Universidad de Xalapa
Ingeniería en Sistemas de Cómputo Administrativo
Ingeniería en Electrónica y Comunicaciones

Informe Técnico

Construcción y Programación de robots limpiadores con Lego Mindstorms y Java

Julio César Sandria Reynoso
Rafael Alarcón Domínguez
Eduardo Salvador Muñoz Castillo
Erick Salvador Cárdenas Bermúdez
Miguel Ángel Alonso Lechuga
Christian Salas Borbolla

Diciembre 2004

Universidad de Xalapa
Informe Técnico de Ingeniería en Sistemas y Electrónica
UX-ISyE-IT-2004-10-15

Construcción y Programación de robots limpiadores con Lego Mindstorms y Java

M.I.A. Julio César Sandria Reynoso
Técnico Académico del Instituto de Ecología, A.C.
Catedrático de la Universidad de Xalapa
sandriaj@ecologia.edu.mx

Rafael Alarcón Domínguez
Eduardo Salvador Muñoz Castillo
Erick Salvador Cárdenas Bermúdez
Miguel Ángel Alonso Lechuga
Christian Salas Borbolla
Estudiantes de la carrera de
Ingeniería en Sistemas de Cómputo Administrativo
de la Universidad de Xalapa

Diciembre 2004

Universidad de Xalapa
Carretera Xalapa-Veracruz Km. 2
Col. Ánimas, C.P. 91190
Xalapa, Veracruz, México
Tel. (228) 812-8191 y 812-8192
http://www.ux.edu.mx/

Resumen

En este Informe Técnico se muestra el proceso de construcción de robots limpiadores usando paquetes Lego Mindstorms Robotics Invention System 2.0 y su programación con el lenguaje Java sobre leJOS (Lego Java Operating System). Los dos robots mostrados son los ganadores del segundo y tercer lugar del Primer Torneo Mexicano de Robots Limpiadores organizado por el Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), la Universidad Veracruzana (UV) y la Sección Puebla del Institute of Electric and Electronic Engineers (IEEE) en las instalaciones del INAOE en Tonantzintla, Puebla, en agosto de 2004. Dichos robots fueron construidos y programados por los autores de este informe, profesor y alumnos de octavo semestre de la carrera de Ingeniería en Sistemas de Cómputo Administrativo de la Universidad de Xalapa (UX). Este trabajo es resultado del curso corto *Programación en Java con robots Lego Mindstorms* y del curso de asignatura *Inteligencia Artificial*, ambos impartidos por el primer autor en la UX.

Contenido

Lista de figuras

Lista de tablas

1 Introducción

En los últimos años, la robótica, un área multidisciplinaria, se ha desarrollado en universidades de todo el mundo, promoviéndose cada vez más su desarrollo mediante concursos de robótica de muchos tipos. A pesar de que el establecimiento de un laboratorio de robótica está fuera del presupuesto de muchas universidades por los costos de robots manipuladores y móviles comerciales, es posible hacer prácticas de robótica con paquetes comerciales de bajo costo y construcciones propias.

1.1 Antecedentes

En este trabajo se muestran los avances que se han realizado en materia de robótica usando el paquete comercial Lego Mindstorms Robotics Invention System 2.0 en la Universidad de Xalapa (UX).

Desde finales de 2003, se empezó a preparar el material para impartir un curso de programación en Java usando como herramienta de apoyo robots Lego Mindstorms. En mayo de 2004 se impartió por primera vez en la UX el Curso-Taller *Programación en Java con Robots Lego Mindstorms*, parte del material de ese curso se usó también en la asignatura *Inteligencia Artificial* (febrero a julio de 2004), en donde, como trabajo final se pidió a los alumnos construir un robot limpiador con la finalidad de seleccionar los mejores trabajos para participar en el *Primer Torneo Mexicano de Robots Limpiadores* que se llevaría a cabo en agosto del mismo año.

Por una parte, el uso de robots Lego en el curso-taller de programación en Java fue una estrategia novedosa en Xalapa de ofrecer un curso teórico-práctico de programación orientada a objetos en Java.

Por otra parte, el uso de robots Lego en la asignatura de inteligencia artificial fue también una estrategia para ofrecer a los alumnos una forma más práctica para aprender algunos conceptos de inteligencia artificial.

Finalmente, la idea de usar robots Lego fue promover la robótica en la Universidad de Xalapa.

1.2 Objetivos

La razón de construir y programar robots limpiadores fue para formar equipos de estudiantes de Ingeniería en Sistemas Computacionales Administrativos y de Ingeniería Electrónica y Comunicaciones de la Universidad de Xalapa para participar en el *Primer Torneo Mexicano de Robots Limpiadores*, organizado por el Instituto Nacional de Astrofísica Óptica y Electrónica (INAOE), la Maestría en Inteligencia Artificial de la Universidad Veracruzana y la Sección Puebla del Institute of Electric and Electronic Engineers (IEEE), en las instalaciones del INAOE en Tonantzintla, Puebla, en agosto de 2004.

Los resultados obtenidos en dicho torneo fueron muy satisfactorios para los autores, ya que los dos robots que se documentan en este informe fueron ganadores del segundo y tercer lugar entre 18 robots concursantes de diversas instituciones de educación superior del país (Tabla 1).

Tabla 1. Equipos e Instituciones participantes en el Primer Torneo Mexicano de Robots Limpiadores 2004

	Equipo	Institución
1	CYR	INAOE
2	Los Victorinos	IPN
3	Limbot	UPAEP
4	Robotrónica	ITE
5	El Pentágono	U. Xalapa
6	Dralion	U. Xalapa
7	IRICAT	UACJ
8	Clon-team	BUAP-INAOE
9	Aguibot	UPAEP
10	Mexicanos al grito -Zorro1	ITQ
11	Mexicanos al grito -Zorro2	ITQ
12	Macrobots	MIA-UV
13	UTM	UTM
14	GIM	INAOE
15	Fegen	U. Xalapa
16	Informática	UV
17	Instrumentación	UV
18	ML-HJ-DISASTER -04	INAOE

2 Robots Lego Mindstorms

El paquete Lego Mindstorms Robotics Invention System para construir y programar robots ofrece una gran versatilidad, desde ser un juguete para niños a todo un sistema de desarrollo de robots que es utilizado por millones de aficionados a la robótica y la ingeniería. Su relativamente bajo costo[1], facilidad para construir diferentes modelos de robots y la variedad de lenguajes de programación disponibles los hacen muy apropiados para gente que se inicia y hasta para investigaciones serias en robótica e inteligencia artificial.

Figura 1. Lego Mindstorms *Robotics Invention System 2.0*

2.1 RIS 2.0

El Lego Mindstorms *Robotics Invention System* 2.0 es un paquete (Figura 1) que contiene:

- Una microcomputadora.
- Un disco compacto con el Software RIS 2.0 (*Robotics Invention System 2.0*).

[1] A diciembre de 2004 su precio al público estaba alrededor de 2900 pesos mexicanos, equivalentes a aproximadamente 250 dólares norteamericanos.

- 718 piezas entre las cuales se encuentran:
  - 2 motores
  - 2 sensores de contacto
  - 1 sensor de luz
- Una guía ilustrada para construir robots, llamada Constructopedia.
- Una torre con transmisor infrarrojo (Torre IR).

La Constructopedia es una guía ilustrada que ofrece sugerencias, ideas y tips para construir robots que ayudan a iniciarse con los retos o desafíos incluidos en el software RIS 2.0, así como para hacer invenciones propias.

2.2 Requerimientos mínimos

Para instalar el Software RIS 2.0 se requiere una computadora personal (PC) con al menos los requerimientos mostrados en la Tabla 2.

Tabla 2. Requerimientos mínimos para el Software RIS 2.0

Sistema operativo	Windows 98/Me
Procesador	Pentium II 233 MHz
Memoria	32 MB
Espacio disponible en disco duro	115 MB
Ratón	Compatible con Windows 98
Sonido	Dispositivo de sonido compatible con Sound Blaster 16 para Windows
CD-ROM	CD 8x
Video	800 x 600 SVGA con 4 MB RAM
Colores	16 Bit
Puerto USB	Un puerto USB disponible para el transmisor infrarrojo
Modem (opcional)	28.8 Kbps
Navegador de Internet (opcional)	Netscape Navigator o Microsoft Internet Explorer

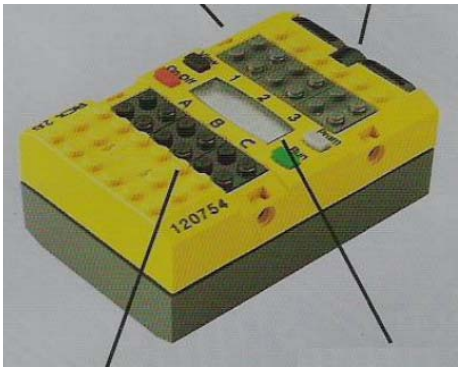
2.3 El RCX

El RCX (Robotics Command System) es una pieza LEGO programable. Como se puede ver en la Figura 2, el RCX tiene tres puertos de entrada, tres puertos de salida, cuatro botones de control, una pantalla de cristal líquido y un transmisor infrarrojo. También tiene un microprocesador para correr los programas, memoria interna para guardar un pequeño sistema operativo y los programas y una bocina integrada para producir pitidos y tonos.

Dado que el RCX es una pieza LEGO, con rebordes y agujeros, éstos le permiten ensamblarse con otras piezas LEGO.

Los puertos de entrada se usan para conectar sensores de contacto y de luz (así como sensores de rotación y temperatura, que no incluye el paquete), son de color gris y están numerados como 1, 2 y 3.

Puertos de entrada Transmisor infrarrojo



Puertos de salida Pantalla LCD

Figura 2. El RCX (Robotics Command System)

Los puertos de salida se usan para conectar motores (así como luces y otros dispositivos de salida, que no incluye el paquete), son de color negro y están nombrados como A, B y C.

Adicionalmente, el RCX tiene tres sensores internos: un temporizador, un contenedor de mensajes para el RCX (para recibir mensajes enviados por otras unidades RCX), y variables definidas por el usuario.

Para programar el RCX, la compañía Lego incluye un disco compacto con el Software RIS 2.0, que contiene un ambiente gráfico para programar visualmente mediante bloques (Figura 3).

Por limitaciones inherentes al entorno de programación del RIS 2.0, que fue diseñado para que niños de 9 años en adelante pudieran programar fácilmente sus creaciones, es necesario recurrir a un lenguaje de programación para crear programas más complejos y de fácil mantenimiento, como C, Java o Visual Basic.

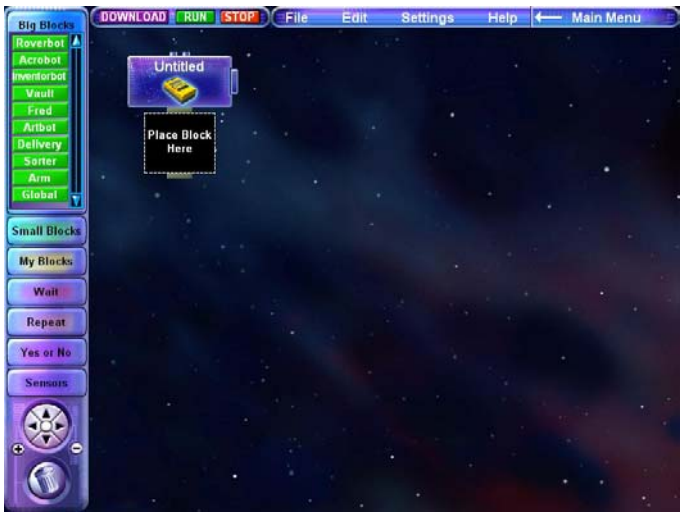


Figura 3. Entorno de programación visual del RIS 2.0

2.3.1 Baterías

La fuente de energía del RCX son 6 baterías AA. Lego recomienda usar baterías alcalinas y no mezclar diferentes tipos de baterías. Cuando las baterías están bajas, el icono se muestra en la pantalla del RCX.

2.3.2 Motores

Los motores que incluye Lego en el paquete son de 9 Volts, y con estos se pueden construir robots móviles, brazos manipuladores y combinaciones de ambos.

Para conectar un motor al RCX, se necesita usar un cable negro con platos conectores, se conecta un extremo del cable al motor y el otro extremo a un puerto de salida negro, como lo ilustra la Figura 4.

El lado en el que se conecte el cable en el motor afecta la dirección de éste. Cuando se construyen modelos basados en una guía, es importante conectar los cables como lo indica la misma.

Figura 4. Conexión de motores

2.3.3 Sensores

Con los sensores, los robots construidos pueden percibir el ambiente, de modo que puedan reaccionar a la luz, temperatura, el movimiento o contacto con un objeto. Los sensores de contacto y de luz se pueden conectar como lo ilustra la Figura 5.

Figura 5. Conexión de sensores

Cuando se construyen modelos basados en una guía, es importante conectar los sensores a los puertos mostrados en las instrucciones de construcción.

2.3.4 Botones

Los botones permiten controlar el RCX y sus programas. La ubicación de los botones se muestra en la Figura 6.

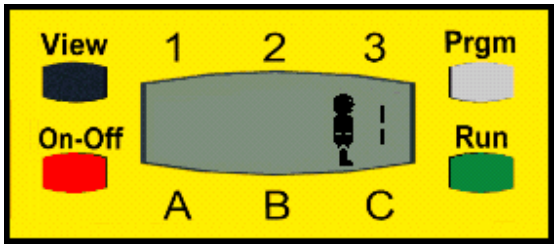


Figura 6. Botones y pantalla del RCX

El botón **On·Off** enciende y apaga el RCX. Los otros tres botones funcionan solamente cuando el RCX está encendido.

El botón **Prgm** (abreviación de program) permite cambiar entre los cinco programas que se pueden cargar en el RCX. El número del programa seleccionado aparece a la derecha del icono en forma de "persona pequeña" mostrado en la pantalla.

El botón **Run** inicia y termina el programa seleccionado. En el modo "run", la "persona pequeña" en la pantalla aparece corriendo.

El botón **View** (activo cuando el firmware Lego está cargado) permite obtener información de los sensores y motores. Se pueden ver las lecturas de sensor en los puertos de entrada 1, 2 ó 3, y dirección de motor en los puertos de salida A, B o C.

2.4 Torre IR

La Torre IR establece un enlace inalámbrico entre una computadora con Windows o Linux y el RCX, como se muestra en la Figura 7. La Torre IR usa señales infrarrojas para enviar mensajes, por lo tanto, a través de ésta se pueden cargar programas desde la computadora al RCX. Estos programas pueden ser ejecutados por el RCX.

Figura 7. Torre IR

Para que ocurra la comunicación, el RCX y la Torre IR deben ser capaces de "verse" uno a la otra. Una distancia de 10 a 12 cm es la adecuada para la carga de programas, aunque, en condiciones óptimas de luz, la comunicación es aún posible a distancias de hasta 30 m.

