Universidad Carlos iii de Madrid

Escuela Politécnica Superior

Ingeniería Industrial

PROYECTO FIN DE CARRERA

ESTUDIO SOBRE LA RESPUESTA DE LEGO MINDSTORMS FRENTE A DIFERENTES LENGUAJES DE PROGRAMACIÓN

Autor: Pablo Pérez de Ciriza Villa
Tutor: Jorge García Bueno

Febrero 2012

A mi padre, que habría disfrutado aún más que yo con este proyecto.

Agradecimientos

A mi madre y mi famila por apoyarme incondicionalmente.

A mi tutor, Jorge García Bueno, por ofrecerme la posibilidad de realizar un proyecto muy entretenido además de enriquecedor.

A Borja Biurrun por tener siempre disponibilidad absoluta.

A Efraim, Alexis y Alberto por haber trasnochado tantas veces conmigo intentando salvar lo insalvable.

A todos los componentes del equipo de los últimos exámenes, que además de trabajar muy duro, siempre han sido capaces de reírse y relajar el ambiente en los momentos clave.

A toda esa difusa red de "tráfico" de apuntes en la que tanto nos hemos apoyado.

En general a todos los que han sufrido, renegado, reido, llorado y maldecido conmigo en algún momento de la carrera.

A aquellos profesores que además hemos sentido como maestros.

Finalmente a Paqui, Mari Carmen, Eva, Rosi, y demás personal de cafetería capaz de hacerme sonreír en los peores días.

Pablo Pérez de Ciriza Villa

A medida que la tecnología avanza, y lo hace de manera exponencial, crece también el temor del hombre que ve como su vida es invadida por los nuevos logros. ¿Podrá la tecnología sustituir a las personas? En algunos aspectos ya lo ha hecho, facilitando y favoreciendo el progreso de la vida doméstica, industrial, científica... Pero en todos los avances conseguidos a lo largo de la Historia han sido imprescindibles las personas, los creadores que con su trabajo, sus ideas, sus reflexiones, consiguen esas mejoras que benefician a toda la humanidad. Son los hombres, y lo serán siempre, los artífices de estos logros.

Resumen

Este proyecto ha sido concebido para ayudar a futuros estudiantes, que vayan a utilizar el Lego Mindstorms NXT, a elegir e iniciarse en el lenguaje de programación que mejor se adecúe a sus necesidades. Para ello se han estudiado cuatro lenguajes de programación (NXT-G, NXC, leJOS monohilo y multihilo, y Matlab teleoperado) y se han ideado una serie de experimentos para evaluarlos. Posteriormente se ha creado una colección similar de programas en cada lenguaje, y finalmente se han comparado los resultados de todos los lenguajes para cada experimento obteniendo una serie de conclusiones. Asimismo se ha realizado una colección de videos para una mejor comprensión de los experimentos.

Abstract

This project has been conceived to help the prospective students, who will use the Lego Mindstorms NXT, to choose and to start learning about the programming language that fulfills their needs. There have been studied four programming languages (NXT-G, NXC, leJOS and leJOS multithreading, and teleoperated Matlab) and have been devised a list of experiments to evaluate them. After that, there has been created a similar collection of programs in each language, and finally the results of all languages have been compared for each experiment obtaining a set of conclusions. Also there has been created a video collection for a better understanding of the experiments.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

INDICE DE TABLAS

CAPÍTULO 1

INTRODUCCIÓN

1.1. Historia de los automatismos

Los automatismos ya existían sobre el año 1500 a.C. Tiempos en los que los egipcios colocaban brazos mecánicos en algunas estatuas de sus dioses, haciendo creer al pueblo que se movían por inspiración divina. También crearon automatismos como el reloj de agua, gracias al cual podían medir el tiempo tanto de día como de noche.

Asimismo los griegos crearon entre otras cosas ingenios hidráulicos con los que movían partes de estatuas para sorprender a su gente. Por ejemplo Ctesibio de Alejandría, al que históricamente se le conoce como el padre de la hidrostática, aplicó la fuerza del aire a diversos instrumentos. Construyó una bomba de compresión aspirante e impelente que utilizaban los bomberos de Alejandría, un órgano hidráulico que se hacía funcionar pasando el aire por distintos tubos impulsándolo mediante una columna de agua, un arcabuz neumático, etc.

En China, alrededor del 500 a. C. King-su Tse fabrica una urraca con

madera y bambú que es capaz de volar, y un caballo también de madera capaz de dar saltos.

En el siglo III a. C. en Grecia, Filón de Bizancio escribe *Mechanike sintaxis* (Tratado de Mecánica) que incluye las secciones *Pneumatica* sobre dispositivos que funcionan por presión del aire o del agua, y *Automatiopoeica* sobre juguetes mecánicos y diversiones. También inventó un autómata acuático y una catapulta automática.

A finales del siglo III a. C., supuestamente fue encontrado un automatismo consistente en una orquesta en la cual los músicos se movían de forma independiente, conocido como *el tesoro de Chin Shih Hueng Ti*

Existen muchos más ejemplos de automatismos en la antigüedad lo que demuestra la curiosidad del ser humano desde sus orígenes y su búsqueda de nuevos elementos que aporten mayor entretenimiento y bienestar a sus vidas. Y así, en ese continuo afán de crecer se llega a la figura del robot que, naturalmente, solo será un nuevo escalón de este inagotable interés de los seres humanos de todos los tiempos.

1.2. Robots

El término robot deriva de "robotnik" que define al esclavo de trabajo. Es utilizado por primera vez en una obra de teatro checa llamada *R.U.R.* (Rossum's Universal Robots) estrenada en 1921 por el dramaturgo Karel Capek.

En 1939 en la Feria Mundial de Nueva York la casa Westinghouse presenta una pareja de Robots: un humanoide llamado Elektro y un perro llamado Sparko. Elektro tenía la capacidad de reproducir unas 700 palabras, fumar, caminar, y distinguir el color rojo del verde, mientras que Sparko era capaz de ladrar, posarse sobre sus patas traseras, y seguir otras ordenes sencillas que se le indicaban.

En 1953, en Inglaterra se crea ELSIE (Electro Light Sensitive Internal External), que se limitaba a seguir una luz utilizando un sistema mecánico realimentado sin ningún tipo de inteligencia.

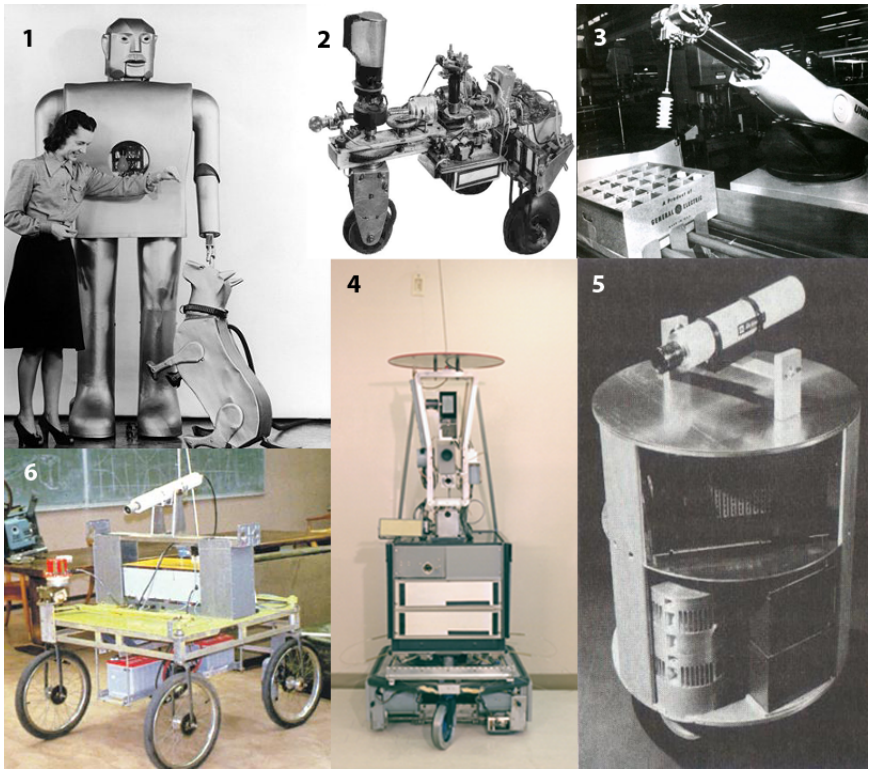

Figura 1.1: *1- Elektro y Sparko. 2- ELSIE. 3- Unimate. 4-Shackey. 5-CMU-rover. 6- Cart.*

En 1961, se produce un salto cualitativo, instalándose *Unimate*, el primer robot industrial, en una cadena de montaje de la General Motors en Nueva Jersey. El robot se encargaba de transportar las piezas fundidas en molde hasta la cadena de montaje y soldarlas sobre el chasis del vehículo. Gracias a ello, este trabajo, que era de alta peligrosidad y nocivo para la salud por los gases de combustión, no lo tenía que realizar ningún humano.

En 1968 el Standford Research Institute presenta a Shackey. Un robot equipado con diferentes sensores y una cámara de visión, capaz de desplazarse por sí mismo. Utilizaba dos computadoras conectadas por radio. Una instalada en el propio robot y la otra remota para procesar las imágenes.

En los años setenta la NASA en cooperación con la Jet Propulsion Laboratory crean el Mars-Rover, provisto de un brazo mecánico, un dispositivo telemático laser, cámaras estéreo y sensores de proximidad.

Enla década de los ochenta aparecen el Cart, del Standford Research

Institute y el CMU-Rover, de la Carnegie Mellon University.

En la actualidad los robots forman parte de nuestra vida cotidiana. Desde robots limpiadores como la roomba, pasando por máquinas expendedoras de tickets de los metros, trenes, etc. hasta robots educativos entre los que se encuentra el LEGO Mindstorms.

1.3. Objetivos

El presente proyecto tiene como meta final ayudar a futuros estudiantes, que vayan a utilizar el Lego Mindstorms NXT, a elegir e iniciarse en el lenguaje de programación que mejor se adecúe a sus necesidades. Los principales objetivos para conseguir esta meta son:

- Aprender los lenguajes de programación NXT-G, NXC, leJOS (monohilo y multihilo) y Matlab (teleoperado).
- Idear una serie de experimentos.
- Crear una colección de programas en cada lenguaje de programación.
- Comparar los resultados de los programas escritos en los diferentes lenguajes y obtener conclusiones.

Para una perfecta comprensión de los experimentos se adjunta un DVD con un video de cada experimento en cada lenguaje. De forma que en un futuro se puedan sacar conclusiones que se hayan pasado por alto en la elaboración del proyecto.

CAPÍTULO 2

ESTADO DEL ARTE

2.1. Historia de LEGO Mindstorms

En 1985 la empresa de juguetes LEGO no pasaba por una buena época. Cuando su presidente leyó el libro *MindStorms: Children, Computers, and Powerful Ideas*, de Seymour Papert, científico, matemático, educador y trabajador del MIT, quedó impactado. La lectura de este libro le impulsó a contactar con el MIT ya que le hizo pensar que compartían las mismas ideas sobre el aprendizaje infantil.

LEGO Y el MIT llegaron a un acuerdo. LEGO financiaría investigaciones del grupo de epistemología (Parte de la filosofía que trata de los fundamentos y los métodos del conocimiento científico) y aprendizaje del MIT sobre cómo aprenden los niños, y a cambio el MIT le ayudaría en la búsqueda de ideas para nuevos productos.

El primer fruto de la colaboración LEGO-MIT fue el LEGO TC Logo. Este sistema consistía en conectar una computadora a una construcción de LEGO que estaba provista de motores, luces y sensores. Tuvo un relativo

éxito comercial, pero se observó que el sistema "imponía restricciones tanto físicas como imaginativas". Posteriormente la idea cambió, y más tarde se pasó de programar una computadora que se conectaba a la construcción, a programar una parte de la construcción. La tecnología para ello era cara y además se necesitaría un ordenador para programarlo, y en aquella época tener un ordenador no era tan común como en la actualidad, e hizo que aparcaran el proyecto por unos años. Finalmente se desarrolló lo que conocemos como el bloque RCX. Con este producto pasaban de crear estructuras estáticas a crear estructuras dinámicas. Las principales pautas de diseño fueron:

- El sistema debía ser sencillo para el nuevo usuario y a la vez debía permitir realizar diseños sofisticados para el iniciado.
- El juguete debía poderse emplear de muchas formas diferentes. Según uno de los creadores, cuando un niño considera que ha "acabado" con el juguete, algo falla en el diseño.
- Simplicidad. El MIT descubrió que al hacer diseños más pequeños y limitados, los usuarios encontraban nuevas aplicaciones más creativas.
- Elección cuidadosa de las cajas negras: El niño tiene a su disposición motores, sensores y microcontroladores, pero no pueden modificarlos o rediseñarlos. El juego trataría sobre como combinarlos para hacer diseños, no sobre cómo diseñar estos componentes.
- Poner énfasis en el aprendizaje de la programación.
- El bloque debía tener un número suficiente de puertos de entrada/salida que pudieran conectarse con diferentes tipos de sensores: de temperatura, de amplitud del sonido o de luz.
- Desatender las sugerencias y observar en cambio en el laboratorio qué es lo que intentan hacer los niños con el juguete, pues apreciaron que de esta forma obtenían mejores ideas.

2.2. Versiones de LEGO Mindstorms

Aunque LEGO Mindstorms no sea un producto joven para ser electrónico, ha tenido muy pocas versiones. Se exponen a continuación.

RCX 1.0, 1.5 y 2.0, que comparten un mismo hardware, pero que tienen actualizaciones de software incompatibles puesto que las tres versiones tienen diferentes regulaciones de voltaje.

Figura 2.1: *Imagen del RCX.*

NXT y NXT 2.0 realmente son el mismo robot. Lo único que cambia son los elementos que configuran el kit.

Figura 2.2: *Imagen del pack NXT.*

Figura 2.3: *Imagen del pack NXT 2.0.*

2.3. Proyectos con LEGO Mindstorms

Actualmente se están continuamente creando más y más proyectos innovadores con el NXT. Ejemplo de ello son:

El robot solucionador de cubos de Rubik.

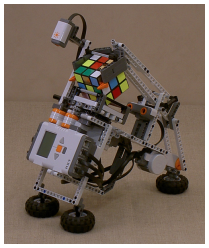

Figura 2.4: *NXT solucionando un cubo de Rubik.*

El robot lancha teledirigida.

Figura 2.5: *NXT convertido en controlador de una lancha teledirigida.*

El robot péndulo invertido.

