Capítulo II. Marco Teórico

2.1. leJOS

El LEGO Java Operating System (leJOS) fue pensado como una pequeña máquina virtual de Java (TinyVM –Tiny Virtual Machine) y fue desarrollada inicialmente por José H. Solórzano en 1999. Se inició como un pasatiempo que culminó en un proyecto de código libre conocido actualmente como leJOS. Varios desarrolladores se unieron al grupo, entre ellos Brian Bagnall, Jürgen Stuber y Paul Andrews, quienes se encargaron de las investigaciones una vez que Solórzano se retiró del proyecto [WEB03].

TinyVM es un *firmware* basado en Java para el microcontrolador del RCX de LEGO MINDSTORMS. Es un código libre y gratuito escrito en C que permite reemplazar el *firmware* de legOS que trae por defecto. La TinyVM ocupa cerca de 10 Kb dentro del RCX, y cada clase de Java es compactada considerablemente antes de ser descargada al RCX.

Características de la TinyVM:

- Lenguaje Orientado a Objetos (Java).
- *Threads.*
- Excepciones.
- Sincronización.
- Arreglos, incluidos los multidimensionales.
- Recursividad.
- Acceso a los botones, altavoz, pantalla de cristal líquido (LCD) y puertos de entrada y salida del RCX.

- Números aleatorios y funciones matemáticas.
- Contadores de tiempo.

Limitaciones:

- No tiene recolector de basura (*garbage collector*).
- No acepta instrucciones *switch* (fácilmente sustituido con un conjunto de *if*).

2.2. RCX

Dentro del RCX se encuentra instalado un microcontrolador Hitachi H8/3292, con un CPU llamado H8/300 CPU Core. Este CPU es el que contiene el programa que va a controlar todo el robot. A través del H8/3292 se tiene acceso a cada uno de los dispositivos de entrada y salida del RCX [WEB04].

El RCX viene equipado con los siguientes dispositivos de entrada y salida:

- Entrada
    - 4 Botones
        - Run.
        - On/Off.
        - View.
        - Prgm.
    - 3 Puertos de entrada
        - Etiquetados como 1, 2 y 3.
    - Nivel de Voltaje de la Batería
    - Contadores de Tiempo
    - Receptor IR

- Salida
    - Pantalla de Cristal Líquido (LCD)
    - Altavoz
    - 3 Puertos de salida
        - Etiquetados como A, B y C.
    - Emisor IR

El RCX interactúa con el ambiente a través de sensores y actuadores conectados a estos puertos de entrada y salida. Los siguientes sensores y actuadores pueden ser conectados a los puertos del RCX:

- Sensores
    - Tacto
    - Luz
    - Rotación*
    - Temperatura*
- Actuadores
    - Motor
    - Lámpara*
    - Sonido

* No incluido en el kit de LEGO MINDSTORMS.

2.3. COMUNICACIÓN

Cuando el botón Run del RCX es presionado, se activa su funcionalidad de interpretar instrucciones. El CPU del robot va a leer y a ejecutar cada comando que se le

ordene. Aquí es donde el concepto de *opcode* comienza a utilizarse. Un *opcode* es una instrucción de bajo nivel que le dice al RCX qué hacer [GUILIO, 2002].

El intérprete de comandos del RCX ejecuta instrucciones provenientes de dos fuentes diferentes, de los programas cargados dentro del RCX y de la comunicación con otro dispositivo compatible. Los *opcode*s pueden ser enviados al RCX desde una computadora, un control remoto, e inclusive otro RCX, utilizando un protocolo serial y a través de un dispositivo emisor de infrarrojos.

Para ejecutar correctamente cada instrucción, el procesador necesita almacenar datos en una memoria estructurada llamada archivo del registro.

