Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Reporte de Proyecto Terminal II

Implementación de Máquina de Turing con robot LEGO

Alumno: Flores Gómez José de Jesús Matrícula:205201483

Trimestre 11-Invierno

Asesora: Dra. Silvia B. González Brambila

Contenido

1 Introducción

En este capítulo se tratarán los conceptos básicos de la Máquina de Turing, qué es LEGO Mindstorms, sus especificaciones y una descripción de las partes que lo componen.

Así mismo, se hablará un poco sobre el proyecto LeJOS, las soluciones que pueden aplicarse en un proyecto y los paquetes disponibles en la API LeJOS.

1.1 Máquina de Turing

A inicios del siglo XX, el matemático David Hilbert se cuestionó la posibilidad de encontrar un algoritmo capaz de determinar la veracidad o falsedad de una proposición matemática, más concretamente, si existiría una forma para determinar si cualquier fórmula en el cálculo de predicados de primer orden, aplicándola a enteros, es verdadera [B1]. El cálculo de predicados de primer orden aplicado a enteros, es suficientemente poderoso como para expresar enunciados del tipo "esta gramática es ambigua", ahora sabemos que para estos problemas no existen algoritmos.
En 1931, Kurt Gödel publicó el teorema de incompletitud, donde construyó una fórmula en el cálculo de predicados aplicado a números enteros, en la que aseguraba que ésta fórmula por sí misma no podría ser probada o refutada dentro del cálculo de predicados. Aun así, el cálculo de predicados no es la única noción matemática para cualquier posible computación [B1].
Alan M. Turing propuso en 1936 a la Máquina de Turing como un modelo de "cualquier posible computación". Este modelo es más parecido a una computadora que a un programa, aun cuando en esa época, las verdaderas computadoras electrónicas o incluso electromecánicas serían creadas años más adelante. Toda propuesta seria para un modelo de cómputo tenía la misma potencia, lo que significa que pueden calcular las mismas funciones o reconocer los mismos lenguajes.
La Máquina de Turing consiste en un control finito, que puede encontrarse en una cantidad finita de estados. Posee una cinta dividida en celdas o bloques, cada uno de los cuales puede almacenar uno de un número finito de símbolos. Una cabeza de lectura y escritura está ubicada siempre por encima de algún bloque de la cinta, se dice que la cabeza está escaneando ese bloque. Inicialmente, la cabeza está ubicada en el primer bloque del lado izquierdo de la cinta que contiene un símbolo o caracter.
La notación formal de la Máquina de Turing[B2] se define por la séptupla:

$$M=(Q,\sum,\Gamma,\delta,q_0,B,F)$$

Siendo

Q: El conjunto finito de estados

∑:El conjunto finito de símbolos de entrada

Γ:El conjunto completo de símbolos de la cinta, siendo Γ un subconjunto de ∑

δ:La función de transición, con argumentos(q, X), siendo q un estado y X un símbolo

de la cinta. Si el valor de δ(q , X) está definido, es la tripleta (p, Y, D), donde:

p es el próximo estado.

Y es el símbolo, contenido en Γ, escrito en el bloque a escanear, reemplazando al anterior valor.

D es una dirección, puede ser L(izquierda) o R(derecha), indicando la dirección en la que la cinta se moverá.

q_0: El estado inicial, miembro de Q, en el que el control estará al inicio.

B: Símbolo en blanco, que está en Γ, más no en ∑, por lo que no es un símbolo de

entrada.

F: El conjunto de estados finales, subconjunto de Q[B1].

Esta Máquina cuenta con un alfabeto de entrada, una cinta y un control finitos. La cinta se divide en celdas o bloques, donde cada una contiene un caracter del alfabeto definido para la cinta. La entrada es una cadena de caracteres de longitud finita, los bloques que están tanto a la izquierda como a la derecha de esa cadena contienen el símbolo nulo [B2].

1.2 LEGO Mindstorms

El LEGO Mindstorms es una serie de productos educativos de la casa LEGO[B3], fabricante de juguetes basados en bloques. Los productos LEGO Mindstorms están diseñados para permitir construir fácilmente robots. La versión NXT es la tercera generación de esta serie de productos [B4].

1.2.1 Historia

En 1988, el grupo LEGO colaboró con el Massachusetts Institute of Technology (MIT) en el desarrollo de un "bloque inteligente". El LEGO Mindstorms RCX (fig. 1.1), producto de este desarrollo, es develado al público durante las ferias del juguete de Nüremberg, Londres y Nueva York de 1998.

Fig. 1.1 Mindstorms RCX

En 2006 nace la nueva generación, llamada NXT (fig. 1.2), que contenía mejoras respecto al modelo anterior, como mejores motores, mejores sensores, interfaz USB y Bluetooth [B4].

Fig. 1.2 Mindstorms NXT

1.2.2 Especificaciones técnicas del NXT

El bloque NXT tiene las siguientes especificaciones:

- Microcontrolador ARM7 de 32 bits
- Memoria Flash de 256 Kbytes
- Memoria RAM de 64 KBytes
- Microcontrolador AVR de 8bits

- Comunicación inalámbrica Bluetooth Class II V 2.0
- Puerto USB de velocidad completa
- 4 puertos de entrada, plataforma digital de 6 cables
- 3 puertos de salida, plataforma digital de 6 cables
- Pantalla LCD de 100x64 píxeles
- Bocina con canal de sonido de 8bits de resolución

1. Fuente de alimentación por 6 baterías AA

1.2.3 Motores y sensores

1.2.3.1 Motor NXT

El motor NXT (fig. 1.3) incluye un sensor de rotación, que permite controlar los movimientos del robot de manera precisa. El sensor de rotación mide los giros ya sea en grados o en vueltas completas. Además, el sensor de rotación permite configurar distintas velocidades de giro [B3].

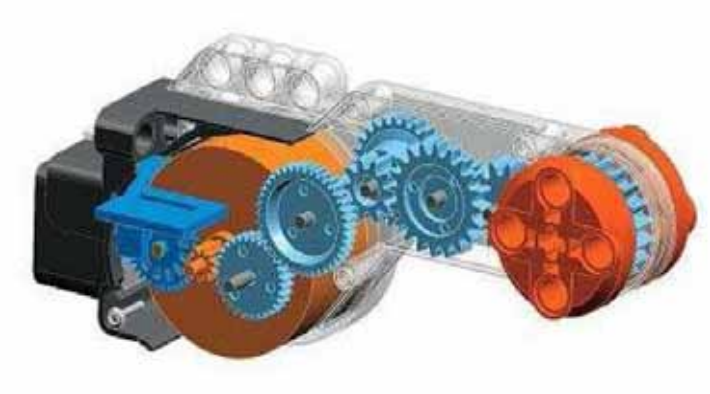


Fig 1.3 Motor NXT

En la tabla 1.1 se enumeran sus características [B7]:

Voltaje	Torque	Velocidad rotacional	Amperaje	Potencia mecánica	Potencia eléctrica	Eficiencia
4.5 V	16.7 N.cm	33 rpm	0.6 A	0.58 W	2.7 W	21.4 %
7 V	16.7 N.cm	82 rpm	0.55 A	1.44 W	3.85 W	37.3 %
9 V	16.7	117 rpm	0.55 A	2.03 W	4.95 W	41 %

	N.cm					
12 V	16.7 N.cm	177 rpm	0.58 A	3.10 W	6.96 W	44.5 %

Tabla 1.1 Tabla de Estado

1.2.3.2 Sensor de luz

El sensor de luz (fig. 1.4) permite distinguir entre luz y obscuridad, midiendo la intensidad de la luz, por ejemplo, en un cuarto o la intensidad de la luz reflejada por objetos o superficies [B4].

Fig. 1.4 Sensor de luz

1.3 El proyecto LeJOS

LeJOS es un proyecto *open source* creado para desarrollar una infraestructura tecnológica para desarrollar software en los productos LEGO Mindstorms, utilizando tecnología JAVA.

Este proyecto ofrece soporte para productos LEGO Mindstorms NXT y su generación anterior, la RCX [B6].

El proyecto LeJOS provee las siguientes soluciones:

- Máquina virtual de JAVA para el módulo NXT
- API LeJOS para el módulo NXT
- Comunicaciones LeJOS para PC
- Comunicaciones LeJOS para JavaME
- Herramientas LeJOS

1.3.1 Soluciones LeJOS

Usando el proyecto LeJOS, se pueden utilizar dos filosofías de desarrollo:

- Solución distribuida
- Solución no distribuida

1.3.2 Solución distribuida

Al utilizar la solución distribuida, se puede distribuir la lógica de la solución en dos partes:

- Parte A:
  - Computadora personal (JAVA SE)
  - Teléfono móvil(JAVA ME)
- Parte B:
  - Módulo NXT (JAVA LeJOS)

Con este tipo de soluciones, se puede desarrollar el sistema de control en una máquina con mejores recursos, por ejemplo, una computadora personal, para enviar y recibir información desde el bloque NXT.

1.3.3 Solución no distribuida

La otra manera de desarrollar un proyecto es colocando la lógica únicamente en la PC, de manera que el bloque NXT es controlado de manera remota por la PC.

1.3.4 Paquetes LeJOS

El proyecto LeJOS tiene disponibles los paquetes [B5] que se muestran en la tabla 1.2:

Paquetes	
java.awt	Paquete mínimo AWT para clases formadas con coordenadas enteras
java.awt.geom	Paquete mínimo AWT para Point2D, Line2D y Rectangle2D
java.io	Soporte de Entrada/Salida
java.lang	Clases de núcleo Java

java.lang.annotation	Soporte básico para anotaciones
java.net	Soporte para sockets vía PC SocketProxy
java.util	Utilerías
javax.bluetooth	Clases estándar Bluetooth
javax.microedition.io	J2ME Entrada/Salida
javax.microedition.lcdui	Clases de interfaz de usuario J2ME LCD
javax.microedition.location	API de locación
LeJOS.addon.gps	Paquete para parseo GPS
LeJOS.addon.keyboard	Soporte para teclados Bluetooth SPP
LeJOS.charset	Conjunto de caracteres simples API para LeJOS
LeJOS.geom	Soporte para formas geométricas de robótica usando coordenadas de punto flotante
LeJOS.io	Soporte especifico de LeJOS para java.io
LeJOS.nxt	Acceso a sensores, motores, etc.
LeJOS.nxt.addon	Acceso a sensores y motores legados y de terceros, además de hardware no incluído en el kit LEGO NXT.
LeJOS.nxt.comm	Clases de comunicación NXT
LeJOS.nxt.debug	Clases para debugging de hilos
LeJOS.nxt.rcxcomm	Emulación de clases de comunicación RCX
LeJOS.nxt.remote	Acceso remoto NXT sobre Bluetooth
LeJOS.robotics	Interfaces de abstracción de hardware para el paquete de robótica
LeJOS.robotics.localization	Soporte para localización

LeJOS.robotics.mapping	Nueva propuesta para navegación
LeJOS.robotics.navigation	Clases de navegación
LeJOS.robotics.proposal	Soporte para mapas
LeJOS.robotics.subsumption	Soporte para arquitectura de subsunción
LeJOS.util	Más clases de utilerías

Tabla 1.2. Listado de los paquetes de la API LeJOS

1.4 Trabajos relacionados

La Máquina de Turing ha sido implementada en varias ocasiones. En particular, ha sido implementada usando el kit LEGO Mindstorms. Una de estas Máquinas de Turing (figs. 1.5 y 1.6) fue realizada por un equipo de estudiantes de la Universidad de Aarhus, Dinamarca en 2008-2009 conformado por los estudiantes Anders Nissen, Martin Have, Sean Geggie y Mikkel Vester [B8].