Figura 2.6: *NXT manteniendose sobre 2 ruedas.*

Y hay, desde robots que utilizan los ultrasonidos para tirar de la cadena al levantarse del w.c., hasta robots que escalan entre dos paredes paralelas.

CAPÍTULO 3

ARQUITECTURA HARDWARE Y SOFTWARE

En esta sección se exponen las herramientas y dispositivos utilizados durante el desarrollo de este proyecto.

3.1. Software

3.1.1. Sistema Operativo

El sistema operativo utilizado para este proyecto ha sido Windows 7. Este sistema operativo destaca por su facilidad de uso, buen comportamiento, y compatibilidad con una gran cantidad de software y hardware.

El presente proyecto podría haberse construido utilizando otros sistemas operativos diferentes, como Linux, dado que todos los dispositivos y programas utilizados para su creación, excepto el BricxCC, tienen versiones com-

patibles con múltiples sistemas operativos. En el caso de querer utilizar el BricxCC, se puede conseguir que funcione en Linux, aunque su versión oficial está en fase de desarrollo.

3.1.2. NXT 2.1 Programing

Este software se adquiere junto al NXT, está basado en Labview y está pensado expresamente para utilizarlo con el hardware de LEGO Mindstoms. Es fácil de usar y muy útil a la hora de hacer los primeros programas para familiarizarse con el robot. Dicho esto, no es una buena opción cuando se quieren hacer programas largos y complicados, ya que, aunque responda bien, para programar una orden como "x++" hay que colocar cuatro cajas.

Desde este programa se instalará el Firmware oficial de LEGO (V1.31), utilizado también en BricxCC y Matlab, cada vez que sea necesario.

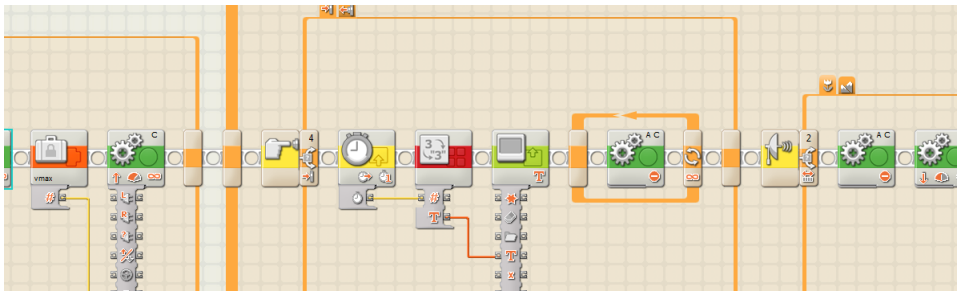

Figura 3.1: *Ejemplo de programación en NXT-G.*

3.1.3. Bricx Command Center

BricxCC es un entorno de desarrollo integrado especialmente dirigido a la programación de los diferentes robots LEGO Mindstorms en NBC (Next Byte Codes), NQC (Not Quite C) y NXC (Not eXactly C). En este caso se ha utilizado este último. Es cómodo, relativamente intuitivo y fácil de usar.

Cabe comentar que para un correcto funcionamiento el robot debe estar conectado al ordenador cuando se inicie el programa.

3.1.4. Eclipse

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma. En este proyecto se va a utilizar como compilador de leJOS. leJOS es un reemplazo del Firmware oficial de LEGO por una pequeña máquina virtual de Java adaptada a LEGO Mindstorms. Su instalación es larga y tiene cierta complicación. Está explicada en detalle en [5].

Desde Eclipse se instalará el Firmware/máquina virtual leJOS cada vez que sea necesario, tan solo apretando un botón.

3.1.5. Matlab 2009a

Matlab es un software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows y Apple Mac OS X.

En este proyecto se va a utilizar Matlab para teleoperar el robot mediante BlueTooth.

RWTH - Mindstorms NXT Toolbox

Es una toolbox para controlar LEGO Mindstorms desde Matlab mediante BlueTooth. Fue creada por estudiantes de la Universidad de Aachen (Alemania), y es de código abierto.

3.2. Hardware

3.2.1. Lego Mindstorms NXT 2.0

Lego Mindstorms NXT es un robot, vendido como juego o herramienta educativa por LEGO, que tiene la capacidad de ser programado en multitud de lenguajes de programación diferentes. Esto lo hace muy atractivo para su

uso académico tanto a nivel escolar como universitario, puesto que el nivel de dificultad puede variar en función de en qué lenguaje se quiera programar. Concretamente con este pack viene incorporado el nuevo Sensor de Color RGB.

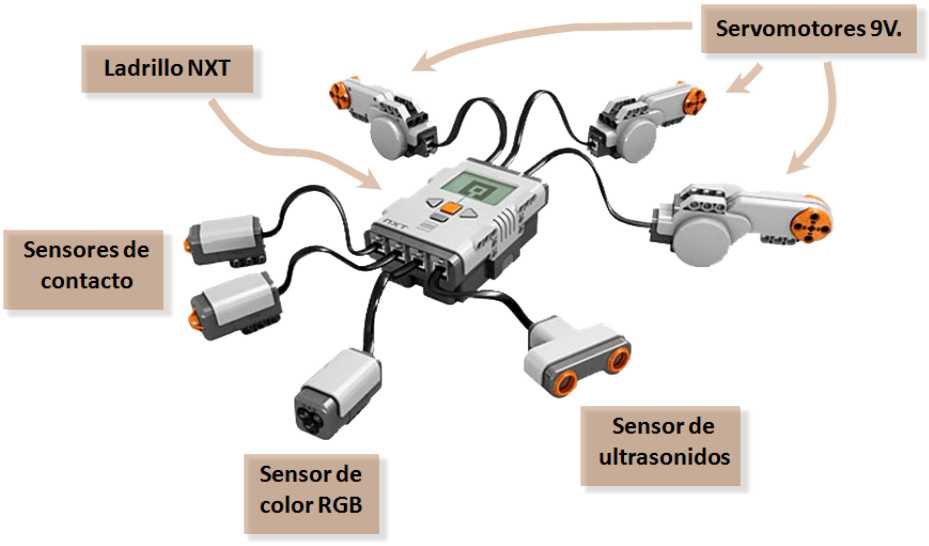

Figura 3.2: *Brick y sensores del pack NXT 2.0.*

3.2.2. Dispositivo BlueTooth

Se denomina Bluetooth al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, con una cobertura baja y basados en transceptores de bajo costo. Su objetivo es facilitar las comunicaciones entre equipos móviles y fijos eliminando cables y conectores, y procurar la sincronización de datos entre equipos personales.

CAPÍTULO 4

EXPERIMENTOS

Antes de la lectura de este capítulo hay que tener en cuenta dos cosas:

- En este capítulo se tratan ex profeso leJOS y leJOS multihilo como si fueran dos lenguajes de programación diferentes.
- Cuando en este capítulo se habla de Matlab, es en condiciones de teleoperación a no ser que se diga expresamente lo contrario.

4.1. Experimento 1 (Error de rotación)

4.1.1. Planteamiento

Este experimento consta de dos partes. La primera consiste en hacer girar un motor 18000º (50 vueltas) a velocidad máxima y comprobar el error que comete en el punto de parada. La segunda parte, es similar a la primera pero esta vez haciendo parar el robot un segundo cada 360º.

4.1.2. Preparación física del experimento

Para preparar el experimento lo primero que hay que hacer es fijar una de las ruedas motrices un poco más salida que su posición normal puesto que se le va a pegar un brazo, y éste no debe rozar la otra rueda. Después, se pega un palito de plástico (de remover el café) con cinta aislante a la llanta de la rueda y se fija bien ya que va a estar en movimiento y podría descolocarse. A éste se pega un palillo de madera pues se necesita algo puntiagudo para poder medir con precisión. Una vez se tiene el brazo bien fijado, se coloca el "campo de pruebas" ,que viene en el paquete del NXT 2.0, justo debajo de una lámpara, y se tumba el robot sobre el lado contrario al de la rueda elegida en el medio del "campo de pruebas", haciendo coincidir el eje de la otra rueda motriz en el medio exacto.

Una vez se tiene todo preparado se cargan los diferentes programas en el NXT, se coloca la punta del palillo de forma que su sombra coincida con el punto de 0º, y se lanzan.

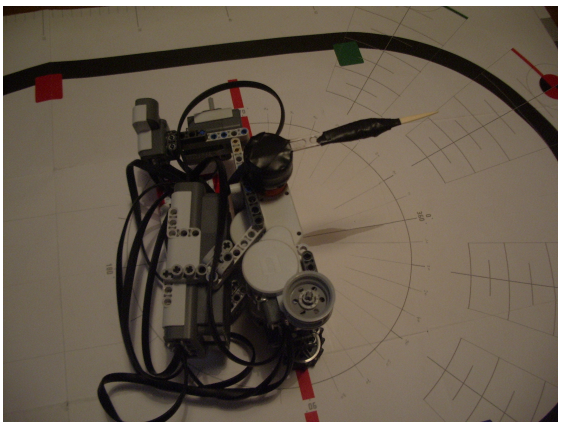

Figura 4.1: *Imagen del experimento 1.*

4.1.3. Estructura de los programas

La estructura de los programas de este experimento (ver Fig. 4.2) es extremadamente simple. Los programas para la primera parte constan de una o dos líneas de código en las que únicamente se les ordena girar a una velocidad concreta durante unos grados determinados. En la segunda parte

se utiliza un bucle para que después de cada giro de 360º espere un segundo e incremente el contador.

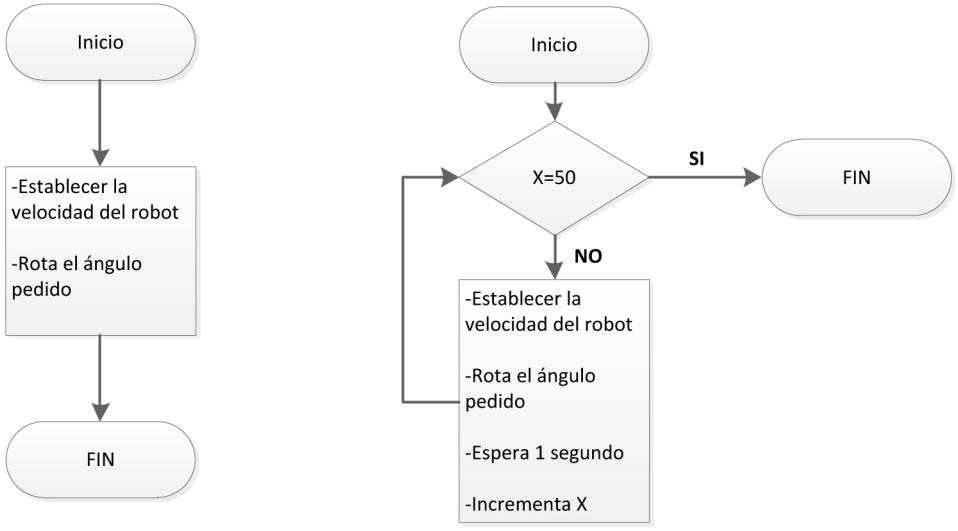

Figura 4.2: *Diagramas de flujo experimentos 1 y 1b.*

4.1.4. Resultados del experimento

En la primera parte del experimento, tanto el programa escrito en NXT-G como el escrito en Matlab son los que mejores resultados han obtenido. El escrito en Matlab tiene una amplitud de rango tan solo de 4 grados y una media de 1,2 grados de error. El escrito en NXT-G tiene unos resultados algo peores, pero muy buenos aun así.

Los otros tres programas tienen unos resultados considerablemente menos precisos. Las medias de error rondan los 10 grados, y en el caso del programa en leJOS multihilo casi llega a 16 grados.

Un dato a destacar es que los dos programas de leJOS, tanto el monohilo como el multihilo, solo han obtenido resultados de errores negativos. Esto quiere decir que en ninguno de los casos se han llegado a completar los grados demandados.

En la segunda parte del experimento se observa que el programa realizado en leJOS monohilo tiene una media de error inferior a un grado, y que la amplitud del rango de error es tan solo de 3 grados.

El programa en NXT-G y el de leJOS multihilo también tienen unos buenos resultados, con medias por debajo de 1,5 grados de error y con amplitudes de rango de error de 6 y 5 respectivamente.

Finalmente NXC y Matlab son los que peores resultados obtienen para sus programas en esta segunda parte, quedando peor el de Matlab con una gran diferencia.

Ensayos	NXT-G	NXC	leJOS	leJOS Multihilo	Matlab
Ensayo 1	6	8	-2	-11	-2
Ensayo 2	-2	12	-10	-17	0
Ensayo 3	-2	14	-4	-20	1
Ensayo 4	0	11	-13	-12	1
Ensayo 5	0	8	-13	-19	2

Tabla 4.1: *Errores del experimento 1 (en grados).*

	NXT-G	NXC	leJOS	leJOS Multihilo	Matlab
Media	2	10,6	8,4	15,8	1,2
Mediana	0	11	-10	-17	1
Mínimo	-2	8	-13	-20	-2
Máximo	6	14	-2	-11	2
Amplitud rango	8	6	11	9	4

Tabla 4.2: *Media, mediana, mín, máx y amplitud de rango de los resultados del experimento 1.*

Ensayos	NXT-G	NXC	leJOS	leJOS Multi	Matlab
Ensayo 1	5	9	-2	2	24
Ensayo 2	1	8	0	-3	20
Ensayo 3	0	7	0	2	26
Ensayo 4	-1	15	0	0	12
Ensayo 5	0	9	1	0	3

Tabla 4.3: *Errores del experimento 1b (en grados).*

	NXT-G	NXC	leJOS	leJOS Multihilo	Matlab
Media	1,4	9,6	0,6	1,4	17
Mediana	0	9	0	0	20
Valor menor	-1	7	-2	-3	3
Valor mayor	5	15	1	2	26
Amplitud rango	6	8	3	5	23

Tabla 4.4: *Media, mediana, mín, máx y amplitud de rango de los resultados del experimento 1b.*

4.1.5. Conclusiones del experimento

Haciendo un pequeño resumen de los resultados, se puede decir que los programas en NXT-G son los que globalmente han tenido una mejor respuesta. Sin embargo no ha sido óptimo en ninguno de los casos. Los más precisos han sido el programa de Matlab en el experimento sin paradas, y el de leJOS monohilo en el experimento con paradas.

Los programas en NXC han sido los que peor respuesta global han tenido. Tampoco han sido los peores en ninguno de los dos casos, pero son los únicos que no tienen un buen resultado en por lo menos uno de los casos.

4.2. Experimento 2 (Velocidad medida mediante contadores)

4.2.1. Planteamiento

Este experimento consiste en lanzar el NXT a velocidad máxima por una pista hasta que se tope con una pared a una distancia determinada. Mientras el NXT esté en marcha un contador correrá en el ladrillo. Una vez llegue a la pared, y el sensor de contacto esté pulsado, se pararán tanto los motores como el contador.