3 Programación en Java

Para programar el RCX con el lenguaje Java, se recurrió a leJOS. En este capítulo se describen los pasos necesarios para programar en Java con lejos, no es un manual de programación en Java.

3.1 leJOS

leJOS es un pequeño sistema operativo que ocupa aproximadamente 16 Kb de los 32 KB de RAM del RCX. leJOS está basado en Java y fue diseñado para el RCX de Lego Mindstorms. leJOS son las siglas de Lego Java Operating System.

leJOS fue producto del proyecto TinyVM, leJOS contiene una máquina virtual para bytecodes Java y software adicional para cargar y ejecutar programas Java. Estas son algunas de las características ofrecidas por la versión 2.1.0 de leJOS:

- Lenguaje orientado a objetos (Java).
- Multihilos (tareas).
- Arreglos, incluyendo multidimensionales.
- Recursión.
- Sincronización
- Excepciones.

Estas características también las proporciona TinyVM. Pero, adicionalmente a éstas, leJOS también ofrece:

- Una versión para Windows, que no requiere CygWin.
- Operaciones de punto flotante (dobles truncados a 32 bits).
- Constantes String.
- Asignación de longs a ints y viceversa.
- Marcas de referencia en la pila (lo que hace factible implementar recolección de basura).
- Descarga multi-programa.
- Clase java.lang.Math con funciones sin, cos, tan, atan, pow, etc.
- Más APIs.

TinyVM sigue siendo simple y es otra opción a usar en vez de leJOS. TinyVM ocupa 7 Kb menos en el RCX.

3.2 Compatibilidad

LeJOS es compatible con:

- RCX 1.0/2.0 en RIS 1.0/1.5/2.0
- Torre serial IR en RIS 1.0/1.5
- Torre IR USB en RIS 2.0

3.3 Requerimientos

Para usar lejos se requiere una computadora con:

- Sistema operativo Linux o Windows.
- Java Development Kit (JDK). leJOS trabaja con el JDK1.1.x a 1.4.x de Sun Microsystems (http://java.sun.com).

3.4 Instalación de leJOS en Windows

Primero es necesario asegurarse de tener instalado el JDK. Si no es así, se puede obtener del sitio de Sun Microsystems e instalarlo; por ejemplo, si se obtiene el programa j2sdk-1_4_2_03-windows-i586-p.exe (usado en este trabajo), al ejecutarlo y seguir los pasos de instalación, éste sugiere instalarlo en el directorio C:\j2sdk1.4.2_03, se puede aceptar dicho directorio o instalarlo en otro directorio, por ejemplo en C:\Java\j2sdk1.4.2_03.

Bajar Win32 leJOS del sito de leJOS (http://lejos.sourceforge.net, para este trabajo se usó la versión 2.1.0) y seguir los siguientes pasos:

- Descomprimir el archivo ZIP bajado. Este archivo proporciona ya un directorio lejos. Por ejemplo, a una carpeta C:\Java\lejos.
- Abrir una ventana de comandos (símbolo de sistema o consola DOS) y cambiar el directorio actual al de lejos:

```
c:                      (cambiar a disco C:)
cd C:\Java\lejos        (cambiar directorio actual a
                        C:\Java\lejos)
```

- Establecer la variable de ambiente RCXTTY con el comando:

```
set RCXTTY=usb          (para Torre IR USB)
set RCXTTY=COM2         (para Torre IR serial conectada en el puerto
                        COM2)
```

- Los directorios bin de JDK y leJOS deben estar en la variable de ambiente PATH. Si no lo están, ejecute los comandos:

```
set path=%path%;C:\Java\j2sdk1.4.2_03\bin
set path=%path%;C:\Java\lejos\bin
```

Bajo Window 95/98, se puede crear un archivo por lotes (batch) que establezca estas variables, por ejemplo, en un editor de texto escriba las siguientes líneas:

```
set RCXTTY=usb
set PATH=%PATH%;C:\Java\j2sdk1.4.2_03\bin;C:\Java\lejos\bin
```

y guarde el archivo como `C:\Java\lejos\variables.bat`.

puede especificar dicho archivo por lotes en las propiedades de programa de un acceso directo a una ventana de comandos, también puede agregar tales líneas al archivo C:\AUTOEXEC.BAT, o cada vez que abra una nueva ventana de comandos ejecutar el archivo por lotes.

3.5 Ejemplo "hola mundo"

Escriba el siguiente programa en un editor de texto, y guárdelo con el nombre C:\Java\lejos\ejemplos\holamundo\HolaMundo.java:

```
import josx.platform.rcx.*;

public class HolaMundo
{
  public static void main (String[] aArg) throws Exception
  {
     LCD.clear();              // Borra la pantalla
     TextLCD.print ("hola");   // Imprime "hola" en la pantalla
     Thread.sleep(2000);       // Espera 2000 milisegundos
     TextLCD.print ("mundo"); // Imprime "mundo" en la pantalla
     Thread.sleep(2000);       // Espera 2000 milisegundos
  }
}
```

Abra una ventana de comandos, inicie las variables de ambiente y cambie el directorio actual a C:\Java\lejos\ejemplos\holamundo, compile el programa con el comando:

```
lejosc HolaMundo.java
```

si el programa no tienen ningún error, lejosc no muestra ningún mensaje. Enlace y cargue el programa en el RCX con el comando:

```
lejos HolaMundo
```

mientras se ejecuta esto, en la línea de abajo se muestra el porcentaje del programa cargado en el RCX, a la vez que en el RCX se muestra el número de bytecodes transferidos. Al terminar de cargarse el programa, se muestra 100% y el RCX emite dos pitidos.

Ejecute el programa oprimiendo el botón Run del RCX y observe lo que se muestra en la pantalla del mismo.

3.6 Compilación, enlazado y ejecución

Para compilar las clases Java programadas para el RCX, se debe usar `lejosc` en vez de `javac`. Si el JDK del PATH es JDK 1.1, entonces se debe usar `lejosc1`.

Nótese que `lejosc` no es un compilador de Java. Es simplemente un pequeño programa que llama a `javac` después de establecer un valor apropiado para `-bootclasspath`. Si se prefiere un compilador de Java diferente a `javac`, se debe definir la variable de ambiente JAVAC.

La manera básica de usar el enlazador es estando seguro de que los archivos `.class` pueden ser localizados en la variable de ambiente CLASSPATH, y ejecutar `lejos` como se ejecutaría `java`, por ejemplo:

```
lejos MiClasePrincipal
```

Esta sentencia enlaza todas las clases que deberán ser usadas por un programa con punto de entrada en `MiClasePrincipal`. El resultante archivo binario enlazado es cargado al RCX. Para simplemente crear un archivo binario conteniendo el programa enlazado, se debe user:

```
lejos -o MiPrograma.bin MiPrograma
```

Se puede cargar un programa enlazado previamente con:

```
lejosrun MiPrograma.bin
```

También se pueden cargar múltiples programas en una sola vez del siguiente modo:

```
lejos Clase1,Clase2,...
```

Los programas serán enlazados juntos, y reemplazarán cualquier programa previo en el RCX. Puede presionar el botón Prgm para seleccionar qué programa ejecutar.

Después de que la figura de la persona quieta aparece en el RCX, puede presionar Run para iniciar un programa. Si el programa termina por sí mismo, la figura de la persona quieta aparecerá otra vez. Para detener un programa que está corriendo, mantenga presionado el botón Run mientras presiona el botón On/Off. Apagando el RCX también detiene un programa corriendo.

4 Construcción de robots limpiadores

4.1 Objetivo de los robots limpiadores

El principal objetivo para construir y programar robots limpiadores fue participar en el Primer Torneo Mexicano de Robots Limpiadores, llevado a cabo en el INAOE, en Tonantzintla, Puebla, del 9 al 13 de agosto de 2004.

Figura 8. Banda del Primer Torneo Mexicano de Robots Limpiadores 2004 (PTMRL-04)

Con objeto de promover la creación de un foro para el intercambio de ideas entre estudiantes universitarios y de posgrado en computación, electrónica y áreas afines, así como para sensibilizar a jóvenes estudiantes y al público en general sobre el impacto que los robots pueden tener en nuestras vidas diarias, el INAOE, la UV y el IEEE convocaron a participar en el Primer Torneo Mexicano de Robots Limpiadores que se celebraría en las instalaciones del mismo INAOE, en Tonantzintla, Puebla, del 9 al 13 de agosto de 2004 de acuerdo a las siguientes bases:

1. Podían participar todos los equipos conformados por al menos un estudiante regular de alguna institución de educación superior en México, y no podía incluir más de un profesor o investigador.
2. Cada equipo podía participar con un robot de hasta 5 Kg de peso, cuyas dimensiones no podían ser mayores a 25 X 25 X 25 cm. El material y el equipamiento de los robots participantes quedaba a juicio de sus diseñadores. Sin embargo, quedaba excluido el uso de sensores láser, de tarjetas de red y de radio-modems.
3. Sólo se permitió la participación de robots autónomos, es decir, de robots que dependen para su operación de los sensores y de los actuadores de los que están equipados, así como del programa de control que se les hubiera cargado previamente. Quedaba excluida la participación de robots tele-dirigidos y de robots que se comunicaran con alguna computadora o dispositivo externo para realizar cualquier tipo de proceso.
4. Los robots participantes debían ser capaces de limpiar de manera autónoma, tres tipos diferentes de basura: *café molido*, *hojas secas* y *confeti*, en un ambiente con y sin obstáculos de las siguientes características: espacio de 100 cm de largo x 200 cm de ancho, cerrado por 4 paredes de cartón blanco

de 50 cm de alto. La superficie del piso de este ambiente era lisa y estaba pintada de color blanco opaco.

5. La limpieza implicaba recoger la basura encontrada en el ambiente y depositarla en un lugar específico del mismo. Dicho lugar, cuya localización podía variar en el transcurso de las pruebas, estaría siempre indicado por un círculo de 14.14 cm. de diámetro de color negro marcado sobre el piso.
6. El ambiente a limpiar no podía ser modificado de ningún modo, más allá de la limpieza de la basura encontrada, por los robots. Quedando excluido el uso de marcas o de cualquier tipo de señal en el piso o en las paredes del ambiente en donde se desarrollarían las pruebas.

Debido a que la Universidad de Xalapa contaba con robots Lego Mindstorms Robotics Invention System 2.0 desde mayo de 2004, y que los estudiantes de la carrera de Ingeniería en Sistemas de Cómputo Administrativo ya habían trabajado con los mismos en la materia Inteligencia Artificial, construyendo y programando robots limpiadores para la parte final del semestre, resultó muy apropiado usar tales robots para el Torneo.

4.2 Fegen: Robot que empuja la basura

4.2.1 Estrategia de limpieza

El robot Fegen (del alemán barrer, barredor) fue diseñado para funcionar como una escoba, empujando la basura. Para poder empujar la basura sin que se escapara por los lados, la parte delantera, donde se acumulaba la basura se diseñó en forma de C cuadrada (Figura 9).

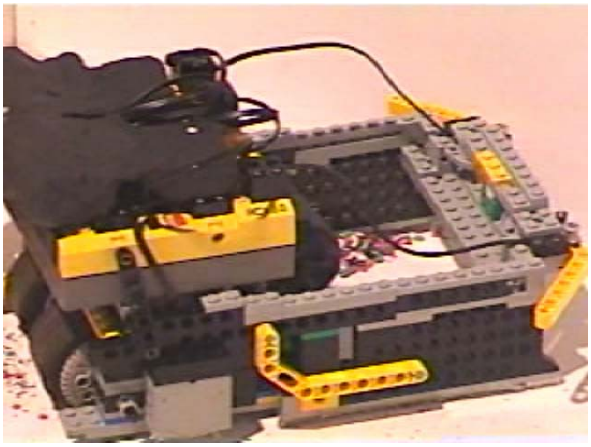
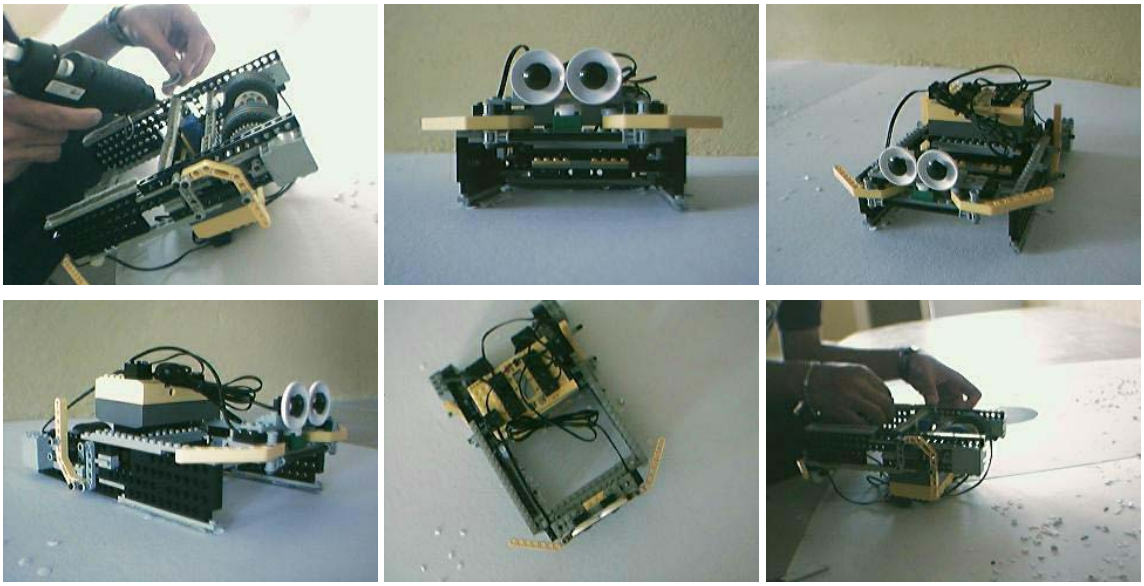


Figura 9. Robot Fegen, ganador del tercer lugar del PTMRL-04

4.2.2 Diseño de un prototipo

El primer prototipo de este robot se diseñó y construyó en la segunda mitad de la asignatura *Inteligencia Artificial* del período Febrero-Julio 2004 (Figura 10). El material empleado para la construcción de este robot fue un paquete Lego Mindstorms Robotics Invention System 2.0. Este paquete cuenta con varias piezas, siendo algunas de las mas pequeñas de alrededor de 2 centímetros, un microprocesador en el RCX, hasta motores y sensores de luz y contacto.