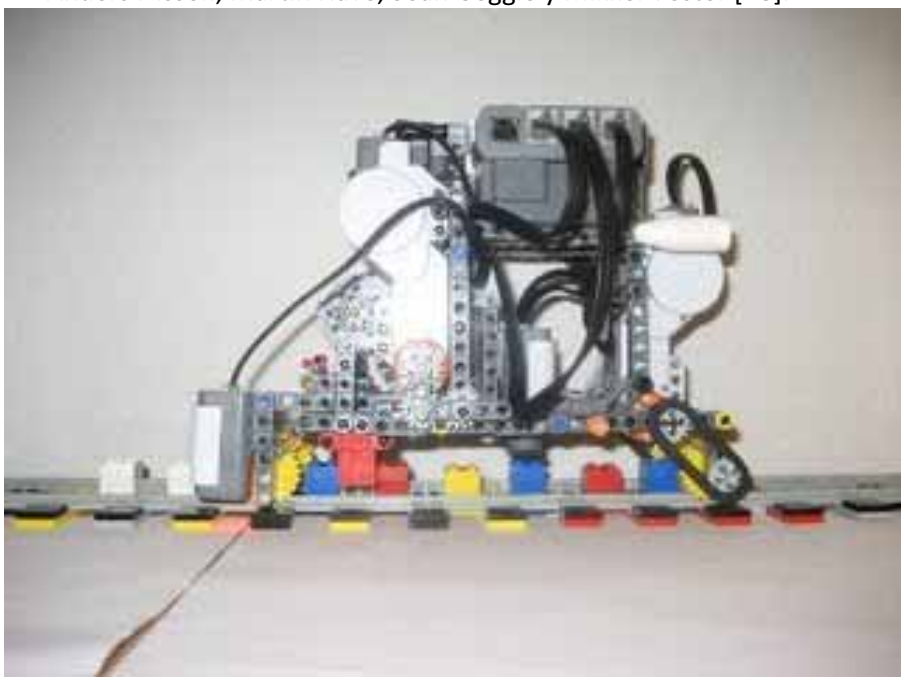


Fig. 1.5 Vista lateral de la Máquina de Turing

Fig. 1.6 Vista de la Máquina y la cinta

Esta máquina utiliza una tabla de transiciones almacenada en un archivo en la computadora personal, el método para controlar la puesta en marcha de la máquina es mediante una interfaz gráfica que se ejecuta en la computadora personal.

Otra Máquina de Turing implementada con robots LEGO es la realizada en 2001 por Denis Cousineau[B9], profesor de ciencia cognitiva y métodos cuantitativos de la Universidad de Montreal, Canadá[B10], utilizando el modelo RCX y lenguaje LOGO. La cinta es sustituida por una pila y los valores se almacenan en cilindros, cada valor está representado por un color en forma de código de barras (fig. 1.7), los cilindros se apilan en una estructura y al momento de ser leídos, se utiliza un actuador que mueve una palanca que empuja al cilindro hacia un sensor de colores, para después ser sacado de la pila. Para introducir un dato, se mueve 90º la estructura, se coloca el cilindro y después se mueve la estructura a la posición original (fig. 1.8).

Fig. 1.7 Bloque con código de barras

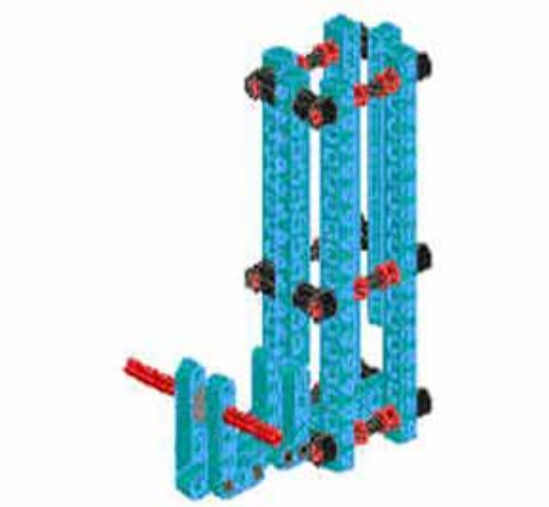


Fig. 1.8 Estructura que funciona como pila

Existen en internet varios ejemplos de Máquinas de Turing, aunque muchos carecen de documentación. La tabla 1.3 muestra las diferencias de los proyectos citados anteriormente con el presente proyecto.

Diferencias entre implementaciones existentes y el presente proyecto

	Máquina de Turing con LEGO NXT, **Equipo de Aarhus[B8]**	**Máquina de Turing con LEGO RXC,** **Denis Cousineau[B9]**	**Implementación de Máquina de Turing,** **Jesús Flores G.**
Lenguaje de programación	*JAVA*	*LOGO*	*JAVA*
Esquema de cinta de entradas/salidas	*Bloques que representan bits y que pueden estar ubicados a la derecha o a la izquierda, cada posición representa un valor*	*Cilindros que representan datos mediante código de barras, varios cilindros puestos uno encima del otro forman una pila*	*Bloques que al inclinarse hacia la derecha o izquierda, representan un valor distinto*
Esquema de lectura de datos	*Dos sensores táctiles que determinan el valor de un bloque en base a la posición del mismo*	*Sensor de color que detecta el valor de los códigos de barras que contienen los cilindros en base a su color(blanco*	*Sensores de luz que al inclinarse el bloque, el sensor de colocado encima de lé, envía un valor numérico que a su vez, es*

		o negro)	*enviado por el módulo NXT a la computadora donde es interpretado.*
Esquema de escritura de datos	*Un actuador ubicado por encima de la cinta, mueve el bloque hacia la posición contraria, cambiando el valor anterior.*	*Un actuador mueve la pila y suelta bloques cilíndricos*	*Los bloques cambian de posición mediante actuadores ubicados a los lados, al cambiar la posición, cambiará el valor de la luz reflejada a los sensores*
Método de comunicación con la PC	*Bluetooth*	*Infrarrojo*	*Bluetooth*

Tabla 1.3 Diferencias entre implementaciones existentes y el presente proyecto

Si bien los tres proyectos pueden usar los mismos sensores y actuadores, la interpretación del funcionamiento de la cinta es distinta. La cinta debe ser tanto de lectura como de escritura, por lo que se representa a través de una serie de objetos físicos, más concretamente, bloques de LEGO que simulan ser bits. En el caso del proyecto de Denis Cousineau[B5], el esquema de memoria elegido consiste en una pila.

Si bien los dos proyectos pueden usar los mismos sensores y actuadores, la interpretación del funcionamiento de la cinta es distinta. En el proyecto creado por el equipo de la Universidad de Aarhus[B4], cada caracter del alfabeto de entrada es representado por dos bloques, cada uno con dos posiciones posibles, por lo que se pueden tener hasta cuatro valores, tres de ellos definidos en el alfabeto de entrada y el restante es el símbolo nulo.

2 Justificación del proyecto

Una máquina de Turing un modelo matemático que permite comprender la importancia y el poder del proceso algorítmico. En la enseñanza es importante mostrar ejemplos visuales de un determinado tema a fin de que el alumno sea capaz de entenderlo más fácilmente. En el caso particular del tema de la Máquina de Turing, se le explica al alumno su funcionamiento de manera teórica, sin embargo, nunca ve su funcionamiento, por lo que este proyecto podría apoyar la comprensión del concepto.

Una Máquina de Turing lee datos de entrada desde la cinta, compara el resultado de esa lectura con un valor definido en la tabla de transiciones para posteriormente decidir a qué estado deberá cambiar o si debe permanecer en el actual, sobrescribir o dejar el valor en la cinta en la posición actual y mover la cinta en la dirección que indica la tabla de transiciones. El alumno entonces, deberá imaginarse estas operaciones, pero resulta ser un proceso bastante mecanizado y complejo, por lo que pueden encontrarse dificultades al tratar de entender el funcionamiento de la Máquina de Turing.

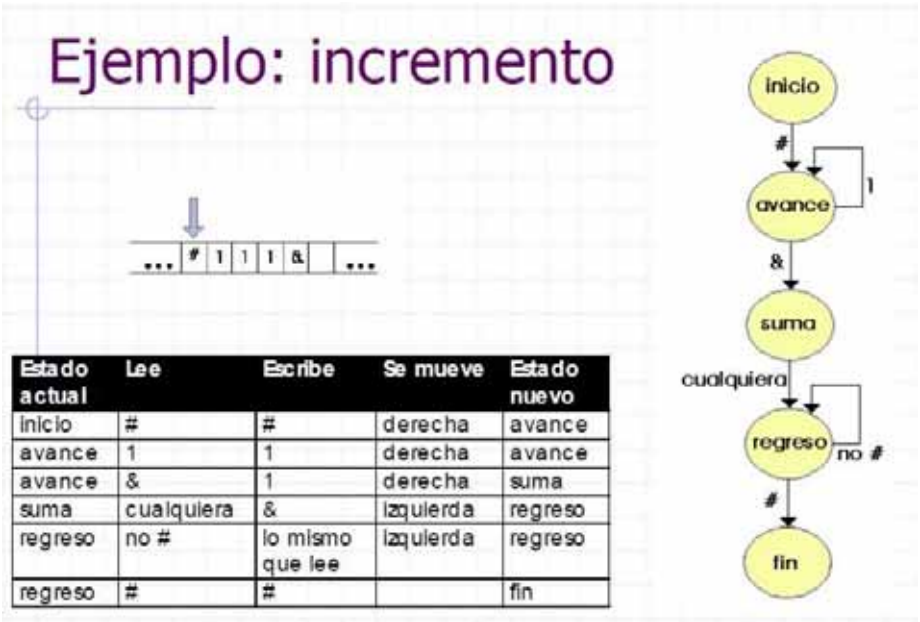
La solución es crear una máquina similar a la planteada originalmente por Alan Turing, cuyo modelo no era realmente un programa, sino una máquina completa con control electrónico, una cinta móvil y una cabeza de lectura/escritura.

El crear la Máquina de Turing con un robot LEGO permite ahorrar tiempo al no tener que construir las estructuras y dispositivos necesarios con materiales que requieran invertir mucho tiempo, como sería el usar metales y/o plásticos que requieren el uso de pegamento, soldadura y/o tornillos. Por lo tanto, el uso de tecnologías LEGO permite emplear mayor tiempo en programar.

2.1 Enseñanza del tema Máquina de Turing

En muchas de las escuelas, universidades e instituciones que imparten carreras relacionadas con la computación, la enseñanza de la teoría de autómatas es un fundamento importante de la carrera. Uno de los temas de la teoría de autómatas es la Máquina de Turing, sin embargo, la forma en que se enseña este tema es muy teórica y poco didáctica en el sentido de que el alumno no presencia el funcionamiento de la Máquina, sino que lo aprende por medio de diagramas y ejemplos que si bien son ilustrados, no muestran el funcionamiento de la Máquina de Turing.

Inicialmente el alumno debe entender el funcionamiento de un autómata y fundamentos como qué es un diagrama de transición y una tabla de estados. Cuando se aborda el tema de la Máquina de Turing, se le explica al alumno que se trata de una máquina mecánica con una cinta infinita donde se almacena la información, una cabeza de lectura y escritura, una tabla finita que almacena las transiciones y un registro de estados que almacena el estado de la Máquina de Turing. Luego entonces, se dan ejemplos de Máquinas de Turing que realizan funciones, por ejemplo, i un valor almacenado en la cinta (fig. 2.1), mostrándole la tabla de estados y el diagrama de transiciones.



Estado actual	Lee	Escribe	Se mueve	Estado nuevo
inicio	#	#	derecha	avance
avance	1	1	derecha	avance
avance	&	1	derecha	suma
suma	cualquiera	&	Izquierda	regreso
regreso	no #	lo mismo que lee	Izquierda	regreso
regreso	#	#		fin

Fig 2.1 Ejemplo de una Máquina de Turing

Una parte complicada de este tema es visualizar a la cinta desplazándose debajo de la cabeza, para que posteriormente la cabeza lea información en dicha cinta y en base al resultado de la lectura la Máquina de Turing decida qué dato debe escribir, en qué posición de la cinta y si debe moverse la cinta hacia la derecha o hacia la izquierda.
Lo común es usar imágenes que muestren paso a paso los movimientos de la Máquina de Turing (fig. 2.2). A primera vista parece simple, sin embargo y dependiendo del valor de los datos almacenados en la cinta con los cuales se va a operar, una Máquina de Turing puede llegar a tener varias decenas de movimientos desde el inicio hasta la finalización de las operaciones. Por ejemplo, en la fig. 2.2 se realizan 12 pasos para incrementar la variable x, cuyo valor inicial es tres, en uno.

