Tiros libres con Lego Mindstorms

Eduardo Díaz Díaz
Jorge Arturo García Moreno

Ingeniería en Tecnologías Estratégicas de Información

Mayo de 2003

Escuela de Ingeniería

Universidad Anáhuac de Xalapa

ÍNDICE

INTRODUCCIÓN

El presente proyecto aplica conocimientos de programación, inteligencia artificial y sistemas operativos en el desarrollo de un par de robots Lego. Estos están programados para simular un entorno en donde uno golpea una pelota en una dirección y el otro robot bloquea el tiro para evitar que la pelota entre en un área llamada portería.

La utilidad del proyecto reside en su valor como aplicación práctica en términos de hardware y software, además de presentar un interesante potencial de comercialización como juguete. Este tipo de usos es uno de los catalizadores más influyentes en el desarrollo de nuevas tecnologías, siendo un requisito este tipo de enfoques para que un producto llame la atención y tenga perspectivas de éxito en el mercado actual.

El fútbol es el deporte más popular del mundo, con lo cual un dúo de robots “inteligentes” que jueguen algún aspecto de dicha actividad llamaría la atención de una gran cantidad de personas. Son muchos los juguetes que hacen uso de la inteligencia artificial en la actualidad, pero en términos de deportes no parecen haber grandes desarrollos.

En el primer capitulo presentaremos el estado del arte en el área, estableciendo el marco teórico y los avances actuales que serán el punto de partida de la tesis. Englobara temas que van desde la robótica reactiva hasta los robots Lego y el lenguaje de programación NQC.

En el segundo capítulo entraremos a una etapa de análisis del problema y diseño de los robots, determinando con precisión sus capacidades y limitaciones, así como las funciones que serán capaces de desempeñar.

El cuarto capítulo será el punto fuerte de la tesis, presentaremos la construcción y la programación de los robots. Es en dicha parte donde veremos a los robots convertirse en realidad y evaluaremos los resultados obtenidos.

Para terminar, tendremos una conclusión en la cual presentaremos nuestros comentarios finales y que es lo que hemos obtenido de este proyecto como experiencia de aprendizaje y desarrollo profesional, además de proponer interesantes aplicaciones alternativas del trabajo realizado.

Capítulo 1

EL ESTADO DEL ARTE

1. Aplicaciones actuales

Los robots juguete tienen muchas manifestaciones destacadas en la actualidad en el mercado comercial y la investigación, veamos algunas:

- Sony AIBO

  AIBO es el nombre otorgado al robot ofrecido por Sony Entertainment Corporation [AIBO], un robot artificialmente inteligente que fue desarrollado para alentar la interacción entre seres humanos y robots.

  Es un robot autónomo con forma de perro que aprende, madura y actúa por sí mismo en respuesta a estímulos externos. Para coexistir por completo con las personas, AIBO posee cuatro piernas, una cabeza, una cola y 20 actuadores que le permiten sentarse o acostarse de una forma natural. También puede utilizar luces, su cola y el movimiento de las orejas para expresar emociones o entretener a personas de distintas formas. AIBO responde de diversas formas cuando es acariciado en la cabeza o espalda. También responde a una pelota especial y realiza distintas acciones cuando se aburre.

  Posee seis emociones: alegría, tristeza, enojo, sorpresa, miedo y descontento. También cuenta con cinco "instintos" o capacidades: dormir, responder a las caricias, embestir, explorar y jugar con alguien, haciendo uso de gestos expresivos para dar a entender lo que desea.

  Un AIBO es capaz de interactuar con otro.

- Furby

  Lanzado al mercado por Tiger Electronics, el Furby [FURB] es una mascota artificial cariñosa que cuenta con un vocabulario de palabras. Su lanzamiento inicial en 1998 incitó un interés increíble ya que el público en general se asombró de su capacidad de interacción. Cuando el usuario toca la panza del Furby, la criatura se ríe y dice "¡Me haces cosquillas!". Cuando el usuario toca la espalda, la criatura ronronea. Si el usuario coloca su dedo en la boca el Furby dice "Mmmm".

Si el usuario intenta mover al Furby como un avión de juguete, emite gritos de emoción.

Cuando una persona compra un Furby por primera vez, la criatura sólo puede hablar su lengua natal, llamada Furbish. Una vez que el dueño acaricia, alimenta y hace cosquillas a la criatura, ésta emite palabras programadas en el idioma del lugar donde fue vendido, simulando que aprende.

La característica más interesante del Furby es que puede interactuar con otros Furbys. Cuando se les coloca uno junto al otro inician largas conversaciones entre sí.

- Toy Robotics Initiative

  Proyecto del Mobile Robot Programming Laboratory del Robotics Institute de la Universidad de Carnegie Mellon [TOYa]. Busca comercializar tecnologías de robótica para educación, juguetes, entretenimiento y arte. Entre sus metas están: descubrir soluciones no tradicionales a problemas de robótica trabajando dentro de las limitaciones de costo del mercado masivo, utilizar fuentes comerciales para financiar investigación y desarrollo en robótica y explotar la fabricación a gran escala en el sector comercial.

  Uno de sus proyectos más destacados es *insect telepresence* [TOYb]. Supongamos que podemos hacer una cámara miniatura cuyo movimiento pueda ser controlado por una persona. Si ponemos ésta cámara en una colonia de insectos, la persona adquiere una perspectiva totalmente nueva de los insectos y su biología. Si agrandamos la imagen lo suficiente para disminuir el tamaño de la persona con respecto a la imagen, y también amplificamos los sonidos de los insectos, tenemos una experiencia de telepresencia interesante y educativa. Este proyecto está siendo actualmente comercializado para museos y próximamente podría ingresar al mercado doméstico.

  Otro proyecto interesante es el de *robot emotion* que parte de investigaciones en modelos para crear agentes robóticos creíbles, haciendo uso de un par de robots que improvisan comedia y actúan historias. Se está colaborando con un fabricante importante de juguetes para darle a los robots modelos emocionales que mejoren la interacción humano-robot.

- Ullanta Performance Robotics

  Compañía teatral en la cual los actores son robots autónomos [ULLA]. La compañía presenta obras tradicionales, piezas de arte interpretativo y bailes. Ha actuado recientemente en Europa, Estados Unidos y Japón.

  Se apoya en el uso de robots basados en comportamiento, con habilidad de parecer ser criaturas vivas e inteligentes que se adaptan a los elementos de sus entornos. Genera una robótica social que hace uso de nuevas tecnologías para crear experiencias innovadoras en teatro, presentando nuevos retos para directores y guionistas. Las presentaciones de los robots deben atraer al público y realizarse en tiempos programados, por lo que demandan un alto nivel de capacidad de

adaptación, robustez e interacción. Asignar papeles dramáticos a robots, como beneficio adicional, permite darles dimensiones más antropomorfas.

Ullanta también cuenta con un equipo de pequeños robots que juegan fútbol y han participado en el RoboCup World Championship.

- RoboCup World Championship

  La RoboCup [RCUP] es un proyecto internacional conjunto que busca promover la inteligencia artificial, la robótica y campos relacionados. Es un intento de alentar la investigación en estos campos proporcionando un problema estándar donde una amplia gama de tecnologías pueden ser integradas y examinadas. La gran meta del proyecto es tener para el año 2050 un equipo de robots humanoides completamente autónomos, capaces de ganarle al equipo de fútbol campeón del mundo.

  Para que un equipo de robots sea capaz de participar en un juego de fútbol se deben utilizar diversas tecnologías, entre las que se encuentran: principios de diseño de agentes autónomos, colaboración multiagente, adquisición de estrategias, razonamiento en tiempo real, robótica y fusión de sensores. La RoboCup es una tarea para un equipo de múltiples robots rápidos bajo un entorno dinámico.

2. La Robótica

2.1. Definición de un robot

El Robot Institute of America [OVRO] define a un robot como:
"un manipulador multifunción programable diseñado para mover material, partes, herramientas o dispositivos específicos a través de movimientos programados variables para la realización de una variedad de tareas."

Esta definición no es muy completa, por lo cual tal vez sea mejor que definamos a un robot como:

"*un agente artificial activo cuyo entorno es el mundo físico*"

La vasta mayoría de los robots tienen muchas características en común. En primer lugar, casi todos los robots tienen un cuerpo móvil. Algunos solo tienen llantas motorizadas, y otros decenas de segmentos móviles, típicamente construidos de metal o plástico. Como los huesos del cuerpo humano, los segmentos individuales están conectados mediante articulaciones.

Los robots hacen girar llantas o giran sobre un eje de articulaciones unidas con algún tipo de actuador. Algunos robots utilizan solenoides y motores eléctricos como actuadores, otros usan un sistema hidráulico o algún tipo de sistema neumático. Un robot puede usar todos estos tipos de actuadores.

Los actuadores están todos conectados a un circuito eléctrico. El circuito proporciona energía a motores eléctricos y solenoides directamente.

La computadora del robot (cuando la tiene) controla todo lo que está conectado al circuito. Para mover al robot, la computadora enciende todos los actuadores necesarios. La mayoría de los robots son reprogramables, requiriendo que sólo se escriba un nuevo programa para modificar el comportamiento del robot.

2.2. El campo de la robótica

La Robótica es una nueva tecnología que surgió como tal alrededor del año de 1960 [STNO], desde entonces han transcurrido pocos años y el interés que ha despertado es superior a cualquier previsión que se hubiera podido formular en su nacimiento, siguiendo un proceso paralelo a la introducción de las computadoras en las actividades cotidianas de la vida del hombre. Aunque si bien los robots todavía no han encontrado la vía de penetración en la vida cotidiana de los hogares, sí son un elemento ya imprescindible en la mayoría de las industrias.

Podemos contemplar a la robótica como una ciencia que aunque ha conseguido grandes avances todavía ofrece un amplio campo para el desarrollo y la innovación, y es precisamente este aspecto el que motiva a muchos investigadores y aficionados a seguir adelante planteando robots cada vez más evolucionados.

Los aficionados a los robots también juegan un papel importante en el desarrollo de la robótica, ya que son estos los que partiendo de una afición firme y con sus particulares ideas, al cabo de un cierto tiempo han podido desarrollar sus teorías y con ello crear un precedente o mejorar un aspecto que se tenía olvidado o con el cual no se había contado desde un principio.

El auge de la robótica, y la imperiosa necesidad de su implantación en numerosas instalaciones industriales, requiere el trabajo en conjunto de un buen número de especialistas y aficionados en la materia.

La robótica es una tecnología multidisciplinaria, ya que ésta hace uso de los recursos que le proporcionan ciencias afines. Solamente hay que pensar que en el proceso de diseño y construcción de un robot intervienen muchos campos pertenecientes a otras ramas de la ciencia, como pueden ser: la mecánica, la electrónica, la informática y la matemática, entre muchas otras que no por omitirlas carecen de importancia.

La robótica es realmente una combinación de todas las disciplinas mencionadas y otras más, más el conocimiento de la aplicación a la que se enfoca. La robótica brinda a investigadores y aficionados un vasto y variado campo de trabajo, lleno de objetivos y en un estado inicial de desarrollo.

2.3. Arquitecturas de los robots

Los dispositivos y mecanismos que pueden agruparse bajo la denominación genérica de robot son muy diversos y por ende es difícil establecer una clasificación coherente de los mismos que resista un análisis crítico y riguroso. La subdivisión de los robots, con base en su arquitectura, puede realizarse en los siguientes grupos [RTIC]:

- Poliarticulados

Bajo este grupo están los robots de muy diversa forma y configuración cuya característica común es la de ser básicamente sedentarios, aunque excepcionalmente pueden ser guiados para efectuar desplazamientos limitados, y estar estructurados para mover sus elementos terminales en un determinado espacio de trabajo según uno o más sistemas de coordenadas y con un número limitado de grados de libertad. En este grupo se encuentran los manipuladores, los robots industriales, los robots cartesianos y algunos robots industriales. Se emplean cuando es preciso abarcar una zona de trabajo relativamente amplia o alargada, actuar sobre objetos con un plano de simetría vertical o deducir el espacio ocupado en el suelo.

- Móviles

Son robots con gran capacidad de desplazamiento, basados en carros o plataformas y dotados de un sistema locomotor de tipo rodante. Siguen su camino por telemando o guiándose por la información recibida de su entorno a través de sus sensores.

Estos robots aseguran el transporte de piezas de un punto a otro de una cadena de fabricación. Guiados mediante pistas materializadas a través de la radiación electromagnética de circuitos empotrados en el suelo, o a través de bandas detectadas fotoeléctricamente, pueden incluso llegar a sortear obstáculos y están dotados de un nivel relativamente elevado de inteligencia.

- Androides

Son robots que intentan reproducir total o parcialmente la forma y el comportamiento cinemático del ser humano. Actualmente los androides son todavía dispositivos muy poco evolucionados y sin utilidad práctica y destinados, fundamentalmente, al estudio y la experimentación.

Uno de los aspectos más complejos de estos robots, y sobre el que se centra la mayoría de los trabajos, es el de la locomoción bípeda. En este caso, el principal problema es controlar dinámica y coordinadamente en tiempo real el proceso y mantener simultáneamente el equilibrio del robot.

- Zoomórficos

Los robots zoomórficos, que considerados en sentido no restrictivo podrían incluir también a los androides, constituyen una clase caracterizada principalmente por sus sistemas de locomoción que imitan a los diversos seres vivos.