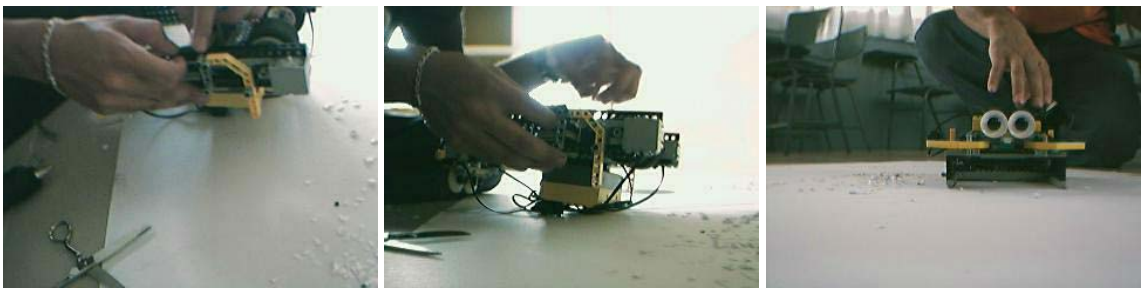


Figura 10. Primer prototipo del robot Fegen

La Figura 11 muestra algunas de las piezas usadas para la construcción del robot.

Figura 11. Algunas piezas usadas para construir Fegen

Con estos tipos de piezas y otras más se construyó un robot que permitiera barrer distintos tipos de basura, los requerimientos principales que debía cubrir eran que pesara menos de 5 Kg y que midiera menos de 25 cm de largo, ancho y alto.

4.2.3 Construcción

El robot Fegen consta básicamente de dos grandes secciones: la parte delantera y la parte trasera, como se puede ver en ambas fotos de la Figura 12.

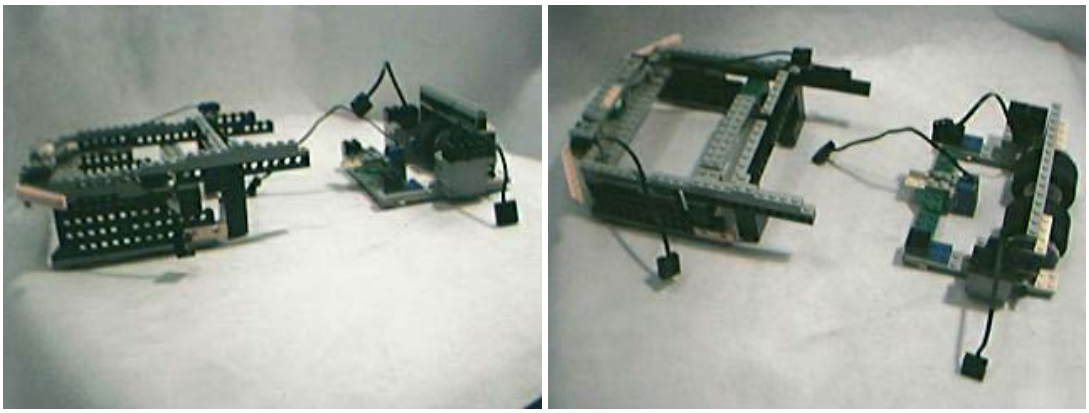
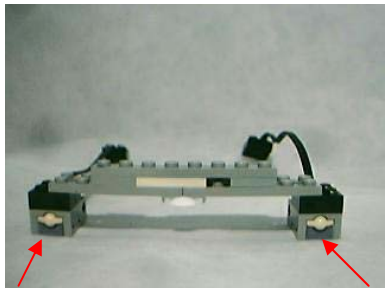


Figura 12. Secciones delantera y trasera del robot Fegen

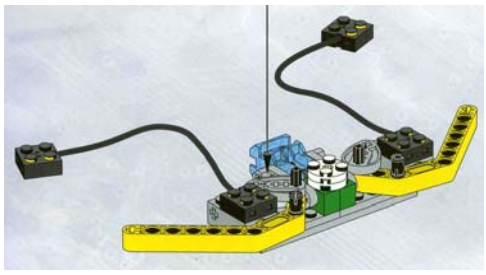
4.2.3.1 Sección delantera

1) Como primer paso reconocer la pieza que se muestra en la siguiente figura como el sensor de contacto, este se monta en la parte frontal de la estructura que conforma el robot con el fin de sensar cuando él toque sobre una superficie.

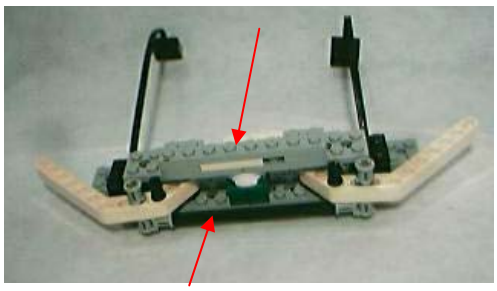
2) Se colocan dos de estos censores de la forma que indica la imagen siguiente, de manera que quede unos de cada lado de la pieza.



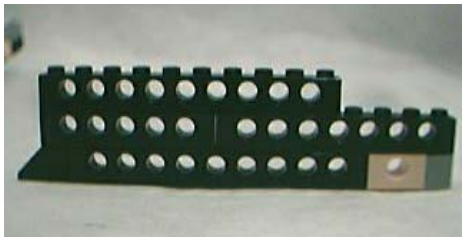
3) Frente a los censores de contacto se colocan dos piezas como las que se muestran a continuación, con el fin de hacer más efectivo el fin de los sensores y aumentar el área de acción de estos.



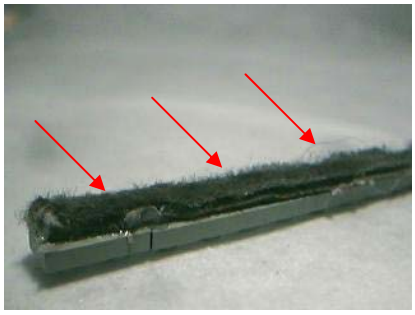
4) Finalmente el censor queda de esta manera sujetándolo firmemente con dos piezas por la parte superior e inferior para que al chocar no sea afectado.

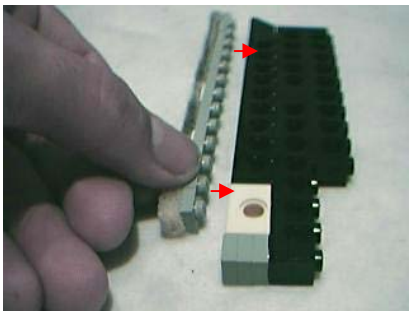


5) Como parte de toda la pieza frontal y como parte fundamental del sistema de barrido y soporte de la pieza de los sensores de contacto, se arma esta nueva parte del robot. De esta pieza se tienen que hacer dos idénticas, una para la parte izquierda y otra para la derecha del robot.

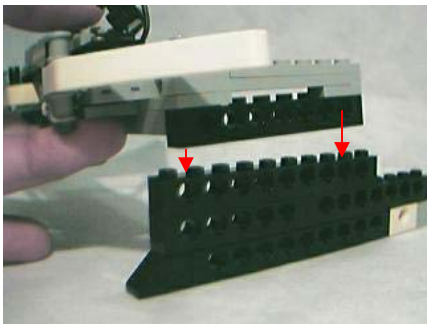


6) A la pieza construida anteriormente se le coloca en la parte de abajo una tira de una especie de cobilla (ver figuras siguientes) que sirve para que el robot al avanzar arrastre la basura que se encuentre en el camino.



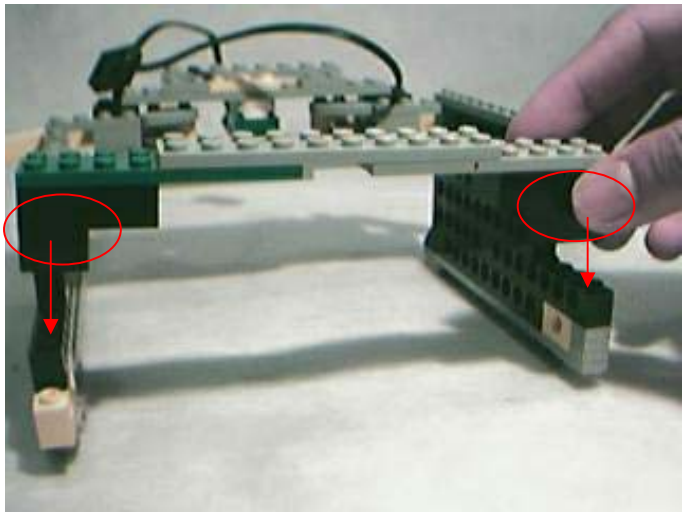


7) A la pieza descrita anteriormente se le coloca sobre de ella la pieza que contiene los sensores de contacto de modo que la soporte sobre sus dos partes.



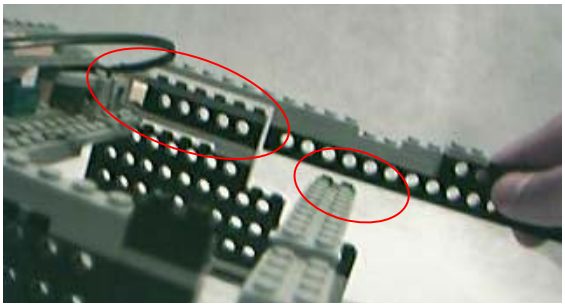
8) Con el fin de hacer mas resistente la pieza frontal del robot se construye esta parte mostrada en la figura de abajo y en el paso siguiente se muestra su montaje.

9) Esta es la manera y el lugar en el que se coloca esta pieza.

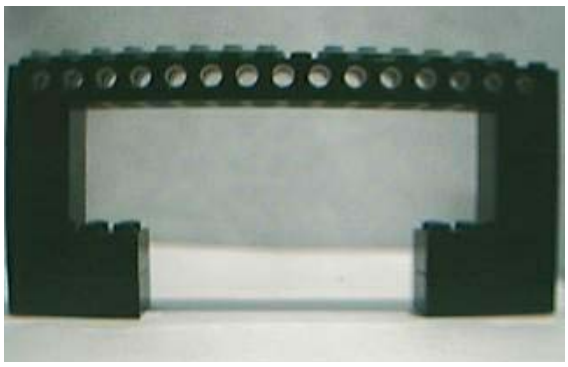


10) Para hacer todavía más resistente la pieza frontal se coloca esta nueva pieza también realizada doble y se coloca como se muestra en el paso anterior.

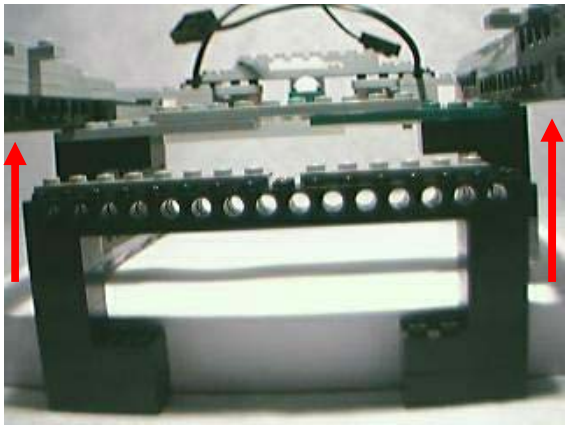
11) Así y en este lugar se coloca la pieza que nos servirá de refuerzo y como parte de ensamblaje con la pieza trasera del robot.



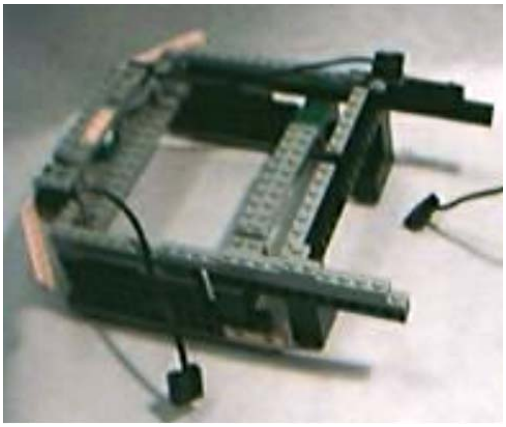
12) Para terminar la parte frontal y también como parte de ensamblaje de la parte delantera con la trasera, se construye esta pieza.



13) La pieza anterior se coloca de esta manera.



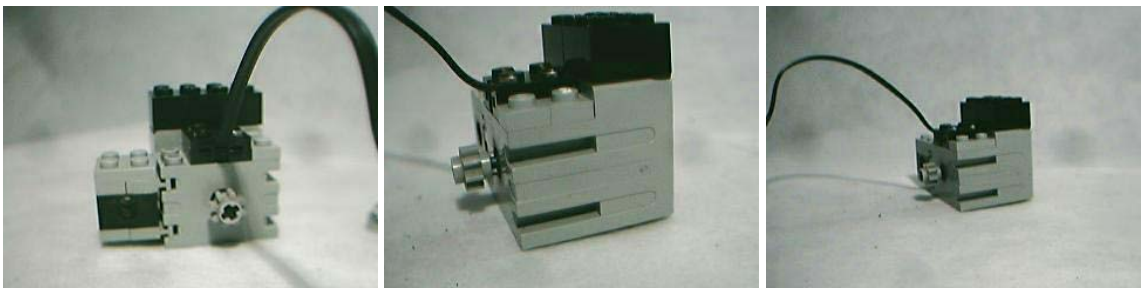
Finalmente la pieza frontal queda entablada de la siguiente manera y lista para ser conectada con el resto del robot.



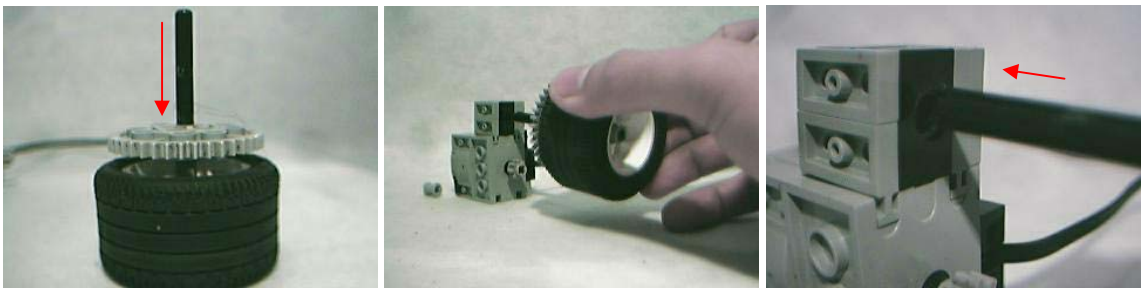
4.2.3.2 Sección trasera

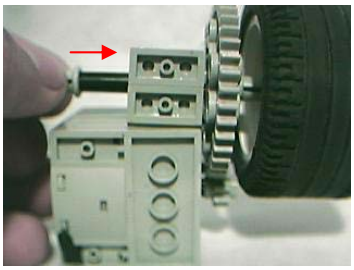
La sección trasera del robot contiene dos motores, dos llantas, un sensor de luz y un RCX. Se describe a continuación su armado.