Fig 2.2 Ejemplo de los pasos a realizar por una Máquina de Turing

Una forma más didáctica de demostrar el funcionamiento de una Máquina de Turing es mediante el uso de implementaciones robóticas, que facilitarán al alumno la comprensión del tema, a la vez que pueden resolverle dudas sobre su funcionamiento. Así, la Máquina de Turing con robot LEGO es una implementación que puede ser usada con fines didácticos (fig 2.)

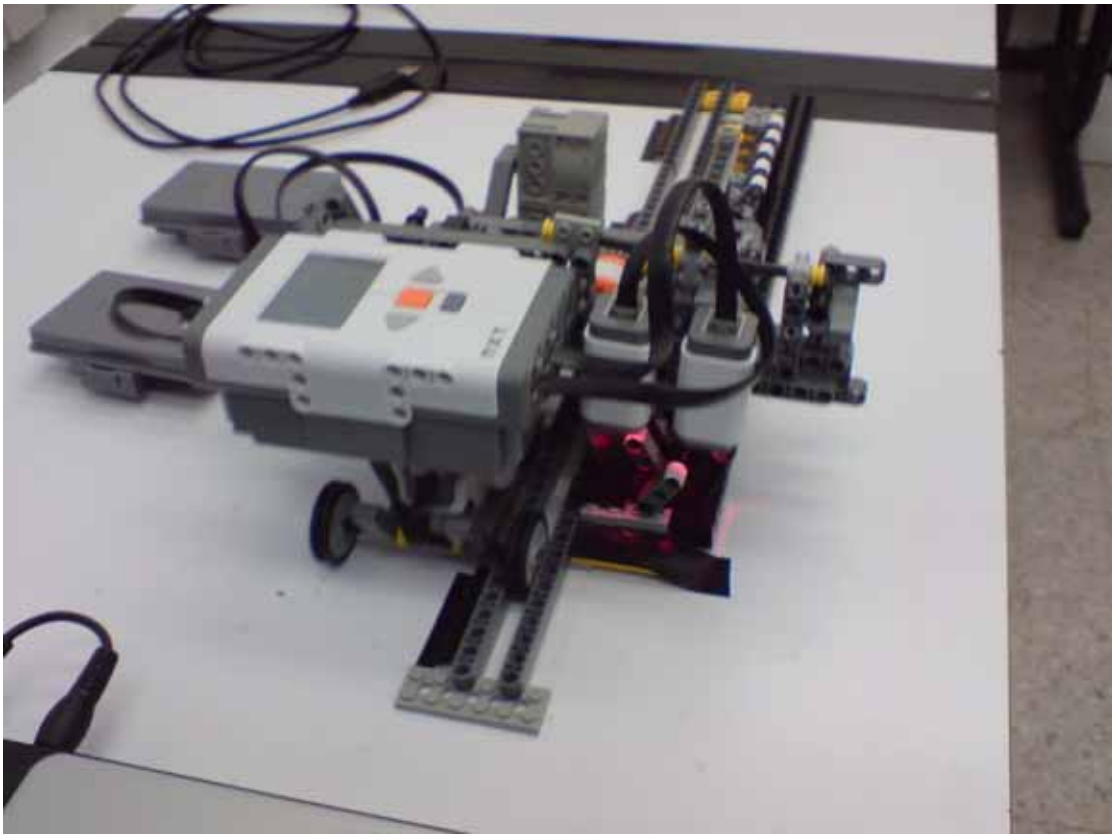


Fig 2.3 Máquina de Turing con robot LEGO

3 Objetivos del proyecto

En este capítulo se enumeran los objetivos, tanto generales como particulares del proyecto. Fue alrededor de esos objetivos que se fue creando el diseño e implementación del proyecto.

3.1 Objetivos generales

Demostrar el uso de una Máquina de Turing física con un alfabeto binario, usando un robot, bloques, sensores y actuadores LEGO y una computadora personal, programando el robot para que pueda ejecutar los movimientos propios de la Máquina de Turing y la computadora sea el medio de entrada para las tablas de transiciones y procese dichas tablas.

3.2 Objetivos particulares

- Construir una estructura con bloques LEGO que cumpla la función de una cinta.
- Construir una estructura que permita el movimiento de una cabeza de lectura y escritura compuesta de sensores y actuadores.
- Programar el módulo NXT para que sea capaz de reconocer y enviar señales a los actuadores y sensores para que la cabeza de lectura y escritura sea capaz de realizar los movimientos definidos en la Máquina de Turing,

• Crear un protocolo de transmisión de datos e instrucciones para que la computadora
y el módulo NXT puedan comunicarse y sincronizarse.
• Programar una interfaz de control en una computadora personal que permita al usuario introducir una tabla de transiciones que represente una Máquina de Turing y que permita iniciar, pausar o detener su funcionamiento.
• Programar una computadora personal para que se comunique con el módulo NXT, reciba los datos leídos por la cabeza de lectura/escritura, procese en base a una tabla de transiciones y envíe los resultados al módulo NXT.

4 Descripción técnica

En este capítulo se describen las partes que componen al robot, el robot completo, diagramas, su funcionamiento y la explicación del código programado tanto para el robot como para la computadora.

4.1 Partes del robot

Para el robot se utilizaron:

- Dos motores, denominados *motor A* y *motor B* alrededor de los cuales se construyeron dos estructuras, denominadas *estructura A* (figs. 4.1 y 4.2) *y estructura B* (figs. 4.3 y 4.4). La *estructura A* permite mover el brazo de la cabeza lectora, mientras la *estructura B* tiene la función de ser el soporte del robot completo, así como desplazarlo a lo largo de la cinta.

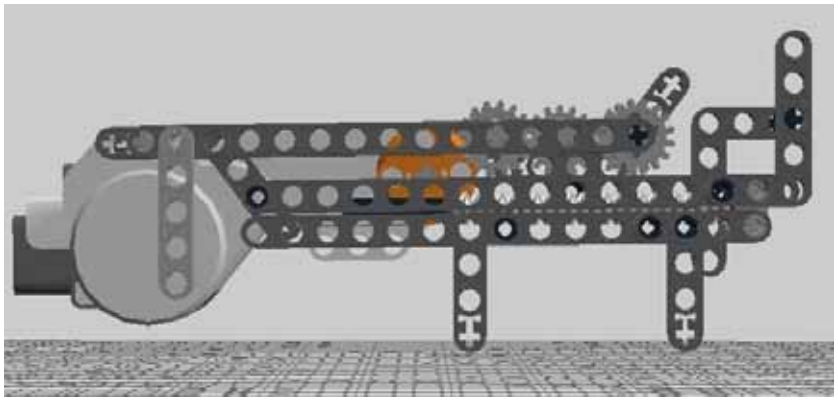


Fig. 4.1 Vista lateral de la estructura A

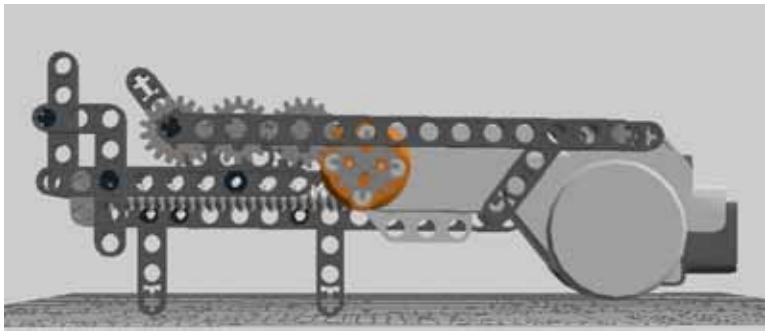


Fig. 4.2 Vista lateral de la estructura B

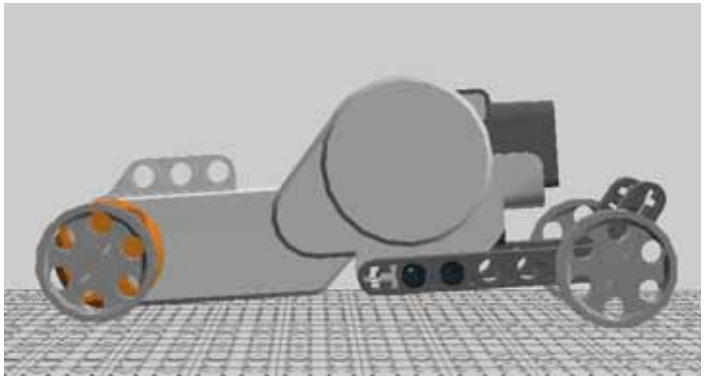


Fig. 4.3 Vista lateral de la estructura B

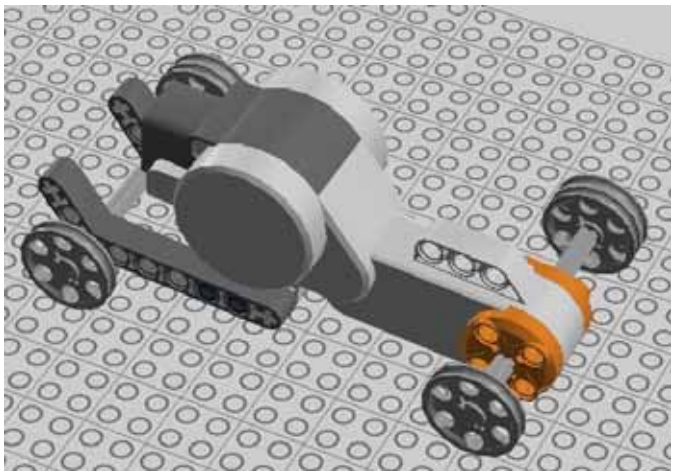


Fig. 4.4 Vista en perspectiva de la estructura B

- Dos sensores ópticos (fig. 4.5), que detectan la intensidad de la luz reflejada por los bloques que conforman a la memoria, el sensor es activado cuando el robot determina que se encuentran encima del bloque que debe leer. Cada sensor está ubicado en una posición tal que permite determinar si en su posición se encuentra inclinado un bloque o no. Para determinar el valor de un

bloque, los resultados de ambos sensores se toman en cuenta, restando los valores de las lecturas entregadas

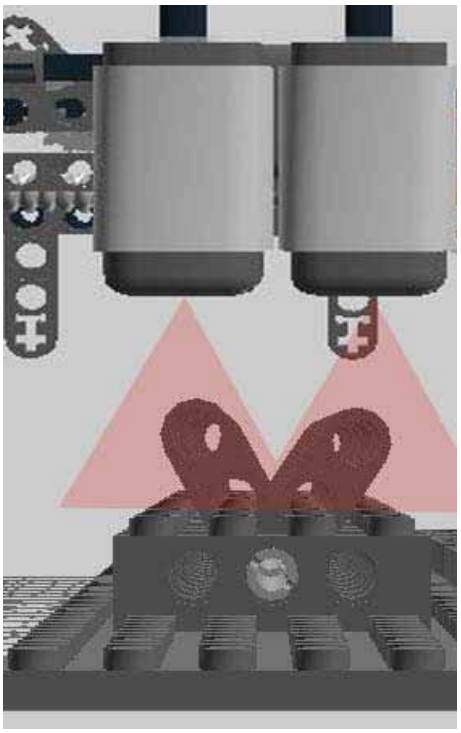


Fig. 4.5 Sensores posicionados sobre la cinta

- Un riel (fig. 4.6), sobre el cual se mueven los ejes delantero y trasero derechos del robot. El riel tiene la función de mantener el movimiento de desplazamiento del robot hacia adelante y atrás, evitando que se salga de una ruta definida.

