de firmware se pudo adquirir otro nuevo, por lo que se procedió el sistema de localización por uno basado en posicionamiento, es decir, en vez de



Anexos

utilizar diferentes colores, se utiliza marcas de producto de un solo color: negro. Gracias a un sistema inicial de 4 estados, y partiendo de una posición inicial, se sabe en qué posición estamos en cada momento:

```
Si ( pos_actual==blanco && pos_anterior==blanco)
    Nos encontramos fuera de zona de productos.
Sino
    Si ( pos_actual==blanco && pos_anterior==negro)
        Hemos salido de zona de producto
    Sino
        Si ( pos_actual==negro && pos_anterior==negro)
            Nos encontramos dentro de zona de producto.
    Sino
        Si ( pos_actual==negro && pos_anterior==blanco)
            Hemos encontrado una nueva zona de
            producto, se procede a añadir uno al
            contador de productos cuya misión es contar
            en qué producto estamos.
```

Está solución a simple vista parece lógica, y parece que no falla, puesto que si la calibración del linefollower funciona perfectamente, también debería funcionar para las marcas de producto, pero como veremos en el siguiente problema, no funcionaba bien.



Anexos

1.8 PROBLEMA 8: SENSOR DE MARCAS DE PRODUCTO

III

Problema: El nuevo sistema de 4 estados no es fiable, porque hay dos estados que falla, es decir, el sensor de luz toma las medidas demasiado rápido, y anterior y siguiente pasan por valores que no corresponden ni a Negro, ni a Blanco.

Según un programa que imprime los valores que lee mientras pasa del blanco al negro, siendo blanco 80 o más, y negro 20 o menos:

104, 103, 102, 101, 100, 100, 97, 97, 95, 95, 94, 92, 90, 88, 86, 85,
83, 82, 82, 81, 81, 83, 86, 88, 87, 83, 80, 77, 76, 74, 72, 69, 67, 66, 66,
66, 64, 61, 58, 54, 50, 47, 42, 42, 40, 38, 36, 35, 33, 32, 30, 29, 27, 26,
26, 23, 23, 22, 19, 17, 16, 16, 14, 13, 12, 11, 10, 8, 8, 7, 7, 6, 5, 5, 5, 4, 4,
4, 4, 4, 4, 3, 3, 3, 3, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1,
1, 1, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 2, 2, 1, 1.

Como se puede apreciar, en el momento en que la variable actual se vuelve negro, es cuando es igual a 19, en ese momento el valor de anterior es 22, por lo que cumpliría la condición Si (pos_actual==negro && pos_anterior==blanco).



Anexos

Solución: La solución fue crear un sistema de dos estados, donde identifique producto, o no producto:

```
actual=light.readValue();

        if(actual>blanco && anterior>blanco){
            System.out.println("actual: blanco- Anterior:
blanco");
            sw2=true;

        }
        else if(actual>blanco && anterior!=blanco){
            System.out.println("actual: blanco- Anterior:
negro");

        }
        else if(actual!=blanco && anterior!=blanco){
            if(sw2==true){
                posicion++;
                sw2=false;
            }
            System.out.println("actual: negro- Anterior:
negro");

        }
        else if(actual!=blanco && anterior>blanco){
            System.out.println("actual: negro- Anterior:
blanco");
        }
        else
            System.out.println("OTRO");

        System.out.println("Posición: " +posicion);
        anterior=actual;
```



UNIVERSIDAD PONTIFICIA COMILLAS

Escuela Técnica Superior de Ingeniería (ICAI)

Anexos

Capítulo 2 ANEXO B: PRUEBAS



Anexos

ANEXO C: ARCHIVO DE CONFIGURACIÓN

SPHINX 4

El archivo de configuración de Sphix 4 define

- El nombre de los tipos de todos los componentes del sistema.
- La conectividad de dichos componentes (la comunicación entre los componenetes)
- Los detalles de configuración de cada componente.

A continuación se explicaran los componentes del fichero de configuración utilizados en la realización del prototipo:

1) Properties:

Las primeras líneas contienen las inicializaciones de los componentes, se suelen encontrar al comienzo del fichero para que sean fácilmente modificables.

2) Recognizer:

El recognizer es el componente que como su nombre indica, se encarga del reconocimiento de voz. Se define el nombre y la clase del reconocedor, esta es la clase con la que la aplicación debe interactuar.



Anexos

Contiene dos propiedades; Decoder y Monitor, que se verán a continuación.

3) Decoder:

El Decoder es una propiedad del recognizer, su misión la descodificación de la voz. Su propiedad principal es searchManger, esta propiedad esta diseñada para reconocer una lista de palabras pequeñas. Para otros tipo de reconocimiento como matemáticos, existen otras tipos de propiedades como logMath. Este es el proceso mas costoso, en términos informáticos, del sistema.

4) Linguist

Este es un componente dentro del SearchManger que se encarga de generar el gráfico de búsqueda usando el archivo de palabras predefinidas y el conocimiento de acústica y diccionarios. Los modelos utilizados para Linguist están definidos en JSAPI.

5) Acoustic Model:

Esta propiedad es parte de Linguist, y define el modelo acústico, es decir, definir la acústica del lenguaje.

6) Front End

El Front End define la frecuencia de emisión de voz, es decir, si va a ser una comunicación de tipo batch donde no hay un punto donde se



UNIVERSIDAD PONTIFICIA COMILLAS

Escuela Técnica Superior de Ingeniería (ICAI)

Anexos

termina, o comunicación “Live” con punto donde se termina. En el proyecto se ha utilizado epFrontEnd que corresponde al modo live.

7) Instrumentation

La Instrumentación es parte del reconocedor de voz, su misión es el seguimiento de la precisión, velocidad y la gestión de la memoria.



Anexos

ANEXO D: PUBLICACIÓN DEL PROYECTO

Con el propósito de fomentar la robótica, se ha creado una página web con el contenido del proyecto, cuya publicación esta pensada en verano de 2011.

The screenshot shows a web browser window titled 'Untitled-1'. The address bar displays the URL: file:///localhost/Users/Miguelo/Library/Caches/TemporaryItems/Untitled-1.html. The main content of the page is a dark-themed website for 'Robonce'. At the top, the word 'Robonce' is written in a large, bold, white sans-serif font. Below it, in a smaller white font, is the name 'Miguel Ruiz Nogués'. To the left, there is a sidebar labeled 'Menú' containing buttons for 'Resumen', 'Objetivos', 'Software', 'Hardware', 'Media', 'Código', and 'Descargas'. To the right of the menu, there is a photograph of a black and silver robotic device with wheels, which appears to be a mobile robot. At the bottom of the page, there is a block of text in Spanish describing the project's purpose and target audience.

La robótica es una combinación de ciencia y tecnología en vías de desarrollo que realiza diferentes tareas para ayudar al ser humano. Éste proyecto pretende *construir un prototipo robot capaz de ayudar y guiar a personas con discapacidad visual o móvil en diferentes entornos* y a un bajo coste para que sirva de ejemplo para centros escolares y universidades.



UNIVERSIDAD PONTIFICIA COMILLAS

Escuela Técnica Superior de Ingeniería (ICAI)

Anexos

La publicación de la página se realiza de acuerdo con los objetivos de fomentar la robótica. La publicación de la página web pretende ser la puerta para los nuevos proyectos como Esmeralda 01.

El código y todos los detalles del robot quedarán a disposición de cualquier persona que tenga interés, también se subirán manuales de instalación y utilización de hardware software, así como una introducción a ROS.