1) Se ensamblan los motores como en las siguiente figuras:

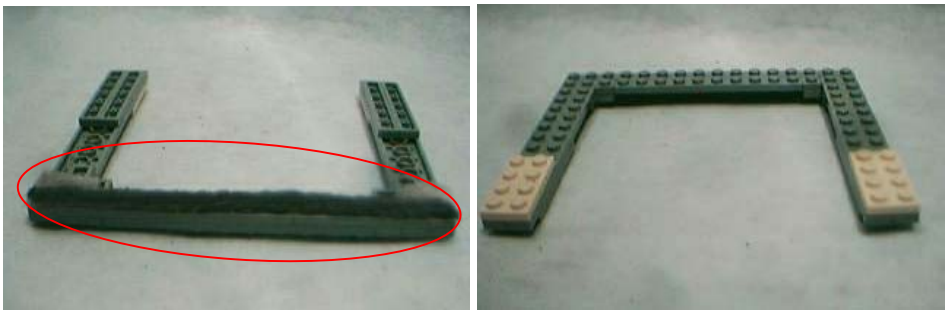


2) Para poder hacer un ensamblaje de la rueda con el motor para mover el robot, se debe poner a la rueda un engrane que se acoplará posteriormente a uno más pequeño que tiene el motor, esto se logra poniendo un eje que sujetará las dos piezas antes mencionadas. Este eje se introducirá en el orificio que tiene el motor para poder ser movido por él.

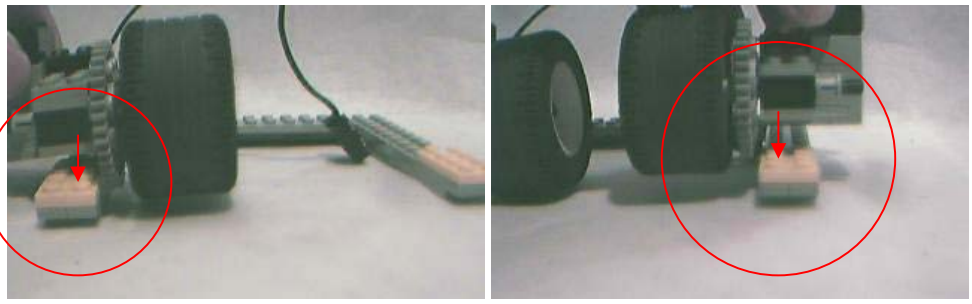




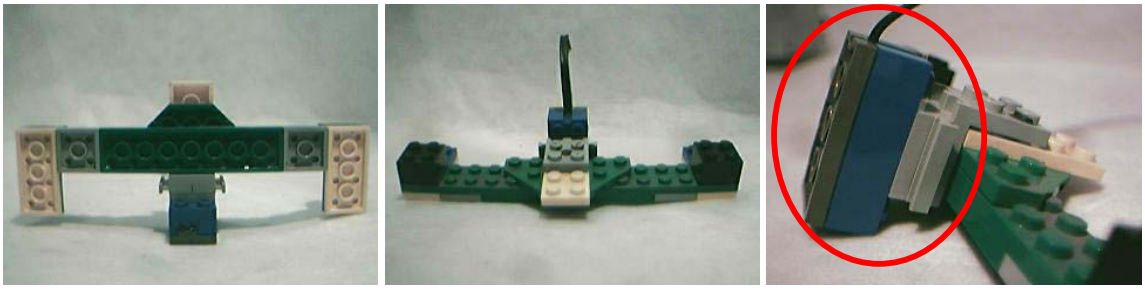
3) A continuación se procede a armar la base en la que se montarán las ruedas. Esta base también servirá como recolector de basura, se puede apreciar en la imagen que la parte que quede hacia abajo tiene escobillas.



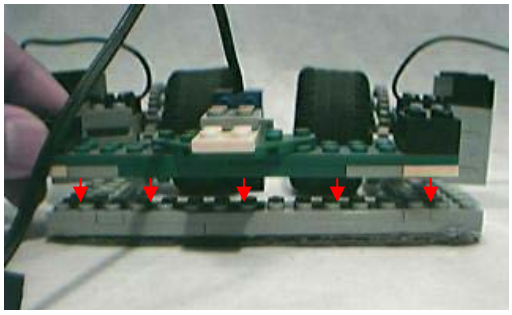
4) Ya que quedaron armadas las llantas junto con el motor se montan sobre la pieza que los soportará de la siguiente manera



5) Algo clave en la construcción es el lugar donde se colocará el sensor de luz, ya que la colocación de este va a ser muy importante para el buen funcionamiento del robot, en las figuras siguientes se muestra la construcción de esta parte.

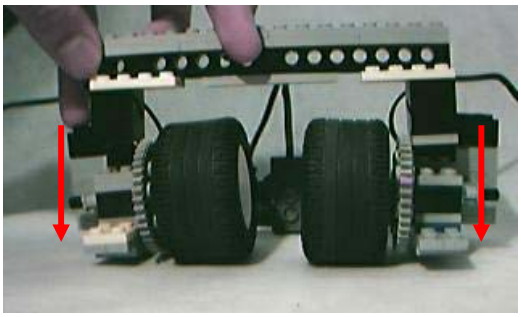


6) A continuación se muestra cómo se monta esta pieza del sensor de luz sobre lo ya armado anteriormente.

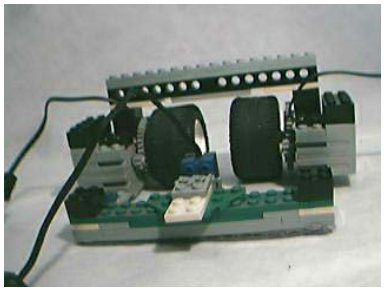
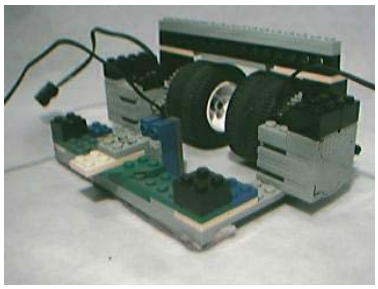
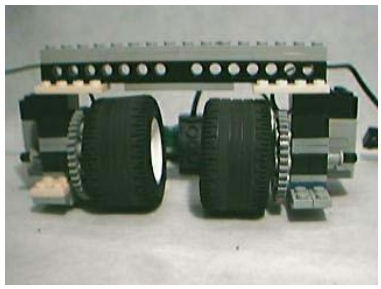


7) Por último se construye una pieza mas, que va a tener la función de hacer consistente la parte trasera del robot y a su vez va a ser parte de lo que va a soportar el RCX.

8) Se coloca esta pieza sobre las llantas de la manera que indica la foto siguiente.

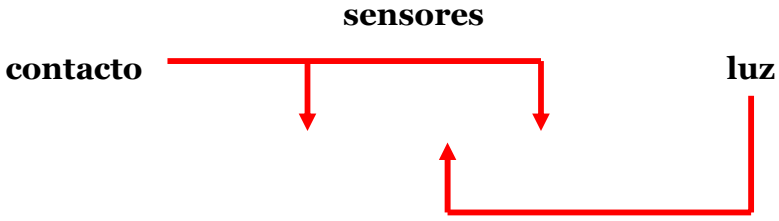


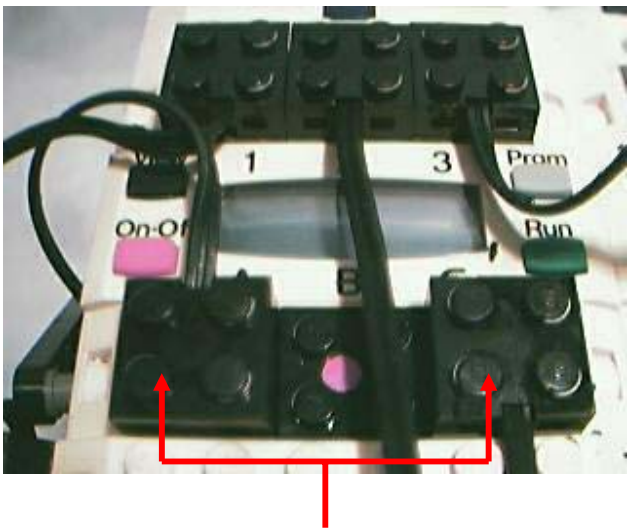
9) La parte trasera del robot se ensambla de la siguiente manera con los motores, llantas y sensor de luz.

El RCX se coloca sobre toda la estructura del robot quedando de esta manera y con las conexiones pertinentes, como se puede apreciar en la imagen siguiente están montados dos RCX juntos, esto es porque se necesitaba mas peso en la parte trasera del robot para que las llantas tuvieran mas tracción a la hora de rodar.

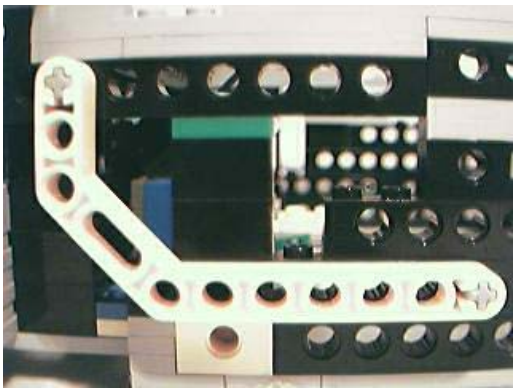
En la imagen siguiente se muestran las conexiones del RCX con los dispositivos



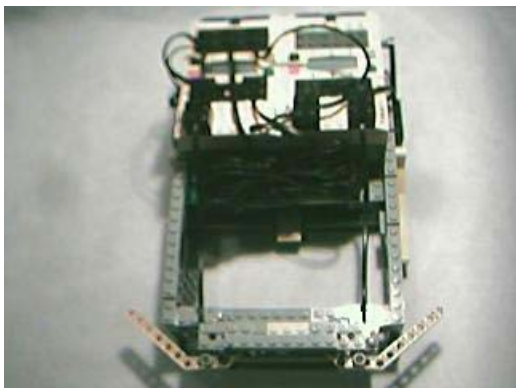


motores

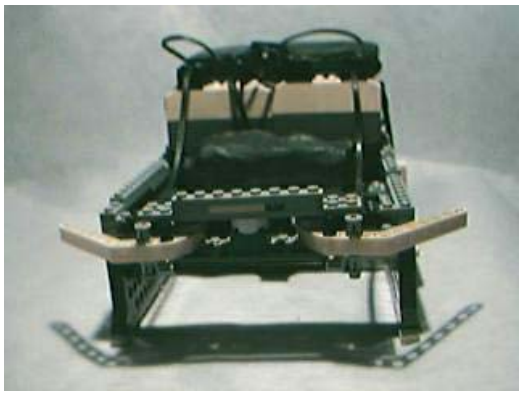
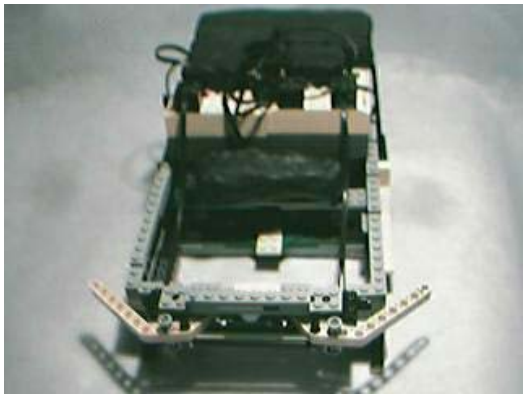
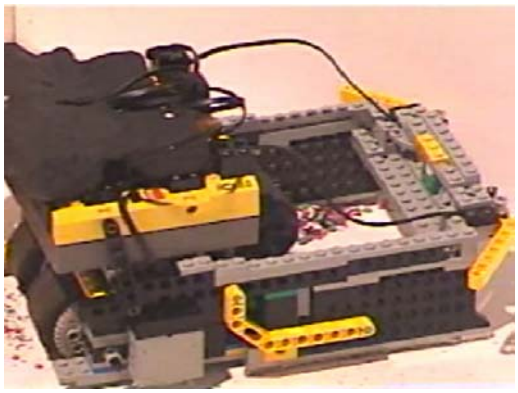
Con la finalidad de sujetar perfectamente tanto la parte frontal como la trasera del robot se coloca esta pieza amarilla en cada uno de los costados para hacer la máquina más fuerte.



Al término de la construcción, el robot barredor queda de esta forma:



Después de hacer una serie de pruebas resultó que se requería mas peso en la parte trasera y en la parte media, así se que le colocó plastilina en dichas partes y la apariencia final del robot fue la siguiente:

4.3 Dralion: Robot que recoge la basura

4.3.1 Estrategia de limpieza

Algunas de las ideas iniciales para construir un robot limpiador consideraban lo siguiente:

- Robot limpiador tipo camión de volteo con aspiradora

- Robot limpiador que empuja la basura
- Robot limpiador que recoge la basura

Al principio este robot se diseñó con la misma estrategia que el anterior, empujar la basura, pero, la mayor parte de las pruebas demostraron que tal estrategia no era la mejor opción, porque, parte de la basura (café molido, confeti y hojas secas) quedaba esparcida en el mismo piso, además de que se le quitaba al robot la opción de moverse hacia atrás, movimiento que resultó muy importante en el Torneo.

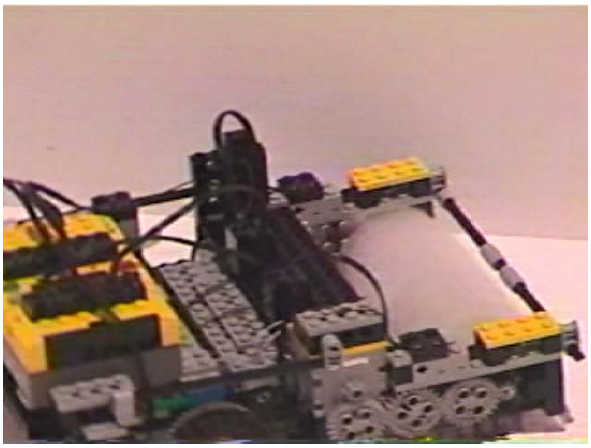


Figura 13. Robot Dralion, ganador del segundo lugar del PTMRL-04

Por lo tanto, este robot se rediseñó completamente, con la idea de usar la estrategia de limpieza con escoba y recogedor. Lo más práctico parecía ser construir un robot que tuviera un recogedor y como escoba un rodillo giratorio (Figura 13).

4.3.2 Diseño

El diseño final del robot consiste de un modelo tipo rover con 4 llantas iguales y un recogedor levadizo para tirar la basura que contiene un cepillo giratorio. La mejor opción de recogedor que se analizó y finalmente se diseñó y construyó se muestra en la Figura 14.