A pesar de la disparidad morfológica de sus posibles sistemas de locomoción, es conveniente agrupar a los robots zoomórficos en dos categorías principales: caminadores y no caminadores. El grupo de los robots zoomórficos no caminadores está muy poco evolucionado. Cabe destacar, entre otros, los experimentos efectuados en Japón basados en segmentos cilíndricos biselados acoplados axialmente entre sí y dotados de un movimiento relativo de rotación. En cambio, los robots zoomórficos caminadores multípedos son muy numerosos y son objeto de experimentos en diversos laboratorios con vistas al desarrollo posterior de verdaderos vehículos terrenos, piloteados o autónomos, capaces de evolucionar en superficies muy accidentadas. Las aplicaciones de estos robots serán interesantes en el campo de la exploración espacial y en el estudio de los volcanes.

- Híbridos

Estos robots corresponden a aquellos de difícil clasificación cuya estructura se sitúa en combinación con alguna de las anteriores ya expuestas, bien sea por conjunción o por yuxtaposición. Por ejemplo, un dispositivo segmentado articulado y con ruedas, es al mismo tiempo uno de los atributos de los robots móviles y de los robots zoomórficos. De igual forma pueden considerarse híbridos algunos robots formados por la yuxtaposición de un cuerpo formado por un carro móvil y de un brazo semejante al de los robots industriales. En parecida situación se encuentran algunos robots antropomorfos y que no pueden clasificarse ni como móviles ni como androides, tal es el caso de los robots personales.

2.4. Contexto actual de la robótica

En el contexto actual la noción de robótica implica una cierta idea preconcebida de una estructura mecánica universal capaz de adaptarse, como el hombre, a muy diversos tipos de acciones y en las que concurren, en mayor o menor grado según los casos, las características de movilidad, programación, autonomía y multifuncionalidad [STNO]. Pero en sentido actual, abarca una amplia gama de dispositivos con muy diversos trazos físicos y funcionales asociados a la particular estructura mecánica de aquellos, a sus características operativas y al campo de aplicación para el que se han concebido. Es además evidente que todos estos factores están íntimamente relacionados, de tal forma que la configuración y el comportamiento de un robot condicionan su adecuación para un campo determinado de aplicaciones y viceversa; ello, a pesar de la versatilidad inherente al propio concepto de robot.

2.5. La construcción de un robot

La construcción de un robot, ya sea una máquina que camine de forma parecida a como lo hace el ser humano, o un manipulador sin rostro para una línea de producción,

es fundamentalmente un problema de control. Existen dos aspectos principales: mantener un movimiento preciso en condiciones que varían y conseguir que el robot ejecute una secuencia de operaciones previamente determinadas. Los avances en ambos campos, siendo el primero esencialmente un problema matemático y el segundo de tecnología, suministran la más grande contribución al desarrollo del robot moderno [RTIC].

3. Los Robots LEGOS

Los Robots LEGO Mindstorms [LEGa] están diseñados como un producto de consumo, lo que les dota de una gran calidad y además de una enorme flexibilidad. El LEGO MindStorms Robotics Invention System es un juguete, basado en la filosofía del grupo LEGO, cuya idea de ser es que el niño debe armar sus propios juguetes. El sistema básico consiste en un paquete de 727 piezas (ladrillos, bandas, engranes, sensores, ejes y motores) con las cuales el niño puede armar diferentes modelos de robots que pueden ser controlados por un programa, también diseñado por el niño, desde una computadora.

El paquete incluye su propio software, el cual se instala fácilmente en una computadora personal. Para disminuir la dificultad en la programación para un niño de 12 años, se diseñó un lenguaje tipo “rompecabezas”, donde cada instrucción viene en una pieza y el niño simplemente las junta con cierta lógica.

El LEGO MindStorms nos permite construir una enorme variedad de robots, desde un brazo hasta un robot recolector, sin necesitar ninguna soldadura. Además, permite una gran libertad a la hora de colocar los sensores y los motores de los robots.

El cerebro de estos robots está basado en un microprocesador Hitachi H8/300 con 32 Kbytes de RAM, que se denomina habitualmente ``ladrillo" o RCX (con la memoria y los puertos de entrada/salida). Además, el *kit* incluye un entorno gráfico de desarrollo y el equipo necesario para descargar el software desde una computadora personal al ladrillo. Dispone de tres puertos de salida para conectar motores y tres puertos de entrada para conectar sensores, además de un puerto de infrarrojos para comunicarse con la computadora que sirve de plataforma de desarrollo.

El robot cuenta con cinco elementos principales:

- El RCX o “ladrillo” ya mencionado, el cual es el controlador del robot. Es un microprocesador que puede ser programado por una computadora.

- Los motores, son los que permiten el movimiento del robot.

- El sensor de luz detecta cuando hay una sombra o muy poca luz en el ambiente.

- El sensor de contacto sirve para detectar que el robot choca con algún objeto.

- La torre IR es un transmisor infrarrojo que permite que el robot pueda interactuar con el ambiente, independientemente de la computadora. Para la instalación de la torre infrarroja IR, se necesita contar con un puerto USB.

3.1. Sensores

Los sensores disponibles son el de luz ambiental, de luz reflejada, de colisión, de temperatura y de rotación [LEGb]. Además cuentan con una interfaz amigable para su control, fácil de usar y con muchos movimientos posibles para el robot, también cuentan con programación de sonidos que son fáciles de manejar.

Los robots Lego MindStorms, que serán controlados por el usuario cliente del sistema a desarrollar, cuentan con sensores de luz y sensores de contacto. Para que un robot pueda ejecutar tareas que ahora son realizadas por los humanos, debe contar con habilidades de detección. El robot emplea el sensor como un dispositivo de medición.

Los sensores en los robots cumplen la misma función que los órganos en la mayoría de los seres vivos. Sin ellos los robots no podrían localizar objetos para poder recogerlos, evitar chocar con obstáculos o comprobar el correcto funcionamiento de una actividad. Además, los sensores ayudan al robot a conocer sus parámetros internos, tales como la posición y la velocidad, entre otros.

Los sensores son en realidad elementos físicos que pertenecen a un tipo de dispositivo llamado transductor.

Los transductores son elementos capaces de transformar una variable física en otra diferente. Los sensores son un tipo concreto de transductores que se caracterizan porque son usados para medir la variable transformada. La magnitud física que suele ser empleada por los sensores como resultado suele ser la tensión eléctrica, debido a la facilidad del trabajo con ella.

Los sensores utilizados en aplicaciones de robótica pueden clasificarse como sensores de contacto o sensores de no contacto. También pueden clasificarse como internos y externos o pasivos y activos. Los sensores de contacto permiten al robot detectar el choque con un objeto.

3.2. NQC

NQC significa *Not Quite C* (No Completamente C) [OVMa], y es un lenguaje sencillo para programar varios productos Lego Mindstorms. Algunas de las características de NQC dependen del producto Mindstorms que se utilice. NQC se refiere a los diferentes ladrillos inteligentes como el *destino*. Actualmente NQC soporta cinco diferentes destinos: RCX, RCX2, CyberMaster, Scout y Spybotics.

Todos los destinos tienen un intérprete de código de bytes (proporcionado por Lego) que puede ser utilizado para ejecutar programas. El compilador NQC convierte un programa fuente en Lego bytecode, el cual puede ser ejecutado en el propio destino. Aunque las estructuras de procesamiento y de control son muy similares a C, NQC no es un lenguaje de propósito general: hay muchas restricciones que son producto de las limitaciones del intérprete de código de bytes Lego.

Lógicamente el NQC se define en dos partes diferenciadas. El lenguaje NQC describe la sintaxis a utilizar al escribir programas. El NQC API describe las funciones del sistema, constantes y macros que se pueden usar en los programas. Este API se define en un archivo especial incluido en el compilador. Por defecto, este archivo siempre se procesa antes de compilar el programa.

3.3. Comunicación entre Legos

Si se dispone de más de un RCX, se pueden escribir programas que permitan a los robots Lego comunicarse entre sí utilizando el puerto de infrarrojos [OVMa]. De este modo, se puede hacer que varios robots colaboren o peleen entre ellos.

El RCX puede enviar y recibir mensajes simples utilizando los infrarrojos. Un mensaje puede tener un valor desde 0 hasta 255, pero no se recomienda utilizar el mensaje 0. El último mensaje recibido es guardado y puede accederse a él mediante Message(). Si no ha sido recibido ningún mensaje, Message() devolverá el valor 0. Debe tenerse en cuenta, que debido a la naturaleza de la comunicación mediante infrarrojos, no se podrán recibir mensajes mientras un mensaje se esté transmitiendo.

Cuando se trabaja con proyectos de comunicación entre Legos, tenemos que elegir a uno como el **líder.** Podemos llamarlo el ***patrón*** o ***jefe***. A veces los esclavos pueden devolver información al patrón. Por ejemplo, el valor devuelto por un sensor o algún valor significativo para nuestro proyecto. Así que se tienen que escribir dos programas: uno para el patrón y otro para el esclavo.

Debemos de tener cierto cuidado cuando manejamos la comunicación por puerto infrarrojo, ya que si los dos robots envían información a la vez ésta puede perderse. Otro problema es que el robot no puede enviar un mensaje en cualquier momento, ya que si dos mensajes son enviados aproximadamente en el mismo momento pueden perderse. Así mismo, un robot no puede enviar y recibir mensajes a la vez.

Capítulo 2

ANÁLISIS Y DISEÑO

1. Recursos

Este proyecto cuenta con los siguientes recursos:

1. Un kit Lego Mindstorms Robotic Invention System 2.0 con RCX versión 2.0, incluye 2 sensores de choque y un sensor de luz.

2. Un kit Lego Dacta Robolab con RCX versión 1.0, incluye 2 sensores de choque y un sensor de luz.

3. Paquete de desarrollo RCX Command Center Versión 3.1 para la programación de los robots en el lenguaje NQC [OVMb].

4. Pelotas de hule, canicas, papel aluminio, pelotas transparentes, pilas, focos, cables de conexión.

5. Una cancha de juego de madera triplay de 85 x 91 cm, con área del portero y puntos de tiro marcados. Cuenta con una portería.

2. Planteamiento inicial

2.1. Definición del problema

Tenemos una cancha de fútbol, en la cual se coloca la pelota en cualquier parte y el portero se encuentra cubriendo su arco. El tirador puede empezar en cualquier lugar de la cancha y su primer objetivo es buscar la pelota y encontrarla, evitando chocar con las paredes. Una vez encontrada la pelota, se coloca para tirar de acuerdo a la ubicación de la portería. En el momento que se coloca el tirador, el portero busca pelear por la pelota y detenerla. La pelota es de hule y mide 2 cm de diámetro. Está recubierta con papel aluminio, buscando con ello que los robots detecten su presencia según la luz que refleje.

Las actividades que deberá realizar el tirador son:

- Detectar la pelota

  La idea es que el portero recorra un ángulo de "visión" en busca de la pelota (objetivo) con ayuda de algún sentido.

  Se propone el uso de un sensor de luz y una pelota de goma forrada de papel aluminio que sirva como superficie reflejante de la luz emitida por el sensor de luz de modo que el sensor detecte su propio reflejo y vaya tras él.

  El resultado de este experimento no fue el esperado. La superficie no era suficiente para hacer que el robot la siguiera.

  La segunda propuesta fue un una pelota plástica transparente en cuyo interior se aloja un foco. La luz emitida fue suficiente para hacer que el sensor la detectara con ayuda de un valor de umbral apropiado que de forma ideal no cambia.

  El resultado de este experimento fue bastante bueno aunque aún queda el problema de tener que reemplazar continuamente la batería que alimenta el foco para evitar un cambio de valor significativo del umbral.

- Evitar obstáculos

  El tirador tiene que continuamente estar alerta mediante su sentido de "tacto" para poder detectar al instante, en caso de ocurrir, que se ha topado con algún obstáculo (pared) que impida su libre desplazamiento en busca de la pelota. En caso de que se encuentre con dicho obstáculo, el tirador es capaz de corregir su desplazamiento para alejarse de él y continuar su movimiento en otra dirección.

  El programa realizado para cumplir con esta actividad funcionó de forma satisfactoria y cumplió por completo con su cometido.

- Comunicarse con el portero

  Una vez encontrada la pelota, el tirador comunica al portero que está listo para tirar y le informa la ubicación del esférico, ocasionalmente proporcionándole información incorrecta que proporcione una ventaja al tirador.

  Esta actividad y las siguientes no fueron implementadas para el planteamiento inicial debido a que éste fue descartado antes de llegar al momento de elaborar el respectivo programa, en la sección 2.4 se discute dicha situación a detalle.

- Colocación para tiro

  De acuerdo a la ubicación del tirador en relación con la portería al momento de encontrar la pelota, éste se coloca apropiadamente para tirar.

- Tirar

  El robot lleva a cabo el tiro de la pelota, presentándose un grado de variabilidad en la potencia y dirección entre distintas ocasiones de tiro.

Las actividades que deberá realizar el portero son:

- Esperar el mensaje del tirador
  El portero se encuentra inmóvil en el centro de la portería, en espera de recibir un mensaje del tirador que le comunique la inminencia del tiro de la pelota y de donde provendrá éste.

