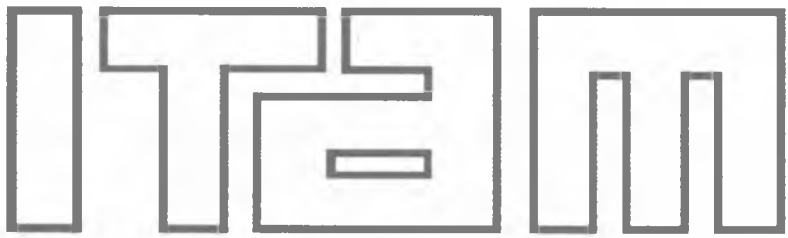
INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO



DISEÑO DE LA ARQUITECTURA Y CONTROL DE TRAYECTORIAS PARA ROBOTS MÓVILES AUTÓNOMOS "SMALL SIZE"

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN TELEMÁTICA

P R E S E N T A

JESÚS GUADALUPE RODRÍGUEZ ORDOÑEZ

MÉXICO, D.F. 2009

A mi madre, abuela y familiares, sabiendo que no existirá forma de agradecer una vida de sacrificio y esfuerzo, quiero que sientan que el objetivo logrado también es de ustedes y que la fuerza que me ayudo a conseguirlo fue su apoyo, cariño y confianza.

A mi asesor Dr. Alfredo Weitzenfeld, en reconocimiento al apoyo brindado durante mi estancia en el Laboratorio de Robótica y durante la realización de esta tesis.

A mis sinodales Dr. Marco Morales y Dr. Fernando Ramírez por su tiempo en las revisiones y sus comentarios.

A los amigos que han estado conmigo durante tantos años y me apoyaron durante mi carrera: Neto, Ferni, Rata, Daniel, Aldo, Mofa, Noe, Yachas, David, Alonso, Jorge, Kuthu, Cartas, Octa, Luis, Paolo, Emir, Gibran, Ruth, etc, etc.

A todos los que hicieron posible este trabajo.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

- CAPÍTULO 1 -

INTRODUCCIÓN

En este capítulo se describen las motivaciones para apoyar el presente proyecto de tesis, así como el problema que pretende resolver. Enseguida se definen los objetivos del proyecto, se acota su alcance, se justifica su realización y se enuncian los trabajos con los que guarda relación. Finalmente se explica la estructura y el contenido de todo el documento.

1.1 Introducción

El continuo aumento en la capacidad de procesamiento de las computadoras ha permitido que la robótica alcance un nivel de madurez bastante elevado. Sin embargo, en áreas como la robótica autónoma, donde el objetivo es desarrollar robots capaces de actuar sin la intervención humana, aún falta mucho camino por recorrer, sobre todo cuando dichos robots tienen que reproducir tareas que para los humanos son muy sencillas, como caminar, correr o manipular objetos. Es por eso que la comunidad científica propuso la creación de un proyecto internacional para promover, a través de competencias entre robots autónomos, la investigación y educación sobre robótica e inteligencia artificial. Así nació "*The Robot World Cup Initiative*", mejor conocido como *RoboCup*.

El objetivo oficial del proyecto *RoboCup* es desarrollar, para el año 2050, un equipo de robots humanoides completamente autónomos capaces de vencer a la entonces selección de futbol campeona del mundo siguiendo sus propias reglas [46].

La iniciativa está dividida en cuatro grandes competencias: *RoboCupSoccer, RoboCupRescue, RoboCupJunior, RoboCup@Home*. Cada una de ellas tiene varias ligas internas dependiendo de la modalidad. Para fines del presente trabajo, nos centraremos en la competencia de fútbol con robots autónomos: *RoboCupSoccer*.

Para conseguir que un equipo de robots juegue futbol es necesario incorporar diversas tecnologías: principios de diseño de robots autónomos, colaboración multiagente, estrategia y procesamiento en tiempo real, sensores para percibir el entorno, etc. Y si a esto añadimos que la actividad se desarrolla en un ambiente dinámico, donde los dos equipos están siempre en movimiento y reaccionando de acuerdo con el desarrollo del juego, entonces, el fútbol proporciona un problema estándar donde una amplia gama de tecnologías e innovaciones pueden ser integradas y examinadas por los investigadores. Por este motivo, y la sencillez de sus reglas de juego, el fútbol fue elegido sobre otros deportes para llevar a cabo las competencias entre robots autónomos.