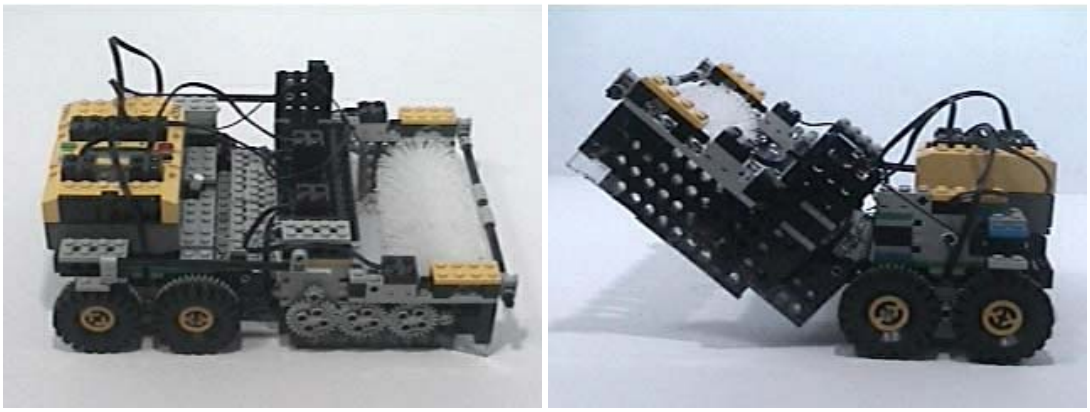


Figura 14. Robot Dralion con cepillo giratorio y recogedor levadizo

Para este diseño el robot requirió 4 motores, 3 sensores de contacto y 2 sensores de luz para usarse del siguiente modo:

2 motores para desplazar al robot.

1 motor para girar el cepillo.

1 motor para levantar el recogedor.

2 sensores de contacto para percibir choques de frente.

1 sensor de contacto para detener el levantamiento del recogedor en su altura máxima.

2 sensores de luz para detectar el círculo negro, lugar donde se levantaría el recogedor depositando la basura.

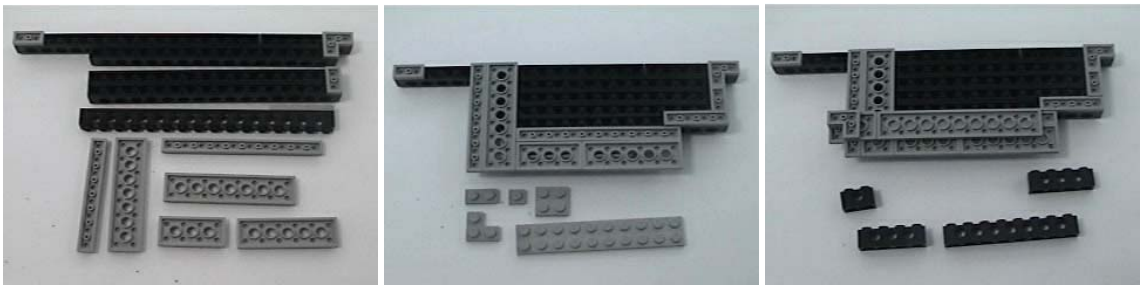
Dado que únicamente se usó un RCX, el cual está limitado con 3 puertos de salida para conectar 3 motores y con 3 puertos de entrada para conectar 3 sensores, se diseñaron arreglos alternativos para controlar los 4 motores y 5 sensores, como se verá en el tema de construcción.

4.3.3 Construcción

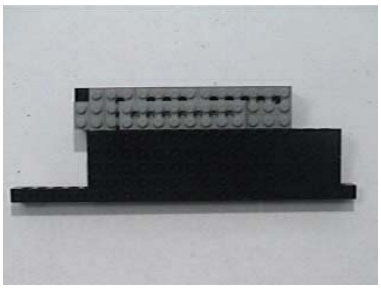
El proceso de construcción de este robot se muestra también mediante fotografías, en las cuales se ven las piezas necesarias y cómo deben ir quedando ensambladas.

Con esta serie de pasos se ensambla primero el armazón que contendrá el cepillo giratorio y el recogedor levadizo.

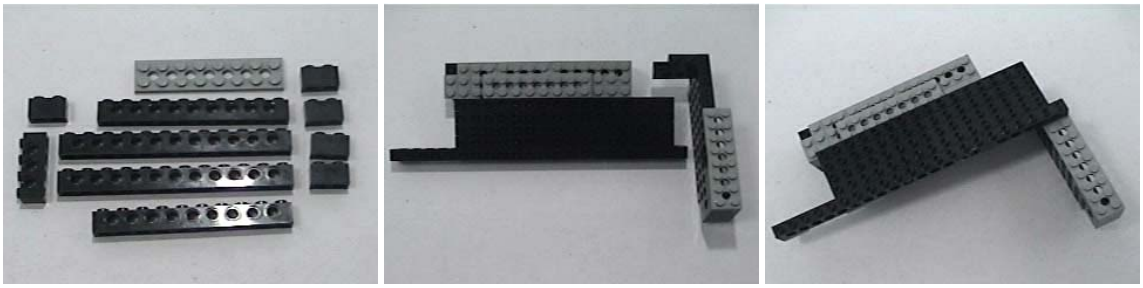
1) Se arma primero una tapa superior con un espacio para el motor que moverá el cepillo.



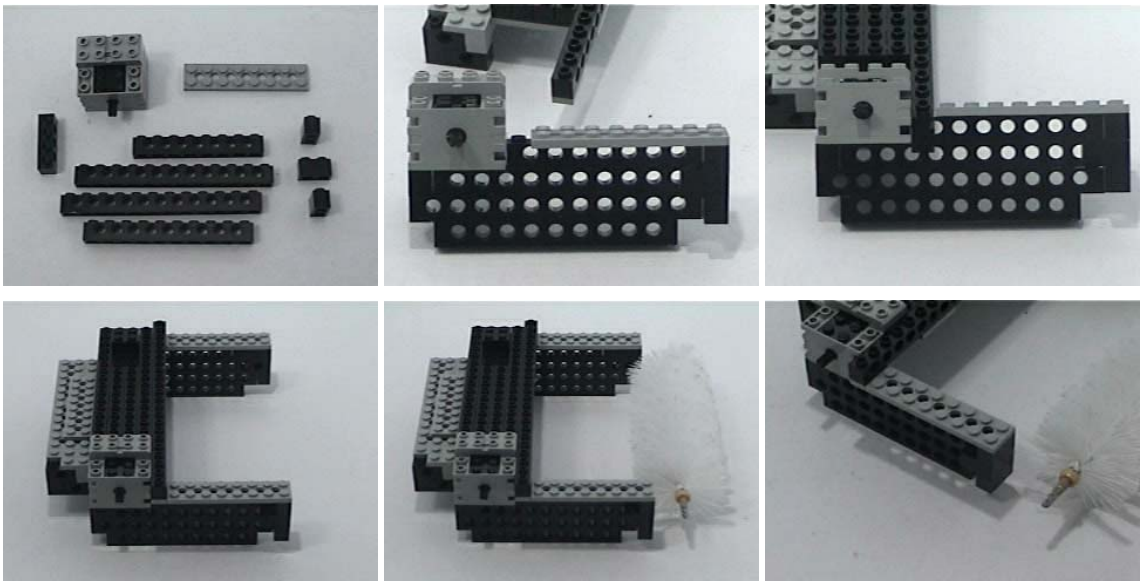
2) Se gira la pieza armada para tener los rebordes hacia arriba.



3) Se arma el lado que no tiene el motor.



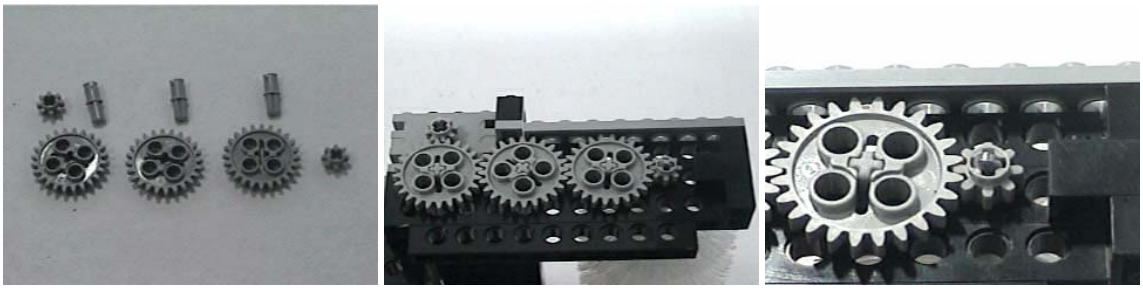
3) Se arma el lado que tiene el motor y engranes para mover el rodillo giratorio.



4) El cepillo giratorio es un escobillón para lavar biberones. Se recorta por ambos extremos para que quepa bien en la estructura armada y se lima con esmeril una de las puntas hasta dejarla plana y poder insertar un engrane chico. Antes del engrane se debe colocar una especie de rondana para que el rodillo no se mueva hacia los lados.

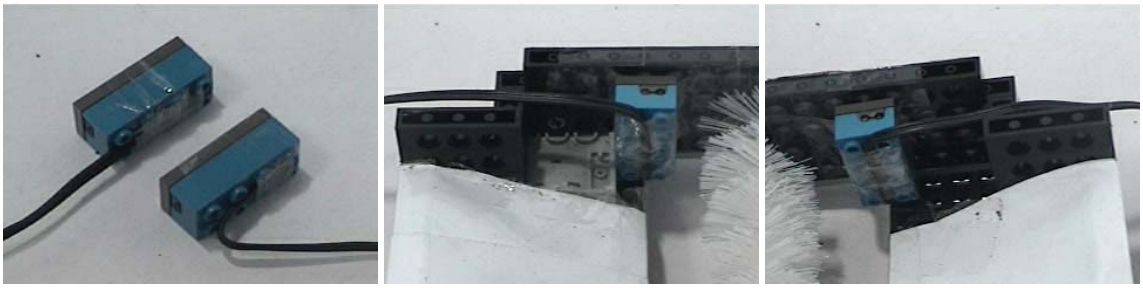
5) Para insertar el cepillo debe desarmarse momentáneamente la estructura ya armada.

6) Se colocan los engranes que van del motor al cepillo.

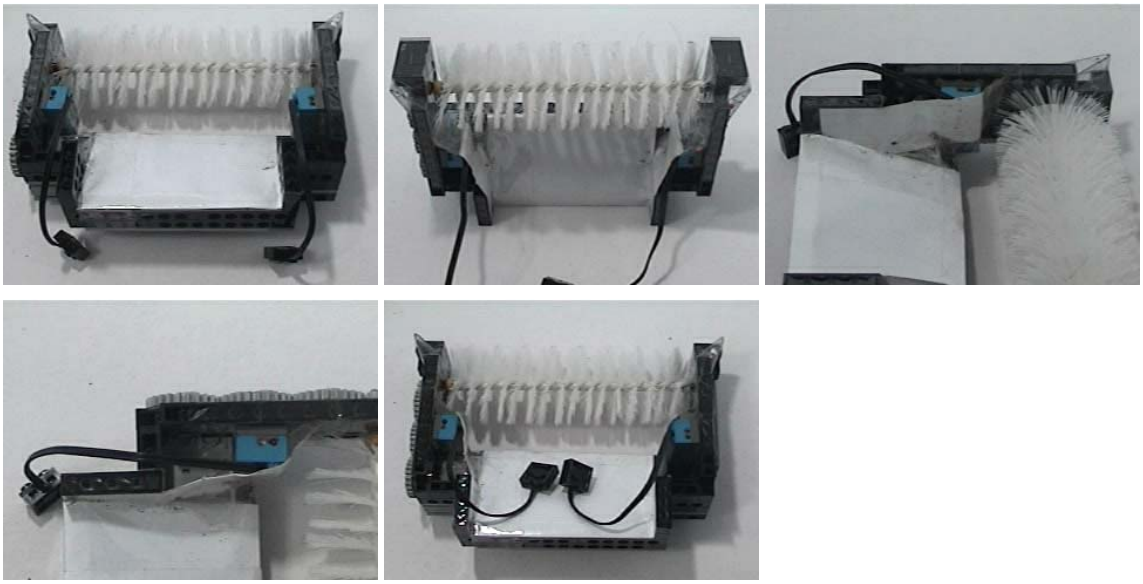


7) Para recolectar el máximo de basura, se recortan dos piezas de plástico para colocarlas en las puntas de la estructura.

8) A los sensores de luz se les pega el cable con cinta adhesiva como se muestra a continuación y se adhieren también con cinta adhesiva a la estructura.

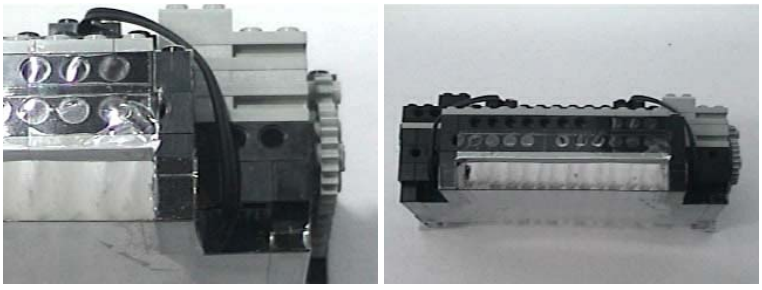


9) Como se ve en las dos última fotos, se coloca una hoja de papel, adhiriéndola con cinta adhesiva. Esta hoja es para que la basura no se meta en los agujeros de las piezas lego. También se ponen trozos de papel a los lados, cubriendo los sensores para evitar acumulación de basura allí.



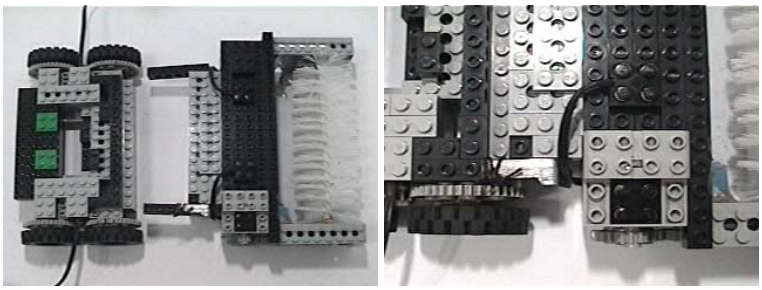
10) Una vez armada la estructura que tiene el cepillo se le agrega una lámina de plástico que actuará como recogedor. Dicha lámina también se pega con cinta ahdesiva.

11) Debe quedar un vacío entre la lámina y la pared contraria al cepillo, ya que por allí saldrá la basura cuando se levante el recogedor.