- Salir a detener la pelota
  El portero sale de su arco y busca desplazarse a la ubicación del punto de tiro, ello con la misión de detener la pelota y evitar que el tirador pueda hacer un gol.

2.2. Programas realizados

Los programas ejecutan las siguientes operaciones (refiérase a la biblioteca de programas que se encuentra en la sección de Anexos para ver el código completo de estos programas):

1. *Sigue la luz*
   El tirador se pone a girar sobre su propio eje hasta detectar la presencia de una luz cuya intensidad sea mayor a un determinado umbral, desplazándose hacia la dirección en que detecte este umbral.

2. *Evitar obstáculos*
   El robot se desplaza en el entorno. Si choca con una pared, es capaz de corregir su desplazamiento para alejarse de ella y continuar moviéndose en otra parte del entorno. En caso de que se encuentre atrapado en un callejón sin salida, es capaz de salir de él.

3. *Encontrar la luz*
   El tirador está en constante búsqueda de una fuente de luz con una luminosidad superior a un umbral determinado. En el momento que la encuentra se acerca a la misma hasta el punto en que el aumento del umbral indica cierta cercanía, deteniéndose el tirador.

4. *Evitar obstáculos mientras se busca la luz*
   El tirador está en constante búsqueda de una fuente de luz según lo especificado en el programa tres, evitando obstáculos según lo establecido en el programa dos.

2.3. Diseño de los robots

Para este planteamiento únicamente llevamos a cabo el diseño y construcción del robot tirador debido a la problemática discutida en la sección 2.4.

Para facilitar ambas tareas, nos apoyamos en el manual de proyectos incluido con el Robotics Invention System 2.0 titulado Constructopedia [CRIS]. Fue de este manual que obtuvimos ideas generales para la construcción del tirador, así como las bases necesarias para poder proponer y probar diseños propios.

Comenzamos con el diseño de doble defensa propuesto para el Roverbot en la Constructopedia, el cual requiere los dos sensores de toque incluidos. Utilizamos esta doble defensa para poder detectar el choque del tirador con un obstáculo ya fuese de su lado izquierdo o derecho, además de poder detectar si el choque incidió en ambos lados. Colocamos el sensor de luz en la parte central de la doble defensa, ya que dicho lugar es el más apropiado del robot para poder detectar la presencia de la pelota. Esta doble defensa con sensor de luz aparece en la figura 2.1.

Figura 2.1: Doble defensa del robot tirador

El resto del cuerpo del robot fue construido alrededor de la doble defensa y el RCX, colocando la doble defensa en la parte frontal inferior del robot y el RCX al centro de su cuerpo. En las figuras 2.2 a 2.5 aparecen distintas vistas del robot tirador.

Figura 2.2: Vista superior del tirador

Figura 2.3: Vista posterior del tirador

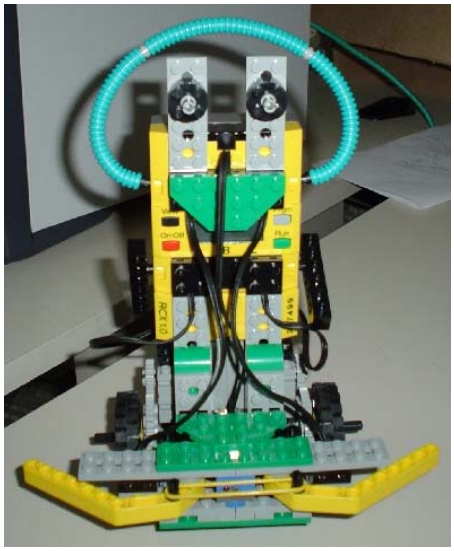

Figura 2.4: Vista frontal del tirador

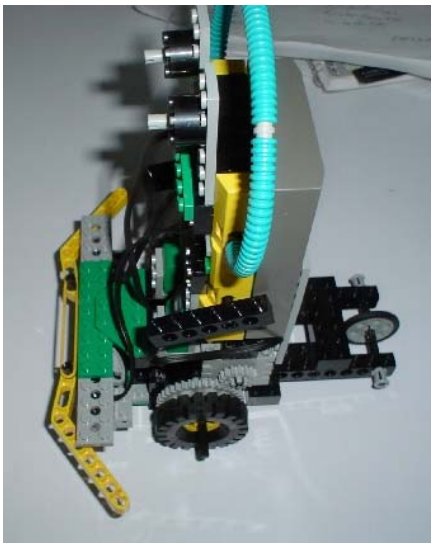

Figura 2.5: Lado izquierdo del tirador

2.4. Problemas suscitados

Para el programa de búsqueda de la pelota, el recubrirla de papel aluminio no originó los niveles de reflexión de luz necesarios para una búsqueda y detección confiable. En un cuarto alumbrado normalmente (con un nivel de luz de promedio de 20 según la medición del sensor) apenas hubo un incremento de 2 a 3 en el umbral de luz al encontrarse el robot a 3 cm de ella, la distancia a la cual se detectaba un incremento de 1 era de aproximadamente 12 cm. Dichas condiciones hicieron inservible el usar un recubrimiento de papel aluminio ya que el nivel de luz dentro de cualquier entorno no se mantiene constante y varía dentro de cierto intervalo de valores; en el caso del cuarto de prueba había variaciones de entre $\pm$ 2-4 de acuerdo a la orientación del robot. Se escudriñó diversas soluciones al problema buscando conservar una pelota recubierta de papel aluminio, pero la probabilidad de error en su búsqueda y detección nos obligó a desechar su posibilidad de uso. Aún haciendo pruebas con superficies planas recubiertas por aluminio, no era confiable su detección ya que al encontrarse el robot a 3 cm de ella el umbral tenía un incremento en alrededor de 5.

Para resolver el problema de la detección de la pelota, planteamos la necesidad de que ésta emitiera luz por sí sola para facilitar su localización e incrementar los niveles del umbral lo suficiente para evitar confundirla con variaciones naturales en el entorno. En el momento que emite luz por sí misma, su localización es sencilla y disminuye drásticamente la posibilidad de no detectar su presencia. Al no contar en el momento con el material necesario para obtener o construir una pelota que emitiese luz, modificamos el programa de ubicación de la misma de acuerdo al umbral generado por

un encendedor, contemplando que el valor de dicho umbral sería modificado posteriormente de acuerdo a las características de emisión de luz de la pelota a construir.

Fue después de hacer estos cambios que se suscitó una complicación inesperada. Funcionando ahora correctamente los programas de evitar obstáculos y búsqueda de la pelota, se requería correrlos en paralelo ya que en la práctica el robot debía estar evitando obstáculos (paredes) mientras estaba en búsqueda de la pelota (programa número 4). El programa de detección de obstáculos funcionaba de forma confiable por sí solo, haciendo uso de los dos sensores de choque para detectar en qué posición se encontraba al toparse con una pared, saliendo del lugar para buscar la pelota en otra dirección y evitando encontrarse atrapado en esquinas. Al momento de correr esta tarea en paralelo con la búsqueda de la pelota, ocurría lo siguiente:

> *En el momento en que el robot chocaba con una pared, se incrementaba sustancialmente el umbral del sensor de luz. Esto se debía a la proximidad del mismo con la pared al momento del choque debido a su ubicación en el robot, por lo que el robot en ese instante malinterpretaba el incremento del umbral como señal de que había encontrado la pelota y se quedaba atorado al querer empujarla, avanzando inútilmente.*

Se buscó diversas alternativas para tratar el problema, pero éstas planteaban la necesidad de contar con sensores adicionales a los cuales no tenemos acceso. Por lo tanto, en vista de la problemática surgida y las restricciones en recursos y tiempo con que se cuenta para el proyecto, hubo un cambio en el planteamiento del problema y los objetivos buscados.

2.5. Resultados obtenidos

- Se requiere usar una pelota con iluminación propia.

- Se restringe el problema al de realizar tiros libres a partir de puntos de tiro establecidos, en vez del de enfrentarse al portero con la pelota sin importar donde ella se encuentre. Este cambio origina una situación más similar a la realidad de los tiros libres, y no de penal, en el fútbol, distinta al planteamiento inicial de confrontación con el portero mientras la pelota está en juego.

- Es menester eliminar la necesidad de evitar obstáculos.

3. Planteamiento modificado (un auténtico tiro libre)

3.1. Planteamiento

Se tiene una cancha de fútbol en la cual se coloca la pelota en cualquiera de tres posiciones preestablecidas en una línea de tiro: izquierda, centro o derecha. El tirador inicia en un punto fijo y preestablecido que se encuentra detrás de la posición de tiro central. El primer objetivo del tirador es buscar la pelota y encontrarla. Una vez encontrada, se coloca para tirar de acuerdo a la posición de la portería en relación con el punto de tiro.

El tirador comunica al portero cual es la posición en que se encuentra, colocándose el último de acuerdo a la situación. En un tiempo aleatorio, el robot tirador empuja el esférico hacia la portería mientras el arquero intenta detectar su movimiento y cercanía mediante el sensor de luz, buscando salir a detenerla. Mediante sus sensores de toque y el de luz, el portero es capaz de detectar si detuvo o no la pelota. El portero le envía al tirador un mensaje para indicarle si la detuvo o no, haciendo el ganador un movimiento de celebración.

La pelota es de plástico transparente y mide 5 cm de diámetro. En su interior cuenta con varios focos alimentados por baterías AAA que facilitan su detección por el sensor de luz de cualquiera de los dos robots.

La cancha mide 81 x 95 cm, teniendo marcados los tres puntos de tiro establecidos y el área de la portería.

Capítulo 3

DESARROLLO

1. Definición detallada del problema

A continuación se define a detalle las actividades que realiza cada uno de los robots para cumplir con el planteamiento establecido en el capítulo anterior.

Las actividades del tirador son:

- Detectar la pelota
  El tirador recorre un ángulo de "visión" en busca de la pelota (objetivo) mediante su sensor de luz. Detecta en cual de tres puntos de tiro posibles se encuentra ésta.

- Aproximación a la pelota
  El robot se acerca hasta cierta distancia de la pelota.

- Colocación para tiro
  Según el punto de tiro en el cual se encuentra la pelota, el robot se coloca debidamente para tirar.

- Comunicarse con el portero
  Una vez encontrada la pelota, el tirador comunica al portero que está listo para tirar y le informa en que punto se ubica el esférico.

- Tirar
  El robot lleva a cabo el tiro de la pelota.

- Esperar mensaje del portero
  El tirador espera la recepción de un mensaje del portero que le indique si este detuvo la pelota o no.

- Acción de celebración o derrota
  El robot realiza una acción de celebración o derrota según el mensaje que haya recibido del portero.

Las actividades del portero son:

- Esperar el mensaje del tirador
  El portero se encuentra inmóvil en el centro de la portería, en espera de recibir un mensaje del tirador que le comunique la inminencia del tiro de la pelota y de que punto provendrá éste.

- Movimiento de búsqueda
  Según el punto de origen del tiro, el portero orienta su sensor de luz hacia esa dirección. El robot comienza a realizar un movimiento cíclico de lado a lado que cubre los 180º frontales en los cuales se puede detectar la presencia de la pelota al acercarse.

- Detección de pelota
  El portero mantiene alerta su sensor de luz para poder detectar el acercamiento de la pelota al encontrarse ésta a una cierta distancia.

- Salir a detener la pelota
  Según la dirección en la cual fue detectada la pelota, el portero se desplaza en línea recta hacia ella para detener la pelota.

- Detectar si se detuvo la pelota
  El portero determina si fue capaz de detener la pelota.

- Enviar mensaje al tirador
  El portero comunica al tirador si fue capaz de detener la pelota.

- Acción de celebración o derrota
  El portero realiza una acción de celebración o derrota según haya evitado o no el gol.

2. Cancha de juego (entorno)

Antes de construir la cancha realizamos un diagrama de la misma para determinar su forma, tamaño y características (vea la figura 3.1).

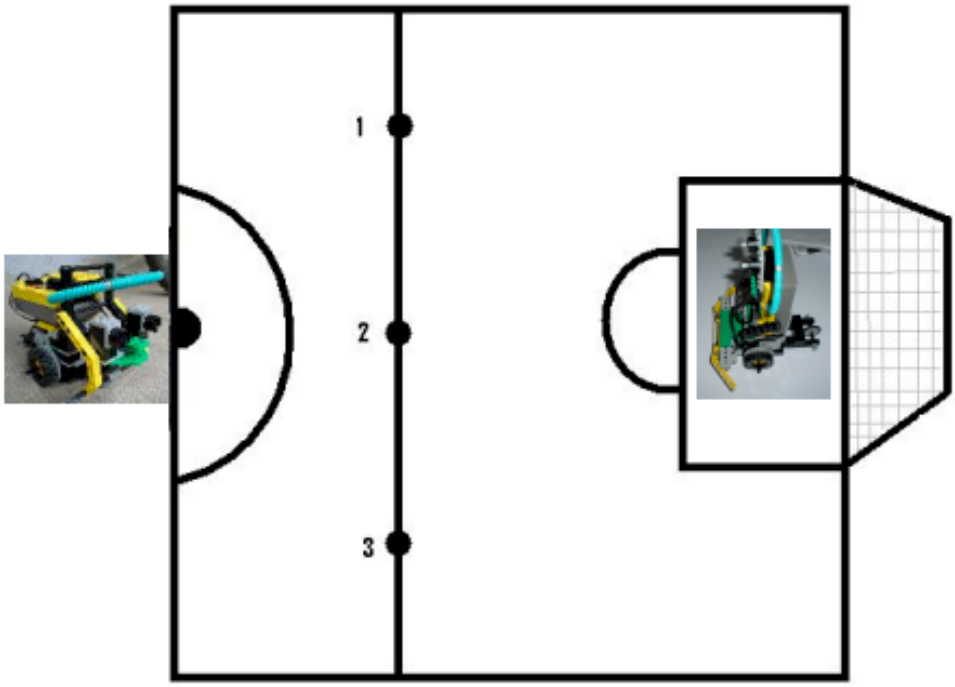

Figura 3.1: Diagrama de la cancha de juego

De acuerdo a este diagrama determinamos construir una cancha de juego a partir de una tabla de madera triplay con dimensiones de 81 cm por 95 cm, teniendo 2 cm de grosor. Dicha tabla está pintada de verde para mejor representar el aspecto de una cancha real, teniendo marcados con color blanco el área de la portería y los tres puntos de tiro (ver figura 3.4).