Las diferentes ligas que conforman *RoboCupSoccer* atienden a la morfología del robot y siguen sus propias reglas:

- *Simulation League*
- *Small Size League*
- *Middle Size League*
- *Standard Plaftorm League*
- *Humanoid League*

El Laboratorio de Robótica del ITAM es parte activa de la iniciativa *RoboCupSoccer* dentro de las ligas *Standard Plaftorm* (SPL) y *Small Size* (SSL) con el equipo *Eagle Knights* [24]. Dicho laboratorio, integrado por alumnos de las Ingenierías del ITAM, promueve el desarrollo de tecnología en las áreas de robótica, inteligencia artificial, visión por computadora y comportamientos autónomos. En adelante, sólo se hará referencia a la liga SSL por ser la que atañe al presente trabajo.

1.2 Descripción del Problema

De acuerdo con las reglas establecidas por *RoboCup* para la liga SSL [51], dos equipos, de 5 robots cada uno, juegan fútbol en una cancha de 5 x 3.5 metros con una pelota de golf en color naranja; las dimensiones de cada robot no deben ser mayores a las de un cilindro de 180mm de diámetro y 150mm de alto. Los robots son totalmente autónomos, es decir, una vez puestos en la cancha prácticamente no hay intervención humana para su control. Lo anterior se logra con la interacción de cuatro sistemas básicos:

- Sistema de Visión: cuyo objetivo es obtener la información del ambiente en que se desarrolla el partido, mediante una o varias cámaras de video colocadas a 4m por encima de la cancha.

- Sistema de Inteligencia Artificial: toma decisiones en cuanto a la estrategia de juego dependiendo del rol asignado a cada robot con base en la información del sistema de visión.

- Módulo de Comunicación: envía la información producida por el sistema de inteligencia a los robots para que sea ejecutada.

- Robots: reciben y ejecutan las instrucciones necesarias para jugar futbol. Para conseguir su objetivo se componen de una estructura mecánica y una arquitectura electrónica orquestada por un procesador central.

El diseño e implementación de la arquitectura electrónica de un nuevo equipo de robots será el problema central que éste trabajo pretende resolver.

Sin embargo, el desarrollo de nuevos robots implica otro problema: es necesario ajustar el software de control motriz a sus nuevas características, tanto mecánicas como electrónicas. Sería deseable que el robot optimizara su comportamiento motriz después de cada cambio mecánico o electrónico y, de esta manera, conseguir que responda con la rapidez y precisión adecuadas para moverse en la trayectoria que se le indica. Desafortunadamente, un modelo analítico de la mecánica del robot no está disponible, de manera que la optimización analítica no es viable. La alternativa al ajuste manual del software de control motriz, tedioso y propenso a errores, es la aplicación de los métodos de aprendizaje.

1.3 Objetivo

Diseñar e implementar la arquitectura electrónica de un nuevo equipo de robots móviles autónomos para la liga *Small Size,* que cumplan con las reglas establecidas por *RoboCup* y puedan participar en competencias oficiales del mismo; así como mejorar el control de trayectorias de los robots ajustando su software de control motriz mediante una técnica de aprendizaje.

1.4 Justificación del Trabajo

Los resultados obtenidos por el equipo *Eagle Knights* en la liga *Small Size* durante el *RoboCup* 2006 [47], celebrado en Bremen Alemania, nos condujeron a la necesidad de contar con nuevos robots que reunieran las características necesarias para ser más competitivos en las próximas competencias. Lo que incluye un diseño electrónico más estable y con mejoras en los sistemas de desplazamiento, pateo y control de trayectorias.

1.5 Alcance del Trabajo

El alcance de este proyecto se centra, en primer lugar, en el diseño e implementación de la arquitectura electrónica para la quinta generación de robots *Small Size* (EK2007) que fue creada en el Laboratorio de Robótica para la competencia *RoboCup* 2007, y a la cual se hicieron pequeñas modificaciones para participar en el *RoboCup* 2008. En segundo lugar, describe un método de aprendizaje por refuerzo, propuesto en [18] y [23], que permite ajustar el software de control motriz de la misma generación de los robots para mejorar su control de trayectorias.

El diseño de la primera solución abarca los módulos de la electrónica que forman parte del robot, como lo muestra la figura 1.1.