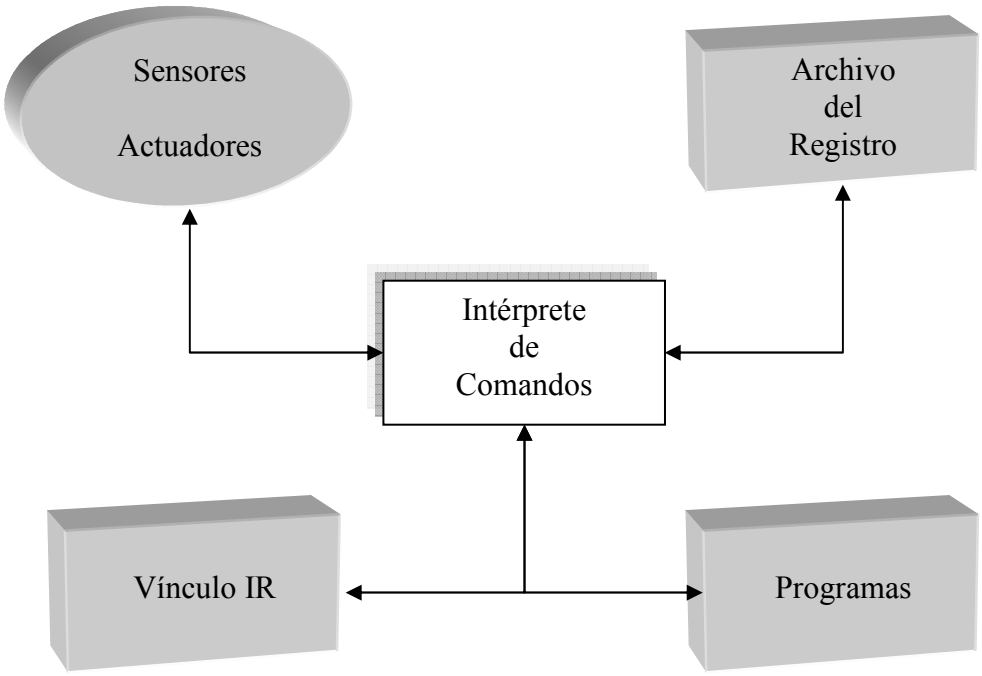

Figura 2.1 Diagrama de la comunicación del RCX [Elaboración Propia].

Kekoa Proudfoot fue uno de los pioneros en investigar cómo funciona el RCX desde su lanzamiento al mercado en 1998. Proporcionó una serie de documentación no oficial que ha permitido a más desarrolladores implementar nuevas aplicaciones hechas en lenguajes de programación diferentes al legOS.

Kekoa llevó a cabo un sistema de ingeniería inversa, donde básicamente descubrió el funcionamiento de lo que hasta entonces fue una caja negra para los usuarios de RCX. En el caso de la comunicación con el RCX, se interpuso en el camino de los mensajes entre la computadora y un dispositivo, en este caso la IR Tower.

2.4. INFRARROJOS

La transmisión de datos por medio de emisores y receptores infrarrojos permite una comunicación entre dispositivos sin la necesidad de un cable que los conecte directamente.

En el *kit* de LEGO MINDSTORMS esta comunicación se da en un modo maestro – esclavo, en donde el RCX es el esclavo y la IR Tower es el maestro. Este esquema cambia sólo cuando se usan mensajes para comunicar dos o más RCX entre ellos [WEB05].

La IR Tower está diseñada para comunicarse sólo con productos LEGO, y no con otros dispositivos infrarrojos, ya que, a diferencia de éstos, la IR Tower no sigue los estándares de la Asociación de Datos Infrarrojos (IrDA). Por esta misma causa tampoco se puede usar el puerto infrarrojo incluido en algunas computadoras para comunicarse con el RCX [WEB06].

2.4.1. PROTOCOLO DE ENVÍO DE BYTES

Cada envío de *bytes* es hecho a través de una comunicación serial entre la computadora y el RCX, cada uno con un emisor y receptor de infrarrojos, uno es la IR Tower conectada a la computadora vía un puerto de comunicación RS232, el otro está instalado dentro del RCX [WEB04].