La finalidad de este experimento es comprobar que la velocidad máxima del NXT es la misma para todos los lenguajes.

4.2.2. Preparación física del experimento

Se pega una pestaña de papel que llegue desde el sensor de color hasta el suelo. Se coloca el robot con el sensor de contacto tocando la pared y se marca en el suelo con cinta aislante donde esté la pestaña. Después se miden dos metros desde la marca de cinta aislante en dirección contraria a la pared y se pone otra marca de cinta aislante. Por último, se cargan los diferentes programas en el NXT, el cual se coloca con la pestaña a la altura de la última marca y se lanza.

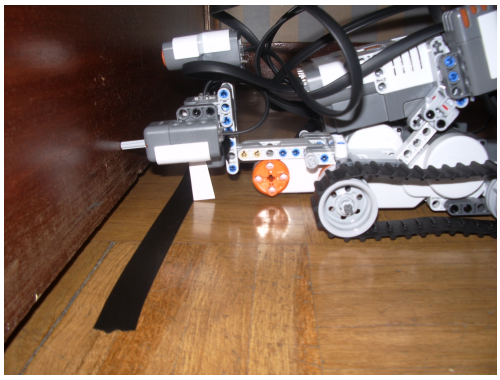

Figura 4.3: *Imagen del experimento 2.*

4.2.3. Estructura de los programas

La estructura de los programas (ver Fig. 4.4) empieza declarando las variables, habilitando los sensores y poniendo los motores en marcha a una velocidad concreta. Después, mientras el sensor de contacto no esté presionado se incrementa el valor del contador y se imprime en pantalla. Una vez que el sensor de contacto haya sido presionado, se paran los motores, pero no se cierra el programa hasta pulsar el botón ESCAPE del NXT, puesto que de otra forma, no se podría leer la pantalla al acabar el experimento.

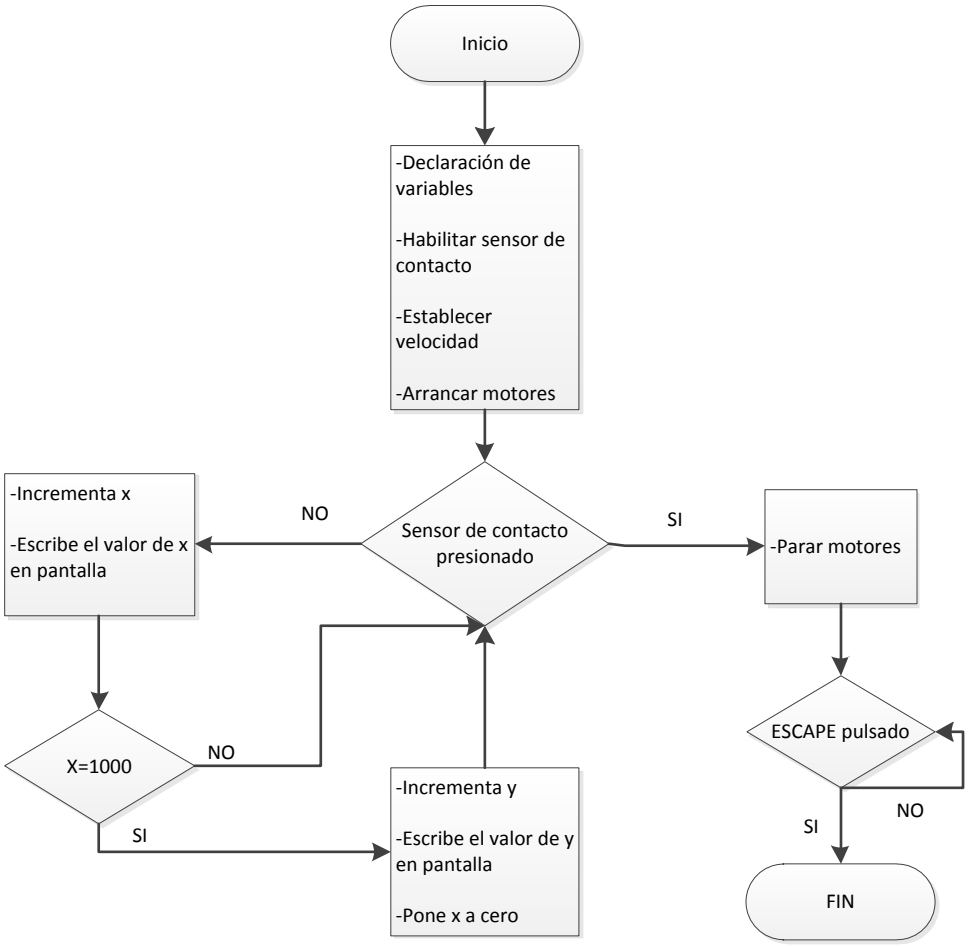

Figura 4.4: *Diagrama de flujo experimento 2.*

4.2.4. Resultados del experimento

Por pantalla salen dos variables. X, que es un contador incremental, e Y que se incrementa cada vez que X llega a 1000.

Los resultados de este experimento son completamente diferentes de lo que se esperaba. Claro está que a partir de estos resultados no se puede estimar si las velocidades máximas de los motores cambian en función de los diferentes lenguajes, dado que cada lenguaje tiene una velocidad de conteo absolutamente diferente.

	NXT-G	NXC	leJOS	leJOS Multihilo	Matlab
Y	7	26	48	12	0
X	382	444	428	995	131

Tabla 4.5: *Valores de X e Y referentes al experimento 2.*

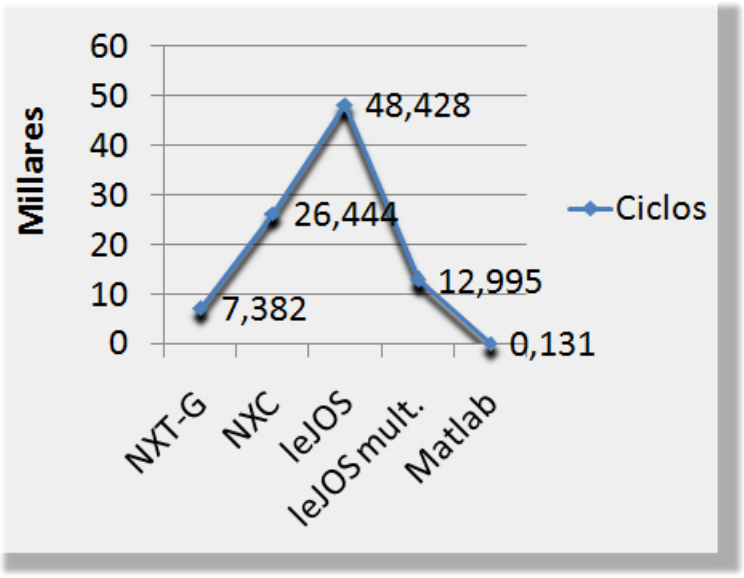

Figura 4.5: *Gráfica comparativa de los resultados del experimento 2.*

4.2.5. Conclusiones del experimento

La conclusión principal obtenida de este experimento es que cada lenguaje tiene una velocidad de conteo completamente diferente. Esto motiva los

experimentos 3 y 4. El primero, para ver realmente que diferencia de velocidad de conteo hay entre los lenguajes, y el segundo para resolver la cuestión que este experimento no ha resuelto: comprobar que la velocidad máxima del NXT es la misma para todos los lenguajes.

4.3. Experimento 3 (Velocidad de los procesos)

4.3.1. Planteamiento

En este experimento se van a comprobar dos cosas. Primero se comprobará cuantas veces se incrementa una variable en cada lenguaje durante 10 segundos. Después se comprobará cuantas veces puede repetir una serie de cálculos compuestos principalmente de raíces cuadradas, también en 10 segundos.

4.3.2. Preparación física del experimento

Este experimento no requiere ningún tipo de preparación física.

4.3.3. Estructura de los programas

En este experimento los programas empiezan tomando una referencia de tiempo. Después entran en un bucle, controlado por una condición de tiempo, en el que se incrementa continuamente un contador (o se realizan una serie de operaciones con raíces, en la segunda parte del experimento) mientras no transcurran 10 segundos. Para ello, dentro del bucle se coge una segunda referencia de tiempo, a la que se le resta la primera, obteniendo el tiempo transcurrido. Una vez transcurridos los 10 segundos se imprime en pantalla el resultado de los incrementos, y no se cierra el programa hasta que el botón ESCAPE no sea pulsado, para posibilitar la lectura de la pantalla. Ver Fig. 4.6.

4.3.4. Resultados del experimento

Como en el experimento anterior, salen dos variables. X, que es un contador incremental, e Y que se incrementa cada vez que X llega a 1000.

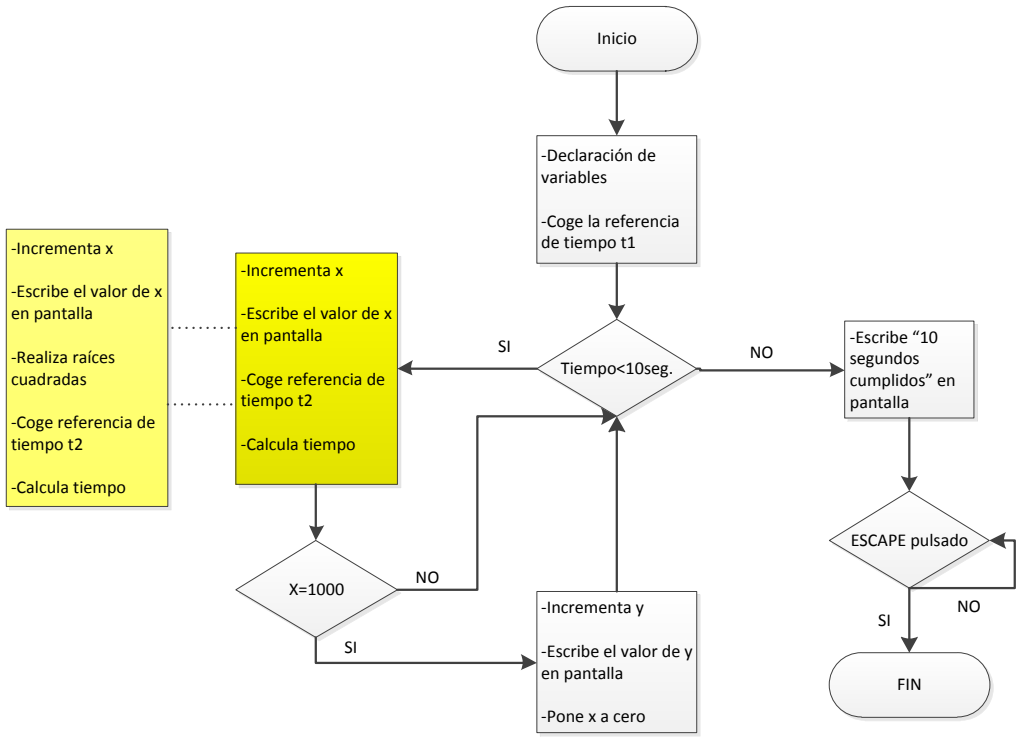

Figura 4.6: *Diagrama de flujo experimento 3. Intercambiando el cuadro amarillo, por el amarillo claro, se obtiene el del 3b.*

En esta prueba se han contemplado dos posibilidades respecto al programa en Matlab. En la primera, se considera que Matlab en cada ciclo tiene que conectar con el NXT. En la segunda, llamada para el experimento "Matlab sin conexión", Matlab no conecta con el NXT durante la ejecución del bucle.

En la primera parte de la prueba el programa en leJOS es con gran diferencia el que más incrementos realiza, seguido a gran distancia por el de leJOS multihilo, que no realiza ni la mitad.

Los otros cuatro programas tienen unas velocidades de conteo muy inferiores. Destaca entre ellos el NXC con una velocidad muy superior a los otros tres. Y por el contrario, destaca el programa de Matlab(con conexión) por ser sin duda el más lento de todos.

En la segunda parte de la prueba, el programa más rápido es el de "Matlab sin conexión", prácticamente a la par del programa de leJOS. El resto de programas no son tan rápidos, pero tampoco hay unas diferencias tan extremas como en la primera parte de la prueba, exceptuando Matlab(con conexión), que queda muy retrasado.

	NXT-G	NXC	leJOS	leJOS Multi	Matlab	Matlab sin conexión
Y	7	29	172	75	0	7
X	608	673	490	386	139	497

Tabla 4.6: *Valores de X e Y referentes al experimento 3.*

	NXT-G	NXC	leJOS	leJOS Multi	Matlab	Matlab sin conexión
Y	2	4	6	3	0	6
X	486	236	789	439	139	800

Tabla 4.7: *Valores de X e Y referentes al experimento 3b.*

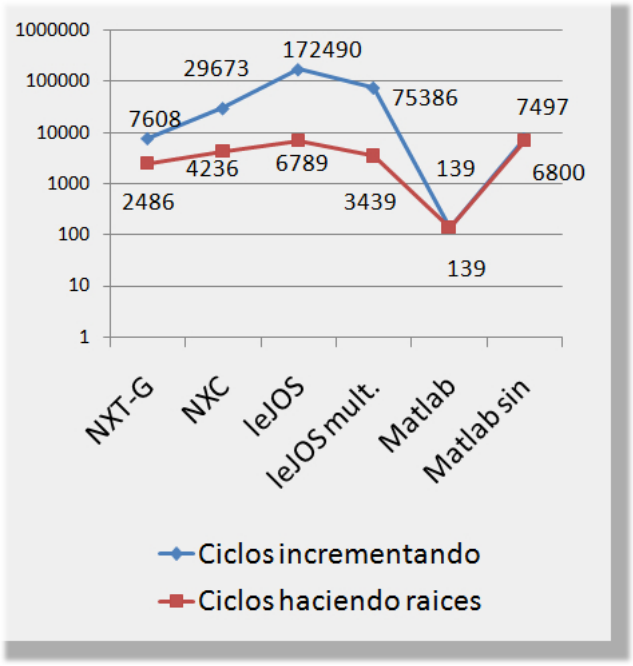

Figura 4.7: *Gráfica comparativa de los resultados del experimento 3. La gráfica es logarítmica para una mejor representación.*

4.3.5. Conclusiones del experimento

En este experimento es notorio el comportamiento del leJOS, que en la primera parte de la prueba es el claro vencedor, y en la segunda está prácticamente a la par del más rápido.

También es importante observar que en la primera parte del experimento el programa de leJOS multihilo es claramente superior al de NXC y que sin embargo en la segunda, el de NXC es superior al primero. De lo que se deduce que las operaciones complicadas, como pueden ser la raíces cuadradas, son mucho más eficientemente procesadas por el NXC.