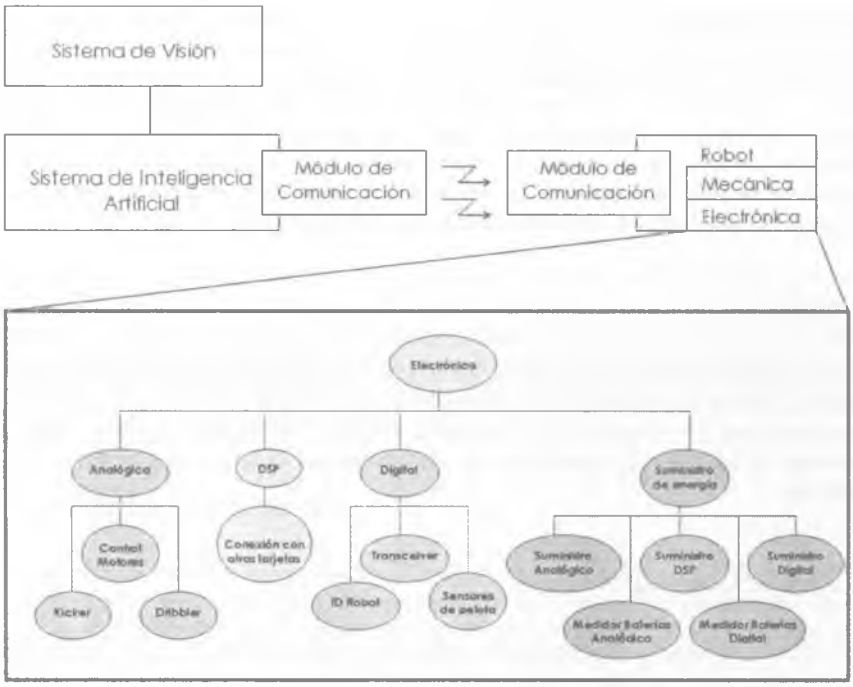


Figura 1.1 - Diagrama del Alcance

Cabe mencionar que en el diseño e implementación de los demás sistemas, generación EK2007, trabajaron los entonces integrantes del Laboratorio: Ernesto Torres Vidal en los Sistemas de Visión e Inteligencia Artificial; Aldo Omar Faba Rodríguez y Christian Chavarría Méndez también en el la arquitectura electrónica de los robots; Misael David Soto Ruíz y David Alejandro Grada Ibáñez en la estructura mecánica; y Luis Alfredo Martínez Gómez que colaboró en todos los sistemas básicos del equipo SSL transmitiendo su conocimiento de las generaciones anteriores.

1.6 Trabajos relacionados

El Laboratorio de Robótica del ITAM ha enfocado sus esfuerzos para representar dignamente a México en las competencias internacionales, prueba de ello son los trabajos de investigación que alumnos involucrados en el proyecto han desarrollado para solucionar los problemas que se plantean en la liga:

Luis Alfredo Martínez Gómez presentó como tesis el "Sistema de Visión Para el Equipo de Robots Autónomos del ITAM" [28], en la que describe una solución al problema de la localización de los robots basada en la interpretación de una señal de vídeo. Francisco

Moneo Soler colaboró con el equipo desarrollando la tesis "Sistema de Planeación de Alto Nivel para RoboCup" [29]; su aportación forma parte del sistema de inteligencia artificial. Juan Pablo Francois Aragón presentó la tesis "Arquitectura de Visión Local para Robots Móviles del ITAM" [16]. Edgar David Sotelo Iniesta trabajó en el diseño y construcción de las primeras generaciones de robots pequeños que participaron en *RoboCup*, con lo cual desarrolló la tesis "Diseño e Implementación de los Robots F180 del ITAM" [41]. Misael David Soto Ruíz diseñó la estructura mecánica que se utilizó en varias generaciones de robots SSL, misma que documentó en la tesis "Diseño e implementación del sistema de control de pelota" [42]. Finalmente, Ernesto Torres Vidal presentó la tesis "Sistema de Inteligencia Artificial para el Control de Robots Autónomos Small Size" [54], en la cual describe a fondo el sistema de Inteligencia Artificial que complementa la arquitectura de un equipo SSL generación EK2007 y EK2008.

1.7 Método de Trabajo

La metodología para la realización de este proyecto de tesis se divide en cinco etapas que se describen a continuación:

1.- Análisis del problema

Se identificaron las debilidades de la cuarta generación de robots *Small Size* (EK2006) para generar los requerimientos de la quinta (EK2007).

2.- Estudio de posibles soluciones

Se estudió la viabilidad de las soluciones propuestas a los requerimientos planteados para decidir una solución única.

3.- Diseño de la quinta generación de robots móviles autónomos

Se diseñó e implementó un prototipo de la solución propuesta para probarlo, hacer las modificaciones necesarias y liberar una versión final.

4.- Construcción de la quinta generación de robots móviles autónomos

Se construyeron los robots con los que el equipo *Eagle Knights* participó en el *RoboCup* 2007.

5.- Fortalecimiento de la quinta generación de robots móviles autónomos.

Se hicieron algunas modificaciones a los robots EK2007 para participar en el *RoboCup* 2008. Después de la competencia se decidió implementar un método de aprendizaje por refuerzo para mejorar el control de trayectorias de los nuevos robots modificados.

6.- Evaluación de resultados

Se identificaron los parámetros de desempeño y evaluación para efectuar un análisis de las soluciones propuestas que guíen al desarrollo nuevos avances en los robots.