El RS232, también conocido como EIA232, es un estándar que especifica el voltaje, sincronización y función de la señal, es un protocolo para intercambio de información que

nació con la finalidad de facilitar este intercambio entre dispositivos hechos por diferentes fabricantes [WEB07].

Tiene una velocidad de 2400 *bits* por segundo. Para diferenciar el inicio y final de cada *byte*, un *byte* es precedido por un *bit* de inicio y seguido por un *bit* de final. También se añade al *byte* un *bit* de paridad, por lo tanto cada *byte* enviado es de 11 *bits*. Además, cada *byte* en un paquete es enviado como dos *bytes*, el *byte* mismo y su *bit* complemento. El paquete resultante tiene igual cantidad de *bits* ceros y unos. Esto le permite al receptor de infrarrojos compensar una mala recepción causada por la luz del ambiente, simplemente sustrayendo el valor promedio de la señal. 8 *bits* de datos son enviados en un paquete de 22 *bits* totales. Con una velocidad de 2400 *bits* por segundo, le tomaría 9 milisegundos enviar cada *byte* [WEB04].

2.4.2. PROTOCOLO DE ENVÍO DE BITS

Cada *bit* transmitido entre los dispositivos es representado como un periodo de 417 microsegundos de un pulso de luz infrarroja con una frecuencia de 38 kHz para un valor de 0, y un periodo de 417 microsegundos sin luz infrarroja para un *bit* de valor 1. La duración del periodo para un *bit* está relacionado a la velocidad de transmisión: 1 *bit*/2400 *bit*/seg. = 417 microsegundos [WEB04].

2.4.3. PAQUETES

Cada paquete transmitido al RCX tiene el mismo formato:

Paquete = encabezado / Mensaje / Suma de Comprobación.

0x55 0xff 0x00 D1 ~D1 D2 ~D2 ... Dn ~Dn C ~C

Encabezado: 0x55 0xff 0x00

Mensaje: D1...Dn

Suma de Comprobación: C = D1 + D2 + ... Dn.

La suma de comprobación es obtenida como el *byte* menos significativo de la suma de los *bytes* del mensaje. El RCX ignora por completo los paquetes que traen una suma de comprobación inválida [WEB08].

Para enviar mensajes infrarrojos al RCX desde la IR Tower se utilizó el paquete josx.rcxcomm de leJOS. Su funcionamiento puede ser transparente para el programador, como una caja negra, pero en este caso sí conviene realizar observaciones a partes específicas para comprender un poco qué está haciendo.

Comenzando con la llamada a la clase Tower.java (ver Apéndice C para más detalles):

```
Tower tower = new Tower();
tower.open("USB");
```

En este caso estamos inicializando la IR Tower conectada a un puerto USB, aunque para versiones anteriores del kit de LEGO MINDSTORMS podría utilizarse COM1 como puerto serial.

```
tower.send(b, b.length);
```

Aquí se observa el envío de un arreglo de *bytes* b hacia el receptor de infrarrojos del RCX.

La codificación de la clase Tower.java está hecha con código nativo, esto se hizo importando código hecho en C a la aplicación de Java [WEB09].

En esta clase se hace un llamado a irtower.cpp, donde se utiliza el método osx_usb_rcx_send del archivo osx_usb.c Aquí es donde se realizan las operaciones para construir el mensaje y enviarlo.

```
msg[msglen++] = 0x55;
msg[msglen++] = 0xff;
```

```
msg[msglen++] = 0x00;
```

En esta parte del código encontramos lo que anteriormente definimos como el encabezado que llevará el mensaje.

```
while (buflen--) {
        msg[msglen++] = *bufp;
        msg[msglen++] = (~*bufp) & 0xff;
        sum += *bufp++;
}
```

Aquí se observa cómo se va construyendo el contenido del mensaje que vamos a enviar, primero agregando el *byte* necesario seguido de su complemento. Se envían 2 *bytes* de información, el primero es para identificar qué acción se quiere realizar, como avanzar hacia delante, y el segundo le pasa los parámetros necesarios, como lo sería la duración de la acción. Conforme se agrega el cuerpo del mensaje, se va calculando una suma de comprobación que le servirá al RCX para confirmar que un mensaje fue bien recibido.

```
msg[msglen++] = sum;
msg[msglen++] = ~sum;
```

Al final del mensaje se añade la suma de comprobación y su complemento, para tener el mensaje completo y listo para ser enviado.