Por último, hay que comentar que los dos programas de Matlab han tenido unos resultados casi exactos en las dos partes del experimento. Lo que hace pensar que el hecho de tener que procesar raíces cuadradas no les afecta.

4.4. Experimento 4 (Velocidad medida mediante referencias temporales)

4.4.1. Planteamiento

Este experimento consiste en lanzar el NXT a velocidad máxima por una pista hasta que se tope con una pared a una distancia determinada. Al arrancar el programa se tomará una referencia de tiempo. Una vez llegue a la pared, y el sensor de contacto esté pulsado, se pararán los motores y se tomará una segunda referencia de tiempo. Restando a la segunda referencia la primera se conseguirá el tiempo consumido.

Al igual que en el experimento 2, la meta de este experimento es comprobar que la velocidad máxima del NXT es la misma para todos los lenguajes.

4.4.2. Preparación física del experimento

La preparación es la misma que para el experimento 2.

Figura 4.8: *Robot corriendo.*

4.4.3. Estructura de los programas

La estructura de este experimento (ver Fig. 4.9) empieza tomando una referencia de tiempo y poniendo en marcha los motores, para entrar justo después en un bucle del que no se saldrá hasta que el sensor de contacto no haya sido presionado. Un vez presionado se toma una segunda referencia de tiempo a la que se le resta la primera obteniendo el tiempo, se imprime en pantalla y se paran los motores. No se cierra el programa hasta que el botón ESCAPE no sea pulsado, para posibilitar la lectura de la pantalla.

4.4.4. Resultados del experimento

Los resultados son prácticamente los mismos en todos los casos. El más elevado es el del programa en NXC, pero esto puede ser debido a que en éste no hace avanzar al robot completamente recto, sino que se arquea un poco la trayectoria.

Por otra parte, las pequeñas diferencias también pueden estar causadas por la diferente carga de las pilas en el momento de realizar cada experimento.

NXT-G	NXC	leJOS	leJOS Multihilo	Matlab
9717	10122	9540	9448	9680

Tabla 4.8: *Tiempos referentes al experimento 4 (en milisegundos).*

4.4.5. Conclusiones del experimento

Se puede concluir que utilizando cualquiera de los lenguajes se puede conseguir una velocidad máxima similar.

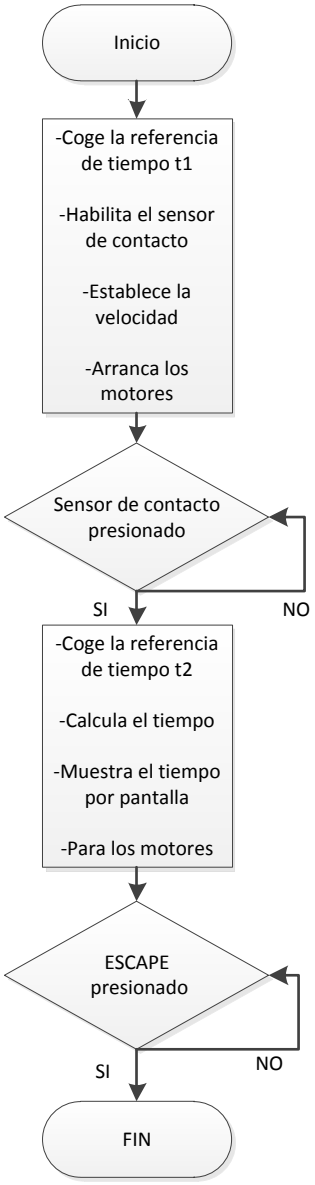

Figura 4.9: *Diagrama de flujo experimento 4.*

4.5. Experimento 5 (Fiabilidad ultrasonidos)

4.5.1. Planteamiento

Este experimento consiste en observar el error que comete el sensor de ultrasonidos en cada rango de valores para cada lenguaje.

4.5.2. Preparación física del experimento

Primero se coloca una pestaña en la parte trasera del robot lo suficientemente larga como para que roce el suelo. Luego se coloca el robot con el sensor de ultrasonidos tocando la pared y se hace una marca en el suelo donde esté la pestaña. Después se pega una cinta métrica al suelo haciendo coincidir el inicio de ésta con el final de la pestaña trasera del robot.

Por último se quitan las orugas de goma de las ruedas para que el NXT pueda deslizarse por el suelo con una mayor facilidad, se cargan los diferentes programas y se desliza el robot por encima de la cinta métrica comprobando los valores.

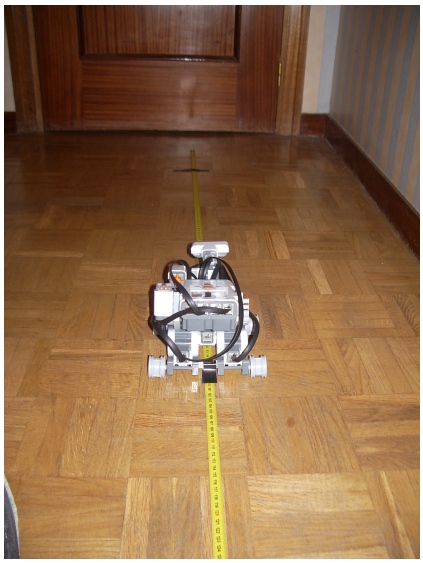

Figura 4.10: *Imagen del experimento 5.*

4.5.3. Estructura de los programas

La programación de este experimento es sencilla (ver Fig. 4.11). Primero se habilita el sensor de utrasonidos, y después se entra en un bucle en el que se muestra por pantalla la distancia percibida por el sensor mientras no se presione ESCAPE para salir del programa.

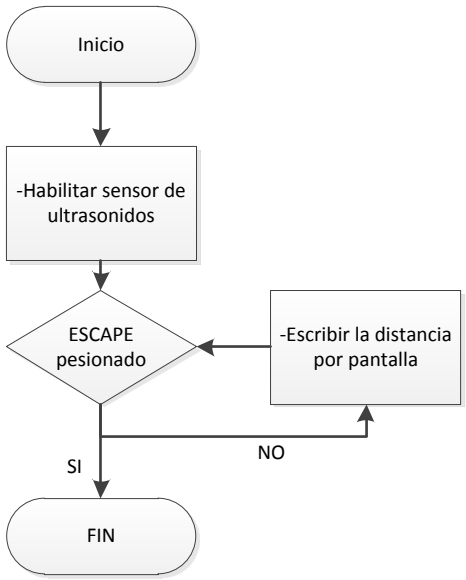

Figura 4.11: *Diagrama de flujo experimento 5.*

4.5.4. Resultados del experimento

Todos los lenguajes de programación han obtenido los mismos resultados exceptuando ciertas anomalías consistentes en detectar que la distancia está fuera de rango cuando realmente no lo está. Por ejemplo el programa en NXC devuelve para las distancias de 181cm. a 201cm. el valor máximo, 255. Todos los lenguajes han sufrido estas anomalías, y en valores muy parecidos. Se ha descubierto que inclinando el sensor de ultrasonidos unos 15 grados hacia arriba estas anomalías desaparecen.

Real	Obtenido
2	5
3	5
4	6
5	7
:	:
20	22
22	23
24	24
:	:
168	168
169	170
:	:

Tabla 4.9: *Medidas del experimento 5 (en centímetros).*

NXT-G	NXC	leJOS	leJOS multihilo	Matlab
180-187	118-189	185-194	180-186	181-201
212-255	220-255	203-213	196-210	208-213
		215-255	219-255	216-255

Tabla 4.10: *Anomalías del experimento 5. Rangos de valores en cm. para los que la respuesta es 255.*

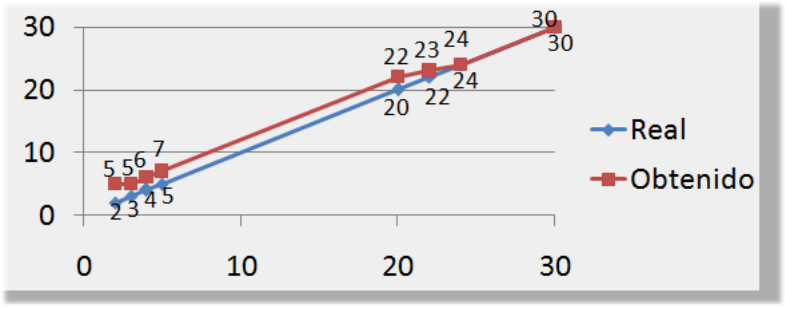

Figura 4.12: *Extracto representativo de la parte baja de la gráfica de valores reales y obtenidos del experimento 5.*

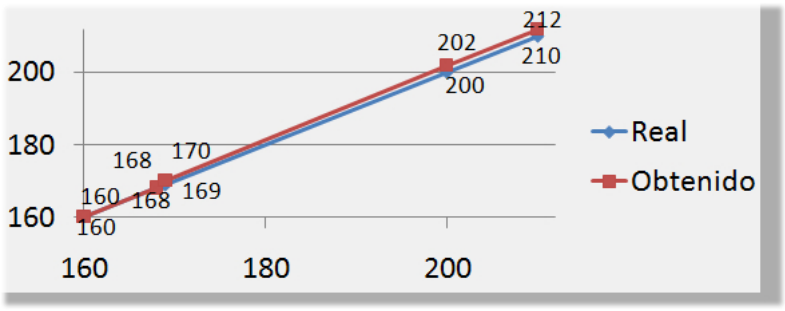

Figura 4.13: *Extracto representativo de la parte media-alta de la gráfica de valores reales y obtenidos del experimento 5.*

4.5.5. Conclusiones del experimento

Parece que todos los lenguajes de programación perciben igual la distancia medida mediante el sensor de ultrasonidos. Si se obvian las anomalías se puede concluir lo siguiente:

Para distancias muy pequeñas (2cm) los programas devuelven valores hasta tres puntos más altos de los debidos. Para distancias pequeñas de entre 3 y 20cm. los valores devueltos son dos puntos mayores que las distancias reales. Para una distancia de 22cm. el valor devuelto solo se incrementa en un punto. Y a partir de 24cm. y hasta 169cm. el valor devuelto es el real. Para distancias mayores a 170cm. los valores devueltos vuelven a estar un punto por encima de la realidad, hasta aproximadamente los 200cm. donde los valores devueltos vuelven a ser dos puntos mayores de lo debido.

4.6. Experimento 6 (Distancia Ultrasonidos-parada)

4.6.1. Planteamiento

Este experimento consiste en ver cuánto espacio utiliza el NXT programado en los diferentes lenguajes para parar yendo a velocidad máxima, desde que el sensor de ultrasonidos detecta una distancia determinada. Se harán dos pruebas diferentes: una parando los motores en seco, y otra dejándolos en flotación.

4.6.2. Preparación física del experimento

Primero se pega una pestaña de papel que llegue desde el sensor de color hasta el suelo. Después se modifican los programas de forma que la velocidad de avance sea una quinta parte de la máxima. Esto hará que el error de frenada sea mínimo. Una vez modificados los programas, se lanza el NXT contra una pared, de forma perpendicular, y se espera a que pare. El punto donde haya parado el NXT es en el que se detecta la pared a 20cm. Se marca como referencia el punto donde esté la pestaña. Se pega una hoja al suelo con una raya paralela dibujada a cada centímetro (ver Fig. 4.14), haciendo coincidir el borde del papel con el punto de referencia. Esto hay que hacerlo para cada programa. Una vez preparado lo anterior, se lanza a velocidad máxima.

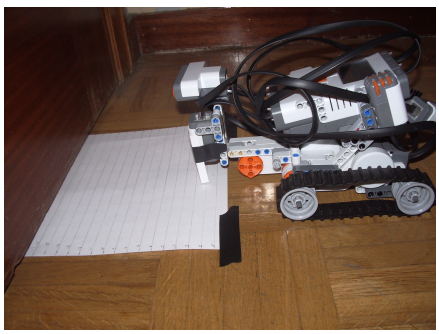

Figura 4.14: *Imagen del experimento 6.*

4.6.3. Estructura de los programas

Los programas empiezan habilitando el sensor de ultrasonidos y poniendo en marcha los motores. Mientras el botón ESCAPE no sea presionado, si el sensor detecta que la distancia a la pared es menor que 20 centímetros, se paran los motores. Ver Fig. 4.15.

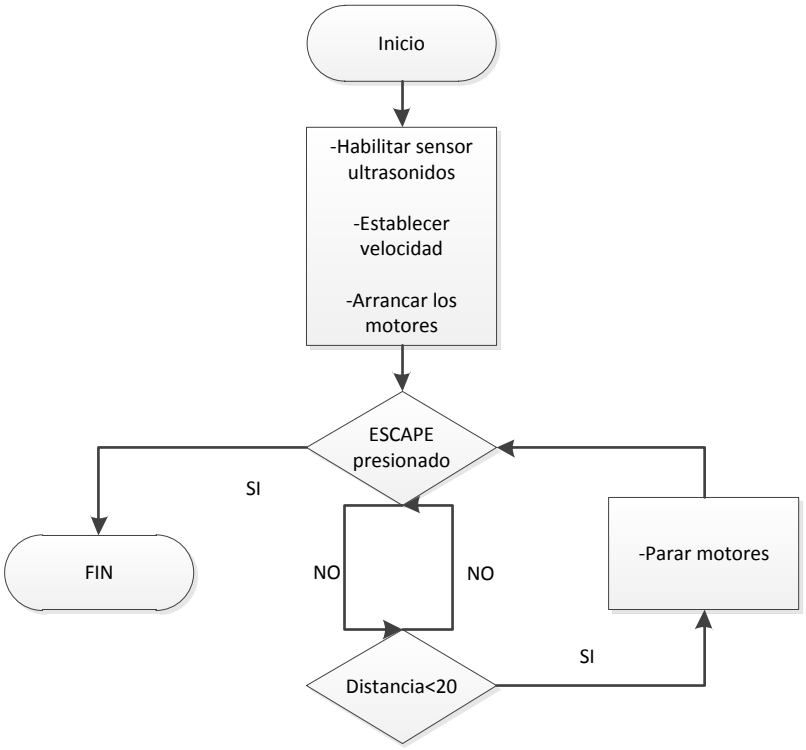

Figura 4.15: *Diagrama de flujo experimento 6.*

4.6.4. Resultados del experimento

En la primera parte del experimento, frenando los motores en seco, los programas que mejores resultados han tenido han sido los escritos en NXT-G y NXC, obteniendo una media de espacio de frenada más baja el primero

que el segundo. Sin embargo, el programa de NXC tiene una dispersión de valores menor que el de NXT-G. Los programas escritos en leJOS multihilo y Matlab consumen más del doble de espacio que los citados anteriormente.