1.8 Estructura del Documento

En el segundo capítulo se expondrá una visión general de la arquitectura de un equipo SSL y se presentará el marco teórico que sirve como base para diseñar los sistemas básicos de un robot *Small Size*, además se describirán los principios fundamentales de una técnica denominada: aprendizaje por refuerzo. En el capítulo tres se establecen los requerimientos de la nueva generación de robots. El capítulo cuatro abordará el diseño de los robots EK2007, describirá las modificaciones hechas a los mismos para participar en el *RoboCup* 2008 y analizará el método de aprendizaje por refuerzo que permite a los robots ajustar su software de control motriz para mejorar el control de trayectorias, mientras que la implementación del diseño electrónico y del método de aprendizaje se detallará en el quinto apartado. En el sexto apartado se evaluarán los resultados obtenidos para finalmente establecer las conclusiones y las líneas futuras del proyecto en el séptimo y último capítulo.

- CAPÍTULO 2 -

MARCO TEÓRICO

En la primera parte de este capítulo se expone una visión general de los sistemas básicos para implementar un equipo de robots *Small Size*, que cumplan con las reglas establecidas por *RoboCup* y que puedan participar en las competencias oficiales del mismo. En seguida se elabora un análisis de la arquitectura de los robots, el cual esta basado en los componentes básicos que les permiten jugar futbol. Finalmente se efectúa una introducción al aprendizaje por refuerzo, ya que es una de las técnicas más aplicadas y con más futuro en el campo de la robótica.

2.1 La Liga Small Size

Como se mencionó en el primer capítulo, *Small Size League* (en adelante SSL) es una de las divisiones de la categoría *RoboCupSoccer*, en la cual se desarrollan competencias de fútbol con robots autónomos. Particularmente, SSL se centra en el problema de la cooperación multi-agente y el control, en un entorno muy dinámico con un sistema híbrido centralizado [51].

Sin lugar a dudas, en ésta liga se desarrollan los partidos más dinámicos e intensos de toda la competencia, debido a la velocidad de juego considerada en relación con las dimensiones del campo y la velocidad de desplazamiento de los robots. Los robots SSL, en promedio, llegan a moverse a más de 2 m/s y la pelota llega a alcanzar una velocidad de hasta 15 m/s en algunos casos extraordinarios.

Las dimensiones de cada robot no deben ser mayores a las de un cilindro de 180mm de diámetro y 150mm de alto. Una vez puestos en la cancha los robots son totalmente autónomos, es decir, no hay intervención humana para su control.

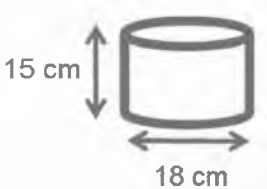


Figura 2.1 - Dimensiones máximas de un robot SSL

En un partido SSL, dos equipos de 5 robots cada uno juegan fútbol en una cancha de 5m de largo por 3.5m de ancho con una pelota de golf en color naranja. Los partidos tienen una duración de 30 minutos divididos en dos tiempos de 15 minutos cada uno. La figura 2.2 muestra las dimensiones del campo de juego autorizadas para la competencia *RoboCup* 2007 [50].

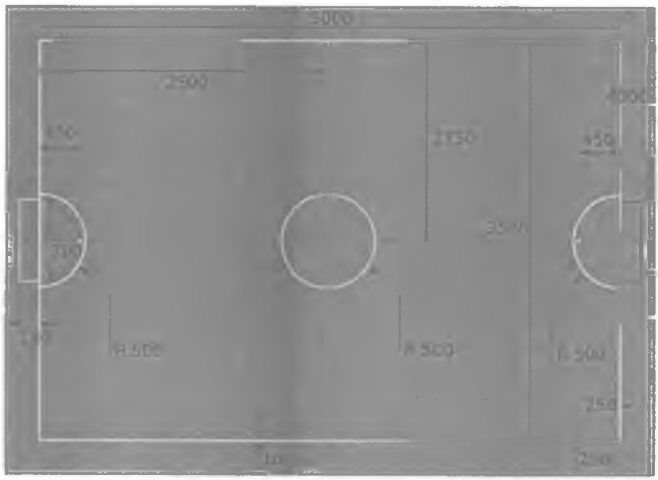


Figura 2.2 - Dimensiones de la cancha de juego para el *RoboCup* 2007

2.2 Arquitectura de un equipo SSL

En general, la arquitectura típica de un equipo SSL consta de cinco sistemas básicos: Sistema de Visión (SV), Sistema de Inteligencia Artificial (IA), Sistema de Control del árbitro (CA), Módulo de Comunicación y los Robots. La figura 2.3 muestra un diagrama de la arquitectura típica.