Finalmente se hace un llamado a la función WritePipe que se encuentra en el archivo IOUSBLib.h. Esta biblioteca pertenece a Apple Computer, Inc. y es la que permite el acceso a los dispositivos conectados la computadora por puertos USB [WEB10].

Existen otras alternativas para enviar los mensajes a través de infrarrojos, una de ellas es usar otros lenguajes de programación como Visual Basic, C y pbForth. En Java se tiene la

opción de utilizar la Interfaz de Programación de Aplicaciones (API por sus siglas en inglés) Java Communications de Sun Microsystems, pero en este caso se optó por la conexión incluida en leJOS, porque se lograron los resultados deseados en las distintas pruebas de velocidad y alcance de la transmisión de datos, además de estandarizar la codificación al utilizar un solo API para la aplicación.

RCXPort es un API que ocupa Java Communications. Pueden enviarse todos los mensajes de manera correcta, pero las pruebas demostraron que no se obtenía un alcance superior a los 50 cm. Esto fue una clara desventaja para lograr el objetivo de esta investigación, ya que se requería colocar la IR Tower a una altura superior a los 2 metros.

2.5. JAVA MEDIA FRAMEWORK (JMF)

El proyecto requiere un medio de visualización de la cancha que esté activo todo el tiempo. Para lograrlo, utilizando tecnología Java, se utilizó el API de Java Media Framework de Sun Microsystems.

Este API permite añadir audio, video y otros tipos de medios en diversos formatos a las aplicaciones diseñadas en Java. Este paquete no viene incluido en el Java 2 Platform, Standard Edition, v 1.4.2 (J2SE) que se utiliza para programar aplicaciones en leJOS, por lo tanto se tiene que descargar por separado [WEB11].

2.6. TRABAJOS SIMILARES

Se encontró que un equipo de investigación realizó un proyecto similar al descrito en este documento, pero tiene varias diferencias que hicieron que no se consultara esta fuente más que para ser mostrada como trabajo previo.

Este trabajo tiene como objetivo lograr que dos robots semi-autónomos construidos con LEGO MINDSTORMS jueguen futbol. Los robots obtienen su posición por medio de mensajes infrarrojos de una computadora, que es calculada a través de un análisis de imágenes obtenidos desde una LEGO Cam.

La principal diferencia es el lenguaje de programación, ya que fue realizado con ROBOLAB [WEB12]. Esta herramienta está basada en iconos, el arrastrarlos y configurarlos permite construir aplicaciones para controlar el RCX, además de que por su sencillez está diseñada para ser usada por niños desde los 8 años de edad.

Otra característica que tiene este trabajo es el manejo de los mensajes infrarrojos. Decidieron usar un *broadcasting*, donde un mensaje de infrarrojo es recibido por un RCX, y este reenviara el mensaje a otro RCX [WEB13].

Los robots están construidos con piezas adicionales a las encontradas en un *kit* básico de LEGO MINDSTORMS, un claro ejemplo de esto es el sensor de rotación. Este les permite controlar con mayor precisión cada movimiento y así pueden hacer al robot más exacto. El sensor de luz es usado sólo para encontrar los límites de la cancha, y la pelota deberá ser encontrada por medio de la LEGO Cam.

Toda su investigación es libre y puede ser descargada gratuitamente de su sitio de Internet, el único inconveniente es que el *software* ROBOLAB no lo es y tiene que ser comprado por separado.