La portería fue construida a partir de piezas sobrantes con que disponíamos en los dos kits Lego que fueron utilizados para el proyecto (uno del robot portero y otro del robot tirador). En la figura 3.2 podrá apreciar una imagen de la portería.

Figura 3.2: Portería construida con piezas Lego

La pelota es una esfera de plástico transparente, dentro de la cual colocamos un pequeño foco alimentado por dos baterías AAA (figura 3.3).

Figura 3.3: Pelota de juego

Figura 3.4: Cancha de juego con portería

3. Construcción de los robots

Para facilitar el diseño y la construcción del robot tirador y el robot portero, nos apoyamos en el manual de proyectos incluido con el Robotics Invention System 2.0 titulado *Constructopedia* [CRIS]. Fue de este manual que obtuvimos ideas generales para la construcción de ambos robots, así como las bases necesarias para poder proponer y probar diseños propios.

- Tirador

Este robot tiene una forma rectangular que le da estabilidad y rapidez. Su diseño fue concebido tomando en cuenta la colocación del sensor de luz, el RCX y la estructura de tiro.

Debido a los requerimientos establecidos por el planteamiento, el único sensor que requiere este robot es el de luz. Dicho sensor fue colocado al frente del tirador y lo más cerca posible al nivel del piso.

El aspecto más destacado de este robot es la estructura de tiro. Ésta incrementa notablemente la longitud del tirador en su parte frontal y cuenta con dos puntos de apoyo reforzados a cada lado del robot. La parte que entra en contacto con la pelota está conformada por varillas de plástico. En la figura 3.5 podrá apreciar una imagen de esta estructura.

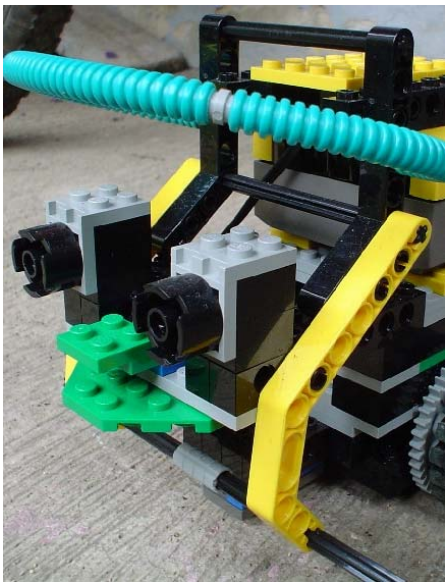

Figura 3.5: Estructura de tiro

En las figuras 3.6 a 3.9 aparecen imágenes con distintas vistas del robot tirador.

Figura 3.6: Vista frontal del tirador

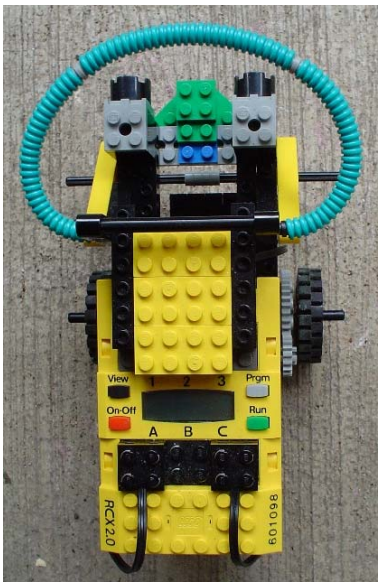

Figura 3.7: Vista superior del tirador

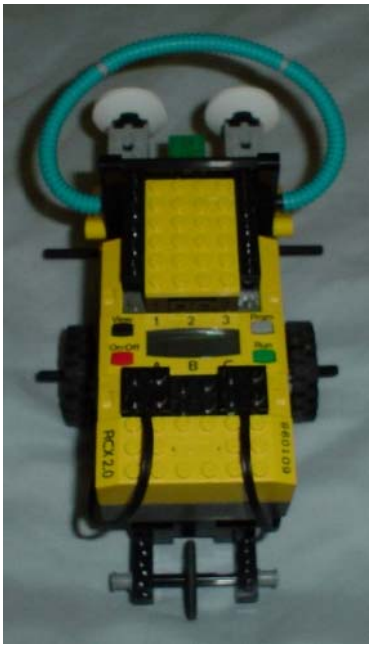

Figura 3.8: Vista posterior del tirador

Figura 3.9: Lado izquierdo del tirador

- Portero

Exactamente el mismo robot diseñado y construido como el tirador del primer planteamiento (sección 2.3) es utilizado como portero del planteamiento final del proyecto. Dicha decisión partió de la conveniencia de utilizar a dicho robot como portero, además de tener la oportunidad de construir un robot tirador con un diseño muy distinto que le permitiera moverse con mayor rapidez y facilidad, pudiendo distinguir con facilidad a un robot de otro.

Para este robot se hizo uso del diseño de doble defensa propuesto para el "Roverbot" en la *Constructopedia* [CRIS], el cual requiere los dos sensores de toque. Utilizamos esta doble defensa para poder detectar el choque del portero con la pelota, ya sea que ocurra éste a su lado izquierdo o derecho. Colocamos el sensor de luz en la parte central de la doble defensa, ya que este lugar es el más apropiado del robot para poder detectar la presencia de la pelota. Esta doble defensa con sensor de luz aparece en la figura 3.10.

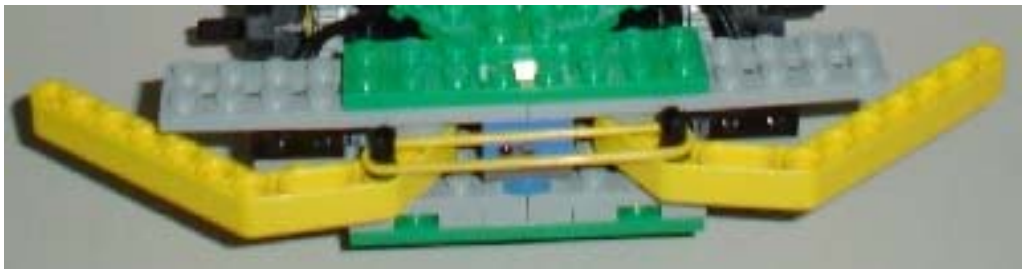

Figura 3.10: Doble defensa del robot portero

Haciendo uso de la doble defensa mencionada con anterioridad, el cuerpo del portero fue construido alrededor de ésta y el RCX, colocando la doble defensa en la parte frontal inferior del robot y el RCX al centro de su cuerpo.

En las figuras 3.11 a 3.14 aparecen distintas vistas del robot portero que construimos.

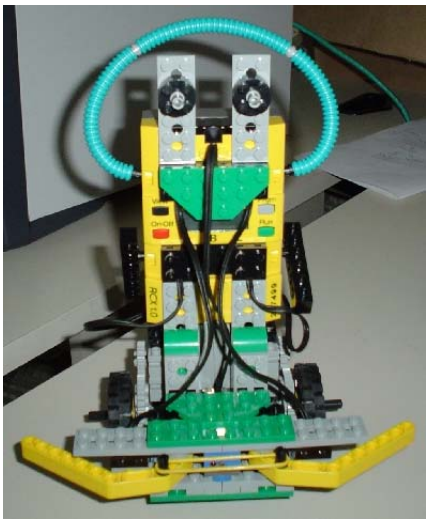

Figura 3.11: Vista frontal del portero

Figura 3.12: Vista superior del portero

Figura 3.13: Vista posterior del portero

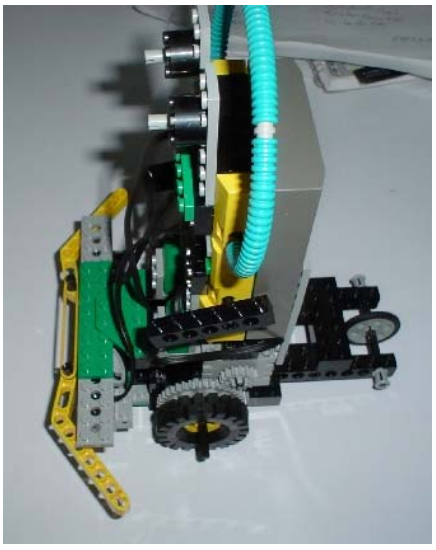

Figura 3.14: Lado izquierdo del portero

4. Diagrama de procesos

Las actividades a realizar por cada uno de los robots fueron agrupadas en módulos, viendo a dichos módulos como procesos independientes que se desarrollan por separado y son conjuntados al final para cada robot. En la figura 3.15 aparece cada uno de estos módulos.

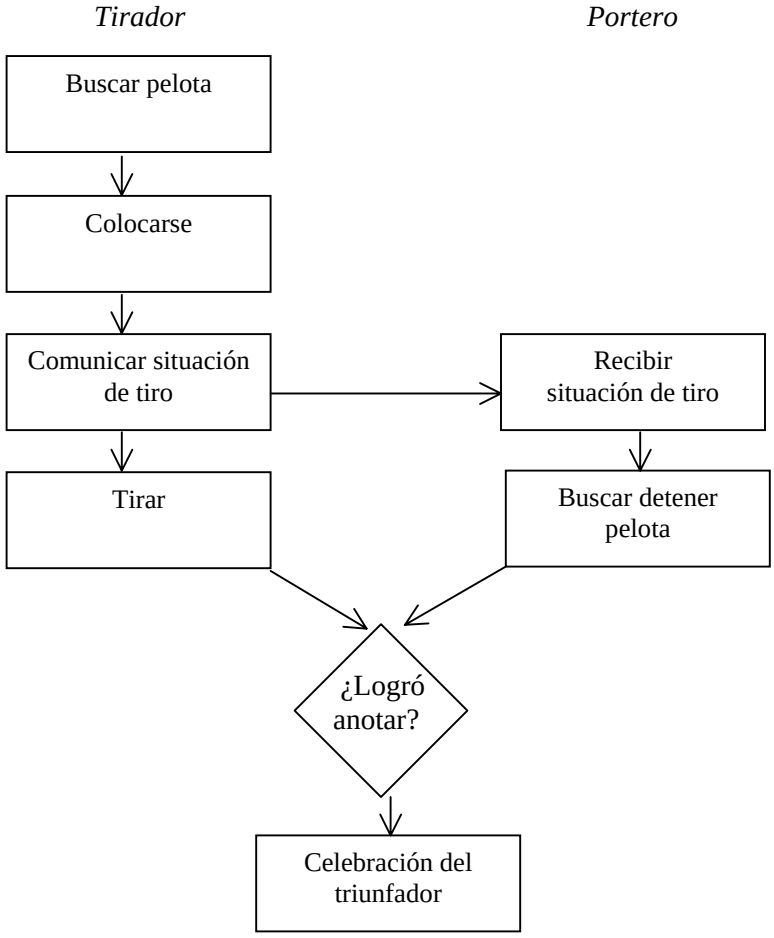

Figura 3.15: Diagrama por módulos (nivel 0)

A continuación se presenta el diagrama detallado de procesos de cada uno de estos módulos:

- Buscar pelota

El tirador primero determina si el punto de tiro es el central, en caso de no encontrar la pelota en este punto gira a la izquierda 90º para detectar si la pelota se encuentra en

el punto de tiro izquierdo. Si no encuentra la pelota, gira 180º a la derecha en búsqueda de la pelota para entonces determinar en donde se encuentra el punto de tiro derecho.

En el momento que el sensor de luz detecta la ubicación de la pelota, el tirador se dirige hacia el punto donde ésta se encuentra, deteniéndose al encontrarse a una distancia indicada al alcanzar el umbral de luz un determinado valor.