En la segunda parte del experimento se descubre que leJOS y leJOS multihilo no responden correctamente a la orden *flt* (float), dado que frenan en seco. Esto hará que no se tengan en cuenta sus resultados. Del resto de los programas, vuelven a ser el de NXT-G y el de NXC los que menos espacio recorren desde que detectan la pared hasta que se paran definitivamente. El programa en Matlab vuelve a ser el que más espacio recorre.

Ensayos	NXT-G	NXC	leJOS	leJOS Multihilo	Matlab
Ensayo 1	3,2	3	3,2	5,1	7
Ensayo 2	2,4	2,7	4,2	5,2	8,5
Ensayo 3	2,5	2,7	4,5	4,7	3,3
Ensayo 4	1,5	2,1	4,3	6	7,3
Ensayo 5	1,9	2,9	3,4	5,6	6,8

Tabla 4.11: *Espacio recorrido en la frenada referente al experimento 6 (en centímetros).*

	NXT-G	NXC	leJOS	leJOS Multihilo	Matlab
Media	2,3	2,68	3,92	5,32	6,58
Mediana	2,4	2,7	4,2	5,2	7
Valor menor	1,5	2,1	3,2	4,7	3,3
Valor mayor	3,2	3	4,5	6	8,5
Amplitud rango	1,7	0,9	1,3	1,3	5,2

Tabla 4.12: *Media, mediana, mín, máx y amplitud de rango de los resultados del experimento 6.*

Ensayos	NXT-G	NXC	leJOS	leJOS Multi	Matlab
Ensayo 1	8	8,2	2,5	3,8	13
Ensayo 2	8,4	8,9	3,8	5,4	17
Ensayo 3	8,2	8,9	4	5,8	13,3
Ensayo 4	8	8,5	4,8	5,4	14,4
Ensayo 5	8,1	7,6	3,8	4,6	17

Tabla 4.13: *Espacio recorrido en la frenada referente al experimento 6b (en centímetros).*

	NXT-G	NXC	leJOS	leJOS Multihilo	Matlab
Media	8,14	8,42	3,78	5	14,94
Mediana	8,1	8,5	3,8	5,4	14,4
Valor menor	8	7,6	2,5	3,8	13
Valor mayor	8,4	8,9	4,8	5,8	17
Amplitud rango	0,4	1,3	2,3	2	4

Tabla 4.14: *Media, mediana, mín, máx y amplitud de rango de los resultados del experimento 6b.*

4.6.5. Conclusiones del experimento

Una vez analizados los resultados en su conjunto, se puede decir que la reacción del conjunto sensor de ultrasonido-motores más rápida es la programada en NXT-G. Sin embargo, la más fiable respecto a dispersión de valores es la programada en NXC.

Sobre la segunda parte del experimento, hay que matizar que no obligatoriamente es peor el resultado de aquel lenguaje que consuma más tiempo en pararse, ya que puede haber diversos motivos para implementar una parada por flotación, y en algunos casos será deseable que recorra el mayor espacio posible.

Objetivamente, el programa en NXT-G es el que menor espacio consume en la segunda parte del experimento, y cuenta con una dispersión de valores extremadamente baja, lo que le otorga una gran fiabilidad.

4.7. Experimento 7 (Distancia Sensor de color-parada)

4.7.1. Planteamiento

En este experimento se tratará de documentar cuanto espacio utiliza el NXT, programado en los diferentes lenguajes, para parar desde su velocidad máxima al cambiar el color del piso de blanco a negro.

Al igual que en el experimento anterior, habrá dos pruebas diferentes, una de parada en seco y otra de frenada en flotación.

4.7.2. Preparación física del experimento

La preparación de este experimento tiene varias similitudes a la del experimento anterior.

Se colocan dos cartulinas juntas, una blanca y una negra, y se fijan bien al suelo. Luego se coloca una pestaña en la parte trasera del robot lo suficientemente larga como para que roce el suelo. Una vez colocada la pestaña, se lanza el robot desde la cartulina blanca hacia la negra con una velocidad de avance igual a una quinta parte de la máxima, y se marca la posición de la pestaña una vez el robot se haya parado. Con esto ya se tiene una referencia fiable de dónde el robot detecta la diferencia de color. Después, se fija la misma hoja que se utilizó en el experimento anterior, haciendo coincidir el inicio con la marca realizada, y se recorta la parte que invade la cartulina negra. Habrá que repetir el proceso para cada lenguaje puesto que cada uno reconocerá el cambio de color en un lugar diferente.

Por último, se lanza el NXT a velocidad máxima.

4.7.3. Estructura de los programas

Los programas empiezan habilitando el sensor de color como sensor de luz y poniendo en marcha los motores. Mientras el botón ESCAPE no sea

presionado, si el sensor detecta que el suelo es de color negro, se paran los motores. Ver Fig. 4.16.

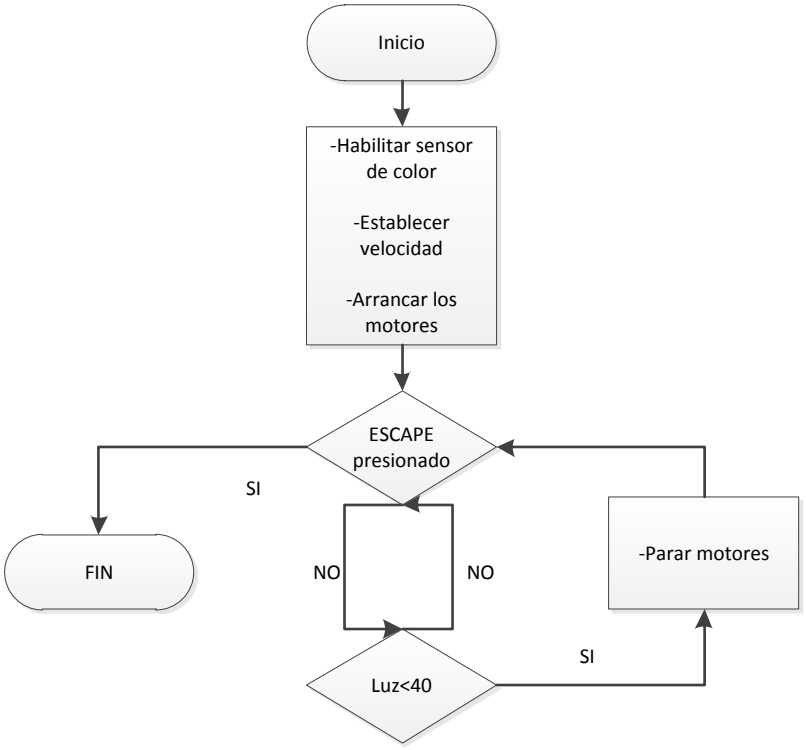

Figura 4.16: *Diagrama de flujo experimento 7.*

4.7.4. Resultados del experimento

Los resultados obtenidos para frenado en seco por el programa en lenguaje NXT-G son realmente buenos. El robot es capaz de parar en 6 milímetros de media, y la amplitud del rango de valores es de 3 milímetros. Los resultados del código NXC también son bastante buenos, y destaca que tiene, al igual que leJOS, una dispersión de valores nula. Los programas en leJOS multihilo y en Matlab tienen unos resultados mucho peores.

En la segunda parte del experimento, dejando los motores en flotación,

como ya se sabía del experimento anterior, los dos programas realizados en leJOS frenan en seco, y por ello no se tendrán en cuenta sus resultados. En esta segunda parte, el programa de NXT-G también es el que menos espacio necesita para pararse, pero la diferencia con el de NXC es solo de 3 milímetros. Este último tiene una dispersión de valores menor, pero ambas son muy bajas. El programa de Matlab vuelve a ser el que más espacio necesita para parar, y con una dispersión de valores alta.

Ensayos	NXT-G	NXC	leJOS	leJOS Multi	Matlab
Ensayo 1	0,8	1,3	3	5,3	8,7
Ensayo 2	0,5	1,3	3	5,3	6,8
Ensayo 3	0,5	1,3	3	6,2	7,4
Ensayo 4	0,6	1,3	3	5,7	6,2
Ensayo 5	0,6	1,3	3	5,1	6,5

Tabla 4.15: *Espacio recorrido en la frenada referente al experimento 7 (en segundos).*

	NXT-G	NXC	leJOS	leJOS Multi	Matlab
Media	0,6	1,3	3	5,52	7,12
Mediana	0,6	1,3	3	5,3	6,8
Valor menor	0,5	1,3	3	5,1	6,2
Valor mayor	0,8	1,3	3	6,2	8,7
Amplitud rango	0,3	0	0	1,1	2,5

Tabla 4.16: *Media, mediana, mín, máx y amplitud de rango de los resultados del experimento 7.*

Ensayos	NXT-G	NXC	leJOS	leJOS Multi	Matlab
Ensayo 1	6	6,1	2,8	6	12,2
Ensayo 2	5,6	6,2	2,8	7,6	14,1
Ensayo 3	5,8	6	2,7	5,9	13,1
Ensayo 4	5,9	6,1	2,9	7,3	10,6
Ensayo 5	5,6	6	2,8	5,8	11

Tabla 4.17: *Espacio recorrido en la frenada referente al experimento 7 (en segundos).*

	NXT-G	NXC	leJOS	leJOS Multihilo	Matlab
Media	5,78	6,08	2,8	6,52	12,2
Mediana	5,8	6,1	2,8	6	12,2
Valor menor	5,6	6	2,7	5,8	10,6
Valor mayor	6	6,2	2,9	7,6	14,1
Amplitud rango	0,4	0,2	0,2	1,8	3,5

Tabla 4.18: *Media, mediana, mín, máx y amplitud de rango de los resultados del experimento 7b.*

4.7.5. Conclusiones del experimento

Analizando aisladamente este experimento, se observa que la frenada del programa en NXT-G es mucho más efectiva que las demás. La del código NXC también es buena, pero no es comparable dado que utiliza más del doble de espacio.

En la segunda parte del experimento se observa que también NXT-G y NXC son los que menos espacio requieren para parar el robot con los motores en flotación.

Hay que señalar que en la segunda parte del experimento, el programa de leJOS multihilo, aun frenando en seco, recorre más espacio que los implementados en NXT-g y NXC.

Si se analizan los resultados de este experimento en conjunto con los del experimento 6, se observa que para los programas escritos en NXT-G y en NXC la respuesta de los motores es notablemente más rápida en este experimento 7. Se deduce por tanto que en estos dos lenguajes el tiempo desde la adquisición del dato hasta la reacción de los motores es mayor en el caso de utilizar el sensor de ultrasonidos que el de color.

4.8. Experimento 8 (Reacción al cambio de color)

4.8.1. Planteamiento

Este experimento consta de dos partes:

La primera consiste en ver cuánto tarda un programa de cada lenguaje en cambiar de un piso blanco a un piso negro o viceversa 100 veces seguidas, avanzando cuando detecte que el suelo es blanco, y retrocediendo cuando lo detecte negro.

La segunda es similar a la primera pero en este caso se activará el motor derecho cuando se detecte blanco y el motor izquierdo cuando se detecte negro. Esto se hace para que el robot avance por la línea que separa los dos colores. En esta segunda parte, solo se cambiará de color 20 veces.

4.8.2. Preparación física del experimento

La preparación es muy básica. Basta con juntar dos cartulinas, una blanca y una negra, y fijarlas bien al suelo. Para la segunda parte del experimento, se necesitarán 2 cartulinas de cada color ya que el robot va a avanzar por ellas.

4.8.3. Estructura de los programas

La estructura básica de los programas (ver Fig. 4.17) de la primera parte del experimento empieza poniendo en marcha los motores para avanzar hacia delante de forma que no tome la primera referencia de tiempo hasta detectar el suelo de color negro. Una vez detectado, se entra en un bucle que se repetirá durante 100 ciclos en el que en cada ciclo se detectará una vez el suelo negro y se retrocederá y otra suelo blanco y se avanzará. Al finalizar los 100 ciclos se sale del bucle y se toma una segunda referencia de tiempo, a la que se le resta la primera para conseguir el tiempo transcurrido.

En esta segunda parte del experimento se empieza girando a la izquierda de forma que no tome la primera referencia de tiempo hasta detectar el suelo de color negro. Una vez detectado se entra en un bucle que se repetirá durante 20 ciclos en el que en cada ciclo se detectará una vez el suelo negro y se girará a la derecha y otra suelo blanco y se girará a la izquierda. Al finalizar los 20 ciclos se sale del bucle y se toma una segunda referencia de tiempo, a la que se le resta la primera para conseguir el tiempo transcurrido.

4.8.4. Resultados del experimento

Hay que destacar que en la primera parte del experimento el programa escrito en NXC es, con diferencia, el más rápido haciendo 100 cambios de color en una media de menos de 10 segundos. El siguiente programa más rápido en completar la prueba es el implementado en NXT-G, haciendo los 100 cambios en menos de 25 segundos de media. Los resultados de los otros tres programas son mucho más discretos, completando la prueba en unas medias de entre 50 y 66 segundos. Es interesante observar que el programa de leJOS multihilo es más rápido que el monohilo.

En la segunda parte del experimento el programa en NXC vuelve a ser el más rápido seguido del de NXT-G, a una distancia considerable de los otros tres lenguajes. En este caso se observa que el programa de leJOS monohilo supera los resultados del de multihilo. Hay que resaltar que en las dos partes del experimento ha habido que ponerle un retardo al programa escrito en NXC debido a que los cambios de color en ocasiones eran imperceptibles. Esto quiere decir que el programa implementado en NXC es todavía más rápido de lo que se muestra.