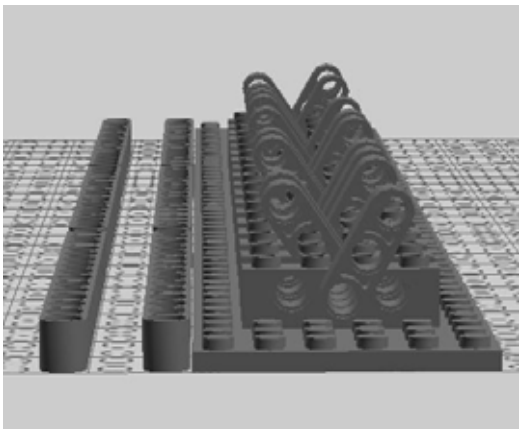


Fig. 4.6 Riel y cinta

- Una cinta (figs. 4.7 y 4.8), compuesta por 18 bloques que mueve el robot hasta en tres posiciones distintas cada uno. Cada bloque representa un valor en la

cinta, si se ve de frente la cinta, quedando el robot y el riel del lado izquierdo, el bloque que está inclinado hacia la derecha vale cero, si está vertical, tiene valor nulo y si está inclinado hacia la izquierda, vale uno, cada celda estará compuesta por un número de bloque dentro de la programación del robot, contados a partir del primer bloque que el robot detecte, de tal manera que el robot puede identificar en qué bloque se encuentra cuando realiza una operación

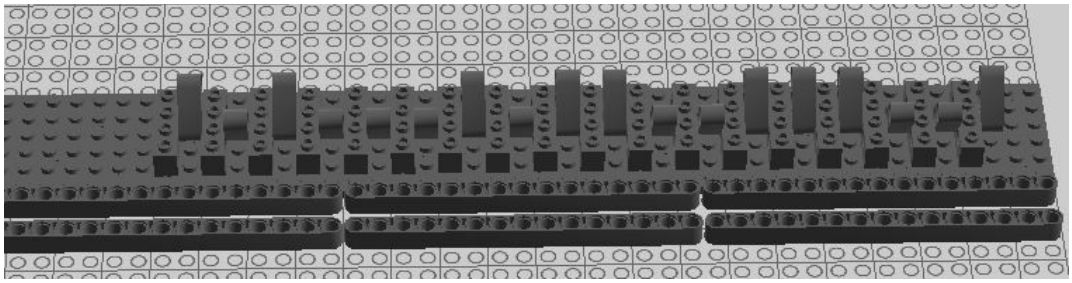


Fig. 4.7 Vista en perspectiva de la cinta

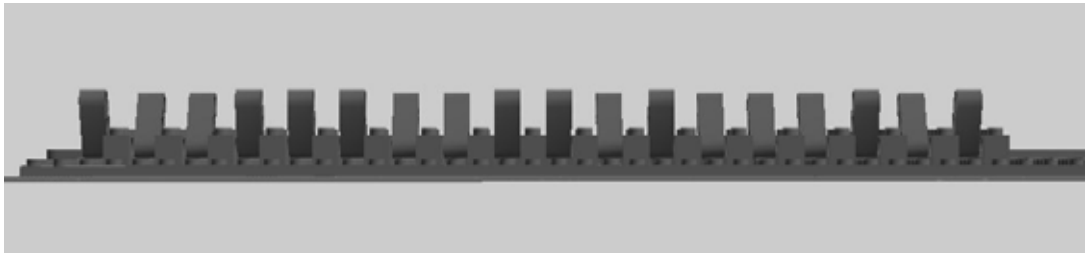


Fig. 4.8 Vista lateral de la cinta

- Módulo LEGO NXT(fig. 4.9), que cumple con la función de control de los sensores y motores, realizar la comunicación bidireccional hacia la computadora personal, recibir instrucciones y enviar valores de lecturas. El módulo NXT está colocado en la parte superior de la estructura B de manera horizontal, de forma que la pantalla es visible.

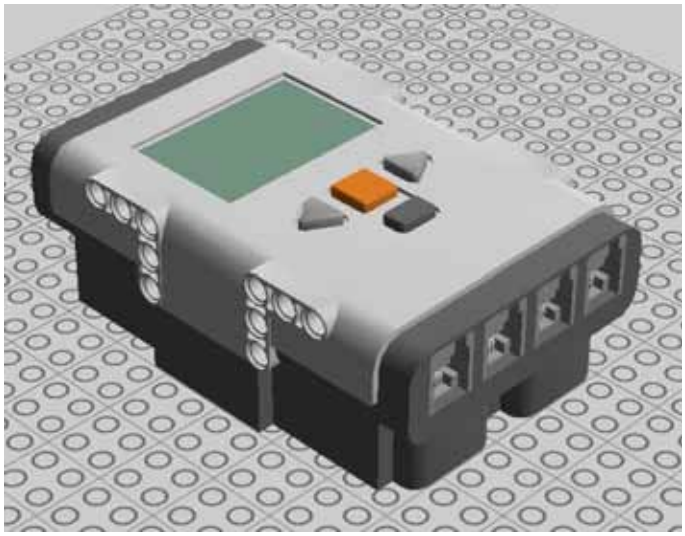


Fig. 4.9 Módulo NXT

4.2 Robot completo

Para construir el robot de la Máquina de Turing de este proyecto se utilizaron 110 piezas. Para el funcionamiento del proyecto se utilizó la solución distribuida, con el robot manejando la parte de la lógica que se encarga de realizar los movimientos, realizar lecturas, escrituras, desplazarse a las celdas indicadas, desplegar en pantalla las operaciones y el resultado de las lecturas y la comunicación de dichas operaciones y resultados a la computadora, mientras la computadora se encarga de la lógica que lee las tablas de estado, decide cual es la operación a realizar, el valor de una escritura y las comunicaciones con el robot.

Los diagramas de dichas estructuras se pueden ver en las figuras 4.10, 4.11, 4.12 y 4.13.