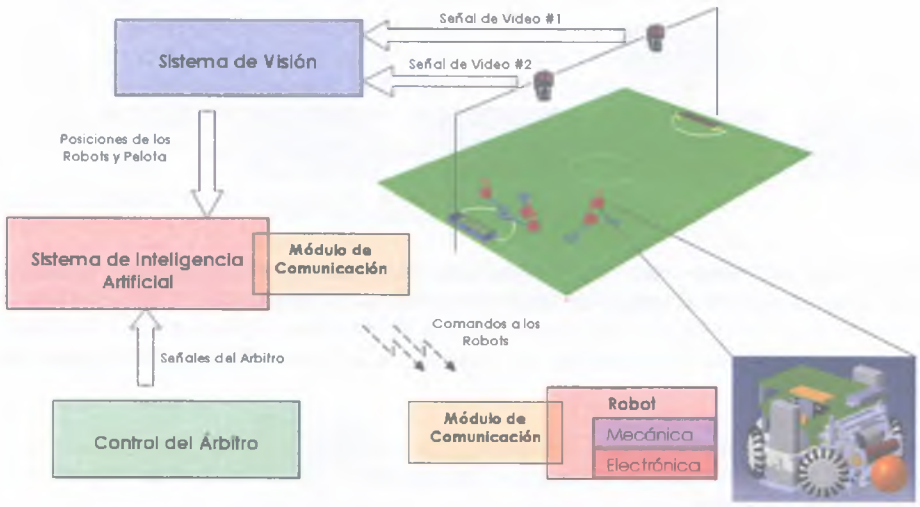


Figura 2.3 - Arquitectura de un Equipo SSL

2.2.1 Sistema de Visión

El objetivo de este sistema es obtener la información del ambiente en que se desarrolla el partido, principalmente la posición de los robots y la pelota, mediante una o varias cámaras de video colocadas a 4m por encima de la cancha.

El sistema de visión es la única fuente de retroalimentación en toda la arquitectura. Este sistema debe ser lo suficientemente robusto para compensar cualquier posible error, como pueden ser la variabilidad en la intensidad de la luz o la proyección de sombras en la cancha de juego. Si la información que proporciona el sistema de visión es incorrecta, el rendimiento global del equipo se verá seriamente afectado [52]. Las principales tareas del sistema de visión son:

- Capturar la imagen de la cancha, video en tiempo real, a través de 2 cámaras que se montan 4m por encima de la cancha
- Reconocer la ubicación de los parches de colores que tienen los robots en la cubierta.

- Identificar y registrar posición y orientación de los robots del nuestro equipo.
- Registrar la posición de los robots del equipo contrario.
- Transmitir la información al sistema de IA.
- Adaptarse a las diferentes condiciones de luz.

Para cumplir con sus tareas, éste sistema cuenta con diversos módulos como se muestra en la Figura 2.4:

Figura 2.4 - Módulos del Sistema de Visión

Calibración por color: Es una herramienta que estabiliza los umbrales de cada componente de acuerdo al espacio de color de los objetos de interés.

Segmentación: Separa cada píxel de las imágenes que pertenecen a los de objetos de interés.

Reconocimiento e identificación: Selecciona las regiones que se ajustan mejor a los objetos buscados. Se busca en la imagen el parche central en la cubierta del robot y los parches alrededor de ése.

Una vez que se ha reconocido e identificado al robot, el siguiente paso es obtener su localización y orientación dentro de la cancha, para esto se utilizan los módulos que se muestran en la Figura 2.5:

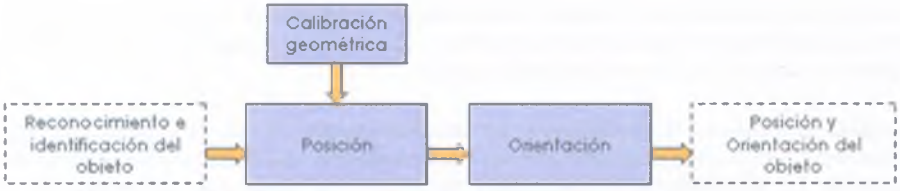


Figura 2.5 - Módulos para la localización

Calibración Geométrica: Permite obtener un sistema de coordenadas correspondientes a las medidas reales en el campo (en el sistema se manejan en milímetros) y la coordenada en píxeles que se lee del buffer de cada una de las imágenes. Además permite resolver los problemas ocasionados por la distorsión del lente gran angular y la intersección existente entre ambas cámaras en la región de la media cancha.

Posición: Calcula la posición de todos los objetos en el campo. Usa parámetros de la cámara que se obtuvieron en el módulo de calibración geométrica con una imagen sin distorsión.

Orientación: Calcula la orientación de los robots de nuestro equipo con ayuda de los parches auxiliares colocados en la cubierta del robot.