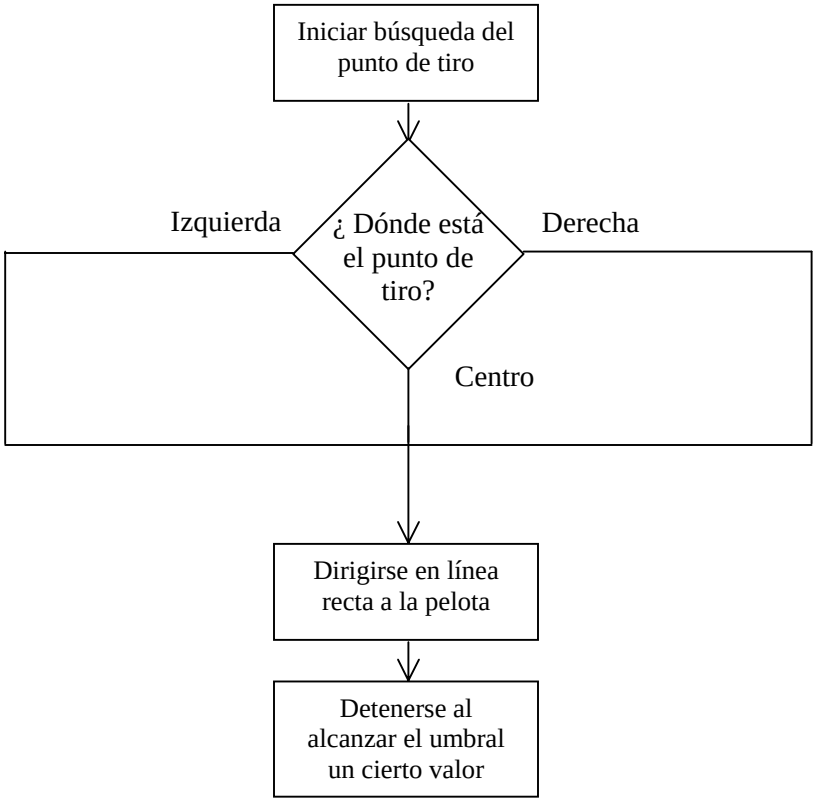

Figura 3.16: Buscar pelota (nivel 1)

- Colocarse

De acuerdo al punto de tiro en el cual fue ubicada la pelota, el robot se acomoda para ocupar su posición de tiro. Si la pelota está en el punto de tiro central, el robot simplemente tira y no se acomoda. En caso de que el punto de tiro sea el izquierdo, el robot se coloca al lado izquierdo de tal forma que la pelota se encuentre a 45º de su nueva posición. En caso de que el punto de tiro sea el derecho, el robot se coloca al lado derecho de tal forma que la pelota se encuentre a 45º de su nueva posición.

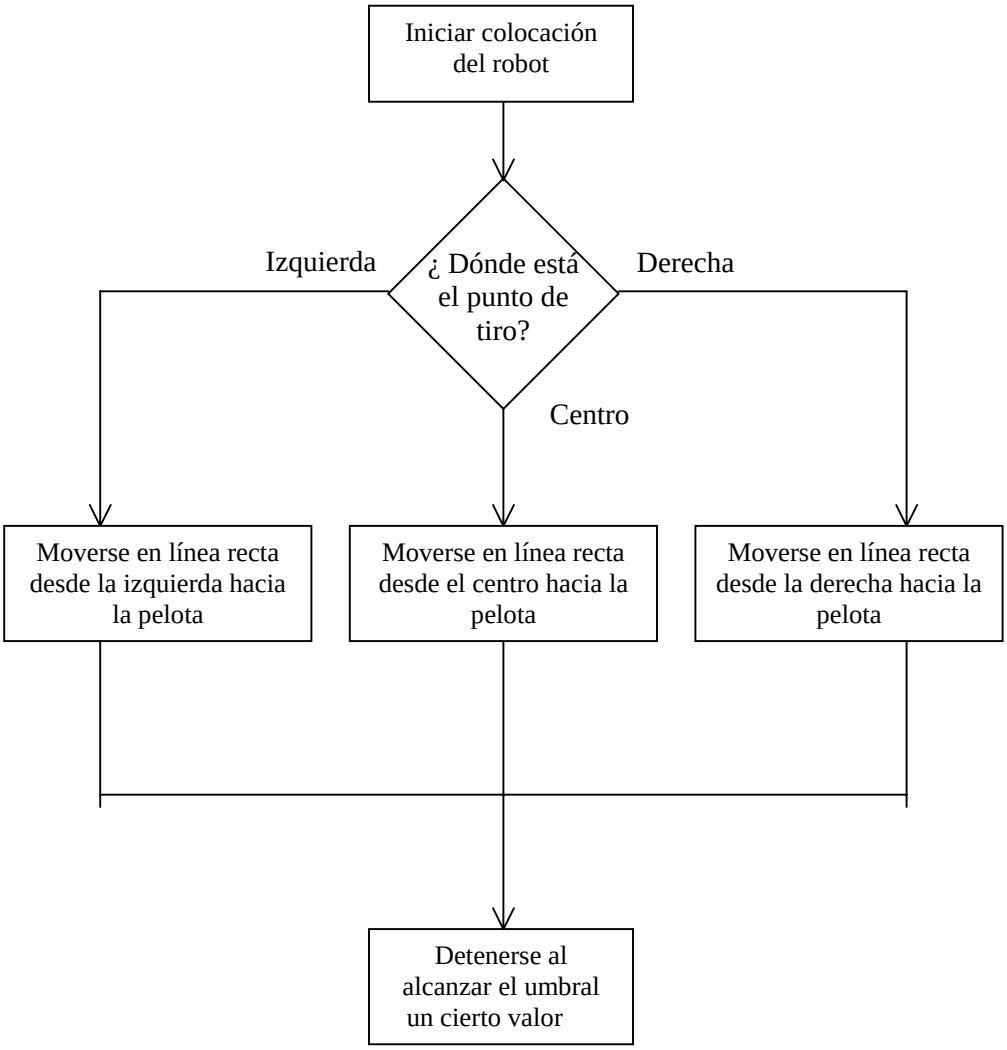

Figura 3.17: Colocarse (nivel 1)

- Comunicar situación de tiro

El tirador envía un mensaje al portero con el cual le informa en cual de los tres puntos de tiro está colocada la pelota y que en cualquier momento intentará meter gol.

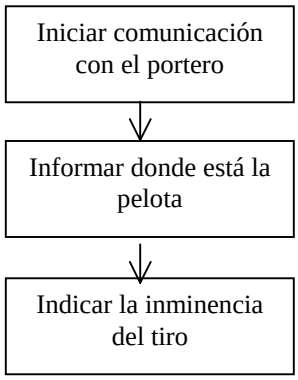

Figura 3.18: Comunicar situación de tiro (nivel 1)

- Tirar

El tirador se desplaza hacia delante a máxima potencia por un tiempo determinado, deteniendo su marcha al cumplirse dicho plazo.

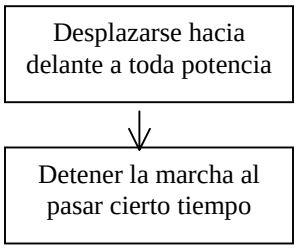

Figura 3.19: Tirar (nivel 1)

- Buscar detener pelota

El portero se orienta hacia el punto de tiro donde se encuentra la pelota. Gira continuamente sobre su propio eje, cubriendo con dicho giro un ángulo de 180º al frente. Continua realizando este giro hasta el momento en que su sensor de luz detecte que el umbral de luz sobrepase un determinado valor (señal de que la pelota está cerca), cuando ocurre dicho suceso detiene su giro y se desplaza por un determinado tiempo hacia la dirección donde detectó el aumento en el umbral. Si el portero detecta un choque con la pelota mediante sus sensores de toque o el de luz antes de que concluya este tiempo, detiene su marcha.

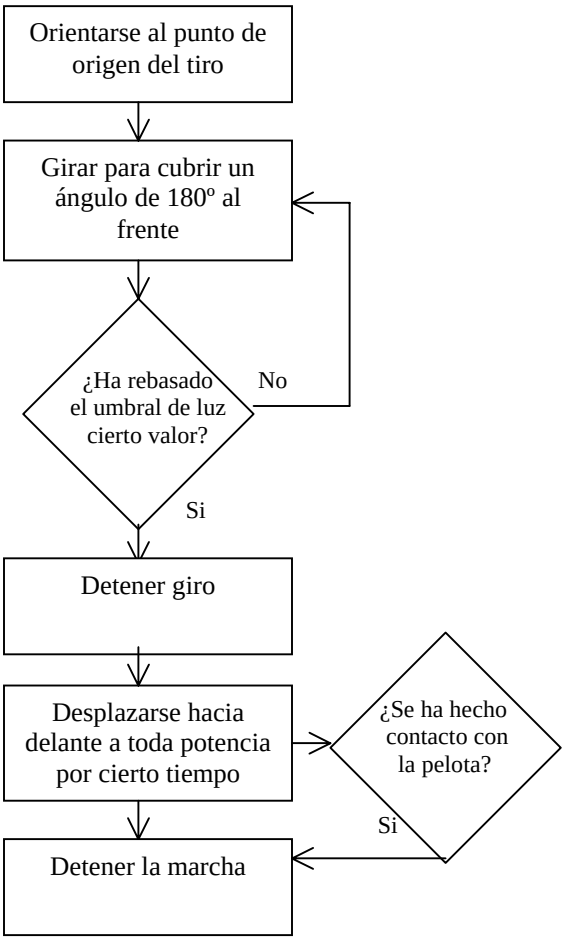

Figura 3.20: Buscar detener pelota (nivel 1)

- Celebración del triunfador

En caso de que el portero haya logrado cumplir con su objetivo de detener la pelota, realiza una acción de celebración. En caso contrario le envía un mensaje al respecto al tirador, siendo este quien entonces realiza la acción de celebración.

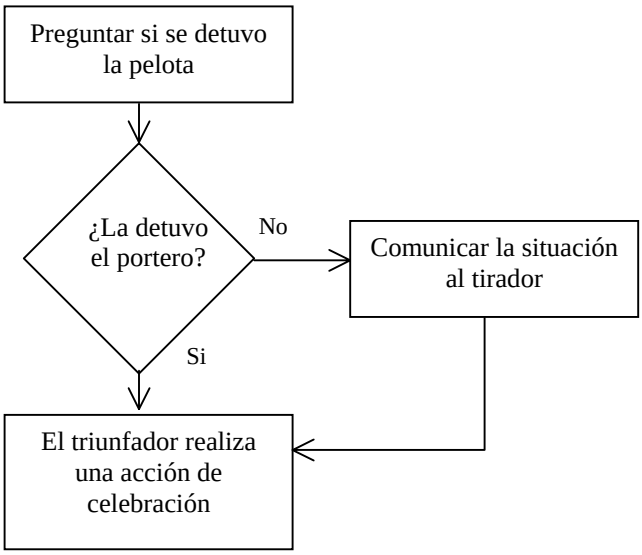

Figura 3.21: Celebración del triunfador (nivel 1)

5. Programas

Realizamos la codificación de los programas requeridos por pasos, codificando y agrupando cada conjunto de acciones según fuese necesario. Ello nos permitió llevar a cabo un desarrollo en el cual se generaron programas aislados que permitieron materializar cada uno de los procesos propuestos en la sección anterior, para luego conjuntar estos programas y obtener uno único para cada robot en donde se reunieran sus respectivas acciones. Son estos últimos programas los que se cargan en la memoria de cada robot, ejecutándose ambos simultáneamente para cumplir con lo propuesto en el planteamiento final.

(Refiérase a la biblioteca de programas al final del capítulo para ver el código completo de estos programas.)

1. *Buscar pelota*
   El tirador busca en cual de los tres puntos de tiro preestablecidos se encuentra la pelota mediante la detección de su luz. Se aproxima a la pelota hasta una distancia a la cual el umbral de luz rebasa cierto valor.

2. *Colocarse y tirar*
   El tirador se coloca para realizar el tiro según cuál de los tres puntos de tiro posibles sea indicado mediante una función aleatoria. Después de que se ha colocado el robot, se realiza el tiro a máxima potencia y los motores detienen su marcha.

3. *Buscar la pelota y tirar*
   El tirador busca en cual de los tres puntos de tiro preestablecidos se encuentra la pelota mediante la detección de su luz, asignando a una variable un valor que indique en que punto se encuentra. El tirador se aproxima a la pelota hasta una distancia a la cual el umbral de luz rebasa cierto valor. A continuación, el tirador se coloca para cobrar el penal según cuál de los tres puntos de tiro posibles sea indicado mediante la variable almacenada. El siguiente paso es realizar el tiro a máxima potencia, después de lo cual los motores detienen su marcha.

4. *Comunicar situación de tiro*
   El tirador indica al portero que está en posición de tirar un penal y le señala en cual de los puntos de cobro preestablecidos se encuentra la pelota.

5. *Buscar detener pelota*
   El portero intenta detener la pelota una vez que detecta su aproximación mediante el sensor de luz. Detecta si fue capaz o no de evitar el gol y le comunica el resultado al tirador.

6. *Movimiento de celebración*
   El robot ganador hace movimientos de celebración para festejar su victoria sobre su contrincante.

7. *Tirador*
   Este programa integra todo el código que requiere el robot tirador para cumplir con el planteamiento del proyecto, conjunta aquellos programas mencionados anteriormente que sean de interés para el tirador.

8. *Portero*
   Este programa integra todo el código que requiere el robot portero para cumplir con el planteamiento del proyecto, conjunta aquellos programas mencionados anteriormente que sean de interés para el portero.

6. Resultados

Obtuvimos los resultados esperados al poner en marcha el planteamiento inicial con todos los aspectos contemplados en el desarrollo, siempre y cuando la cancha fuese colocada en un cuarto oscuro con bajos niveles de luz natural que no obstaculicen la detección precisa de la pelota.