Ensayos	NXT-G	NXC	leJOS	leJOS Multi	Matlab
Ensayo 1	28,604	9,23	58,1	51,98	67,02
Ensayo 2	24,446	10,15	59,2	51,147	65,03
Ensayo 3	22,539	11,015	62,4	51,814	66,02
Ensayo 4	24,253	9,145	61,4	52,015	64,8
Ensayo 5	22,546	9,53	59,5	52,636	65,9

Tabla 4.19: *Tiempos referentes al experimento 8 (en segundos).*

	NXT-G	NXC	leJOS	leJOS Multi	Matlab
Media	24,4776	9,814	60,12	51,9184	65,754
Mediana	24,253	9,53	59,5	51,98	65,9
Valor menor	22,539	9,145	58,1	51,147	64,8
Valor mayor	28,604	11,015	62,4	52,636	67,02
Amplitud rango	6,065	1,87	4,3	1,489	2,22

Tabla 4.20: *Media, mediana, mín, máx y amplitud de rango de los resultados del experimento 8.*

Ensayos	NXT-G	NXC	leJOS	leJOS Multi	Matlab
Ensayo 1	3,53	3,26	7,528	9,279	15,367
Ensayo 2	3,513	3,26	7,683	9,329	16,455
Ensayo 3	3,778	3,12	7,645	9,703	15,71
Ensayo 4	3,46	3,18	7,916	9,771	14,526
Ensayo 5	3,381	3,12	7,638	9,956	16,329

Tabla 4.21: *Tiempos referentes al experimento 8b (en segundos).*

	NXT-G	NXC	leJOS	leJOS Multihilo	Matlab
Media	3,5324	3,188	7,682	9,6076	15,6774
Mediana	3,513	3,18	7,645	9,703	15,71
Valor menor	3,381	3,12	7,528	9,279	14,526
Valor mayor	3,778	3,26	7,916	9,956	16,455
Amplitud rango	0,397	0,14	0,388	0,677	1,929

Tabla 4.22: *Media, mediana, mín, máx y amplitud de rango de los resultados del experimento 8b.*

4.8.5. Conclusiones del experimento

Se ha comprobado que hay dos lenguajes capaces de hacer programas muy competentes para ambas partes de este experimento, que son el NXC y el NXT-G. Los otros tres lenguajes tienen unos resultados mucho peores de los esperados.

El lenguaje NXC, en este ámbito, es muy superior a los demás.

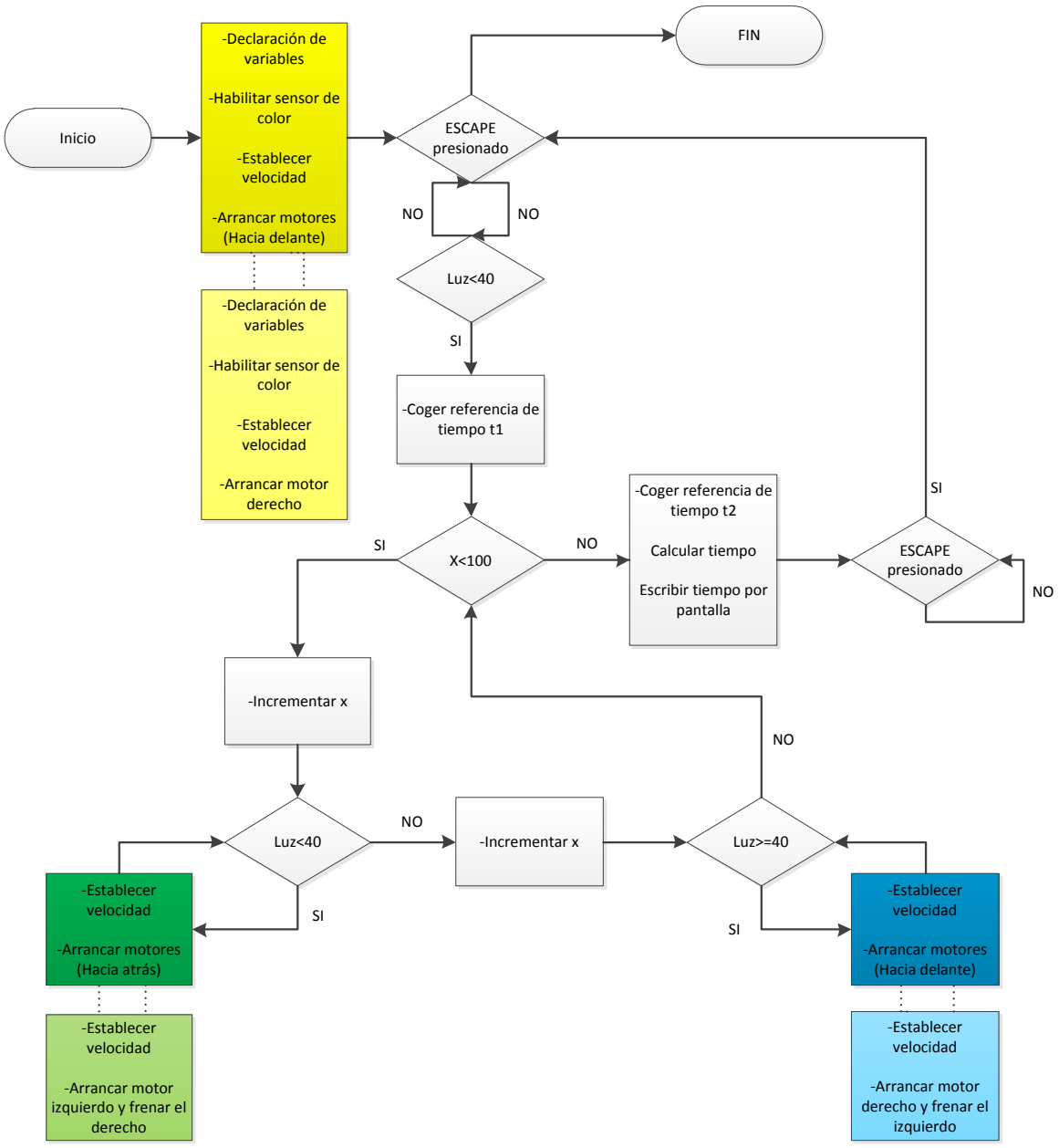

Figura 4.17: *Diagrama de flujo experimento 8. Intercambiando los cuadros de colores, por los claros, se obtiene el del 8b.*

4.9. Experimento 9 (Cambio de color de LEDs)

4.9.1. Planteamiento

El objetivo de este experimento consiste en averiguar cuanto tiempo necesita cada programa para cambiar de color 100 veces los LEDs del sensor de color.

4.9.2. Preparación física del experimento

Este experimento no requiere de una preparación física concreta.

4.9.3. Estructura de los programas

Los programas comienzan declarando las variables, habilitando el sensor de color y tomando una referencia de tiempo. Posteriormente entran en un bucle en el que repetirán 20 ciclos, en cada uno de los cuales, cambiarán 5 veces de color el LED del sensor. Una vez finalizados los 20 ciclos se toma una segunda referencia de tiempo, a la que se le resta la primera referencia para conseguir el tiempo transcurrido e imprimirlo en pantalla. El programa no se cerrará hasta que el botón ESCAPE no sea presionado, con el fin de poder leer los resultados en pantalla. Ver Fig. 4.18.

4.9.4. Resultados del experimento

Los más rápidos son los programas de NXT-G y de NXC. Los de leJOS y Matlab son más lentos pero no mucho. Sin embargo, el programa escrito en leJOS multihilo es considerablemente más lento.

	NXT-G	NXC	leJOS	leJOS Multihilo	Matlab
Milisegundos	16901	16901	18000	35101	18855

Tabla 4.23: *Tiempos del experimento 9 (en milisegundos).*

4.9.5. Conclusiones del experimento

Se ha observado que los programas de NXT-G y NXC, aparte de ser los más rápidos, tienen exactamente el mismo resultado. Esto puede querer decir que los procesos internos realizados por ambos para este experimento pueden ser los mismos, dado que de otra forma coincidir en un resultado de cinco cifras es más que improbable.

Destaca el de leJOS multihilo por ser el más lento, con casi el doble de tiempo que los anteriores.

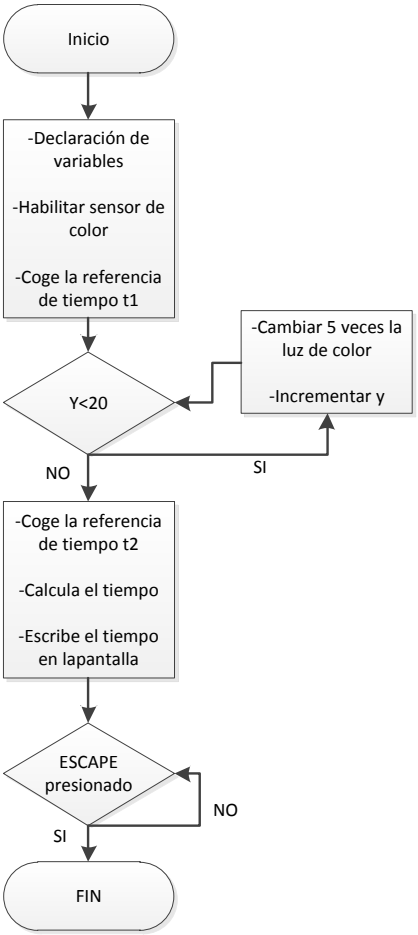

Figura 4.18: *Diagrama de flujo experimento 9.*

4.10. Experimento 10 (Reconocimiento de color)

4.10.1. Planteamiento

Este experimento consiste en intentar reconocer seis colores básicos (negro, azul, verde, amarillo, rojo y blanco) mediante el sensor de color RGB del NXT, y comprobar si hay diferencias entre los diferentes lenguajes de programación a la hora de hacerlo.

4.10.2. Preparación física del experimento

Para este experimento se necesitarán cartulinas de los colores citados en el apartado anterior.

4.10.3. Estructura de los programas

La estructura de los programas de este experimento es muy sencilla (ver Fig. 4.19). Principalmente consiste en un bucle en el que se detecta el color y se imprime por pantalla constantemente.

4.10.4. Resultados del experimento

Se ha comprobado que el sensor no reconoce bien los colores y se ha cubierto la zona entre el sensor de color RGB y el suelo con un cilindro de papel. Una vez hecha esta modificación, se ha proseguido con el experimento.

Todos los programas son capaces de reconocer los 6 colores. La principal diferencia entre ellos, es qué números devuelven para cada color. Los programas de NXT-G y NXC devuelven los colores según la tabla 4.24. Sin embargo los programas escritos en leJOS y leJOS multihilo devuelven los colores según la tabla 4.25. Y los programas escritos en Matlab devuelven directamente el nombre del color.

Color	Número Asignado
Negro	1
Azul	2
Verde	3
Amarillo	4
Rojo	5
Blanco	6

Tabla 4.24: *Equivalencia de número para colores en NXT-G y NXC.*

Color	Número Asignado
Ninguno	-1
Rojo	0
Verde	1
Azul	2
Amarillo	3
Magenta	4
Naranja	5
Blanco	6
Negro	7
Rosa	8
Gris	9
Gris claro	10
Gris oscuro	11
Cyan	12

Tabla 4.25: *Equivalencia de número para colores en leJOS y leJOS multihilo.*

4.10.5. Conclusiones del experimento

La conclusión más importante obtenida con este experimento es que el sensor de color RGB tiene que estar muy cerca del suelo (aproximadamente a un centímetro) si se quieren tener resultados fiables.

También hay que remarcar que en función de en qué lenguaje se programe, se tendrán unos resultados numéricos diferentes a la hora de reconocer los diferentes colores, excepto en el caso de Matlab, que devuelve directamente el nombre de los colores.

Coincide, no se sabe si casualmente, que los lenguajes que devuelven los colores según la tabla 4.24 comparten el Firmware oficial de LEGO, y que sin embargo los que devuelven los colores según la tabla 4.25 comparten el Firmware de LeJOS.

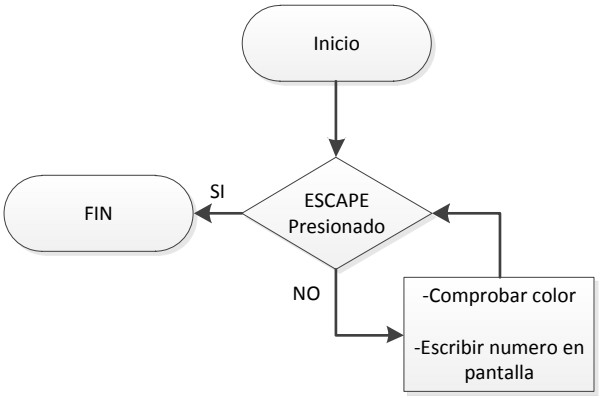

Figura 4.19: *Diagrama de flujo experimento 10.*

4.11. Experimento 11 (Canción de colores)

4.11.1. Planteamiento

Este experimento consiste en lanzar el robot por una pista creada con diferentes franjas de colores de tal forma que según avance, vaya reconociendo los colores y a su vez vaya haciendo sonar diferentes frecuencias asociadas a cada color. El resultado debería ser una canción.

4.11.2. Preparación física del experimento

Para la realización de este experimento habrá que crear previamente una pista con franjas de colores, tratando cada color como una nota musical, y la longitud de la franja como el tiempo que tenga que sonar esa nota. El color blanco se tratará como silencio, o lo que es lo mismo, frecuencia cero. En este caso se ha creado un camino de forma que al recorrerlo suene la Marcha Imperial de "La guerra de las galaxias". Los motores del robot tienen que moverse a la mitad de la velocidad máxima.

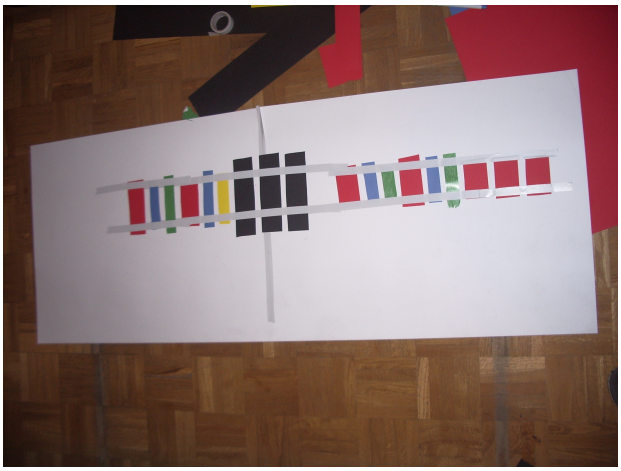

Figura 4.20: *Imagen del experimento 11.*

4.11.3. Estructura de los programas

Los programas comienzan poniendo en marcha los motores para entrar después a un bucle en el cual de forma continua se lee el sensor de color, se asigna una frecuencia al color detectado y finalmente se hace sonar esa frecuencia. Ver Fig. 4.21.

4.11.4. Resultados del experimento

Tanto el programa escrito en leJOS como el escrito en leJOS multihilo son los que mejor han hecho sonar la Marcha Imperial. En el resto de los programas, se percibe demasiado la transición entre franjas de colores, donde se detecta un tercer color no presente, que hace que suenen notas de más que no deben existir en la canción.

4.11.5. Conclusiones del experimento

El principal problema de este experimento ha sido que no se ha podido dar una velocidad mayor al robot ya que el sonido de los motores sonaba por encima de la música. Con una mayor velocidad de avance se podrían haber hecho franjas de color más largas y la transición entre los colores se habría notado mucho menos.

En definitiva, los programas con Firmware leJOS, los creados en leJOS y leJOS multihilo, han sido los que más definición han tenido.