Cuando se logra obtener la posición y orientación de todos los robots, en el caso de los robots del equipo contrario sólo la posición, es necesario transmitir dicha información al Sistema de Inteligencia Artificial (IA). Generalmente, cada sistema se ejecuta en una computadora diferente por lo que es necesario utilizar una conexión directa de red (UDP) a través de un cable cruzado.

2.2.2 Sistema de Inteligencia Artificial

El objetivo central de este sistema es tomar decisiones en cuanto a la estrategia de juego dependiendo del rol asignado a cada robot con base en la información del sistema de visión (posición y orientación de los robots) [52]. Este sistema está formado por los módulos que aparecen en la Figura 2.6:

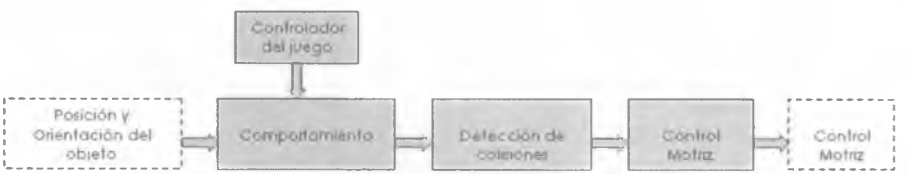


Figura 2.6 - Módulos del Sistema de Inteligencia Artificial

Comportamiento: El módulo de comportamiento o de Inteligencia Artificial, como también se le conoce, recibe la posición en la que se encuentran los robots y la pelota, los ángulos de orientación de los robots de nuestro equipo, el estado de juego (proveniente de un árbitro), y la dirección de la portería contraria. Con esta información el sistema calcula la posición futura y la acción que deben ejecutar los robots para seguir una estrategia durante el partido, basada en el rol asignado a cada uno de ellos (normalmente se emplean tres roles: portero, defensa y delantero).

Detección de colisiones o evasión de obstáculos: Este módulo recibe un vector de movimiento en el que se desplazará el robot, así como la posición de la pelota, y regresa un nuevo vector para evitar que el robot se impacte contra otros cuando ejecuta su movimiento.

Control Motriz: Calcula el vector de movimiento en el cual desea que se desplace el robot.

Controlador del juego: recibe todos los comandos del árbitro y regresa un estado de juego.

2.2.3 Sistema de Control del Árbitro

Los partidos de la liga SSL son comandados por un árbitro humano y un asistente del árbitro. Como en un partido de fútbol normal, el árbitro se encarga de vigilar que el partido transcurra de acuerdo con las reglas establecidas utilizando un silbato y su voz para hacer las indicaciones correspondientes. Mientras tanto, el asistente recibe las indicaciones del árbitro y opera un sistema que se encarga de controlar el estado de juego llamado *Referee-box*. Por ejemplo, si el árbitro indica que salió la pelota, entonces el asistente cambia el estado de juego a *Kickoff* y el sistema de control del árbitro envía una señal de *Kickoff* a los sistemas de IA de los equipos que disputan el partido. El sistema permite ingresar estos comandos y generar algunos automáticamente como combinación de dos botones consecutivos. Por ejemplo *KickoffReady* a partir de *Kickoff* seguido de *Ready*. La interpretación de todas las combinaciones de comandos para control y reinicio de juego se muestran en la Figura 2.7:

Figura 2.7 - Diagrama de estados del Referee-box

2.2.4 Módulo de Comunicación

La principal función de éste módulo es construir paquetes, en los cuales las instrucciones producidas por el sistema de IA son enviada a los robots para que estos las ejecuten. La información que se envía a cada robot consta del vector de movimiento en que éste se desplazará, la velocidad angular respectiva y las decisiones acerca del control y pateo de la pelota. La comunicación entre el sistema de IA y los robots es inalámbrica y para ello se utiliza un dispositivo que combina un transmisor y un receptor que comparten el mismo canal de radio frecuencia, llamado *Transceiver*.

2.2.5 Robots

Su función consiste en recibir y ejecutar las instrucciones necesarias que les permitan jugar fútbol. Para lograrlo deben ofrecer la siguiente funcionalidad básica [41]:

1. Recibir la información enviada por IA.
2. Procesar y ejecutar la información recibida.
3. Capacidad de desplazamiento dentro de la cancha.
4. Golpear la pelota para enviar pases y marcar goles.
5. Controlar la pelota, de modo que se puedan desplazar sin perderla.

Cada función es posible gracias a la interacción de uno o varios sistemas: un Sistema de Comunicación y uno de Identificación permiten conseguir la primera; un Procesador Central hace posible la segunda, ya que interpreta los comandos recibidos por IA y envía señales a los demás sistemas para efectuar alguna acción; para la tercera se ocupa un Sistema de Control Motriz; y, para las dos últimas, un Sistema de Pateo y uno de Control de Pelota, ambos auxiliados por Sensores. Además, cada sistema debe recibir la energía adecuada para su funcionamiento. La figura 2.8 representa un diagrama de los sistemas básicos de un robot *Small Size*, mientras que la figura 2.9 muestra la forma en que interactúan los mismos.