El tirador detecta con un grado acepta7ble de exactitud la ubicación inicial de la pelota, siendo capaz de acomodarse tal y como se planteó que lo hiciera. Aunque el tiro que realiza tiene una potencia mucho menor a la que hubiésemos deseado, a pesar de hacerlo con la mayor potencia disponible para el robot, la pelota encuentra su camino para llegar a gol, ser detenida por el portero o salir desviada.

Los resultados obtenidos nos hicieron ver que no era necesario contemplar una aleatoriedad en la programación del tiro para aumentar la posibilidad de goles, ya que las leves variaciones entre uno y otro tiro originadas por la superficie de la cancha y el ruedo de la pelota, además de que los robots no se acomodan exactamente de la misma manera a pesar de siempre iniciar su movimiento en un mismo punto, le otorgan al planteamiento la variabilidad requerida.

3.22: Momento de tiro

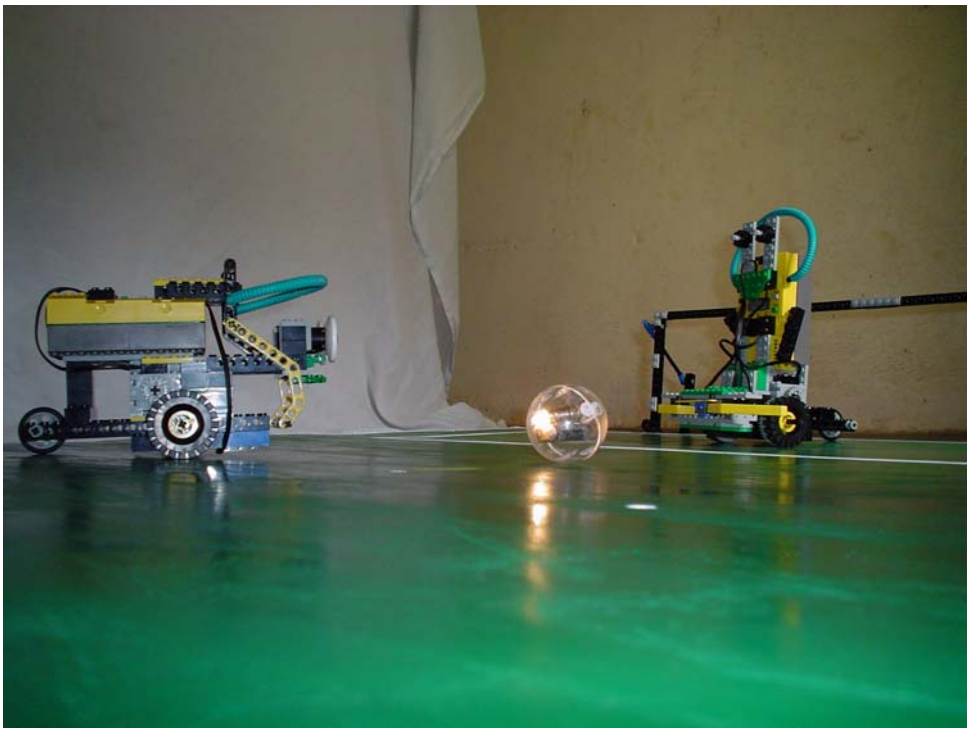

Figura 3.23: Pelota en juego

7. Código de programas (comentado)

7.1. Buscar pelota

```
// El tirador busca en cual de los tres puntos de tiro preestablecidos
// se encuentra la pelota mediante la detección de su luz. Se aproxima
// a la pelota hasta una distancia a la cual el umbral de luz rebasa
// cierto valor.

#define gira_der(s,t)
SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A);OnRev(OUT_C);Wait(t);
#define gira_izq(s,t)
SetPower(OUT_A+OUT_C,s);OnRev(OUT_A);OnFwd(OUT_C);Wait(t);
#define adelante(s,t)
SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A+OUT_C);Wait(t);
#define atras(s,t)
SetPower(OUT_A+OUT_C,s);OnRev(OUT_A+OUT_C);Wait(t);
```

```
int ban,indicador;
int i,max;
//si indicador es = 1 tiro desde la izquierda
//si indicador es = 2 tiro desde el centro
//si indicador es = 3 tiro desde Derecha

task main()
{
  SetSensor(SENSOR_1,SENSOR_TOUCH);
  SetSensor(SENSOR_3,SENSOR_TOUCH);
  SetSensor(SENSOR_2,SENSOR_LIGHT);
  max=0;
  ban=0;
  //si ban == 0 entonces gira a la izquierda
  //si ban == 1 entonces gira a la Derecha
  if (SENSOR_2 > 45){ ban = 1; indicador = 2;}
  Wait (50);

  while (max < 40  && ban == 0){
      gira_izq(1,1);
      max+=1;
      if (SENSOR_2 > 45){ ban = 1; indicador = 1;}
  }
  Off(OUT_A+OUT_C);
  Wait (50);
  max =0;
  while(ban == 0 && max < 80) {
       gira_der(1,1);
       max +=1;
       if (SENSOR_2 > 45){ ban = 1; indicador = 3;}
  }
      Off(OUT_A+OUT_C);
      OnFwd(OUT_A+OUT_C);
      until(SENSOR_2 >= 65);
      Off(OUT_A+OUT_C);
}
```

7.2. Colocación y tiro

```
// El tirador se coloca para cobrar el penal según cual de los tres
// puntos de tiro posibles sea indicado mediante una función aleatoria.
// Después de que se ha colocado el robot, se realiza el tiro a máxima
```

```
// potencia y el motor detiene su marcha.

#define gira_der(s,t)
SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A);OnRev(OUT_C);Wait(t);
#define gira_izq(s,t)
SetPower(OUT_A+OUT_C,s);OnRev(OUT_A);OnFwd(OUT_C);Wait(t);
#define adelante(s,t)
SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A+OUT_C);Wait(t);
#define atras(s,t)
SetPower(OUT_A+OUT_C,s);OnRev(OUT_A+OUT_C);Wait(t);
#define _360 400
#define _180 200
#define _90  100
#define _45  50
int dir;
//si dir == 0 entonces pelota a la izquierda del punto de partida
//si dir == 1 entonces pelota al frente del punto de partida
//si dir == 2 entonces pelota a la derecha del punto de partida

task main()
{
  SetSensor(SENSOR_1,SENSOR_TOUCH);
  SetSensor(SENSOR_3,SENSOR_TOUCH);
  SetSensor(SENSOR_2,SENSOR_LIGHT);
  dir=Random(2);
  if (dir==0){     //  PELOTA a IZQUIERDA del punto de partida
     atras(1,20);
     gira_der(3,_45);
     gira_der(3,_90);
     atras(1,200);
     gira_izq(3,_45);
     SetOutput(OUT_A+OUT_C,OUT_OFF);
  }

  if (dir==1){    // PELOTA al FRENTE del punto de partida
     int ban;
     ban=Random(2);
     if (ban==0)  // colocar a la izquierda
     {
      atras(1,20);
      gira_der(3,_90);
      atras(1,90);
      gira_izq(3,_45);
```

```
     SetOutput(OUT_A+OUT_C,OUT_OFF);
    }
    if (ban==1) // colocar de frente
    {
     atras(1,20);
     SetOutput(OUT_A+OUT_C,OUT_OFF);
    }
    if (ban==2) // colocar a la derecha
    {
     atras(1,20);
     gira_izq(3,_90);
     atras(1,90);
     gira_der(3,_45);
     SetOutput(OUT_A+OUT_C,OUT_OFF);
    }
  }

  if (dir==2){     //  PELOTA a DERECHA del punto de partida
    atras(1,20);
    gira_izq(3,_45);
    gira_izq(3,_90);
    atras(1,200);
    gira_der(3,_45);
    SetOutput(OUT_A+OUT_C,OUT_OFF);
  }

  // Hace el tiro
  adelante(8,100);
  SetOutput(OUT_A+OUT_C,OUT_OFF);
}
```

7.3. Buscar la pelota y tirar

```
// Combina los programas 1. BUSCAR PELOTA y 2. COLOCARSE Y TIRAR en uno

#define gira_der(s,t)
SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A);OnRev(OUT_C);Wait(t);
#define gira_izq(s,t)
SetPower(OUT_A+OUT_C,s);OnRev(OUT_A);OnFwd(OUT_C);Wait(t);
#define adelante(s,t)
SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A+OUT_C);Wait(t);
```

```
#define atras(s,t)
SetPower(OUT_A+OUT_C,s);OnRev(OUT_A+OUT_C);Wait(t);
#define _360 400
#define _180 200
#define _90  100
#define _45  50
int dir;
int i,ban,max;
//si dir == 0 entonces pelota a la izquierda del punto de partida
//si dir == 1 entonces pelota al frente del punto de partida
//si dir == 2 entonces pelota a la derecha del punto de partida

task main()
{
  SetSensor(SENSOR_1,SENSOR_TOUCH);
  SetSensor(SENSOR_3,SENSOR_TOUCH);
  SetSensor(SENSOR_2,SENSOR_LIGHT);

  // INICIA LA BUSQUEDA DE LA PELOTA
    max=0;
    ban=0;
    if (SENSOR_2 > 45){ ban = 1; dir = 2;}
    Wait (50);
    while (max < 40  && ban == 0){
      gira_izq(1,1);
      max+=1;
      if (SENSOR_2 > 45){ ban = 1; dir = 1;}
  }
  Off(OUT_A+OUT_C);
  Wait (200);
  max=0;
  while(ban == 0 && max < 80) {
       gira_der(1,1);
       max +=1;
       if (SENSOR_2 > 45){ ban = 1; dir = 3;}
  }
  Off(OUT_A+OUT_C);
  SetPower(OUT_A+OUT_C,2);
  OnFwd(OUT_A+OUT_C);
  until(SENSOR_2 >= 65);
  Off(OUT_A+OUT_C);
  //trae  el valor de dir
  //si es = 1 tiro desde la izquierda
```

```
//si es = 2 tiro desde el centro
//si es = 3 tiro desde Derecha

// DE ACUERDO A LA DIRECCION EN QUE SE ENCONTRO A LA PELOTA,
// SE COLOCA PARA TIRAR
Wait(50);
if (dir==1){     //  PELOTA a IZQUIERDA del punto de partida
   atras(1,20);
   gira_der(3,_45);
   gira_der(3,_90);
   atras(1,200);
   gira_izq(3,_45);
   SetOutput(OUT_A+OUT_C,OUT_OFF);
}

if (dir==2){    // PELOTA al FRENTE del punto de partida
   int ban;
   ban=1;         // siempre tirara de frente
   if (ban==0)  // colocar a la izquierda
   {
    atras(1,20);
    gira_der(3,_90);
    atras(1,90);
    gira_izq(3,_45);
    SetOutput(OUT_A+OUT_C,OUT_OFF);
   }
   if (ban==1) // colocar de frente
   {
    atras(1,20);
    SetOutput(OUT_A+OUT_C,OUT_OFF);
   }
   if (ban==2) // colocar a la derecha
   {
    atras(1,20);
    gira_izq(3,_90);
    atras(1,90);
    gira_der(3,_45);
    SetOutput(OUT_A+OUT_C,OUT_OFF);
   }
}

if (dir==3){     //  PELOTA a DERECHA del punto de partida
   atras(1,20);
```

```
    gira_izq(3,_45);
    gira_izq(3,_90);
    atras(1,200);
    gira_der(3,_45);
    SetOutput(OUT_A+OUT_C,OUT_OFF);
  }

  // TIRA LA PELOTA
  adelante(10,100);
  SetOutput(OUT_A+OUT_C,OUT_OFF);
}
```

7.4. Comunicar situación de tiro

```
// El tirador indica al portero que está en posición de tirar un penal
// y le señala en cual de los puntos de cobro preestablecidos se
// encuentra la pelota.

int dir;
//si dir == 0 entonces pelota a la izquierda del punto de partida
//si dir == 1 entonces pelota al frente del punto de partida
//si dir == 2 entonces pelota a la derecha del punto de partida

task main()
{
  SetSensor(SENSOR_1,SENSOR_TOUCH);
  SetSensor(SENSOR_3,SENSOR_TOUCH);
  SetSensor(SENSOR_2,SENSOR_LIGHT);

  //Valor de dir
  //si es = 1 tiro desde la izquierda
  //si es = 2 tiro desde el centro
  //si es = 3 tiro desde Derecha

  dir=Random(2)+1;   // Genera al azar un valor de dir entre 1 y 3
  SendMessage(dir);  // Envía dir al portero, indicándole un punto
                     // de tiro según el valor de esta variable
}
```