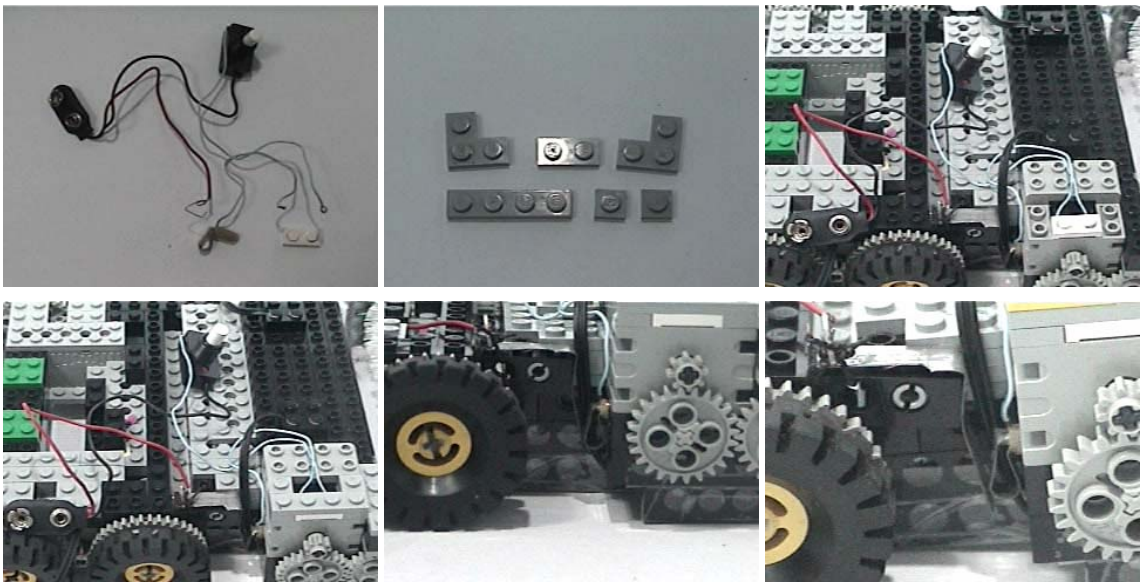


12) Se recorta una lámina metálica para forma un interruptor que hará que el motor del cepillo se apague cuando se levante el recogedor.

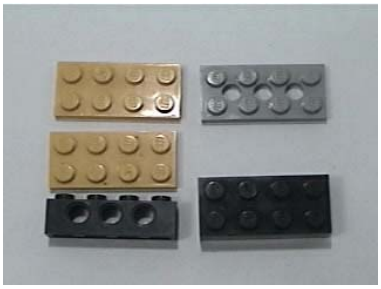
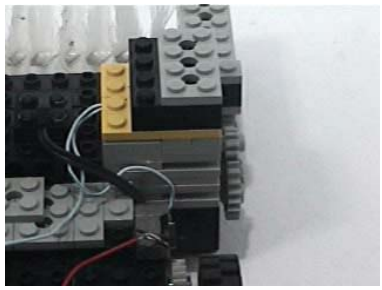
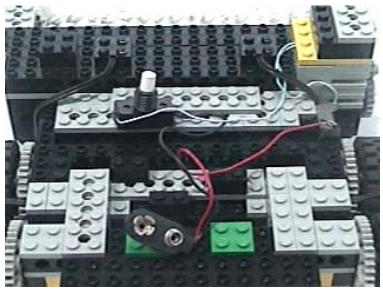
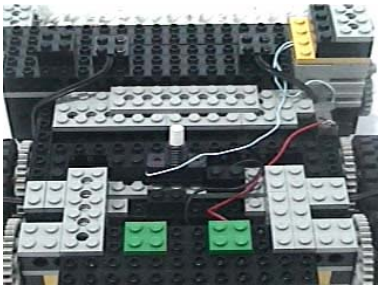
13) Como se mencionó anteriormente el robot consiste de un modelo tipo rover bastante modificado para adaptarlo al recogedor. El recogedor se ensambla al robot como se ilustra a continuación. El recogedor quedará arriba del suelo aproximadamente 1mm en el extremo unido al rover y por su peso, la parte frontal será arrastrada en el suelo.



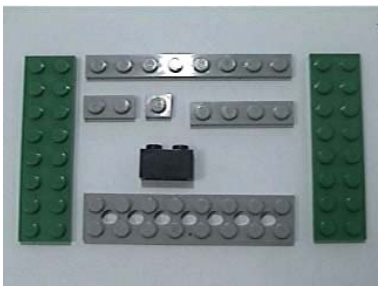
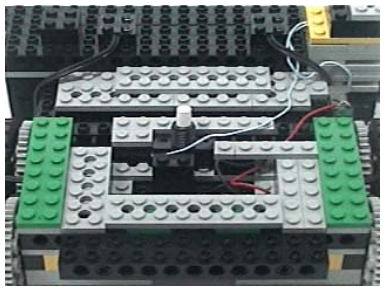
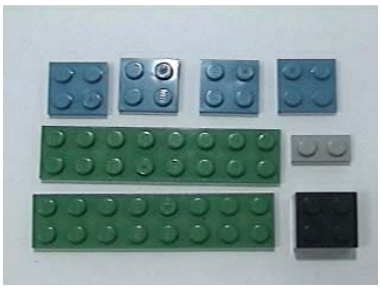
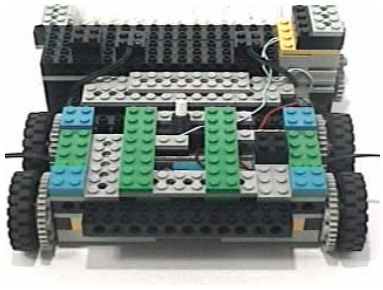
14) Se arma un sencillo circuito para encender y apagar con un interruptor de presión el motor del cepillo cuando el recogedor está abajo. El interruptor estará a un lado del RCX y será fácilmente manipulable cuando se encienda el mismo RCX y se ejecuten sus programas.



15) Se agregan piezas para sujetar bien las piezas conectadas al motor.

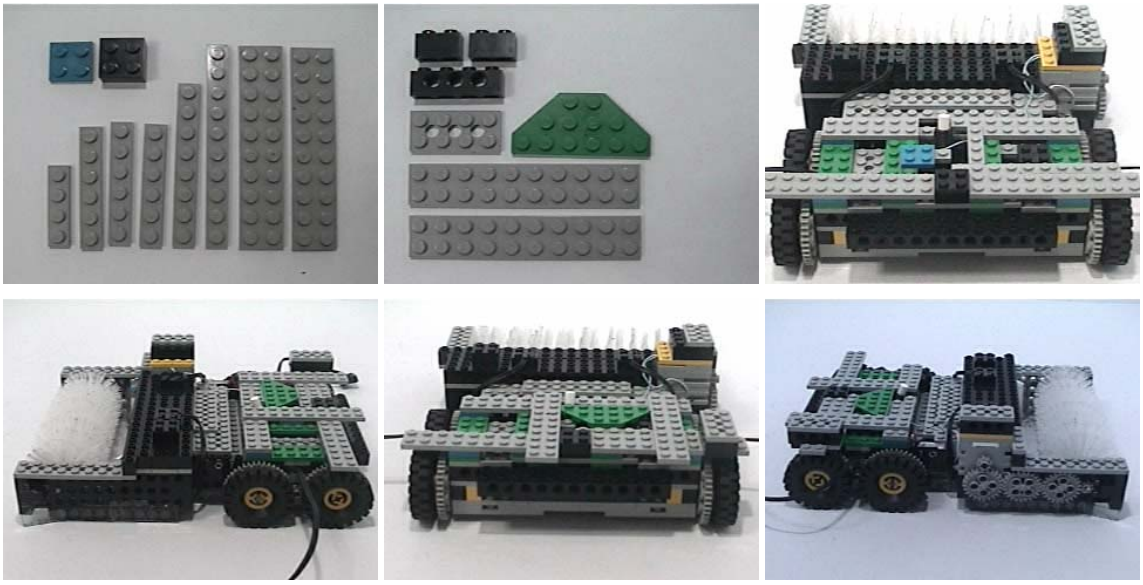





16) Se agregan más piezas para sujetar los cables que se conectan a la batería adicional para encender el motor del rodillo.

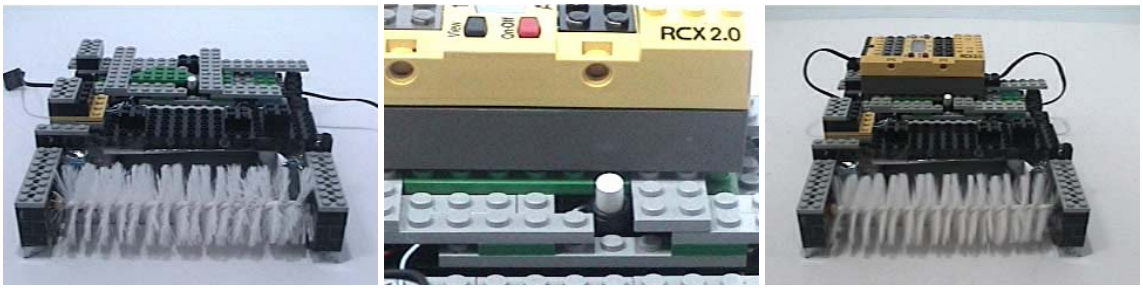





17) El robot debe estar diseñado para quitar y poner fácilmente la batería del motor que gira el rodillo.

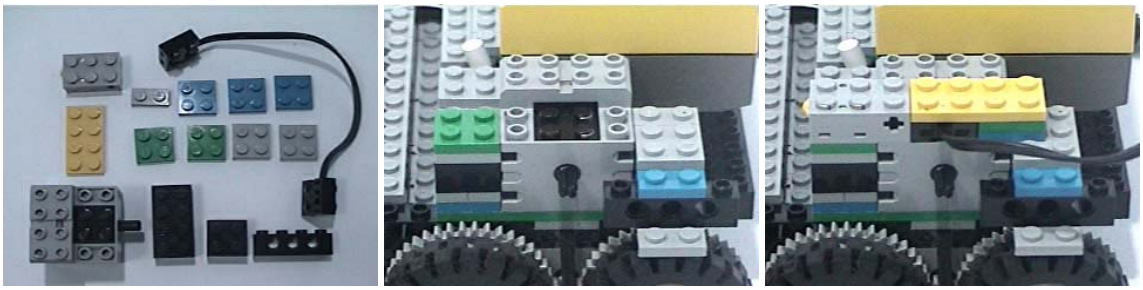
18) Se agregan piezas para iniciar la base del RCX y del motor que levanta el recogedor.



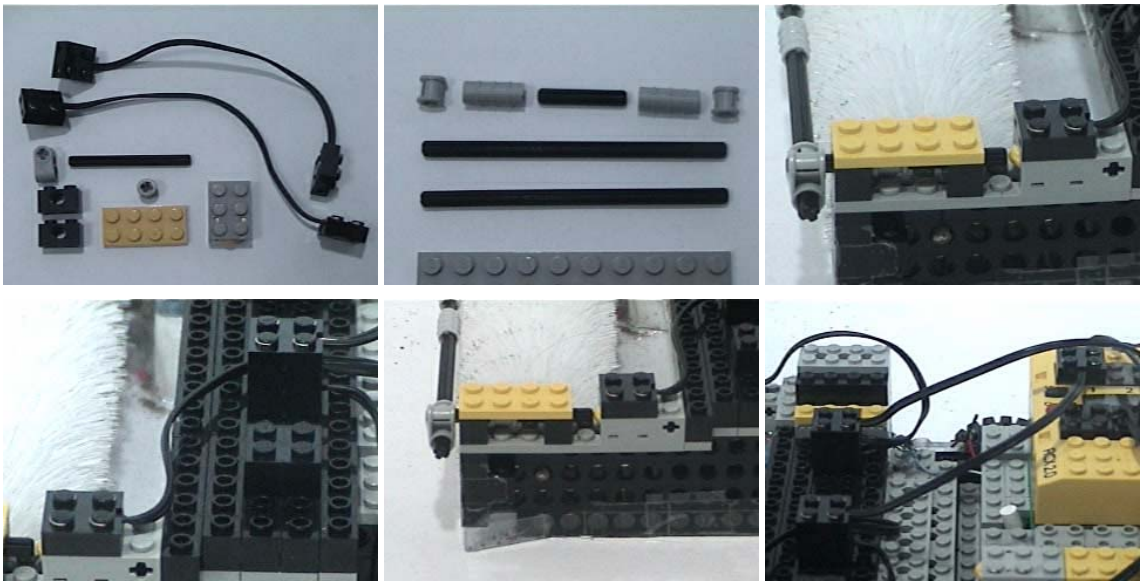
19) Se coloca el RCX sobre el robot.



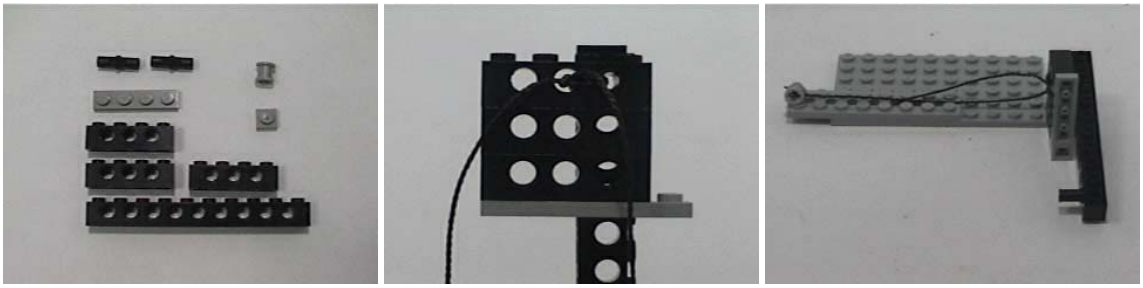
20) Se coloca el motor que levanta el recogedor.

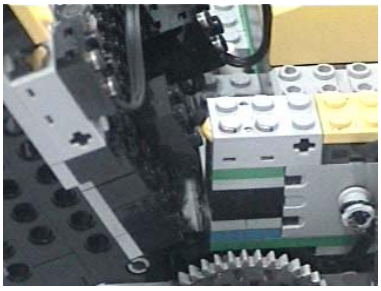
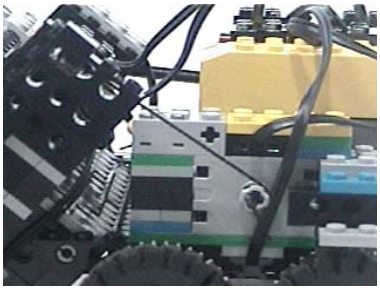


21) Se colocan los sensores de contacto al frente del recogedor, junto con una varilla para detectar contactos al centro. Los sensores se conectan a los puertos 1 y 3 del RCX.