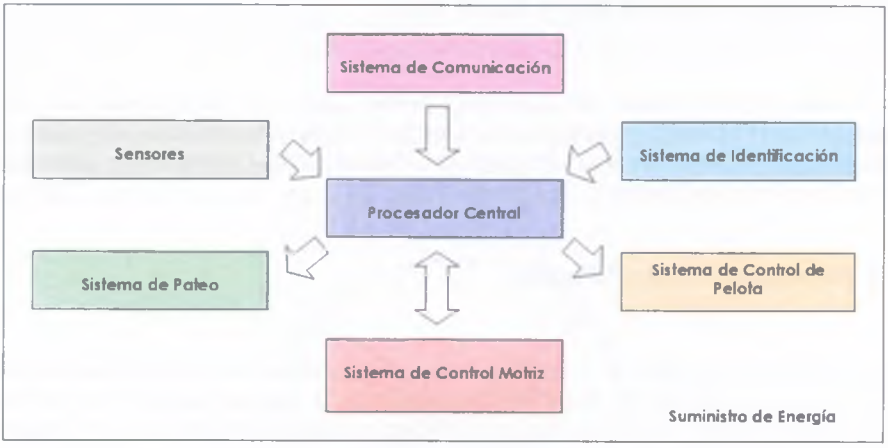


Figura 2.8 - Sistemas básicos de un robot SSL

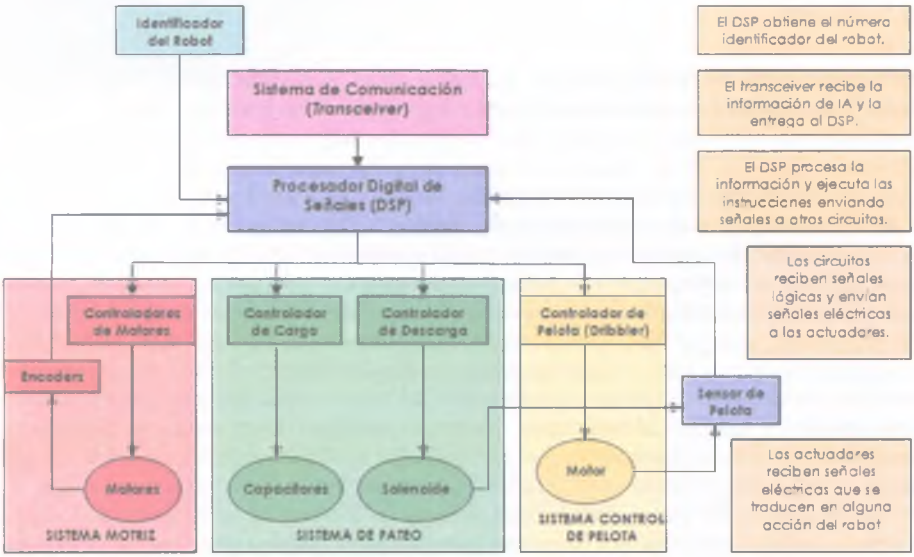


Figura 2.9 - Interacción entre los sistemas de un robot SSL

2.3 Arquitectura de un Robot Small Size

Para entender mejor la interacción entre los sistemas, figura 2.9, las próximas secciones estarán dedicadas a describir los principios en los que se basa cada uno, haciendo énfasis en el los Sistemas de Control Motriz, Comunicación, Procesamiento Central, Pateo, Control de Pelota y Suministro de Energía.

2.3.1 Sistema de Control Motriz

Como primer punto se elabora un análisis de los componentes críticos para la locomoción del robot, ya que, además de permitirle desplazarse en la cancha, guardan una estrecha relación con el control de trayectorias, que también es parte central del presente trabajo. Siendo así, podemos identificar tres componentes críticos para el control motriz: el Modelo Cinemático Omnidireccional, que nos permite desplazar al robot en cualquier dirección sin tener que rotarlo; los puentes H y las Señales PWM, mediante los cuales se puede controlar la dirección y la velocidad de giro de los motores; y los Controladores PID, que compensan el error entre la velocidad deseada y la velocidad real de los motores.

2.3.1.1 Modelo Cinemático Omnidireccional

El desplazamiento omnidireccional, con cuatro motores, ha sido utilizado en nuestros robots *Small Size* desde la tercera generación (EK2005), porque permite que el robot se desplace desde un punto origen hacia cualquier otro punto, sin tener que rotar antes de desplazarse. Adicionalmente, la traslación sobre la ruta deseada se puede combinar con una rotación, de modo que el robot llega a su destino en el ángulo correcto. Este tipo de desplazamiento requiere de ruedas que se puedan mover en más de una dirección, la figura 2.10 muestra el diseño de una rueda omnidireccional.