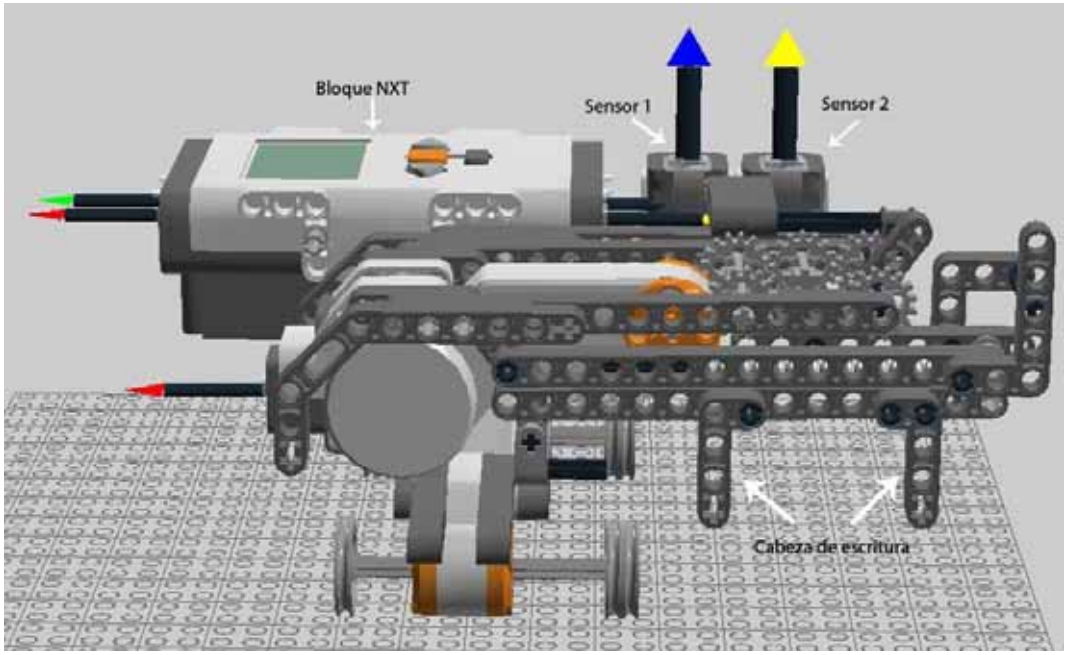


Fig. 4.10 Diagrama lateral del robot completo

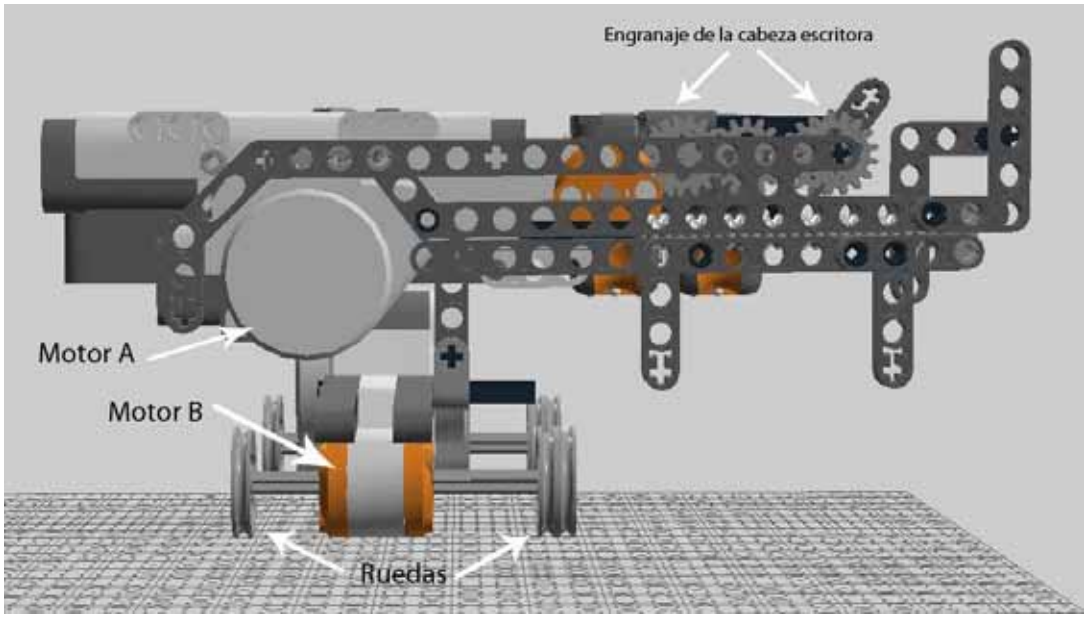


Fig. 4.11 Diagrama lateral del robot completo

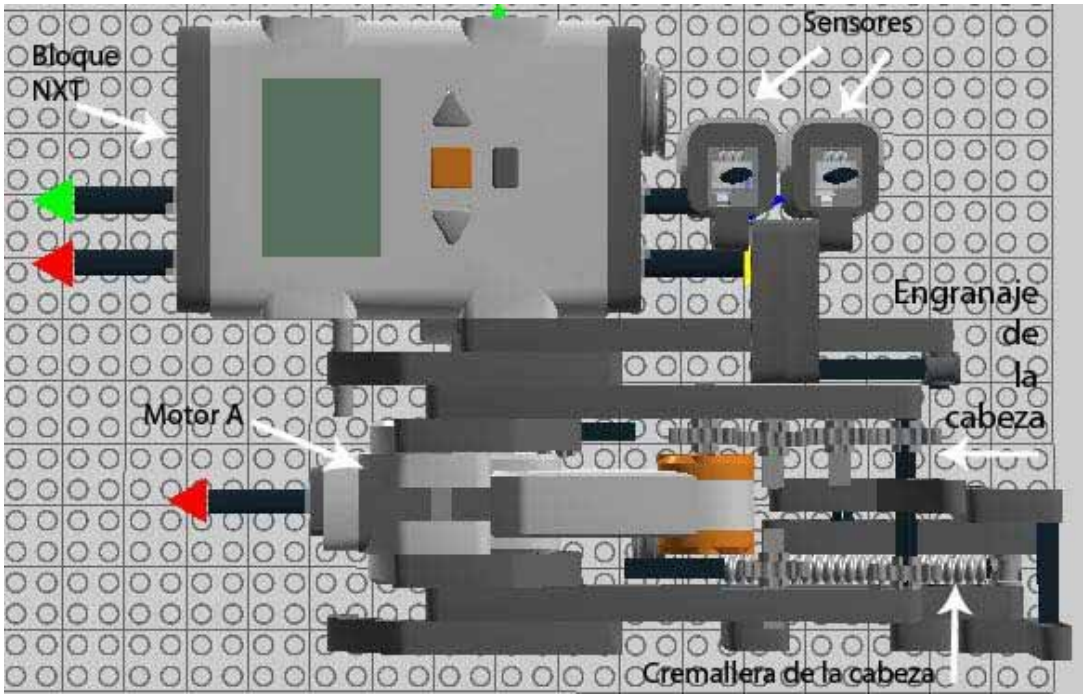


Fig. 4.12 Diagrama superior del robot completo

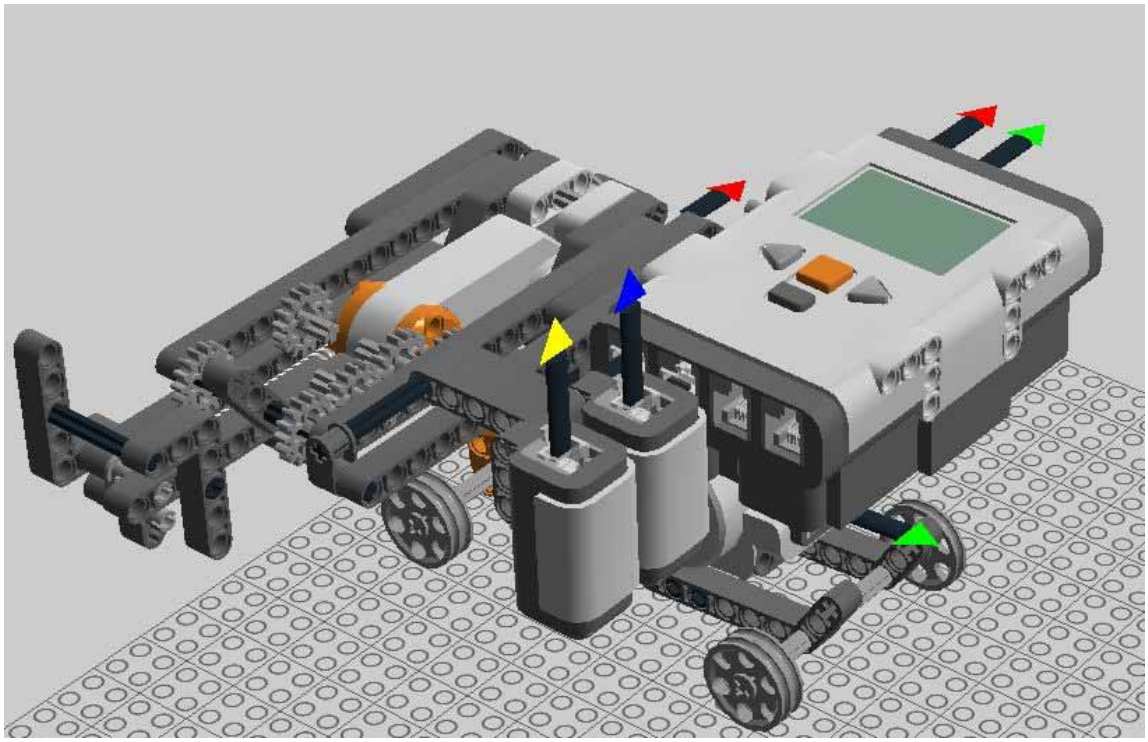
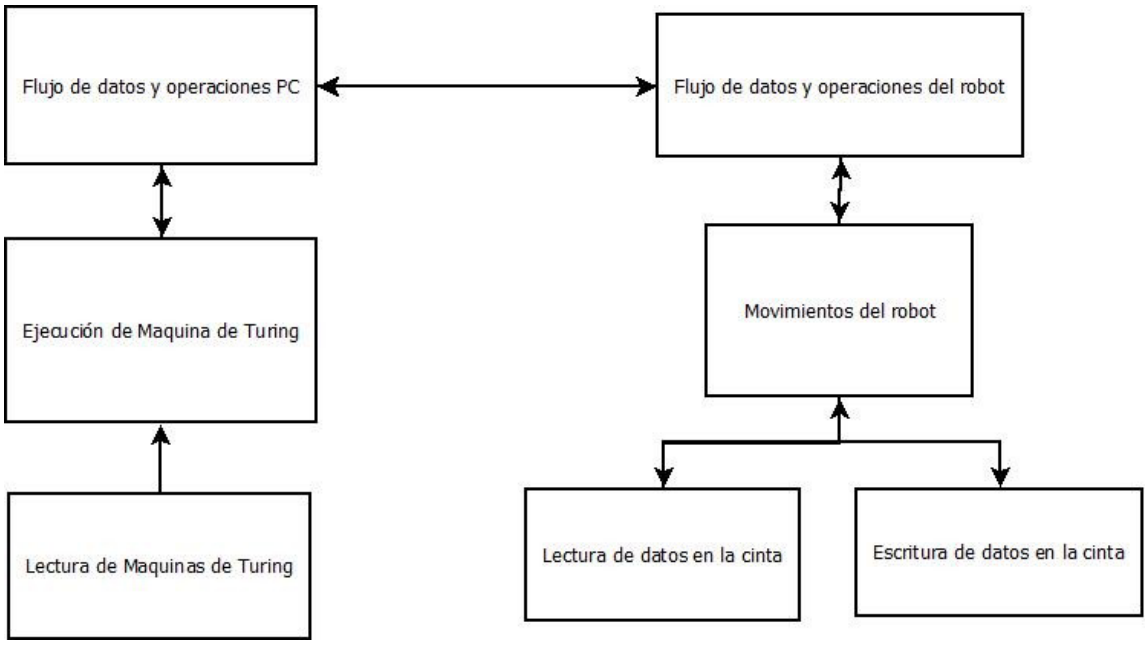


Fig. 4.13 Diagrama en perspectiva del robot completo

4.3 Funcionamiento general

El software de la Máquina de Turing construida con robot LEGO NXT está constituida por dos partes: el programa que se ejecuta en el robot, llamado RobotTuring y un programa JAVA llamado TuringPC que corre en una computadora personal, el cual se encarga de buscar los archivos con extensión *.tur* que están almacenados en el directorio de trabajo. Una vez leídos estos archivos, el programa despliega sus nombres al usuario y le permite elegir cuál ejecutar. Una vez elegido un archivo, el programa se encarga de cargar el archivo a memoria, separar la información y generar unas tablas de estado que contienen 5-tuplas, estas tuplas contienen las operaciones que deben ejecutarse en la Máquina de Turing, siendo estas almacenadas en un arreglo de enteros bidimensional llamado *tabla*. Una vez realizado esto, el programa intenta buscar al robot NXT mediante bluetooth, si el robot se está ejecutando, acepta la conexión y ambos crean *streams* o flujos de entrada y salida. Es entonces que inicia un protocolo para el intercambio de datos y operaciones entre el robot y la computadora personal.



Diag. 4.1 Diagrama por bloques del proyecto

4.4 Tablas de estado

En la teoría de autómatas, una tabla de estados o tabla de transición de estados es una tabla que muestra a qué estado se deberá mover una máquina de estado finito, basándose en su estado actual y a otras entradas [B1].

La Máquina de Turing con robot LEGO utiliza tablas de estado que se cargan a memoria desde un archivo con extensión *.tur* una vez elegida la Máquina de Turing con la que se desea trabajar.

La estructura del archivo consiste en múltiples líneas de texto simple, cada línea contiene cinco números separados por comas, que representan tablas de estado. El programa TuringPC que se ejecuta en la computadora personal se encarga de leer línea por línea, generando un arreglo de enteros en memoria que posteriormente servirá para hacer funcionar a la Máquina de Turing.

El siguiente es un ejemplo de un archivo *.tur* que representa a un sumador:

```
1,0,0,0,1
1,1,1,0,2
2,0,1,0,3
2,1,1,0,2
3,0,0,1,4
3,1,1,0,3
4,0,0,1,4
4,1,0,1,5
5,0,0,0,0
5,1,1,1,5
```

Tabla 5.1 Ejemplo de tablas de estado

El sumador de este ejemplo tiene 5 estados y realiza suma unaria de dos números. Los números están escritos en la cinta y separados por un cero. Esta Máquina de Turing escribirá el resultado en la cinta, separando del resto de la cinta mediante ceros.

Los valores de la primera columna indican el valor del estado actual; la segunda columna indica el valor leído, si coincide ese valor con el que leyó el robot, se debe usar el valor de la tercera columna, que indica el valor que el robot deberá escribir, mientras que la cuarta columna le indica hacia qué dirección se deberá mover para realizar dicha escritura, que en el caso de ser un cero se moverá hacia la derecha y en el caso de un uno, hacia la izquierda. La última columna indica que

si las condiciones de esa línea se cumplen, el estado actual deberá cambiar al estado indicado por dicha columna.

4.5 Cinta

La cinta contiene un total de 18 celdas, pero gracias a la forma en que se implementó la Máquina de Turing con robot LEGO, puede ser de cualquier tamaño. Es capaz de tener tres valores, dos de ellos usables para realizar operaciones, los cuales son cero y uno y un valor nulo que indica que es una celda sin usar. Cuando se van a iniciar las operaciones, el robot busca desde el inicio, celda por celda a la primera que contenga el valor de uno, mientras el cero sirve como separador.

Por ejemplo, en un sumador, el primer grupo de celdas a sumar está separado del segundo grupo por un cero, mientras que las demás celdas tienen valor nulo.

4.6 Protocolo de intercambio de datos y operaciones

El protocolo creado para este proyecto está dividido en dos partes. La primera parte, o protocolo 1(Diag 4.2), se encarga de encontrar la primera celda válida, mientras la segunda parte, o protocolo 2(Diag 4.3), ejecuta las acciones de la Máquina de Turing en base a las tablas de estado.

En la computadora personal, el programa TuringPC carga las tablas de estado, pero para poder iniciar las operaciones en el robot, debe indicarle a este que busque a lo largo de la cinta la primera celda que contenga un valor válido para comenzar las operaciones, este valor es igual a uno. El protocolo se inicia cuando TuringPC entra en un ciclo *do-while*, envía el valor de una variable llamada *fin,* que inicialmente vale uno; cuando RobotTuring la recibe, determina qué debe iniciar, en este instante ambos lados tienen una variable *fin* distinta de cero, por lo que se envía la posición de lectura, que inicialmente es la celda uno, desde la computadora hacia el robot. El robot ejecuta la instrucción de lectura en la celda indicada y una vez obtenido el valor de la lectura, que puede ser cero, uno o nulo, envía el valor a la computadora. Si el valor que la computadora recibe no es válido para comenzar las operaciones, esto es, es distinto de uno, entonces incrementa el valor de la variable que contiene la celda a leer, y reinicia el ciclo, reenviando el valor de *fin y* pidiéndole al robot que lea la siguiente celda. El ciclo continúa hasta que el valor de la lectura del robot sea igual a uno, en cuyo caso el programa TuringPC cambia el valor de la variable *fin* a un valor distinto de cero después de

recibir el valor de la lectura, al reiniciar el ciclo, el valor cambiado hace que el ciclo *do-while* finalice.