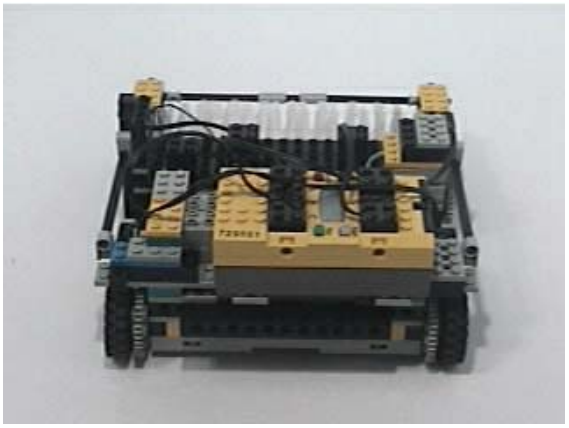


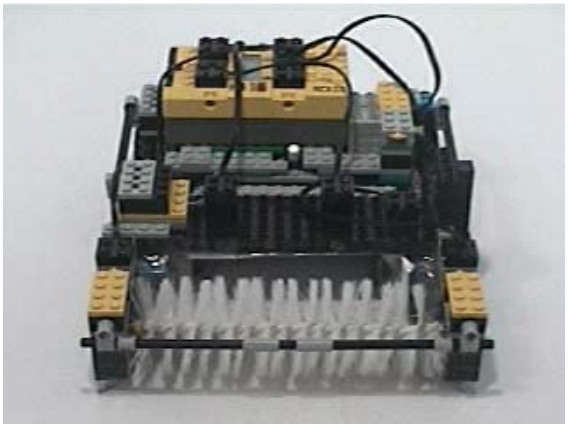
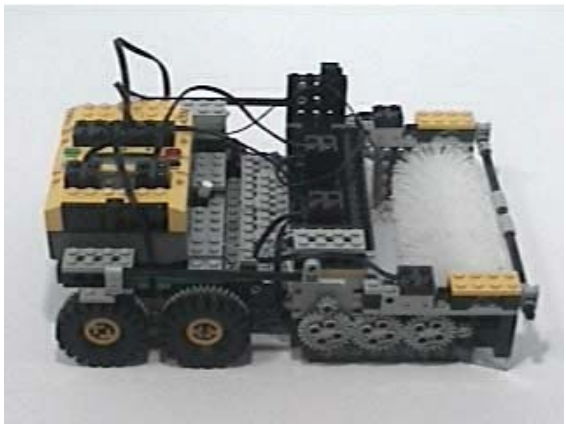
22) Se arma la pieza mediante la cual se levantará el recogedor. Esta se amarra al motor correspondiente mediante un hilo resistente.




Con esto queda construido el robot Dralion.



El motor izquierdo debe conectarse al puerto A, el derecho al puerto C y el motor que levanta el recogedor al puerto B.

El sensor de contacto frontal izquierdo debe conectarse al puerto 1, el sensor de contacto frontal derecho al puerto 3 y el sensor de contacto que está junto al motor del recogedor debe conectarse al puerto 2.

El sensor de luz izquierdo debe conectarse al puerto 1 y el sensor de luz derecho debe conectarse al puerto 3.

4.4 Otros robots limpiadores

En el Primer Torneo Mexicano de Robots Limpiadores participaron otros robots. Las figuras siguientes muestran los robots ganadores del primero y cuarto lugar del torneo.

Figura 15. Robot Zorro 1, ganador del primer lugar del PTMRL-04

Figura 16. Robot Zorro 2, ganador del cuarto lugar del PTMRL-04

5 Programación de robots limpiadores

En este capítulo se presenta el código usado para mover los dos robots construidos en el capítulo anterior.

5.1 Código general

El código general está definido para robots Lego del tipo rover, en el que el motor izquierdo está conectado al puerto A y el motor derecho al puerto C del RCX.

Se presentan dos clases generales, la clase roverbot2 y la clase roverbot2mas, en la cual la última hereda de la primera.

5.1.1 Clase roverbot2

La clase roverbot2 define los movimientos básicos del robot con los métodos *potencia*, *avanza*, *retrocede*, *izquierda*, *derecha* y *alto*.

```
// Programación en Java con robots Lego Mindstorms, Mayo 2004
// Autor: M.I.A. Julio César Sandria Reynoso
// Robot: Roverbot

import josx.platform.rcx.Motor;

public class roverbot2 {
   public static void potencia( int potencia ) throws InterruptedException
   {
      Motor.A.setPower (potencia);
      Motor.C.setPower (potencia);
   }

   public static void avanza( int milisegundos ) throws InterruptedException
   {
      Motor.A.forward ();
      Motor.C.forward ();
      Thread.sleep ( milisegundos );
   }

   public static void retrocede( int milisegundos ) throws InterruptedException
   {
      Motor.A.backward ();
      Motor.C.backward ();
      Thread.sleep ( milisegundos );
   }

   public static void derecha( int milisegundos ) throws InterruptedException
   {
      Motor.A.forward ();
      Motor.C.backward ();

      Thread.sleep ( milisegundos );

      Motor.A.forward ();
      Motor.C.forward ();
   }
```

```
   public static void izquierda( int milisegundos ) throws InterruptedException
   {
      Motor.A.backward ();
      Motor.C.forward ();

      Thread.sleep ( milisegundos );

      Motor.A.forward ();
      Motor.C.forward ();
   }

   public static void alto( ) throws InterruptedException
   {
      Motor.A.stop ();
      Motor.C.stop ();
   }

} // class roverbot2
```

5.1.2 Clase roverbot2mas

La clase roverbot2mas se definió como una clase que extiende a roverbot2, como un ejemplo de herencia, en la que se definen los métodos *zigzag*, *baila*, *zumba* y *pita*. Nótese que los métodos *baila* y *zumba* están definidos dos veces cada uno, pero con un número de argumentos distintos, mostrando el uso de polimorfismo en los métodos.

```
// Programación en Java con robots Lego Mindstorms, Mayo 2004
// Autor: M.I.A. Julio César Sandria Reynoso
// Robot: Roverbot

import josx.platform.rcx.Motor;
import josx.platform.rcx.Sound;
import josx.platform.rcx.LCD;

public class roverbot2mas extends roverbot2 {

   // Los métodos avanza, derecha, retrocede, izquierda y alto
   // son heredados de la clase roverbot2

   public static void zigzag( int milisegundos ) throws InterruptedException
   {
      int tiempo = 0, avance = 500;
      while( tiempo < milisegundos ) {
         avanza( avance );    tiempo = tiempo + avance;
         derecha( avance );   tiempo = tiempo + avance;
         avanza( avance );    tiempo = tiempo + avance;
         izquierda( avance ); tiempo = tiempo + avance;
      }
      /* PROBLEMA: Aunque el método sea llamado con un tiempo de 500
         milisegundos zigzag tarda 2000 milisegundos en ejecutarse */
   }

   public static void baila( int milisegundos ) throws InterruptedException
   {
      baila( milisegundos, true );
   }
```

```
   public static void baila( int milisegundos,
                             boolean mostrarEnPantalla ) throws
                                                          InterruptedException
   {
      int tiempo = 0, avance = 500, paso;

      while( tiempo < milisegundos ) {
         paso = (int) ( Math.random() * 7 );
         if( mostrarEnPantalla )
            LCD.showNumber( paso );  // Muestra paso en la pantalla LCD
         if ( paso == 1 ) {
            avanza( avance );        tiempo = tiempo + avance;
         }
         if ( paso == 2 || paso == 5 ) {
            derecha( avance );       tiempo = tiempo + avance;
         }
         if ( paso == 3 ) {
            avanza( avance );        tiempo = tiempo + avance;
         }
         if ( paso == 4 || paso == 6 ) {
            izquierda( avance );     tiempo = tiempo + avance;
         }
         if ( paso == 7 ) {
            alto();
            Thread.sleep ( avance ); tiempo = tiempo + avance;
         }
      }
   }

   public static void zumba( int milisegundos ) throws InterruptedException
   {
        Sound.buzz ();
       Thread.sleep ( milisegundos );
   }

   public static void pita() throws InterruptedException
   {
      pita(1);
   }

   public static void pita( int pitidos ) throws InterruptedException
   {
      for( int i=0; i<pitidos; i++ ) {
         Sound.beep ();
         Thread.sleep ( 400 ); // Damos tiempo para escuchar el pitido
      }
   }

} // class roverbot2mas
```

5.2 Código del robot Fegen

El robot Fegen fue programado según las bases del concurso, en breve se trababa de que los robots recogieran tres tipos de basura (hojas secas, confeti y café molido) y que la depositaran en un círculo negro que simulaba las veces el basurero.

El robot Fegen fue programado utilizando programación orientada a objetos para que se facilitara el manejo de algunas partes del robot como objetos y se

pudieran programar como es el caso de los sensores de luz y de contacto, usando el lenguaje Java. El funcionamiento de este robot era el siguiente: estaba programado para que los sensores del robot detectaran el valor del color blanco de la plataforma y mientras no encontrara el círculo negro andaría recogiendo basura, cuando se topaba con una esquina o topaba con la pared retrocedía 250 milisegundos y giraba hacia la derecha 1800 milisegundos y si el sensor de luz detectaba el color negro se detenía, retrocedía 2600 milisegundos y giraba hacia la derecha 1000 milisegundos y seguía recogiendo basura.

A continuación se muestra el código fuente del robot Fegen, que consiste en la clase *Fegen* (que hereda de roverbot2mas) con el método *main*.

```
// Evento:   Primer Torneo Mexicano de Robots Limpiadores
// Organiza: Instituto Nacional de Astrofísica Óptica y Electrónica (INAOE)
// Lugar:    Tonantzintla, Puebla, México
// Fecha:    Agosto de 2004
// Lenguaje: Java sobre leJOS
// Autores:  Equipo Fegen de la Universidad de Xalapa:
//           Rafael Alarcón Domínguez
//           Eduardo Salvador Muñoz Castillo
//           Erick Salvador Cárdenas Bermúdez
//           Miguel Ángel Alonso Lechuga
//           Christian Salas Borbolla
// Robot:    Fegen

import josx.platform.rcx.Sensor;
import josx.platform.rcx.SensorConstants;
import josx.platform.rcx.SensorListener;
import josx.platform.rcx.LCD;

public class Fegen3 extends roverbot2mas {

   public static void main (String[] args) throws InterruptedException
   {
      int valorLuz = 0;
      int umbral = 0;

      // El sensor de luz debe estar conectado al puerto 2
      Sensor.S2.setTypeAndMode (SensorConstants.SENSOR_TYPE_LIGHT,
                                SensorConstants.SENSOR_MODE_RAW);

      // El sensor de toque  debe estar conectado al puerto 1
      Sensor.S1.setTypeAndMode (SensorConstants.SENSOR_TYPE_TOUCH,
                                SensorConstants.SENSOR_MODE_BOOL);
      // El sensor de toque  debe estar conectado al puerto 3
      Sensor.S3.setTypeAndMode (SensorConstants.SENSOR_TYPE_TOUCH,
                                  SensorConstants.SENSOR_MODE_BOOL);

      Sensor.S2.activate();

      Thread.sleep ( 250 );

      valorLuz = Sensor.S2.readRawValue();
      umbral = valorLuz + 50;
      potencia( 7 );

      // Mientras el valor de luz sea menos que 800 que es
      // el negro entonces que siga recogiendo basura
```

```
      while ( valorLuz < 800 ) {
         valorLuz = Sensor.S2.readRawValue(); // Leemos valor del sensor
                                              // [0...1023]
         LCD.showNumber( valorLuz );          // Muestra valorLuz en la
                                              // pantalla LCD
         // Si el sensor de toque muestra un valor quiere decir
         // que toco con algo y tiene que //retroceder y girar
         if( Sensor.S3.readBooleanValue() ) {
            retrocede(250);
             derecha(1800);
         }
         // Si el sensor de toque muestra un valor quiere decir
         // que toco con algo y tiene que //retroceder y girar
         if( Sensor.S1.readBooleanValue() ) {
            retrocede(250);
             derecha(1800);
         }
         avanza( 30 );
         // Si el valor de luz es mayor a 800 quiere decir que
         // encontró el circulo negro y se tiene que detener y
         // retroceder para avanzar por la derecha para evadir
         // el circulo negro y dejar la basura
         if (valorLuz > 800){
            alto();
            retrocede(2600);
            derecha(1000);
            valorLuz = Sensor.S2.readRawValue();
            LCD.showNumber( valorLuz );
         }
      } // while

      Sensor.S2.passivate();
      alto();
      Thread.sleep ( 2000 );

   } // main

} // class
```

5.3 Código del robot Dralion

El robot Dralion consiste de la clase *Dralion*, la cual hereda de *roverbot2mas* y que contiene los métodos *calibrar*, *vuelta*, *dejarBasura*, *daVuelta* y *main*.

El funcionamiento del robot es el siguiente:

1. Se debe encender primero el motor del cepillo giratorio y después oprimir el botón **Run** del RCX para iniciar el programa.
2. El robot realiza un proceso de calibración, de modo que, como inicia siempre en suelo blanco, toma la medida del sensor de luz, y cuando después lee una medida un poco mayor a la calibrada, determina que está ya sobre el círculo negro.
3. En un ciclo infinito avanza 100 ms y revisa sensores.
4. Si hay contacto en el sensor 1 (izquierdo) entonces retrocede 150 ms, da vuelta a la derecha y continúa avanzando.

5. Si hay contacto en el sensor 3 (derecho) entonces retrocede 150 ms, da vuelta a la izquierda y continúa avanzando.
6. En los puntos 4 y 5, si el robot ha avanzado más de 3 segundos sin percibir contacto, entonces da una vuelta completa (de casi 180°) hacia el lado opuesto a la última vuelta completa para hacer un recorrido en forma de zigzag y continúa avanzando.
7. Si el robot hace más de 10 contactos alternados izquierdo y derecho consecutivamente, seguramente está en una esquina, por lo que da una vuelta completa.
8. Si el robot está avanzando y lleva más de 20 segundos sin percibir ningún contacto, es porque, seguramente no se está moviendo, lo que significa que está atorado de alguna manera; entonces, retrocede, da una vuelta completa y continúa avanzando.
9. Cuando uno de los sensores de luz percibe el color negro, el robot inicia un giro hasta que ambos sensores perciben dicho color, entonces, avanza un poco, levanta el recogedor hasta percibir contacto con el sensor correspondiente (apagándose automáticamente el motor del cepillo), entonces, el robot gira a izquierda y derecha para tirar más basura, retrocede, baja el recogedor, da una vuelta completa y continúa avanzando.
10. Finalmente, cuando el robot llega al tiempo máximo de operación (15 minutos para el Torneo) se detiene.