7.5. Buscar detener pelota

```
// El portero intenta detener la pelota una vez que detecta su
// aproximación mediante el sensor de luz. Detecta si fue capaz o no
// de evitar el gol y le comunica el resultado al tirador.

#define gira_der(s,t)
SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A);OnRev(OUT_C);Wait(t);
#define gira_izq(s,t)
SetPower(OUT_A+OUT_C,s);OnRev(OUT_A);OnFwd(OUT_C);Wait(t);
#define adelante(s,t)
SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A+OUT_C);Wait(t);
#define atras(s,t)
SetPower(OUT_A+OUT_C,s);OnRev(OUT_A+OUT_C);Wait(t);
#define __NOTETIME   10
#define __WAITTIME   12
#define _360 400
#define _180 200
#define _90  100
#define _45  50
int time,time2,time3;
int flag=0,flag2=0;
task main (){
  while(true){      // El portero se mantiene en un ciclo infinito de
    ClearMessage(); // espera hasta el momento en que recibe un mensaje
  until (Message()!= 0);
// Si el mensaje es igual a:
// 1 entonces pelota a la izquierda del punto de partida
// 2 entonces pelota al frente del punto de partida
// 3 entonces pelota a la derecha del punto de partida

    if (Message() == 1){  //Ubicarse para tiro desde el punto izquierdo
      gira_der(1,_45/2); Off(OUT_A+OUT_C);detenertiro(); }
    if (Message() == 2){  //Ubicarse para tiro desde el punto central
      gira_izq(1,_45);Off(OUT_A+OUT_C);detenertiro();}
    if (Message() == 3){ //Ubicarse para tiro desde el punto derecho
      gira_izq(1,_45+_90);Off(OUT_A+OUT_C);detenertiro();}

  }

}
```

```
sub detenertiro(){   // Subrutina con acciones para detener un tiro.
flag=0;              // Es el mismo para cualquier punto de tiro ya que
time=0;              // el portero ya se orientó a él.
  while((time < 200) && (flag==0)){   // Realiza la búsqueda de la
        time = time +1;               // pelota 200 veces antes de
        time2=0;                      // detenerse.
        time3=0;
    while ((flag==0) && (time2 < 25)){  //Busca la pelota a la derecha
          gira_der(1,1);
          time2=time2+1;
               if (SENSOR_2 > 40) {flag=1;} // Si detecta un umbral
    }                                       //  mayor a 40 al girar
                                            //  encontró la pelota

    while ((flag==0) && (time3 < 25)){ //Busca la pelota a la izquierda
       gira_izq(1,1);
        time3=time3+1;
               if (SENSOR_2 > 40) {flag=1;} // Si detecta un umbral
    }                                       //  mayor a 40 al girar
                                            //  encontró la pelota
  }
     Off(OUT_A+OUT_C);        // El portero detiene su giro y sale
     SetPower(OUT_A+OUT_C,2); // en línea recta a enfrentar la pelota
     OnFwd(OUT_A+OUT_C);
     time2=0;
     flag2=0;   // Si se venció el tiempo de espera para detener
                // la pelota gana el tirador y almacena el resultado en
                // la variable flag2
  while (time2 < 200 )
   {
     if ((SENSOR_2 >= 70) || (SENSOR_1 == 1) || (SENSOR_3 == 1 ))
       { Off(OUT_A+OUT_C);  // Si detecta que la pelota está justo
         time2=20;          // enfrente del portero mediante alguno de
         flag2=1;           // sus dos sensores de toque o el de luz
} else time2+=1;       // se detiene y almacena en flag22 su victoria
   }
                     // Envía mensaje al tirador comunicándole
   if (flag2==0){    //  si detuvo la pelota.
       SendMessage(2);   // Si el mensaje es 2 no la detuvo.
   } else SendMessage(1); // Si el mensaje es 1 si la detuvo.

   Off(OUT_A+OUT_C);
   SelectDisplay(flag2); Wait(600);  // Input 1
```

```
}
```

7.6. Movimiento de celebración

```
// El robot ganador hace movimientos de celebración para festejar su
// victoria sobre su contrincante.

int time,time2,time3;
int flag=0,flag2=0;
task main () {
       start musica;           // Inicia la tarea musica
       start celebracion;      // Inicia la tarea celebracion
       Wait(800);
       stop musica;            // Detiene la tarea musica
       stop celebracion;       // Detiene la tarea celebración
}

task musica()        // Toca música para celebrar su victoria
{
 while(true)
 {
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(311,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(349,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(440,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(311,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(349,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(415,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(311,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(349,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(587,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(494,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(415,2*__NOTETIME); Wait(2*__WAITTIME);
```

```
  PlayTone(349,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(294,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(262,2*__NOTETIME); Wait(2*__WAITTIME);
 }
}

task celebracion()   // Realiza movimientos de celebración
{
    OnFwd(OUT_A);
    OnRev(OUT_C);
    Wait(500);
    Off(OUT_A+OUT_C);
}
```

7.7. Tirador

```
// Programa que ejecuta todas las actividades del robot tirador. Integra todos los programas
// anteriores y es el que se ejecuta para cumplir con el planteamiento propuesto.

#define gira_der(s,t)  SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A);OnRev(OUT_C);Wait(t);
#define gira_izq(s,t)  SetPower(OUT_A+OUT_C,s);OnRev(OUT_A);OnFwd(OUT_C);Wait(t);
#define adelante(s,t)  SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A+OUT_C);Wait(t);
#define atras(s,t)     SetPower(OUT_A+OUT_C,s);OnRev(OUT_A+OUT_C);Wait(t);
#define __NOTETIME   10
#define __WAITTIME   12
#define _360 400
#define _180 215
#define _90  110
#define _45  55
int dir;
int i,ban,max;
//si dir == 1 entonces pelota a la izquierda del punto de partida
//si dir == 2 entonces pelota al frente del punto de partida
//si dir == 3 entonces pelota a la derecha del punto de partida

task main()
{
 SetSensor(SENSOR_1,SENSOR_TOUCH);
 SetSensor(SENSOR_3,SENSOR_TOUCH);
 SetSensor(SENSOR_2,SENSOR_LIGHT);
```

```
 // INICIA LA BUSQUEDA DE LA PELOTA
  max=0;
  ban=0;
  if (SENSOR_2 > 40){ban = 1; dir = 2;}   // Pelota en el punto de tiro central

  Wait (50);
  while (max < 40  && ban == 0){
   gira_izq(1,1);
   max+=1;
   if (SENSOR_2 > 45){                    // Pelota en el punto de tiro izquierdo
    ban = 1; dir = 1;
    gira_izq(1,50);    //tiempo de espera para ajuste
   }
  }
 Off(OUT_A+OUT_C);
 Wait (200);
 max=0;
 while(ban == 0 && max < 80) {
     gira_der(1,1);
     max +=1;
     if (SENSOR_2 > 45){ ban = 1; dir = 3;       // Pelota en el punto de tiro derecho
       gira_der(1,50);   //tiempo de espera para ajuste
     }
 }
 Off(OUT_A+OUT_C);                  // Detiene su giro al encontrar la pelota
 SetPower(OUT_A+OUT_C,2);
 OnFwd(OUT_A+OUT_C);              // Se desplaza en línea recta hacia la pelota
 until(SENSOR_2 >= 80);       // Espera que el umbral sea 80, lo cual indica que la pelota
 Off(OUT_A+OUT_C);         // está al frente. Al ocurrir esto detiene su marcha.

//trae  el valor de dir
 //si es = 1 tiro desde la izquierda
 //si es = 2 tiro desde el centro
 //si es = 3 tiro desde Derecha
 // De acuerdo a la direccion en que se encontro a la pelota, se coloca para tirar
 Wait(50);
 if (dir==1){     //  PELOTA a IZQUIERDA del punto de partida
   atras(1,20);
   Off(OUT_A+OUT_C);
   Wait(30);
   gira_der(3,_45);
   Off(OUT_A+OUT_C);
   Wait(30);
```

```
    gira_der(3,_90);
    Off(OUT_A+OUT_C);
    gira_izq(3,45);
    Off(OUT_A+OUT_C);
    Wait(30);
    atras(1,150);
    Off(OUT_A+OUT_C);
    Wait(30);
    gira_izq(3,_45);
    SetOutput(OUT_A+OUT_C,OUT_OFF);
  }

  if (dir==2){    // PELOTA al FRENTE del punto de partida
//    SendMessage(2);
    int ban;
    ban=1;        // siempre tirara de frente
    if (ban==0)  // colocar a la izquierda
    {
     atras(1,20);
     gira_der(3,_90);
     atras(1,90);
     gira_izq(3,_45);
     SetOutput(OUT_A+OUT_C,OUT_OFF);
    }
    if (ban==1) // colocar de frente
    {
     atras(1,20);
     SetOutput(OUT_A+OUT_C,OUT_OFF);
    }
    if (ban==2) // colocar a la derecha
    {
     atras(1,20);
     gira_izq(3,_90);
     atras(1,90);
     gira_der(3,_45);
     SetOutput(OUT_A+OUT_C,OUT_OFF);
    }
  }

  if (dir==3){    //  PELOTA a DERECHA del punto de partida
//    SendMessage(3);
    atras(1,20);
    Off(OUT_A+OUT_C);
```

```
   Wait(30);
   gira_izq(3,_45);
   Off(OUT_A+OUT_C);
   Wait(30);
   gira_izq(3,_90);
   Off(OUT_A+OUT_C);
   gira_der(3,45);
   Off(OUT_A+OUT_C);
   Wait(30);
   atras(1,150);
   Off(OUT_A+OUT_C);
   Wait(30);
   gira_der(3,_45);
   SetOutput(OUT_A+OUT_C,OUT_OFF);
 }

 SendMessage(dir);                    // Envía mensaje al tirador indicándole donde está la pelota
 Wait(50);
 adelante(7,100);                     // Tira la pelota
 SetOutput(OUT_A+OUT_C,OUT_OFF);      // Detiene su marcha
while(true)
{                                     // Espera recibir mensaje del portero que le indique si este
 ClearMessage();                      // detener la pelota
 until (Message() != 0);
 SelectDisplay(Message());

  if (Message()==2){                 // Si la etiqueta del mensaje que recibe es 2 entonces el tirador
       start musica;                 // es el ganador y toca música de celebración mientras se mueve.
       Off(OUT_A+OUT_C);
       start celebracion;
       Wait(800);
       stop musica;
       stop celebracion;
  }
 }
}

task musica()        // Música de celebración
{

 while(true)
 {
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
```

```
  PlayTone(311,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(349,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(440,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(311,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(349,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(415,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(311,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(349,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(587,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(494,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(415,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(349,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(294,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(262,2*__NOTETIME); Wait(2*__WAITTIME);
 }
}

task celebracion()              // Movimiento de celebración
{
   OnFwd(OUT_A);
   OnRev(OUT_C);
   Wait(500);
   Off(OUT_A+OUT_C);
}
```

7.8. Portero

```
// El portero intenta detener la pelota una vez que detecta su
// aproximación mediante el sensor de luz. Detecta si fue capaz o no
// de evitar el gol y le comunica el resultado al tirador.

#define gira_der(s,t)  SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A);OnRev(OUT_C);Wait(t);
```

```
#define gira_izq(s,t)  SetPower(OUT_A+OUT_C,s);OnRev(OUT_A);OnFwd(OUT_C);Wait(t);
#define adelante(s,t)  SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A+OUT_C);Wait(t);
#define atras(s,t)     SetPower(OUT_A+OUT_C,s);OnRev(OUT_A+OUT_C);Wait(t);
#define __NOTETIME   10
#define __WAITTIME   12
#define _360 400
#define _180 200
#define _90  100
#define _45  50
int time,time2,time3;
int flag=0,flag2=0;
task main (){
  while(true){      // El portero se mantiene en un ciclo infinito de
    ClearMessage(); // espera hasta el momento en que recibe un mensaje
  until (Message()!= 0);
// Si el mensaje es igual a:
// 1 entonces pelota a la izquierda del punto de partida
// 2 entonces pelota al frente del punto de partida
// 3 entonces pelota a la derecha del punto de partida

   if (Message() == 1){  //Ubicarse para tiro desde el punto izquierdo
           gira_der(1,_45/2); Off(OUT_A+OUT_C);detenertiro(); }
   if (Message() == 2){  //Ubicarse para tiro desde el punto central
           gira_izq(1,_45);Off(OUT_A+OUT_C);detenertiro();}
   if (Message() == 3){ //Ubicarse para tiro desde el punto derecho
           gira_izq(1,_45+_90);Off(OUT_A+OUT_C);detenertiro();}

  }

}


sub detenertiro(){   // Subrutina con acciones para detener un tiro.
flag=0;              // Es el mismo para cualquier punto de tiro ya que
time=0;              // el portero ya se orientó a él.
  while((time < 200) && (flag==0)){   // Realiza la búsqueda de la
      time = time +1;                 // pelota 200 veces antes de
      time2=0;                        // detenerse.
      time3=0;
   while ((flag==0) && (time2 < 25)){  //Busca la pelota a la derecha
       gira_der(1,1);
       time2=time2+1;
              if (SENSOR_2 > 40) {flag=1;} // Si detecta un umbral
```

```
   }                                                    //  mayor a 40 al girar
                                                       //  encontró la pelota

   while ((flag==0) && (time3 < 25)){          //Busca la pelota a la izquierda
     gira_izq(1,1);
      time3=time3+1;
                if (SENSOR_2 > 40) {flag=1;} // Si detecta un umbral
   }                                                    //  mayor a 40 al girar
                                                       //  encontró la pelota
 }
    Off(OUT_A+OUT_C);       // El portero detiene su giro y sale
    SetPower(OUT_A+OUT_C,2); // en línea recta a enfrentar la pelota
    OnFwd(OUT_A+OUT_C);
    time2=0;
    flag2=0;   // Si se venció el tiempo de espera para detener
                        // la pelota gana el tirador y almacena el resultado en
                        // la variable flag2
  while (time2 < 200 )
   {
    if ((SENSOR_2 >= 70) || (SENSOR_1 == 1) || (SENSOR_3 == 1 ))
      { start musica;   // Empieza a tocar música de celebración
        Off(OUT_A+OUT_C);
        start celebracion;    // Empieza a hacer movimientos de celebración
        Wait(800);            // Espera 8 segundos
        stop musica;         // Detiene acciones de celebración
        stop celebracion;
                                  // Si detecta que la pelota está justo
        time2=20;          // enfrente del portero mediante alguno de
        flag2=1;                  // sus dos sensores de toque o el de luz
}      else time2+=1;               // se detiene y almacena en flag22 su victoria
   }
                                  // Envía mensaje al tirador comunicándole
   if (flag2==0){    //  si detuvo la pelota.
      SendMessage(2);   // Si el mensaje es 2 no la detuvo.
   } else SendMessage(1); // Si el mensaje es 1 si la detuvo.

   Off(OUT_A+OUT_C);
   SelectDisplay(flag2); Wait(600);  // Input 1
}

task musica()         // Música de celebración
{
```

```
 while(true)
 {
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(311,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(349,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(440,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(311,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(349,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(415,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(311,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(349,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(587,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(494,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(415,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(349,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(330,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(294,2*__NOTETIME); Wait(2*__WAITTIME);
  PlayTone(262,2*__NOTETIME); Wait(2*__WAITTIME);
 }
}

task celebracion()              // Movimiento de celebración
{
   OnFwd(OUT_A);
   OnRev(OUT_C);
   Wait(500);
   Off(OUT_A+OUT_C);
}
```

Capítulo 4

CONCLUSIONES

Durante la realización del presente proyecto nos encontramos con diferentes tipos de problemas. Estos, más allá de ser de índole intelectual, fueron más bien problemas de obtención de recursos; ya que elaborar un proyecto de ésta magnitud con los recursos disponibles, durante el corto lapso de tiempo establecido, es un reto para nuestra imaginación e ingenio.