En la siguiente parte del protocolo el valor de la variable *fin* se reinicializa con el valor de uno, después se carga el primer valor de la tabla, que indica el estado inicial. Una vez hecho esto, se entra en un ciclo *while* que finalizará cuando *fin* sea igual a cero. La computadora enviará al robot la posición que deberá leer, posición que con anterioridad el robot habría enviado al encontrar la primera celda válida. El robot entonces leerá de nuevo el valor de la celda y lo enviará a la computadora, la cual realizará una búsqueda en el arreglo que contiene a la tabla de estados, fila por fila. El primer criterio de la búsqueda consiste en encontrar el valor del estado actual, almacenado en una variable *e,* en la primera columna de la tabla, que representan el valor de los estados actuales, el segundo criterio es el valor de la lectura realizada que será buscada en la segunda columna de la tabla, si ambos criterios coinciden, se elige a esa fila como la 5-tupla a usar en ese momento. Es entonces que se lee la tercera columna, que contiene al valor que se debe escribir, se envía ese valor al robot, el cual realiza la operación de escritura, entonces se lee la cuarta columna y esta indica hacia qué lado se deberá mover el robot, después se lee la última columna y el valor de esta indica el siguiente estado. El ciclo continuará hasta que el valor del estado siguiente sea cero, condición con la cual el valor de la variable *fin* se cambia a cero y el ciclo *while* finaliza.

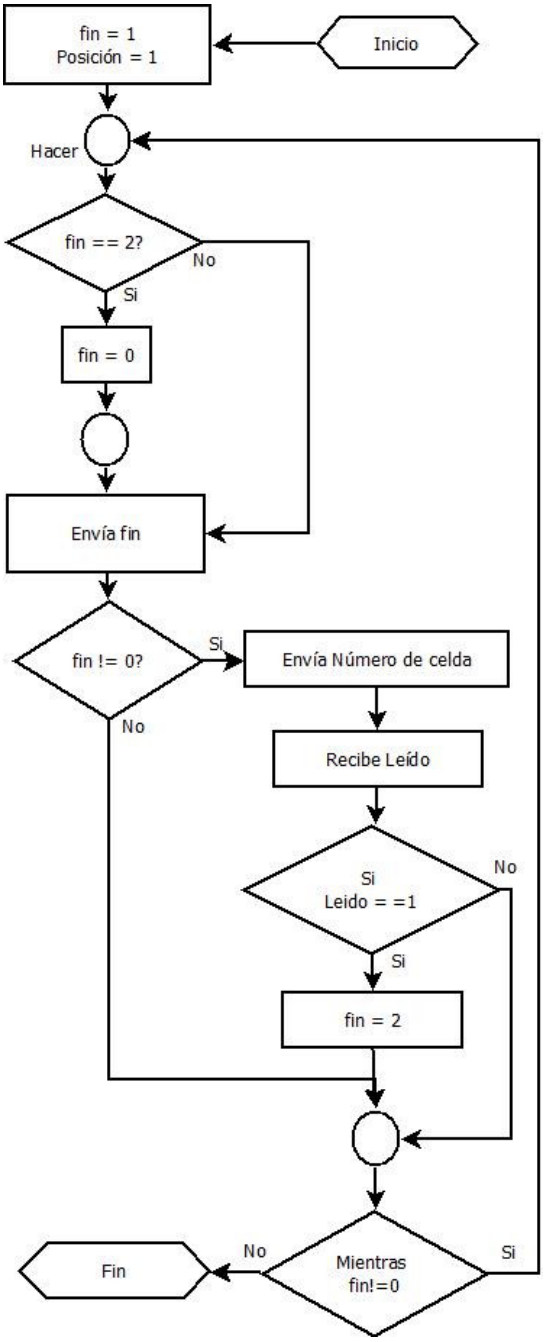


Diagrama 4.2 Protocolo 1

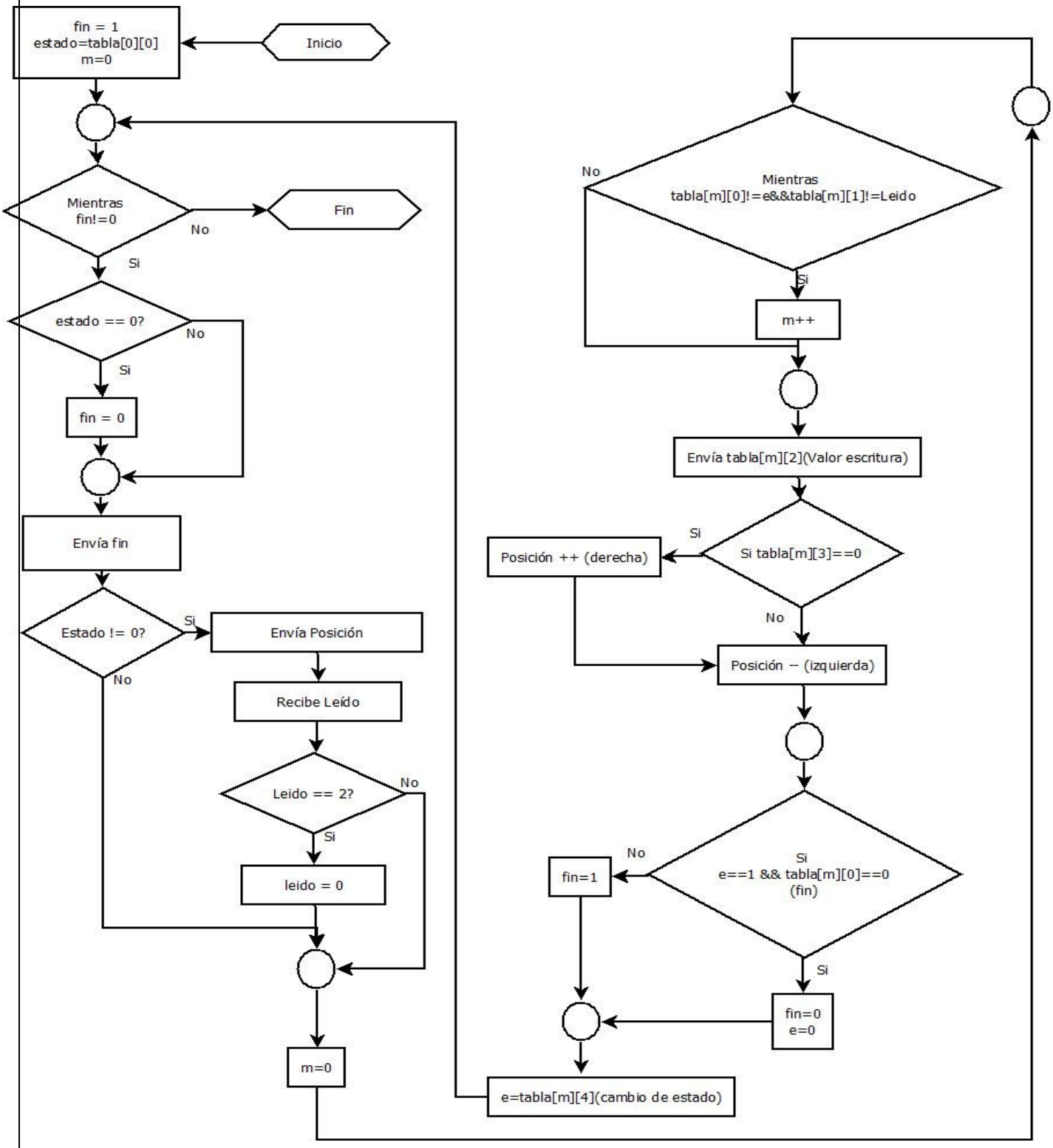


Diagrama 4.3 Protocolo 2

5 Componentes de software

Para programar tanto el robot como la computadora personal, se usó el lenguaje JAVA y una API llamada LeJOS, que contiene todas las funciones necesarias para programar los movimientos de motores, lectura de sensores, el uso de la pantalla LCD y los sonidos que puede reproducir el robot, así como las comunicaciones entre el robot y una computadora personal o dos robots, vía Bluetooth o USB.

5.1 Software del robot

La clase que hace funcionar al robot se llama RobotTuring, y contiene los siguientes métodos:

- escribeCeros
- escribeUnos
- todosNulos
- uno
- cero
- nulo
- escribe
- lee
- mov

5.1.1 Función principal

En la clase principal es donde se realizan las operaciones de comunicación entre el robot y la computadora personal:

```
public class RobotTuring{

    int c=0,avanza=61,centro=220,derecha=90,izquierda=340;

    public static int digitos=18;

    static int vel_a=500,vel_b=100;
```

int posicion;

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

int x=0,y,z,val_leido=2,fin=0;

LCD.clear();

Constructor:

RobotTuring robot = new RobotTuring();

Inicialización de las velocidades:

Velocidad del motor A

Motor.A.setSpeed(vel_a);

Velocidad del motor B

Motor.B.setSpeed(vel_b);

Inicialización de Strings e información en la pantalla:

LCD.clear();

String connected = "Conectado";

String waiting = "Esperando...";

String closing = "Cerrando conexion";

LCD.drawString("Turing", 0, 0);

LCD.drawString(waiting,0,1);

LCD.refresh();

Inicialización de la conexión bluetooth, al tiempo que se espera que la computadora realice la conexión:

BTConnection btc = Bluetooth.waitForConnection();

LCD.clear();

LCD.drawString(connected,0,1);

LCD.refresh();

Inicialización de los JavaStreams para envío y recepción de datos:

DataInputStream dis = btc.openDataInputStream();

DataOutputStream dos = btc.openDataOutputStream();

La siguiente parte del código hace al robot avanzar de celda en celda, buscando la primera celda que contenga un valor válido para comenzar las operaciones. La primera operación es recibir el valor de la variable *fin* que indica si se deben realizar las operaciones, es decir, si se ha se ha encontrado o no la primera celda; en el primer ciclo, el valor de *fin* es distinto de cero:

do{

Recibe el valor de la variable *fin*:

fin=dis.readInt();

Inicia el ciclo *if* cuando el valor de la variable fin es distinto a cero:

if(fin!=0)

{

En esta parte del el ciclo se espera la transmisión de las operaciones y datos.

Recibe el número de celda a leer:

```
x = dis.readInt();
```

Devuelve el valor de la lectura a la variable *val_leido*:

```
val_leido=robot.lee(x);

LCD.drawString("Leido:", 0, 1);

LCD.drawInt(val_leido, 0, 10);

y = dis.readInt();

LCD.drawString("y:", 1, 1);

z = dis.readInt();

LCD.drawString("z:", 2, 1);

LCD.drawInt(z, 2, 10);
```

Si *y* es distinto de cero, indica que es una instrucción de escritura, por lo que se ejecuta el ciclo en el que se escribe sobre la cinta en la posición:

```
if(y!=0)

  robot.escribe(y,z);

dos.writeInt(val_leido);

dos.flush();

        }

}while(fin!=0);
```

El siguiente ciclo implementa un protocolo de transmisión y recepción de datos, además de realizar los movimientos de la Máquina de Turing:

```
do{
```

Reutiliza la variable fin, asignándole el valor de uno durante la primera ejecución

```
fin=1;
```

Recibe de la computadora el valor de la variable fin:

fin=dis.readInt();

if(fin!=0)

{

Recibe la posición a leer y la asigna a x

x = dis.readInt();

Lee la posición indicada por x y asigna el valor a la variable *val_leído*

val_leido=robot.lee(x);

Imprime en pantalla el valor leído:

LCD.drawInt(val_leido, 4, 4);

Envía el valor de la lectura a la computadora:

dos.writeInt(val_leido);
dos.flush();

Recibe la posición a escribir:

y = dis.readInt();

Recibe el valor a escribir:

z = dis.readInt();

Recibe el valor de la variable *fin*:

fin=dis.readInt();

Si *y* vale cero, determina que no es una instrucción de escribir

if(y!=0)

Escribe en la posición indicada por la variable *y,* el valor de la variable *z:*

robot.escribe(y,z);

}

Este ciclo se realiza mientras el valor de fin sea distinto a cero:

}while(fin!=0);

Se imprime en la pantalla del robot que se han finalizado las operaciones:

LCD.refresh();

LCD.drawString("Terminado", 0, 4);

Regresa el motor B a la posición inicial:

Motor.B.rotateTo(0);

Regresa el motor A a la posición inicial:

Motor.A.rotateTo(0);