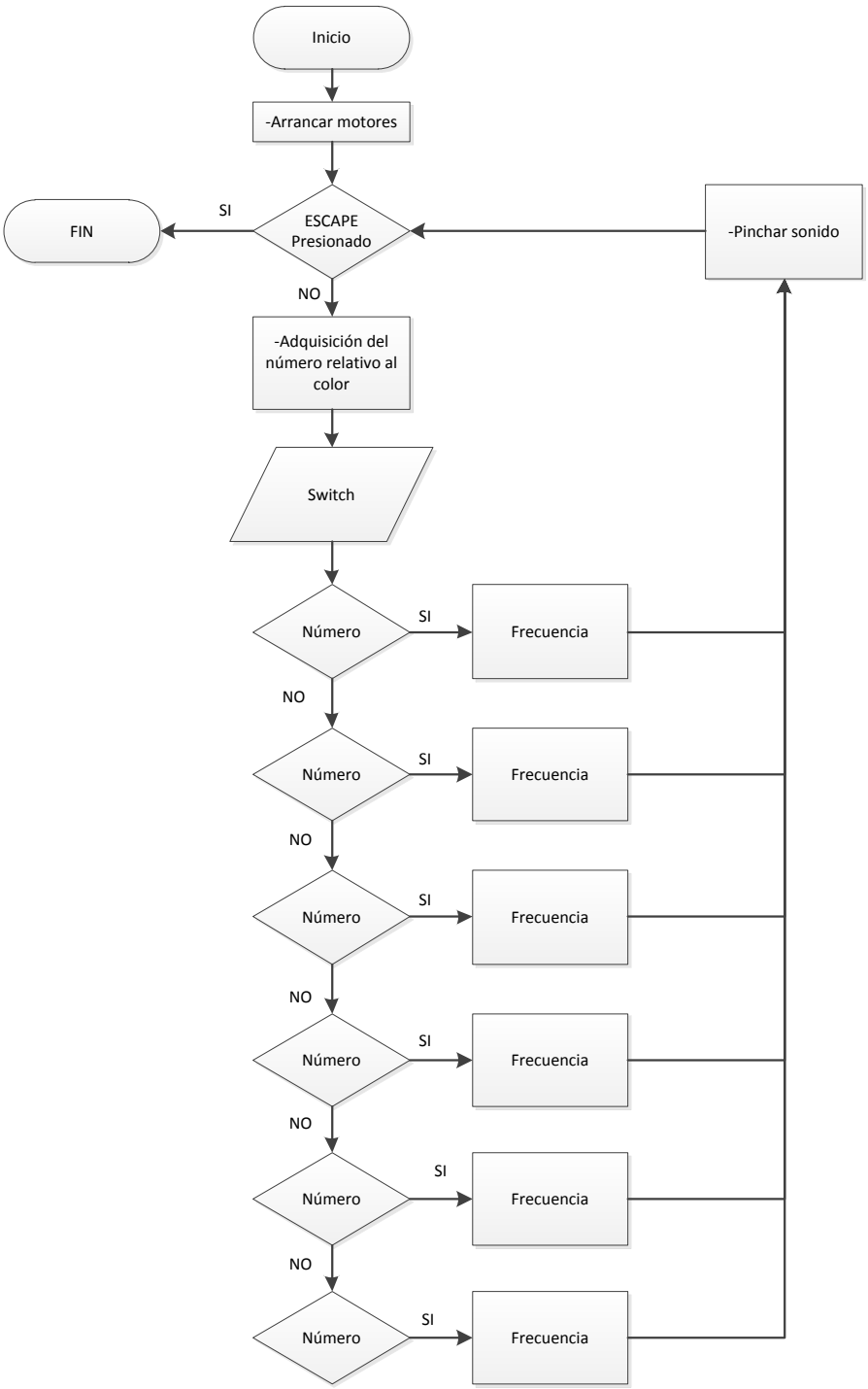

Figura 4.21: *Diagrama de flujo experimento 11.*

4.12. Experimento 12 (Sigue líneas avanzado)

4.12.1. Planteamiento

Este último experimento consiste en crear un programa que capacite al robot para que siga una línea, salve los obstáculos que se encuentre por el camino, y que finalmente nos diga el tiempo que ha tardado en llegar a la meta.

Hay que aclarar que el robot realmente no sigue una línea de un color, sino el borde de ella, girando a la izquierda al percibir el piso de color blanco, y a la derecha al percibir el piso de color negro.

El método utilizado para esquivar los obstáculos consiste en detectarlos por ultrasonidos, y hacer pasar al robot por la derecha del obstáculo dibujando un arco a su alrededor. Una vez se llega a la línea negra se sigue de frente hasta rebasarla y volver a llegar al piso de color blanco. Rebasada esta línea se hace girar solo el motor izquierdo, lo que provocará un giro cerrado hacia la derecha, hasta que el robot perciba otra vez el piso de color negro. Cuando el robot haya llegado de nuevo a la línea todo seguirá con normalidad.

Para medir el tiempo que tarda el robot en completar el recorrido, no se empieza a contar el tiempo hasta que se percibe que el color del piso es negro, y no se para de contar hasta que el sensor de contacto es presionado por el zócalo situado al final del recorrido.

Figura 4.22: *Imagen del circuito del experimento 12.*

4.12.2. Preparación física del experimento

Para preparar la pista en la que se va a probar el programa, hay que utilizar 4 cartulinas blancas, cinta aislante negra, un obstáculo, que en este caso es un botellín de agua, y un zócalo de una altura que llegue al sensor de contacto pero lo suficientemente bajo como para que no sea percibido por el sensor de ultrasonidos. En este caso como zócalo se han utilizado dos paquetes de tabaco, con algo pesado dentro, envueltos en cinta aislante.

Una vez montado todo el circuito deberá situarse el NXT a la altura del comienzo de la línea, pero posicionando el sensor de color sobre suelo blanco, a la derecha de la línea.

4.12.3. Estructura de los programas

La estructura básica de este último experimento (ver Fig. 4.23) se empieza girando a la izquierda, de forma que no se tome la primera referencia de tiempo hasta que se detecte suelo negro. En el momento en que es detectado se toma la referencia, se da la orden de girar a la derecha y se entra automáticamente al bucle principal del programa. En este bucle primero se comprueba si está pulsado el sensor de contacto para, en caso afirmativo, tomar la segunda referencia de tiempo y parar los motores. Después se comprueba si el sensor de ultrasonidos detecta algo a menos de 15 centímetros, y en caso de ser así se inicia la siguiente secuencia: Parar motores, hacer sonar una exclamación, girar sobre sí mismo hacia la derecha, girar alrededor del obstáculo dejando éste a la izquierda mientras el suelo sea blanco, seguir recto al detectar la línea negra y finalmente al volver a detectar blanco girar a la derecha, de forma que al volver a detectar negro por segunda vez el robot esté posicionado sobre la línea para continuar. Finalmente sale del bucle del sensor de ultrasonidos.

Una vez comprobado el sensor de ultrasonidos, habiendo pasado por el bucle o no, si se detecta el suelo negro se gira a la derecha, y si por el contrario se detecta blanco se gira hacia la izquierda. Una vez terminado el ciclo se vuelve al principio del bucle principal.

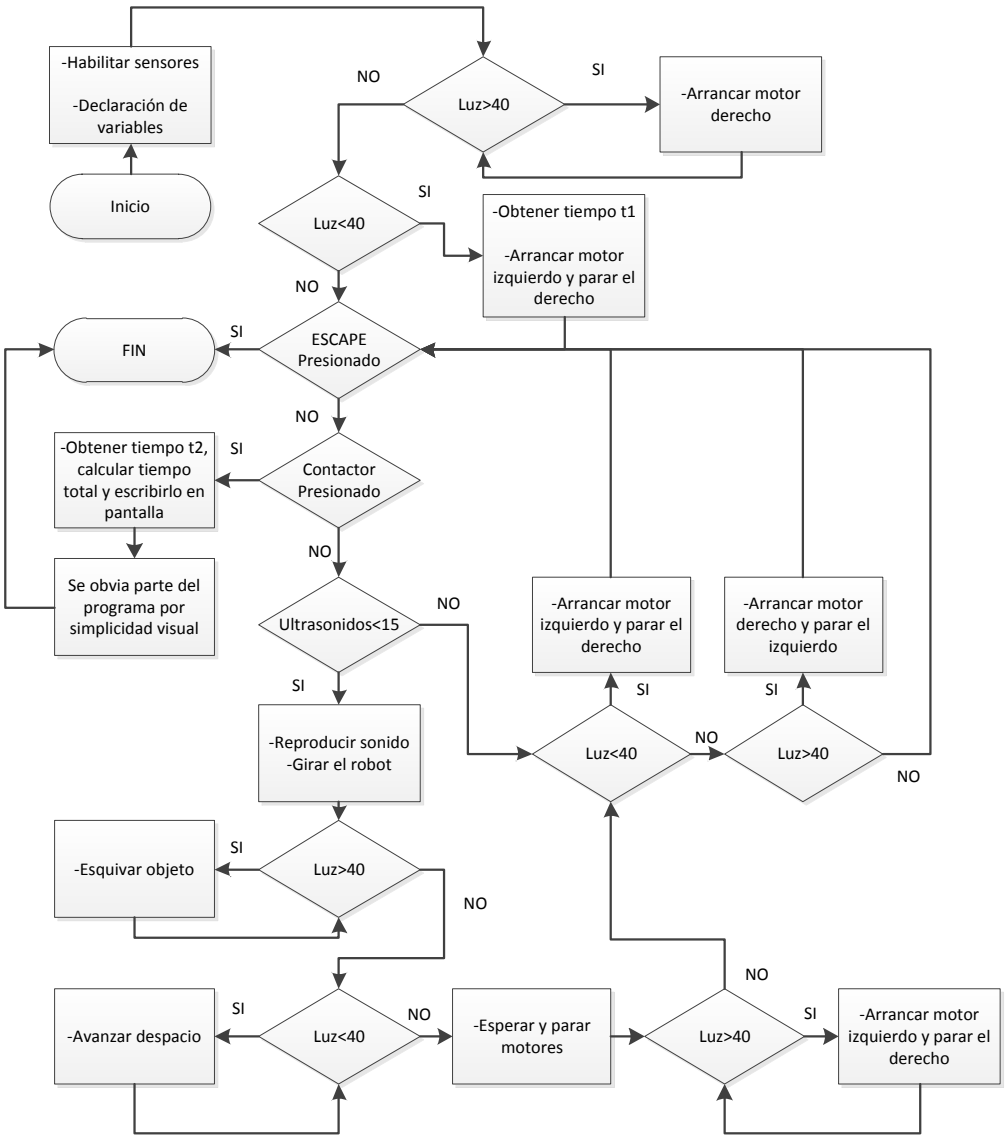

Figura 4.23: *Diagrama de flujo experimento 12.*

4.12.4. Resultados del experimento

Los resultados de este experimento no se deben medir únicamente por los tiempos presentados dado que aunque los resultados obtenidos por el programa en leJOS son parecidos a los obtenidos en los programas en NXT-G y NXC, estos dos últimos son mucho más precisos y fieles a la trayectoria propuesta. Es más, la anchura de la línea del experimento es equivalente al ancho de dos tiras de cinta aislante (aprox. 19mm. cada una) porque los programas en leJOS, leJOS multihilo y Matlab no son capaces de seguir una línea de menor anchura. Estos programas son incapaces de detectar la línea, frenar y corregir la trayectoria antes de salirse de ella. Sin embargo, los programas en NXT-G y NXC son capaces de seguir líneas de un grosor mucho menor al de una tira de cinta, por lo que si se dispusiera de más potencia, podrían ser mucho más veloces de lo que se entiende por los resultados.

También se observa que los implementados en NXT-G y NXC responden mucho antes ante un obstáculo, lo que denota que la comunicación entre el sensor de ultrasonidos y los motores es mucho más rápida.

Ensayos	NXT-G	NXC	leJOS	leJOS Multihilo	Matlab
Ensayo 1	32839	31049	32689	36783	55702
Ensayo 2	32735	31575	35098	39347	59211
Ensayo 3	33082	31400	35243	37761	57020
Ensayo 4	33620	31976	46140	37784	62387
Ensayo 5	33152	31883	37946	38905	56218

Tabla 4.26: *Tiempos del experimento 12 (en milisegundos).*

	NXT-G	NXC	leJOS	leJOS Multihilo	Matlab
Media	33085,6	31576,6	37423,2	38116	58107,6
Mediana	33082	31575	35243	37784	57020
Valor menor	32735	31049	32689	36783	55702
Valor mayor	33620	31976	46140	39347	62387
Amplitud rango	885	927	13451	2564	6685

Tabla 4.27: *Media, mediana, mín, máx y amplitud de rango de los resultados del experimento 12.*

4.12.5. Conclusiones del experimento

Las conclusiones principales respecto a este experimento son tres:

Los programas en NXT-G y NXC son capaces de seguir una línea de un anchura menor que el de una cinta aislante y a velocidad máxima, mientras que los otros tres ni siquiera son capaces de seguir a velocidad máxima una línea del doble de anchura.

Los programas en NXT-G y NXC son mucho más precisos respecto a la trayectoria que los demás, y podrían seguir siéndolo a una velocidad mucho mayor a la máxima que ofrece el robot.

Los programas en NXT-G y NXC tienen una comunicación más rápida entre el sensor de ultrasonidos y los motores.

CAPÍTULO 5

CONCLUSIONES

En este capítulo se recogerán y analizarán las conclusiones de los experimentos en su conjunto. De este modo se podrá diferenciar qué lenguajes de programación positiva o negativamente en cada campo. También se realizará un análisis general de cada lenguaje, donde se plasmarán los puntos fuertes y flacos de cada uno.

5.1. Conclusiones generales

Ante las pruebas realizadas sobre la precisión de los motores al parar después de rotar un ángulo concreto, Matlab y NXT-G son los que tienen una respuesta más precisa. El resto de los programas no son fiables en este aspecto. Sin embargo Matlab acumula error al hacerlo con paradas intercaladas, y leJOS, leJOS multihilo y NXT-G son los fiables.

Respecto a la velocidad de procesamiento, cuando a operaciones sencillas se refiere, el ganador indiscutible es el leJOS. En caso de complicación en las operaciones, se iguala la velocidad de leJOS con la de Matlab, siempre y

cuando este último no esté en contacto con el NXT durante el proceso. En caso de estar en contacto, la velocidad baja drásticamente.

Se ha comprobado que la velocidad máxima alcanzable con los comandos habituales es la misma utilizando cualquiera de los lenguajes.