Nuestra investigación inicial nos mostró que existen varios proyectos de robots que juegan fútbol realizados a partir de Lego Mindstorms, pero estos no se apoyan meramente en el kit básico que viene con un robot Lego, sino que lo complementan con otros sensores, cámaras y hardware especializado que facilita enormemente la tarea de hacerlos jugar fútbol. Para este proyecto tuvimos que aprovechar al máximo lo que el software nos podía proporcionar, ya que las carencias en hardware lo demandaban.

Como se pudo observar en el planteamiento inicial, nuestros objetivos eran mucho más ambiciosos, pero se vieron aminorados por la necesidad de optimizar los recursos disponibles. Sin embargo, estamos más que satisfechos con el trabajo realizado y consideramos que la propuesta final es muy buena e interesante, ya que pudimos vencer satisfactoriamente los problemas que tuvimos con el factor más difícil de dominar: la detección certera de los niveles de luz de la pelota en relación con el entorno.

En cuanto a la experiencia de aprendizaje obtenida, pudimos resolver problemas relacionados con umbrales de luz a través del software y no del hardware como inicialmente lo habíamos pensado, al mismo tiempo que reforzamos conocimientos de procesos paralelos a pesar de haber sido este paralelismo causa del replanteamiento inicial del proyecto. Comparando lo que sabíamos antes del proyecto con lo que sabemos al momento de concluirlo, no nos queda duda alguna de que la experiencia práctica obtenida acrecentó nuestras habilidades de análisis y resolución de problemas.

Consideramos que los logros conseguidos con el proyecto podrían tener la siguiente aplicación útil:

- La capacidad programada para detectar y aproximarse a objetos luminosos podría ser de provecho para el desenvolvimiento del robot en entornos

peligrosos o inaccesibles al ser humano, siendo capaz éste de dirigirse a un sitio con determinadas características de iluminación y realizar ciertas actividades preprogramadas. Sería una alternativa mucho más barata que robots más complejos y caros existentes en el mercado.

Además de la útil experiencia de aprendizaje obtenida, del desarrollo de este proyecto obtuvimos interesantes ideas para desarrollos futuros:

- Desarrollar un sistema de comunicación más avanzado para robots Lego, en el cual no se manejen simples etiquetas previamente acordadas, si no que se logre un verdadero paso de mensajes mediante el intercambio de caracteres unitarios que en su conjunto conformarían el mensaje. Ya que la comunicación entre Legos permite manejar 255 etiquetas diferentes, podríamos construir un sistema en el cual cada una de estas etiquetas representaran un carácter del código ASCII, con lo cual un robot podría comunicar cualquier mensaje alfanumérico a otro.
- El siguiente paso para nuestro proyecto sería lograr que éste cumpla con el planteamiento inicial presentado en la etapa de análisis y diseño, sujeto a la posibilidad de contar en el futuro con los suficientes recursos.
- Utilizando conocimientos de electrónica, crear un circuito capaz de incrementar las potencias de las terminales de salida del Lego Mindstorms de tal forma que se pueda operar grandes motores a través del lenguaje NQC.

Tuvimos una gran experiencia con los robots Lego Mindstorms y se la recomendamos a cualquiera que éste cuando menos un poco interesado en ellos. Las posibilidades que estos ofrecen son increíblemente diversas, siendo el mayor requerimiento de todos, como dice el Lego Group en su publicidad, simplemente imaginar.

REFERENCIAS

[AIBO] Sony Corporation. *Sony Aibo*.
http://www.us.aibo.com

[CRIS] *Constructopedia*, Robotics Invention System 2.0, The LEGO Group, Dinamarca, 2000.

[FURB] *The Fabulous Furby and Poo-Chi Fan Site*
http://www.cs.umd.edu/~mstark/furby/welcome.html

[LEGa] The LEGO Group. *LEGO MINDSTORMS*.
http://mindstorms.lego.com

[LEGb] *LEGO MindStorms*.
http://www.robotbooks.com/Lego-Mindstorms.htm

[OVMa] Mark Overmars. *Programming Lego Robots using NQC*. Universidad de Utrecht, Departamento de Ciencias Computacionales, Marzo de 1999.
http://www.cs.uu.nl/people/markov/lego/tutorial.pdf

[OVRO] *Overview of Robotics*, Introduction to Artificial Intelligence, University of the West Indies.
http://athena.uwimona.edu.jm:1104/archives/site/DMCS/compsci/CS33Q/projects/Robotics1.pdf

[OVMb] Mark Overmars. *RCX Command Center*.
http://www.cs.uu.nl/people/markov/lego/rcxcc/index.html

[RCUP] *RoboCup Official Website.*
http://www.robocup.org

[RTIC] Víctor Hugo Pérez Cordero, *La Robótica: conceptos, definiciones e historia*.
http://www.geocities.com/Eureka/Office/4595/robotica.html

[STNO] Stuart Russell y Peter Norvig, *Artificial Intelligence: a modern approach*, Prentice-Hall, Estados Unidos de América, 1995.

[TOYa] *Toy Robotics Initiative*. Carnegie Mellon University, Robotics Institute. http://www-2.cs.cmu.edu/~illah/EDUTOY/index.html

[TOYb] *Insect Telepresence*, Toy Robotics Initiative Projects. http://www-2.cs.cmu.edu/~illah/EDUTOY/projects.html#insect

[ULLA] *ULLANTA Performance Robotics*. http://www-robotics.usc.edu/~barry/ullanta/

ANEXOS

1. Programas del planteamiento inicial

1. Sigue la luz

```
/*   SIGUELUZ
EL BOT SE PONE A GIRAR SOBRE SU MISMO EJE HASTA DETECTAR
LA PRESENCIA DE UNA LUZ CUYA INTENSIDAD SEA MAYOR A
LIM, DESPLAZANDOSE HACIA LA DIRECCION DE LA MISMA
*/

#define LIM 40
#define POT 3

task main()
{
  SetSensor(SENSOR_2,SENSOR_LIGHT);
  SetPower(OUT_A+OUT_C,POT);
  SetDirection(OUT_A+OUT_C,OUT_FWD);
  SetOutput(OUT_A+OUT_C,OUT_ON);
  while (true)
  {
    if (SENSOR_2 < LIM)
    {
      SetDirection(OUT_C,OUT_REV);
      Wait(10);
      until (SENSOR_2 >= LIM);
      SetDirection(OUT_A+OUT_C,OUT_FWD);
    }
  }
}
```

2. Evitar obstáculos

```
// created by Eduardo Diaz, Jorge Garcia Moreno
// 21 de Marzo del 2003
// Programa EVITA ESQUINAS, el robot se desplaza en el entorno
// Puede salir de un callejon sin salida

#define gira_der(s,t)
SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A);OnRev(OUT_C);Wait(t);
#define gira_izq(s,t)
SetPower(OUT_A+OUT_C,s);OnRev(OUT_A);OnFwd(OUT_C);Wait(t);
#define adelante(s,t)
SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A+OUT_C);Wait(t);
#define atras(s,t)
SetPower(OUT_A+OUT_C,s);OnRev(OUT_A+OUT_C);Wait(t);
#define _360 200
#define _180 60
#define _90  30
#define _45  25
int ban;
int i;
//si ban == 0 entonces gira a la izquierda
//si ban == 1 entonces gira a la Derecha

task main()
{

  SetSensor(SENSOR_1,SENSOR_TOUCH);
  SetSensor(SENSOR_3,SENSOR_TOUCH);

   while (true){
    adelante(6,100);
    choquefrontal();
    choqueizq();
    choqueder();
    Wait(10);
   }
}

sub choqueizq(){
  if ((SENSOR_1 == 1) && (SENSOR_3 == 0)){
     atras(1,50);
     gira_der(3,_90);
  }
```

```
    }

    sub choqueder(){
      if ((SENSOR_1 == 0) && (SENSOR_3 == 1)){
         atras(1,50);
         gira_izq(3,_90);
      }
    }



    sub choquefrontal(){
        if ((SENSOR_1 == 1) && (SENSOR_3 == 1)){
         atras(1,150);
         ban= Random(1);
            if (ban==0){
                  gira_izq(3,_90);
                  Wait(50);
                    if (SENSOR_1 == 1){
                        gira_der(3,_180);
                        Wait(25);
                        }
              }else{
                     gira_der(3,_90);
                     Wait(50);
                     if (SENSOR_3 == 1){
                       gira_izq(3,_180);
                       Wait(25);
                       }
                  }
             }
}
```

3. Encontrar la luz

```
    task main()
    {
    while (true){
    if (SENSOR_2 >= 30)
        {
          SetDirection(OUT_A+OUT_C,OUT_FWD);
          until (SENSOR_2 >= 45 );
          SetOutput(OUT_A+OUT_C,OUT_FLOAT);
        }
```

```
}
}
```

4. Evitar obstáculos mientras se busca la luz

```
// created by Eduardo Diaz, Jorge Garcia Moreno
// 21 de Marzo del 2003
// Programa EVITA OBSTACULOS Y BUSCA LUZ, el robot se desplaza
en el
// entorno buscando una luz. Puede salir de un callejon sin
salida

#define gira_der(s,t)
SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A);OnRev(OUT_C);Wait(t);
#define gira_izq(s,t)
SetPower(OUT_A+OUT_C,s);OnRev(OUT_A);OnFwd(OUT_C);Wait(t);
#define adelante(s,t)
SetPower(OUT_A+OUT_C,s);OnFwd(OUT_A+OUT_C);Wait(t);
#define atras(s,t)
SetPower(OUT_A+OUT_C,s);OnRev(OUT_A+OUT_C);Wait(t);
#define _360 200
#define _180 60
#define _90  30
#define _45  25
int ban;
int i;
//si ban == 0 entonces gira a la izquierda
//si ban == 1 entonces gira a la Derecha

task main()
{

  SetSensor(SENSOR_1,SENSOR_TOUCH);
  SetSensor(SENSOR_3,SENSOR_TOUCH);
  SetSensor(SENSOR_2,SENSOR_LIGHT);

       start moverse_en_casa;
       start buscar_umbral;
}

task buscar_umbral()
{
```

```
while (true){
if (SENSOR_2 >= 48)
    { stop moverse_en_casa;
      Wait(300);
      SetDirection(OUT_A+OUT_C,OUT_FWD);
      until (SENSOR_2 >= 60 );
      SetOutput(OUT_A+OUT_C,OUT_FLOAT);
    }
//    Wait(30);
}
}

task moverse_en_casa()
{
  while(true){
    adelante(6,100);
    choquefrontal();
    choqueizq();
    choqueder();
    }
//    Wait(30);
}

sub choqueizq(){
  if ((SENSOR_1 == 1) && (SENSOR_3 == 0)){
     atras(1,50);
     gira_der(3,_90);
  }
}

sub choqueder(){
  if ((SENSOR_1 == 0) && (SENSOR_3 == 1)){
     atras(1,50);
     gira_izq(3,_90);
  }
}

sub choquefrontal(){
    stop buscar_umbral;
    if ((SENSOR_1 == 1) && (SENSOR_3 == 1)){
     atras(1,150);
     ban= Random(1);
        if (ban==0){
```

```
            gira_izq(3,_90);
            Wait(50);
              if (SENSOR_1 == 1){
                  gira_der(3,_180);
                  Wait(25);
                  }
        }else{
              gira_der(3,_90);
              Wait(50);
              if (SENSOR_3 == 1){
                gira_izq(3,_180);
                Wait(25);
                }
            }
      }
      start buscar_umbral;
}
```