Cierra la conexión de entrada

dis.close();

Cierra la conexión de salida:

dos.close();

Se limpia la pantalla

Thread.sleep(1000);

LCD.clear();

LCD.drawString(closing,0,0);

LCD.refresh();

Se cierra la conexión bluetooth:

btc.close();

}

5.1.2 Métodos de la clase

escribeCeros: Escribe en todos los bloques el valor de cero. Utiliza un ciclo while que llama al método mov para realizar los movimientos desde el bloque cero o inicial hasta el bloque final, determinado por la variable digitos:

```
public void escribeCeros()
{
        c=0;
         while(c<digitos){
```

Llamada al método mov:

```
            mov(centro,avanza*(c+1),derecha,derecha);
            c++;
           }
       c=0;
      Motor.B.rotateTo(0);
      Motor.A.rotateTo(0);
}
```

escribeUnos: Escribe en todos los bloques el valor de uno. Utiliza un ciclo while que llama al método mov para realizar los movimientos desde el bloque cero o inicial hasta el bloque final, determinado por la variable digitos

```
public void escribeUnos()
        {
                c=0;
                while(c<digitos){
```

Llamada al método mov:

```
                mov(centro,avanza*(c+1),izquierda,izquierda);
                c++;
          }
           c=0;
          Motor.B.rotateTo(0);
```

todosNulos: Escribe en todos los bloques el valor nulo, útil para inicializar toda la cinta, de manera que se puedan escribir los valores para realizar operaciones. Utiliza un ciclo while que llama al método mov para realizar los movimientos desde el bloque cero o inicial hasta el bloque final, determinado por la variable digitos:

```
public void todosNulos()
```

```
        {
        c=0;
        while(c<digitos){
```

Llamada al método mov:

```
                                    mov(centro,avanza*(c+1),izquierda-
30,derecha+27);
                    c++;
                    }
          Motor.B.rotateTo(0);
         }
```

uno: escribe el valor de uno en el bloque con el número dado como parámetro al método. Utiliza al método mov para realizar la operación:

```
        public void uno(int p)
        {
```

Llamada al método mov:

```
              mov(centro,p*avanza,izquierda,izquierda);
          }
```

cero: escribe el valor de cero en el bloque con el número dado como parámetro al método. Utiliza al método mov para realizar la operación:

```
       public void cero(int p)

        {
```

Llamada al método mov:

```
              mov(centro,p*avanza,derecha,derecha);
          }
```

nulo: escribe el valor nulo en el bloque con el número dado como parámetro al método. Utiliza al método mov para realizar la operación:

```
     public void nulo(int p)
        {
```

Llamada al método mov:

```
              mov(centro,p*avanza,izquierda-30,derecha+27);
          }
```

escribe: ejecuta una operación de escritura, utiliza dos parámetros, el primero es la posición a la cual se va realizar la escritura y el segundo es el valor a escribir:

```
public int escribe(int p,int valor)
   {
```

Si la variable valor es igual a cero, escribe un cero en la posición *p*:

```
        if(valor==0)
                cero(p);
```

Si la variable valor es igual a uno, escribe un uno en la posición *p:*

```
        if(valor==1)
                uno(p);
```

Si la variable valor es igual a dos, escribe un nulo en la posición *p:*

```
        if(valor==2)
                nulo(p);
```

Escribe en pantalla la operación:

```
        LCD.drawString("Escribiendo", 5, 0);
    }
```

mov: puede usarse para realizar varias operaciones, como escribir y leer. Contiene cuatro parámetros, entregados por los métodos que ocupan al método mov, estos son: int cent, int avance, int empuja e int jala, donde:

- cent: determina el valor de la posición central del brazo, de forma que permite al robot desplazarse por encima de los bloques sin tocarlos
- avance: indica al robot a qué posición debe moverse
- empuja: indica al brazo hasta qué posición deberá moverse hacia la izquierda a partir de la posición central cent
- jala: indica al brazo hasta qué posición deberá moverse hacia la derecha a partir de la posición central

Este método inicializa a los dos sensores ópticos para que puedan realizar lecturas, cuando los valores de empuja y jala son idénticos, el método reconoce que se le está solicitando una lectura, por lo que procede a colocar los sensores justo encima del bloque a leer y ejecuta la operación de lectura. El valor resultante es impreso en la pantalla, además de que una alarma sonora identifica el resultado de la lectura:

```
public int mov(int cent, int avance, int empuja, int jala)
        {
```

Inicializa los sensores de luz:

```
LightSensor luz1 = new LightSensor(SensorPort.S1);
LightSensor luz2 = new LightSensor(SensorPort.S2);
int sensor1 = 0,sensor2 = 0,cont=0,valor=5;
```

Mueve el motor A a la posición central, definida en la variable cent:

```
Motor.A.rotateTo(cent);
```

Mueve el motor B y por lo tanto, al robot a la posición de la celda donde se va a realizar la operación:

```
Motor.B.rotateTo(avance);
```

Si el valor de empuja es igual al de jala y ambos son igual al valor de la posición central, significa que no es una operación de escritura, sino, de lectura:

```
if(empuja==centro&&jala==centro){
LCD.clear();
cont++;
```

Indica a los sensores realizar operación de lectura

```
  sensor1=SensorPort.S1.readRawValue();
  sensor2=SensorPort.S2.readRawValue();
```

En base a los valores de las lecturas, realiza el cálculo de la posición de la celda en base a la diferencia de lecturas de ambos sensores, si la diferencia es menor a 34, significa que el valor leído es cero:

```
  if((sensor1-sensor2)>34){
  LCD.drawString("Cero", 11, 6);
  LCD.drawString("Leido:", 0, 6);
Sound.playTone(1000, 100);
valor=0;
}
```

Si el valor de la diferencia es menor a -34, el valor leído es uno:

```
                else if((sensor1-sensor2)<-34){
                        LCD.drawString("Uno", 11, 6);
                        LCD.drawString("Leido:", 0, 6);
                    Sound.playTone(1000, 500);
                    valor=1;
            }
```

Si el valor de la diferencia es mayor a -70, el valor leído es nulo:

```
            else if((sensor1-sensor2)>-70||(sensor1-sensor2)<70){

                        LCD.drawString("Nulo", 11, 6);
                        LCD.drawString("Leido:", 0, 6);
                        Sound.playTone(500, 500);
                        valor=2;
            }
              }
```

Mueve la cabeza de escritura a la posición que indica la variable *empuja* y posteriormente a la de jala, para luego ir la posición central, en caso de que las tres variables sean idénticas, no se realizará escritura alguna:

```
            Motor.A.rotateTo(empuja);
            Motor.A.rotateTo(jala);
            Motor.A.rotateTo(cent);
```

Regresa como valor de retorno a la variable *valor*:

```
            return valor;
        }
    }
```

5.2 Software de la computadora personal

5.2.1 Función principal

El software programado en la computadora personal contiene una función principal:

```
public static void main(String[] args) {
```

En esta parte se crea un conector NXT, que se encarga de buscar a los robots NXT disponibles:

```
NXTConnector conn = new NXTConnector();
boolean connected = conn.connectTo("btspp://");
```

Se declaran algunas variables:

```
int i=1,j=0, e=1,leido,p=1,fin=1,l=0,m=0;
```

Se crea un *stream* y un buffer para el archivo:

```
InputStreamReader isr = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader (isr);
```

```
DataInputStream datais = null;
int valor=0;
String record = null;
int x = 0;
```

Se crea un arreglo bidimensional de enteros que contendrá a la tabla de estados:

```
int[][] tabla=new int[20][5];
```

Se crea un nuevo objeto tipo archivo para posteriormente cargar en él al archivo que se elegirá más adelante:

```
File fichero = new File(".");
```

Se crea un filtro para poder usar únicamente archivos con extensión *.tur,* creando un objeto de la clase filtro:

```
FilenameFilter filter = new FilenameFilter() {
   public boolean accept(File dir, String name) {
```

```
        return name.endsWith(".tur");
    }
};
```

Se solicita al usuario que elija un archivo de los que cumplen con el criterio de la extensión *.tur,* los nombres de los archivos se guardan en la variable *lista*:

```
System.out.println("Elija un archivo");
String[] listaArchivos=fichero.list(filter);
for(i=0; i<listaArchivos.length; i++){
        System.out.print(i+1 +".-" );
        System.out.print( listaArchivos[i]);
        System.out.println("");
}
```

Se captura la elección del usuario y se almacena en la variable *valor*:

```
try{
String texto = br.readLine();
valor = Integer.parseInt(texto);
valor--;
System.out.println(listaArchivos[valor]);
}
catch (Exception exc)
{
   System.out.println(exc);
   exc.printStackTrace();
}
```

Se crea una variable tipo String con el valor del nombre del archivo elegido:

```
String archivo=new String(listaArchivos[valor]);
try {
        String[] estado=new String[20];
```

Se abre el archivo elegido:

```
   File f = new File(archivo);
   FileInputStream fis = new FileInputStream(f);
   BufferedInputStream bis = new BufferedInputStream(fis);
   datais = new DataInputStream(bis);
```

Se lee el archivo línea por línea, cada una se almacena en un arreglo llamado *estado*:

```
   while ( (estado[j]=datais.readLine()) != null ) {
            j++;
     System.out.println(j + ": " + estado[j-1]);
```

```
        }
```

Se toman tokens del arreglo *estado*, separándolos por comas y se convierten de String a enteros, se van copiando a un arreglo bidimensional de manera que van conformando a la tabla de estados:

```
        for(x=0;x<j;x++)
        {
        record=estado[x];
        StringTokenizer st=new StringTokenizer(record,",");

        tabla[x][0] = Integer.parseInt(st.nextToken());
        tabla[x][1]= Integer.parseInt(st.nextToken());
        tabla[x][2]  = Integer.parseInt(st.nextToken());
        tabla[x][3]= Integer.parseInt(st.nextToken());
        tabla[x][4] = Integer.parseInt(st.nextToken());
```

Se imprimen en pantalla los valores de cada lectura:

```
        System.out.println("Linea:          "+(x+1));
        System.out.println("Estado Actual:   " + tabla[x][0]);
        System.out.println("Valor leido:     " + tabla[x][1]);
        System.out.println("Valor a escribir:" + tabla[x][2]);
        System.out.println("Movimiento:      " + tabla[x][3]);
        System.out.println("Siguiente estado:" + tabla[x][4]);
        System.out.println("");
        }
      } catch (IOException er) {
        System.out.println("error IOException" + er.getMessage());
      } finally {

        if (datais != null) {
          try {
                datais.close();
          } catch (IOException ioe) {
          }
        }
      }
```

Se intenta la conexión, si falla, se produce un error:

```
      if (!connected) {
                System.err.println("No se pudo conectar con el NXT");
                System.exit(1);
      }
```

Si la conexión es exitosa, se crean dos *streams*, uno para entrada y otro para salida de datos y operaciones:

```
DataOutputStream dos = conn.getDataOut();
DataInputStream dis = conn.getDataIn();
```

Se inicia un ciclo en el cual se solicitará al robot que busque en la cinta el primer valor válido para iniciar operaciones:

```
try {
   do{
```

Si en algún ciclo la variable fin toma el valor de dos, al llegar a este punto del ciclo, se le asigna el valor de cero

```
if(fin==2)
        fin=0;
```

Se envía el valor de la variable *fin* al robot:

```
dos.writeInt(fin);
dos.flush();
```

Si la variable *fin* es distinta de cero, se ejecuta el ciclo, donde se envía el valor de la posición de lectura, definida por la variable *p* e inicialmente vale uno:

```
if(fin!=0){
dos.writeInt(p);
System.out.println("Pos Lectura: " + (p));
dos.flush();
dos.writeInt(0);
dos.flush();
dos.writeInt(0);
dos.flush();
```