A continuación el código completo del robot Dralion.

```
// Evento:   Primer Torneo Mexicano de Robots Limpiadores
// Organiza: Instituto Nacional de Astrofísica Óptica y Electrónica (INAOE)
// Lugar:    Tonantzintla, Puebla, México
// Fecha:    Agosto de 2004
// Lenguaje: Java sobre leJOS
// Autores:  Equipo Dralion de la Universidad de Xalapa:
//           Julio César Sandria Reynoso
//           Sheila Reyes Guerrero
// Robot:    Dralion

import josx.platform.rcx.Sensor;
import josx.platform.rcx.SensorConstants;
import josx.platform.rcx.Sound;
import josx.platform.rcx.LCD;
import josx.platform.rcx.*;
import java.lang.Math;
import josx.util.Timer;

public class Dralion extends roverbot2mas {
   public static boolean sensor1Contacto = false;
   public static int     sensor1Luz = 0;
   public static boolean sensor2Contacto = false;
   public static boolean sensor3Contacto = false;
   public static int     sensor3Luz = 0;
   public static int     DERECHA = 1;
   public static int     IZQUIERDA = 2;
```

```
public static int     ultimaVuelta = 0;
public static int     ultimoContacto = 0;
public static int     contactosIzquierda = 0;
public static int     contactosDerecha = 0;
public static int     contactosAlternados = 0;
public static int     minBlanco = 1023;
public static int     maxBlanco = 0;
public static int     minNegro = 1023;
public static int     maxNegro = 0;
public static int     tiempoInicial = (int) System.currentTimeMillis() / 1000;
public static int     tiempoTranscurrido =
                                  (int) System.currentTimeMillis() / 1000;
public static int     tiempoMinimo = 60 * 7; // segundos
public static int     tiempoMaximo = 60 * 15; // segundos
public static int     tiempoParaPrimerDeposito = 60 * 5; // segundos
public static int     minDepositos = 3;
public static int     depositos = 0;

public static void calibrar() throws InterruptedException
{
   for(int i=0; i<5; i++) {
      sensor1Luz = Sensor.S1.readRawValue(); // Leemos sensor [0...1023]
      sensor3Luz = Sensor.S3.readRawValue(); // Leemos sensor [0...1023]
      minBlanco = Math.min( sensor1Luz, minBlanco );
      minBlanco = Math.min( sensor3Luz, minBlanco );
      maxBlanco = Math.max( sensor1Luz, maxBlanco );
      maxBlanco = Math.max( sensor3Luz, maxBlanco );
      avanza( 200 );
      retrocede( 200 );
   }
   minNegro = maxBlanco + 15;
   maxNegro = minNegro + 10;
}

public static void vuelta(int direccion, int tiempo)
                                        throws InterruptedException
{
   int t=0;
   while(t<tiempo) {
      sensor1Luz      = Sensor.S1.readRawValue(); // Leemos sensor [0...1023]
      sensor1Contacto = ( sensor1Luz < 100 );
      sensor2Contacto = Sensor.S2.readBooleanValue();
      sensor3Luz      = Sensor.S3.readRawValue(); // Leemos sensor [0...1023]
      sensor3Contacto = ( sensor3Luz < 100 );

      if(sensor1Contacto || sensor3Contacto)
         retrocede(100);
      else {
         if(direccion==DERECHA) derecha(100);
         else izquierda(100);
         t = t + 100;
      }
   }
   ultimaVuelta = direccion;
}

public static boolean dejarBasura() throws InterruptedException
{
   boolean depositaYA = false;
   sensor1Luz      = Sensor.S1.readRawValue(); // Leemos sensor [0...1023]
```

```
   sensor1Contacto = ( sensor1Luz < 100 );
   sensor2Contacto = Sensor.S2.readBooleanValue();
   sensor3Luz      = Sensor.S3.readRawValue(); // Leemos sensor [0...1023]
   sensor3Contacto = ( sensor3Luz < 100 );

   if( sensor1Luz >= minNegro || sensor3Luz >= minNegro ) {
      while( true ) {
         sensor1Luz      = Sensor.S1.readRawValue();
         sensor1Contacto = ( sensor1Luz < 100 );
         sensor2Contacto = Sensor.S2.readBooleanValue();
         sensor3Luz      = Sensor.S3.readRawValue();
         sensor3Contacto = ( sensor3Luz < 100 );

         if( (sensor1Luz >= minNegro &&
              sensor3Luz >= minNegro) || depositaYA ) {
            // Ambos sensores están sobre el círculo,
            // entonces ya depositamos la basura
            avanza(500);
            alto();
            Motor.B.setPower(3);
            while( !sensor2Contacto ) {
               Motor.B.forward();
               Thread.sleep( 50 );
               sensor2Contacto = Sensor.S2.readBooleanValue();
            }
            Motor.B.stop();
            for(int i=0; i<3; i++) {
               izquierda(200);
               derecha(200);
               avanza(200);
               retrocede(200);
               alto();
            }
            retrocede(2000);
            alto();
            Motor.B.flt(); // float motor: liberamos motor para
                           // bajar recogedor
            depositos++;
            return (true );
         } else if( sensor1Luz >= minNegro ) {
            izquierda( 50 );
         } else if( sensor3Luz >= minNegro ) {
            derecha( 50 );
         } else
            return( false );
      }
   } else
      return( false );
}

public static void daVuelta() throws InterruptedException
{
   int tiempoVuelta = 1500;
   if(ultimaVuelta==DERECHA) {
      if(ultimoContacto==IZQUIERDA && contactosIzquierda>2)
          vuelta(DERECHA,tiempoVuelta);
      else
         vuelta(IZQUIERDA,tiempoVuelta);
   } else {
      if(ultimoContacto==DERECHA && contactosDerecha>2)
```

```
            vuelta(IZQUIERDA,tiempoVuelta);
         else
            vuelta(DERECHA,tiempoVuelta);
      }
      contactosIzquierda = 0;
      contactosDerecha = 0;
      contactosAlternados = 0;
   }

   public static void main (String[] args) throws InterruptedException
   {
      Sensor.S3.setTypeAndMode (SensorConstants.SENSOR_TYPE_LIGHT,
                                SensorConstants.SENSOR_MODE_RAW);

      Sensor.S2.setTypeAndMode (SensorConstants.SENSOR_TYPE_TOUCH,
                                SensorConstants.SENSOR_MODE_BOOL);

      Sensor.S3.setTypeAndMode (SensorConstants.SENSOR_TYPE_LIGHT,
                                SensorConstants.SENSOR_MODE_RAW);

      Sensor.S1.activate();
      Sensor.S3.activate();

      int     EdoIniciando = 1;
      int     Estado = EdoIniciando;
      int     TiempoAvanzando = 0;

      potencia(6);

      calibrar();

      avanza(100);
      TiempoAvanzando = 100;
      while ( true ) {
         sensor1Luz      = Sensor.S1.readRawValue(); // Leemos sensor [0...1023]
         sensor1Contacto = ( sensor1Luz < 100 );
         sensor2Contacto = Sensor.S2.readBooleanValue();
         sensor3Luz      = Sensor.S3.readRawValue(); // Leemos sensor [0...1023]
         sensor3Contacto = ( sensor3Luz < 100 );

         //LCD.showNumber( sensor1Luz ); // Muestra valorLuz en la pantalla LCD
         //LCD.showNumber( FechaHora.getTime() );
         //LCD.showNumber( TiempoAvanzando );

         if(sensor1Contacto) {
            if( ultimoContacto == DERECHA ) contactosAlternados++;
            retrocede(150);
            if(TiempoAvanzando>=2000)
               daVuelta();
            else
               derecha(100);
            TiempoAvanzando = 0;
            ultimoContacto = IZQUIERDA;
            contactosIzquierda = contactosIzquierda + 1;
         } else if(sensor3Contacto) {
            if( ultimoContacto == IZQUIERDA ) contactosAlternados++;
            retrocede(150);
            if(TiempoAvanzando>=2000)
               daVuelta();
            else
```

```
            izquierda(100);
         TiempoAvanzando = 0;
         ultimoContacto = DERECHA;
         contactosDerecha = contactosDerecha + 1;
      } else if((sensor1Luz >= minNegro || sensor3Luz >= minNegro) &&
                tiempoTranscurrido >= tiempoParaPrimerDeposito ) {
         alto();
         if( dejarBasura() ) {
            daVuelta();
            TiempoAvanzando = 0;
         }
      } else if(TiempoAvanzando > 1000 * 20) {
         // Si lleva "avanzando" 20 segundos seguramente
         // está atorado en algún lado.
         retrocede(3000);
         daVuelta();
         TiempoAvanzando = 0;
      } else {
         avanza(100);
         TiempoAvanzando = TiempoAvanzando + 100;
      }
      if( contactosAlternados > 10 ) {
         daVuelta();
         TiempoAvanzando = 0;
      }
      tiempoTranscurrido =
        ((int) System.currentTimeMillis()/1000) - tiempoInicial;
      LCD.showNumber( tiempoTranscurrido ); // Muestra segundos en la LCD

      if( tiempoTranscurrido >= tiempoMinimo && depositos >= minDepositos ) {
         alto(); pita(5); break;
      }
      if( tiempoTranscurrido >= tiempoMaximo ) {
         alto(); pita(5); break;
      }

   } // while

  } // main

} // class
```

5.4 Compilación y carga de los programas

Desde una ventana de comandos (Figura 17), se ejecutan el compilador (`lejosc`) de los programas `.java` y el programa que carga las clases `.class` al RCX (`lejos`).

Antes de compilar y cargar los programas al RCX, asegúrese de que todos los archivos `.java` estén en el mismo directorio y que se encuentre trabajando en ese mismo directorio en la ventana de comandos. Las variables de ambiente deben estar definidas también, como se indica en el apartado 3.4 (Instalación de leJOS en Windows). La torre IR debe estar conectada a la computadora y colocada frente al puerto infrarrojo del RCX y éste último debe estar encendido (Figura 17).

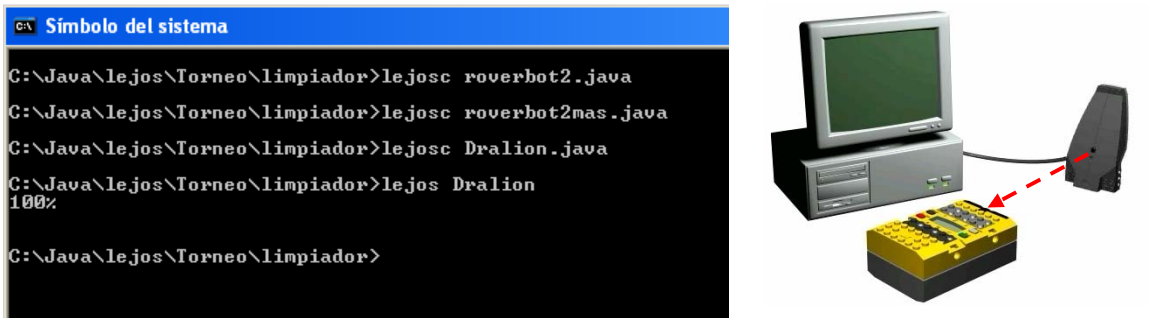


Figura 17. Compilación y carga de programas al RCX

Para el robot **Fegen**:

`lejosc roverbot2.java` `lejosc roverbot2mas.java`	Con el programa `lejosc` debe compilar primero las clases roverbot2 y roverbot2mas que están en los archivos roverbot2.java y roverbot2mas.java respectivamente. El compilador debe crear los archivos roverbot2.class y roverbot2mas.class.
`lejosc Fegen3.java`	Se compila la clase Fegen3 que debe estar dentro del archivo Fegen3.java, creando el archivo Fegen3.class.
`lejos Fegen3`	Por último, el programa `lejos` carga la clase Fegen3 (archivo Fegen3.class) en el RCX.

Para el robot **Dralion**:

`lejosc roverbot2.java` `lejosc roverbot2mas.java`	Con el programa `lejosc` debe compilar primero las clases roverbot2 y roverbot2mas que están en los archivos roverbot2.java y roverbot2mas.java respectivamente. El compilador debe crear los archivos roverbot2.class y roverbot2mas.class.
`lejosc Dralion.java`	Se compila la clase Dralion que debe estar dentro del archivo Dralion.java, creando el archivo Dralion.class.
`lejos Dralion`	Por último, el programa `lejos` carga la clase Dralion (archivo Dralion.class) en el RCX.

Si hay algún error de sintaxis en un archivo .java, el compilador lo muestra y no genera el archivo .class.

Si hay algún error en la carga de las clases al RCX, el cargador lo muestra y no se carga el programa al RCX.

6 Conclusiones

La robótica es un área multidisciplinaria que está en pleno desarrollo. Aún falta mucha investigación para lograr hacer robots inteligentes que nos faciliten la vida o trabajen por nosotros en tareas peligrosas sin arriesgar vidas humanas.

Muchas instituciones de educación superior tienen laboratorios de robótica, en los que los alumnos pueden realizar prácticas y experimentos con robots comerciales o de creación propia. En las instituciones donde se inicia formalmente un área de robótica es posible experimentar con pequeños paquetes comerciales, de bajo costo y que permitan crear diversos tipos de robots, como es el caso de los Lego Mindstorms Robotics Invention System 2.0.

El análisis y diseño del robot son fases muy importantes durante su construcción, porque una vez analizado el problema a atacar con un robot, se debe proponer un modelo de análisis que finalmente se pueda diseñar y construir con el material disponible. Y una vez construido el robot, lo que resta es programarlo.

La programación del robot es fundamental en su desempeño. El uso de un lenguaje de programación de alto nivel, como Java y de herramientas de desarrollo como un ambiente integrado de desarrollo que resalte la sintaxis del mismo, disminuye el tiempo de programación y posterior mantenimiento del programa. El uso de buenas prácticas de programación es importante, porque facilitan la programación, corrección de errores, mejoras y mantenimiento de los programas. Así también, una buena experiencia en programación, puede hacer más propicia la implementación de algoritmos de inteligencia artificial en un robot.

Finalmente, la participación en concursos de robótica es importante, porque proporcionan a profesores y estudiantes metas bien definidas a seguir en la construcción y programación de robots y posteriormente, muchas veces, según los resultados obtenidos en la competencia, una visión general del nivel de competitividad que una institución puede dar a sus estudiantes.

Bibliografía

Ferrari, Guilio, Andy Gombos, Soren Hilmer, Jürgen Stuber, Mick Porter, Jamie Waldinger and Dario Laverde (2002), **Programming Lego Mindstorms with Java**, Syngress Publishing.

Lego Mindstorms RIS 2.0 Constructopedia.

leJOS 2.1.0 README.

Sandria Reynoso, Julio César (2004), CD de recursos del Curso-Taller Programación en Java con robots Lego Mindstorms.