En relación al sensor de ultrasonidos todos los programas responden igual. Es recomendable posicionarlo con una inclinación de unos 15 grados positivos, puesto que en caso contrario pueden aparecer anomalías a partir de los 45 centímetros. Además se documenta que incrementa el resultado en 2 centímetros cuando detecta algo entre los 3 y los 22 centímetros, aunque a partir de los 24 se corrige, volviendo a aparecer desde los 169 centímetros un incremento de un centímetro. Por último, al sobrepasar los 2 metros el error se incrementa en otro centímetro más.

A la hora de frenar respondiendo a un dato enviado por el sensor de ultrasonidos, NXT-G y NXC son los más eficaces, siendo el primero algo más rápido. Y si el dato es enviado por el sensor de color vuelven a ganar los dos mismos, esta vez destacando claramente el NXT-G sobre el NXC. Sin embargo, al testar el comportamiento de los programas teniendo que cambiar las órdenes a sus motores ante los repetidos cambios de color el NXC es mucho más rápido que el NXT-G. Es algo bastante inesperado teniendo en cuenta la conclusión anterior. Se deduce que el NXC optimiza las transiciones entre los estados de los motores, haciendo al robot más ágil.

En relación a la iluminación, se observa que menos el de Matlab, todos los demás programas son capaces de cambiar de color un LED en unos 170 ó 180 milisegundos. De todas formas, esto es demasiado tiempo, y es debido a que para encender el LED se activa todo el sistema de detección de color. Existe una función de la toolbox de la universidad de Aachen que sirve para encender las lámparas enchufando el sensor de color a los puertos de los motores, pero únicamente sirve para encenderlo o apagarlo, sin posibilidad de elegir los colores.

Y finalmente, en las pruebas de reconocimiento de color se descubre que los lenguajes que utilizan el Firmware oficial de LEGO son capaces de reconocer 6 colores, mientras que los que utilizan la máquina virtual de leJOS reconocen 12 además de la ausencia de color. Hay que comentar que para un buen reconocimiento, en cualquiera de los lenguajes, el sensor debe estar a una distancia aproximada de un centímetro del suelo.

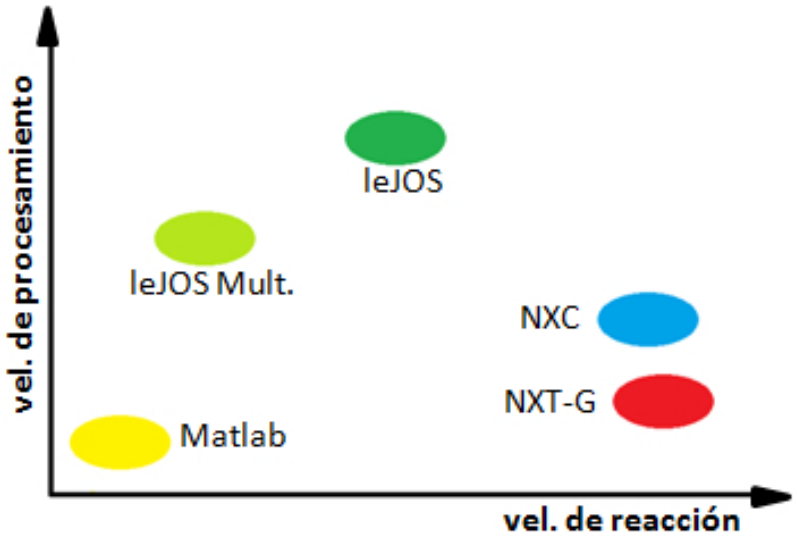

Figura 5.1: *Gráfica de los lenguajes de programación en función de la velocidad de procesamiento y la de reacción.*

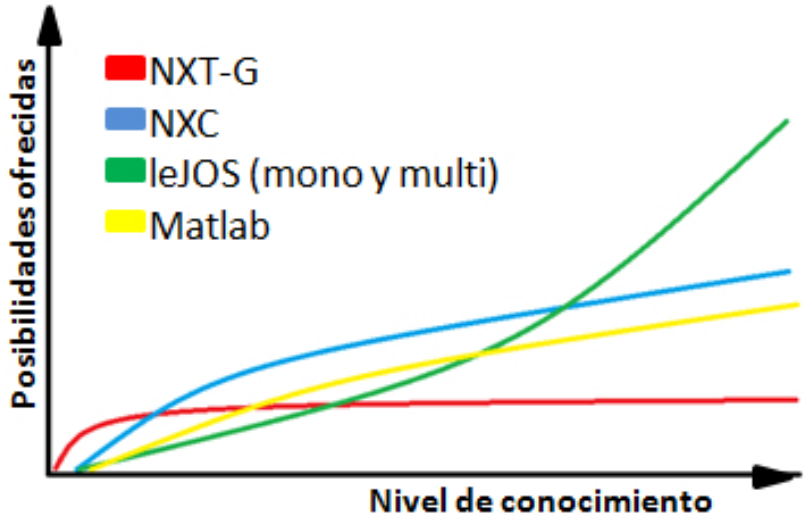

Figura 5.2: *Gráfica sobre las posibilidades ofrecidas en función del nivel de conocimiento del lenguaje de programación.*

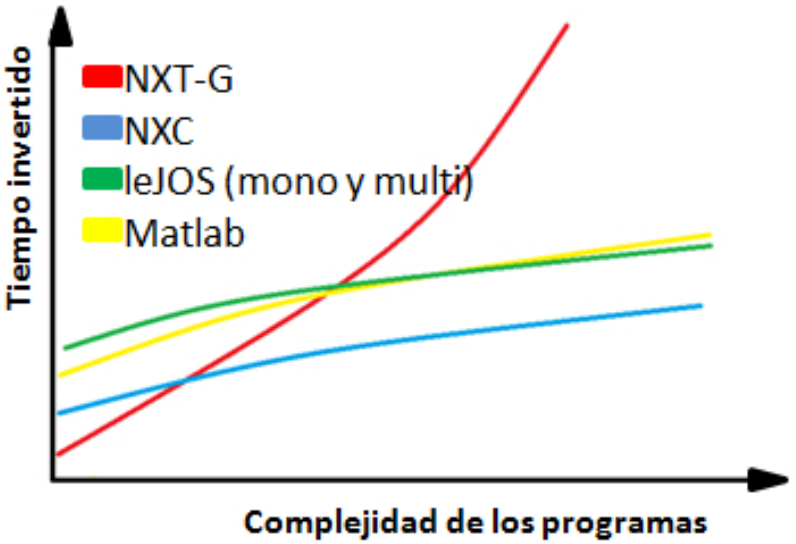

Figura 5.3: *Gráfica sobre el tiempo invertido en programar en función de la complejidad de los programas.*

5.2. Aspectos destacables de cada lenguaje

5.2.1. NXT-G

Tiene una buena precisión tanto a la hora de parar un motor después de rotar un ángulo concreto, como a la de pararlo después de varias rotaciones con paradas intercaladas.

Es el lenguaje más lento en relación al procesamiento junto a Matlab, sin contar Matlab siendo teleoperado, que lógicamente es el más lento con diferencia.

Es el más rápido frenando en respuesta a un dato tomado tanto por el sensor de ultrasonidos como por el de color. Aunque sea superado por el NXC cuando es necesario hacer cambios rápidos y bruscos en los motores atendiendo a los datos del sensor de color.

5.2.2. NXC

Tiene una mala precisión tanto a la hora de parar un motor después de rotar un ángulo concreto, como a la de pararlo después de varias rotaciones con paradas intercaladas. En todos los casos rota más de lo que se le ha pedido.

Es el lenguaje de procesamiento más rápido de los que utilizan el Firmware oficial de LEGO, aunque dista mucho de los lenguajes con Firmware/máquina virtual de leJOS.

Hay que destacar que es el lenguaje que peor estabilidad direccional tiene utilizando órdenes sencillas, sin entrar en regulación de motores.

Tiene buenos resultados frenando en respuesta a un dato tomado tanto por el sensor de ultrasonidos como por el de color, aunque no llega a ser tan rápido como el NXT-G. Y supera a todos con diferencia cuando es necesario hacer cambios rápidos y bruscos en los motores atendiendo a los datos del sensor de color.

5.2.3. leJOS

Tiene una mala precisión a la hora de parar un motor después de rotar un ángulo concreto, no llegando a cubrir en ninguno de los casos el ángulo deseado. Sin embargo es el lenguaje que tiene una mejor precisión a la hora de pararlo después de varias rotaciones con paradas intercaladas.

Es con mucha diferencia el lenguaje más rápido en cuanto a procesamiento de operaciones simples. También es el más rápido, junto a Matlab (siempre y cuando este último no esté compartiendo datos con el robot durante el proceso), procesando operaciones más complejas.

Tiene unos resultados aceptables frenando en respuesta a un dato tomado tanto por el sensor de ultrasonidos como por el de color, aunque no es comparable al NXT-G ni al NXC. Y es demasiado lento cuando es necesario hacer cambios rápidos y bruscos en los motores atendiendo a los datos del sensor de color, tardando más de seis veces más que el NXC.

Hay que destacar que en este lenguaje no se ha conseguido dejar los motores en flotación en ningún caso.

5.2.4. leJOS multihilo

Tiene una pésima precisión a la hora de parar un motor después de rotar un ángulo concreto, no llegando a cubrir, en ninguno de los casos, el ángulo deseado. Sin embargo es uno de los mejores en el momento de pararlo después de varias rotaciones con paradas intercaladas.

Tarda más del doble de tiempo que leJOS en procesamiento de operaciones simples. Aun así supera con mucha diferencia a los lenguajes que utilizan el Firmware oficial de LEGO. Sin embargo, en el procesamiento de operaciones más complejas, el NXC también le supera.

Tiene unos resultados notablemente malos frenando en respuesta a un dato tomado tanto por el sensor de ultrasonidos como por el de color. Y es demasiado lento cuando es necesario hacer cambios rápidos y bruscos en los motores atendiendo a los datos del sensor de color, aunque sorprendentemente supera a leJOS monohilo.

Hay que destacar que en este lenguaje no se ha conseguido dejar los motores en flotación en ningún caso.

5.2.5. Matlab

Tiene la mejor precisión a la hora de parar un motor después de rotar un ángulo concreto. Sin embargo es el peor en el momento de pararlo después de varias rotaciones con paradas intercaladas.

Es, con diferencia, el lenguaje más lento en relación al procesamiento mientras comparte información con el robot. Aunque durante el procesamiento no se comparta información con el robot, seguiría siendo el más lento a la par del NXT-G.

Tiene los peores resultados frenando en respuesta a un dato tomado tanto por el sensor de ultrasonidos como por el de color. Y vuelve a ser el peor

haciendo cambios rápidos y bruscos en los motores atendiendo a los datos del sensor de color.

Cabe destacar, que frente a sus malos resultados en algunos ámbitos, tiene la ventaja de que es teleoperado.

CAPÍTULO 6

TRABAJOS FUTUROS

Hay una infinidad de posibilidades. Dado que el NXT 2.0 es relativamente nuevo, y además contando con que cada poco tiempo se van creando nuevos sensores compatibles con él, se pueden hacer muchos proyectos e investigaciones.

Por ejemplo, podría ser interesante hacer un estudio análogo a éste pero utilizando los sensores legacy del RCX, predecesor del NXT. Esto arrojaría luz sobre qué lenguajes de programación están preparados para utilizarlos, cómo hacerlo, y ayudaría a ver si los sensores legacy tienen un rendimiento mejor, igual o peor conectados al ladrillo del NXT que los propios.

También sería interesante completar este trabajo utilizando otros sensores diferentes del propio NXT, o de la serie Hitechnic, como podrían ser el sensor magnético, el sensor buscador de infrarrojos, el sensor electro-óptico detector de proximidad, el sensor receptor de infrarrojos, el sensor brújula, el sensor giroscópico, etc.

Otra opción diferente sería estudiar más profundamente el funcionamiento de los sensores, tanto a nivel software como hardware, ya que esto ayudaría

a mejorar y crear nuevas funciones para los sensores.

Tampoco estaría mal hacer estudios sobre las capacidades, sensores, actuadores, etc. del NXT en comparación a sus competidores en el mercado como podrían ser los robots de Fischertechnik, el Protobot VEX.

BIBLIOGRAFÍA

[1] Laurence Vanhelsuwé. «*La biblia de Java*». Anaya Multimedia, 1997.

[2] Matthias Paul Scholz. «*Advanced NXT: the Da Vinci inventions book*». Apress, 2007.

[3] John C. Hansen. «*Lego Mindstorms NXT Power Programming: Robotics in C*». Variant Press, 2007

[4] Emilson Pereira Leite. «*Matlab - Modelling, Programming and Simulations*». A B M Nasiruzzaman.

[5] Página web donde se explica como instalar leJOS. «*http://robotlego.wordpress.com/2008/06/15/tutorial-como-instalar-y-correr-java-con-un-robot-lego-mindstorms-nxt-usando-eclipse-metodo-alternativo/*». Página activa en diciembre de 2011.

[6] Página web oficial de Mindstorm de la Universidad de Aachen. «*http://www.mindstorms.rwth-aachen.de/*». Página activa en diciembre de 2011.

[7] Pagina sobre como instalar la toolbox de RWTH. «*http://people.clemson.edu/ñwatts/engr141/instructions/*». Página activa en diciembre de 2011.

[8] Página web oficial de Lego.
«*http://mindstorms.lego.com/en-us/bluetooth/Default.aspx?domainredir=www.mindstorms.com*». Página activa en diciembre de 2011.

[9] Manual de NXC en inglés.
«*http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_tutorial.pdf*». Página activa en diciembre de 2011.

[10] Tutorial de nxc en castellano.
«*http://www.iesantoniodenebrija.es/robotica/images/stories/departamento/NXC/nxccastellano.pdf*». Página activa en diciembre de 2011.

[11] Página oficial de leJOS.
«*http://lejos.sourceforge.net/*». Página activa en diciembre de 2011.

[12] Blog con información sobre LEGO Mindstorms.
«*http://blog.electricbricks.com*». Página activa en diciembre de 2011.

[13] Tutorial de Java en Inglés.
«*http://docs.oracle.com/javase/tutorial/essential/regex/intro.html*». Página activa en diciembre de 2011.

[14] Manual de Java.
«*http://www.webtaller.com/manual-java/clases.php*». Página activa en diciembre de 2011.

[15] Presentación sobre multihilo en java.
«*http://www.slideshare.net/czelada/hilos-en-java*». Página activa en diciembre de 2011.

[16] Guía de Iniciación al Lenguaje JAVA.
«*http://zarza.usal.es/fgarcia/doc/tuto2/Index.htm*». Página activa en diciembre de 2011.

[17] Multi threading en JAVA.
«*http://www2.sys-con.com/itsg/virtualcd/java/archives/0301/cunninham/index.html*». Página activa en diciembre de 2011.