Recibe el valor leído por el robot, si vale uno, se determina que es el valor válido para iniciar operaciones, así que se asigna el valor de 2 a la variable *fin* y la siguiente vez que se inicie el ciclo, esa variable cambiará a cero, por lo que se saldrá del ciclo:

```
leido=dis.readInt();
System.out.println("Leido:" + (leido));
    if(leido==1){
        fin=2;
   }else p++;
     }
}while(fin!=0);
```

Se asigna el valor de uno a la variable *fin*, ya que se volverá a usar:

```
fin=1;
```

Se copia el primer valor de la tabla a la variable *e*, que almacena el estado actual:

```
e=tabla[0][0];
```

En tanto la variable fin sea distinta de cero, el ciclo se ejecuta:

```
while(fin!=0){
```

El estado inicial vale uno, pero si en algún ciclo el estado cambia a cero, se cambia el valor de la variable fin para salir del ciclo:

```
if(e==0)
        fin=0;
```

Se envía el valor de fin, para este punto , si fin vale cero, tanto el robot como la computadora saldrán de sus respectivos ciclos:

```
dos.writeInt(fin);dos.flush();
System.out.println("Valor de fin:" + fin);
```

Si el estado actual es distinto de cero, se envía el valor de la posición a leer, almacenada en la variable *p* y que en el primer ciclo es igual al valor que se había obtenido en la parte donde se determina la primera celda válida:

```
if(e!=0){

System.out.println("------------------------------------------------------------");
dos.writeInt(p);dos.flush();
System.out.println("Posición a leer:" +p );
```

Se recibe el valor de la lectura:

```
leido=dis.readInt();
if(leido==2)
        leido=0;
System.out.println("Valor leído:" + leido);
m=0;
l=0;
```

Se busca en cada fila los valores de la primera columna, que son los que contienen a los distintos estados posibles, con la variable *e,* que contiene al

estado actual, y que también coincida el valor de la segunda columna, que contiene a los posibles valores leídos, con el valor de la variable *leido,* que contiene el resultado de la última lectura hecha por el robot, el número de la fila se va guardando en la variable *m*:

```
while(!((tabla[m][0]==e)&&(tabla[m][1]==leido)))
{
        m++;
        l=m;
}
```

Una vez determinada la línea que contiene el valor a escribir, se envía el número de la celda que se va a escribir y posteriormente se envía el valor de dicha escritura al robot, valor determinado por la tercera columna de la tabla de estados:

```
        System.out.println("Linea:"+ l);
        System.out.println("Estado:"+ e);
        dos.writeInt(p);dos.flush();
        dos.writeInt(tabla[l][2]);dos.flush();
        System.out.println("Valor a escribir:"+tabla[l][2]);
```

Se busca entonces en la cuarta columna el valor del siguiente movimiento, si ese valor es igual a cero, significa que el siguiente movimiento de la cabeza es hacia la derecha, si vale uno, el siguiente movimiento es hacia la izquierda

```
        if(tabla[l][3]==0){
                p++;

                System.out.println("Movimiento a la derecha->");
                        }
        if(tabla[l][3]==1){
                p--;

                System.out.println("Movimiento a la izquierda<-");
                        }
        if(e==1&&tabla[l][0]==0){
                fin=0;
                e=0;
                }
else fin=1;
```

Se cambia el valor del estado con el valor de la última columna, que contiene el siguiente estado:

```
e=tabla[l][4];
System.out.println("Siguiente estado:"+ e);
                        }
```

```
                    }
                }
        catch (IOException ioe) {
                System.out.println("IO Exception writing bytes:");
                System.out.println(ioe.getMessage());
                    }
```

Se cierra entonces la conexión:

```
        try {
                dis.close();
                dos.close();
                conn.close();
        } catch (IOException ioe) {
                System.out.println("IOException closing connection:");
                System.out.println(ioe.getMessage());
                            }
        }
```

5.2.2 Clase filtro

La clase filtro se utiliza para poder seleccionar en la clase principal únicamente a los archivos que cumplan con el filtro determinado:

```
        public class Filtro implements FilenameFilter{
          String extension;
          Filtro(String extension){
            this.extension=extension;
          }
          public boolean accept(File dir, String name){
            return name.endsWith(extension);
          }
        }
}
```

5.3 Problemas encontrados

En todo proyecto es común encontrar problemas, este proyecto no fue la excepción.

5.3.1 Problemas con los sensores de luz

Los sensores de luz detectan la luminosidad del reflejo de los objetos a los cuales están apuntando, a mayor reflejo, mayor el valor numérico que devuelven los sensores. Los sensores se posicionan sobre los bloques que van a leer, una vez que ejecutan su lectura, lo que se hace es determinar hacia qué lado está inclinado el bloque. El problema radica en que el bloque es de color gris, igual que los bloques que lo rodean, así que la diferencia entre el valor que retorna el sensor ubicado

sobre el bloque y el valor del otro sensor es muy baja. La solución fue pintar la punta del bloque de tal forma que cuando esté inclinado, el sensor que está ubicado del lado inclinado del bloque obtiene una lectura mayor que el otro bloque

5.3.2 Problemas con el riel y la cinta

Si bien el riel y la cinta son elementos grandes, están hechos de materiales plásticos que les proporcionan ligereza. La superficie donde se construyó el robot resultó tener poca fricción, por lo que el riel y la cinta podrían ser movidas de su posición por la fuerza del robot. La solución fue utilizar cinta adhesiva para evitar que el robot moviera a estos elementos.

5.3.3 Problemas de tracción

Una delas características de un motor eléctrico es su torque elevado. Esta característica puede ser problemática especialmente en superficies con poca fricción y cuando mueven objetos con peso relativamente bajo.

La función del riel en el proyecto es la de mantener al robot en una trayectoria, pero las ruedas están en contacto con la superficie, que como se mencionó anteriormente, tiene poca fricción. Se trató de solucionar el problema reduciendo la velocidad del motor B, que es el que desplaza al conjunto, sin embargo, esto no fue suficiente y seguía existiendo derrape en las ruedas, muy a pesar de estar hechas de hule. Además de la reducción de la velocidad, se tuvo que colocar un contrapeso en la parte delantera del robot, que es donde están ubicadas las ruedas que proporcionan tracción, con lo que se mejoró el agarre.

6 Conclusiones

La implementación de la Máquina de Turing con robot LEGO permite entender el funcionamiento de un robot y su programación. Si bien los sensores que se utilizaron son relativamente poco complejos, son más que suficientes para los propósitos del proyecto.

La API JAVA LeJOS que se utilizó para programar al robot es poderosa y permite explotar muy bien las capacidades del robot mucho mejor que la herramienta CASE MindStorms, muy sencilla y básica, orientada básicamente para niños. Los resultados de las lecturas de los sensores son enviadas como números enteros, los motores tienen integrados tacómetros y los métodos que manipulan esos motores saben cuántos grados han recorrido, lo que facilita el retorno a la posición original. Además, la velocidad de los motores se puede variar, de hecho se configuró una velocidad baja para obtener mejor precisión al momento de mover el robot a lo largo de los rieles.

El manejar comunicaciones entre el robot y la computadora permitió reforzar la comprensión de lo aprendido en las UEAs de redes: transmisión, recepción, sincronización, inicios de conexión, finalización de conexión, entre otros conceptos.

La utilización de la tecnología inalámbrica bluetooth y de JAVA streams simplifica el uso de las conexiones y permite aprovechar las ventajas de las tecnologías inalámbricas, así como entender algunas de sus desventajas.

La Máquina de Turing con robot LEGO muestra claramente y de manera física el funcionamiento de esta herramienta, que si bien es un modelo teórico, se continúa su enseñanza para que los profesionistas del área de computación entiendan algunas de las bases más importantes que llevaron al desarrollo de la computación, así, la Máquina de Turing construida resulta un modelo didáctico útil para tal propósito. En la manera como se realizó la implementación, se puede ver en la pantalla las operaciones y los cambios de estado que se están realizando, al mismo tiempo que se pueden ver las operaciones de escritura, lectura y movimientos a lo largo de la cinta, también, se pueden escuchar los resultados de las lecturas del robot por medio de sonidos claramente diferenciables.

Existen dos formas de implementar soluciones con el LEGO Mindstorms, la elegida para el proyecto claramente fue la solución distribuida. Por lo tanto, el proyecto no solo permitió aplicar conocimientos aprendidos durante la carrera, sino también adquirir nuevos conocimientos y ampliar algunos ya existentes. No cabe duda que la robótica representará en un futuro tal vez no muy lejano una herramienta que permitirá mejorar la calidad de vida de las personas, disminuir costos de producción de muchos productos, pero además, será una herramienta de aprendizaje para distintos niveles educativos.

Por último, se alcanzaron los siguientes objetivos:

- Se consiguió programar un robot LEGO Mindstroms NXT, con firmware modificado para funcionar como una máquina virtual JAVA, de tal forma que el robot fuera capaz de ejecutar los movimientos propios de una Máquina de Turing.
- Se construyó la estructura necesaria para que dicho robot pudiera funcionar, estructura que en conjunto implicó el uso de sensores y motores, además de elementos mecánicos que permiten a la máquina realizar movimientos que emulan a una Máquina de Turing en funcionamiento, además de una estructura que representa a una cinta de datos donde se almacenan los valores sobre los cuales se realizarán las operaciones de lectura y escritura, datos representados por celdas individuales que a su vez representan dos valores según su posición.
- Se logró que una cabeza de lectura fuera capaz de determinar la posición de cada celda, basándose en el lado hacia el cual se encuentran inclinadas las piezas que representan a celdas.
- Se programó al robot tal que es capaz de realizar los movimientos propios de una Máquina de Turing.
- Se implementó una interfaz de control que permite a un usuario elegir la Máquina de Turing que desea ejecutar mediante un menú, que a su vez es tomada de archivos existentes en el directorio de trabajo. Esta interfaz es capaz de interpretar tablas de transición que contengan quíntuplas que describan los movimientos de una Máquina de Turing.
- Se implementó un protocolo de transmisión y recepción de datos que permite a la computadora personal y al robot enviar y recibir operaciones y datos.

7 Bibliografía

[B1] John E. Hopcroft, Raveev Motwani, Jeffrey D. Ullman *Automata Theory, Languages, and Computation*, 3rd Edition, Pearson Education, 2007, pp. 324-330

[B2] Henry Hamburger, Dana Richards, *Logic and language models for computer science*, 2nd Edition, Prentice Hall 2002, pp. 313-317

[B3] The LEGO group, "LEGO.com About us", 2010. [En linea]. Disponible en: http://aboutus.lego.com/en-us/default.aspx [Consultada: 8/Abril/2011]

[B4] The LEGO group, "LEGO Mindstorms", 2010. [En linea]. Disponible en: http://mindstorms.lego.com/ [Consultada: 8/Abril/2011]

[B5]P. Andrews, J. Stuber , L. Griffiths, B. Bagnall, M.P. Scholz, "Overview (leJOS NXJ API documentation)", [En linea]. Disponible en: http://lejos.sourceforge.net/nxt/nxj/api/index.html [Consultada: 8/Abril/2011]

[B6]Juan Antonio Breña Moral, "*Develop LeJOS Programs Step by Step",* 2009,[En linea].Disponible en:[Consultada: 8/Abril/2011] *http://www.juanantonio.info/lejos-ebook/*

[B7] Philippe E. Hurbain, "LEGO 9V Technic Motors compared characteristics",[En linea]. Disponible en http://www.philohome.com/motors/motorcomp.htm [Consultada: 8/Abril/2011]

[B8]S. Geggie, M. Vester, A. Nissen, M. Have, "LEGO of doom", enero 2009, http://legoofdoom.blogspot.com/ [Consultada: 9/Abril/2011]

[B9] Denis Cousineau, "Turing Machine", 2000. [En linea]. Disponible en: http://www.mapageweb.umontreal.ca/cousined/lego/5-machines/turing/turing.html [Consultada: 9/Abril/2011]

[B10] Denis Cousineau, "Denis Cousineau home page", 17/Diciembre/2009. [En linea]. Disponible en: http://www.mapageweb.umontreal.ca/cousined/home/ [Consultada: 9/Abril/2011]