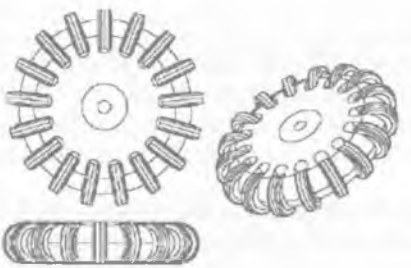


Figura 2.10 - Diseño de una rueda omnidireccional

Los diseños omnidireccionales requieren un procesamiento adicional para convertir las velocidades de rotación y traslación del robot en velocidades individuales para cada motor. Para ello es necesario establecer un modelo cinemático omnidireccional. La descripción detallada de dicho modelo puede consultarse en la sección 2.3.1.4 *Modelo cinemático omnidireccional*, de la tesis: *Diseño e Implementación de los Robots F180 del ITAM* [41], o bien en [35].

Para fines del presente trabajo sólo se analizan las nuevas expresiones que transforman las velocidades en el plano euclidiano en velocidades de motores y viceversa, ya que la disposición de los motores se modifico como en la figura 2.11

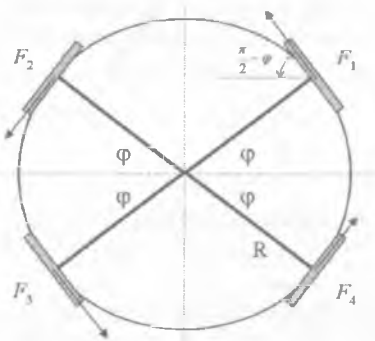


Figura 2.11 - Disposición de las ruedas y distribución de fuerzas

Tal como lo muestra la figura anterior los robots tienen dos ejes de simetría. Llamamos φ al ángulo de las ruedas con respecto al eje horizontal (eje x). Cuando los cuatro motores están activados, obtenemos cuatro fuerzas de tracción F_1, F_2, F_3, F_4, que se suman a una fuerza de translación y un torque de rotación. Cada fuerza de tracción F_i es el torque del motor multiplicado por el radio de la rueda. La suma de las fuerzas depende de la disposición exacta de las ruedas.

2.3.1.2 De Velocidades Euclidianas a Velocidades de Motores y Viceversa

Con ayuda de ciertos sensores se puede calcular la velocidad real de las ruedas, y con el Sistema de Visión es posible obtener la velocidad del robot mientras este se desplaza, así como su velocidad angular. Sin embargo, se tiene que pensar al robot en el espacio euclidiano, calcular su trayectoria en el mismo, y obtener de este la velocidad de cada rueda. Para esto es necesario analizar la geometría del problema tal como se indica en [35]:

Hay que empezar por agrupar la velocidad individual de los cuatro motores en el vector $(v_1, v_2, v_3, v_4)^T$ y las velocidades euclidianas y de rotación del robot en el vector $(v_x, v_y, R\omega)^T$. Por ejemplo, si el robot se mueve según lo determinado por el vector $(1, 0, 0)^1$ significa que se está moviendo sobre el eje x, izquierda o derecha sin rotación.

Cuando el robot se mueve con velocidad de 1 a la derecha, las ruedas giran con velocidad $\sin\varphi$ con el respectivo signo. Esto es fácil de ver en el diagrama de la figura 2.12 si se considera el diseño de una rueda omnidereccional como el de la figura 2.10. La rueda de mayor diámetro proporciona uno de los componentes del movimiento horizontal, es decir $\sin\varphi$, mientras que las pequeñas ruedas periféricas proporcionan el otro componente ortogonal, es decir $\cos\varphi$.

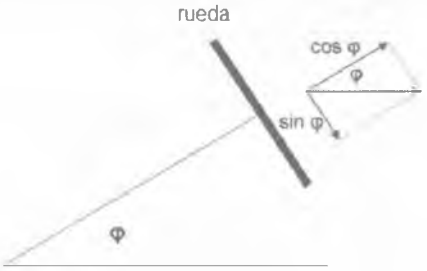


Figura 2.12 - Rotación de la rueda principal y las ruedas periféricas, cuando el robot se mueve sobre el eje x.

El mismo tipo de cálculo se puede hacer cuando el robot se mueve sobre el eje y, al frente o atrás sin rotar. El movimiento de la rueda es entonces la componente $\cos\varphi$. Usando la convención de que la dirección de rotación positiva es la dirección en que gira la rueda

cuando tomamos el motor con la mano derecha y el pulgar apuntando al eje del motor, se obtiene la siguiente expresión para las correspondencias entre las velocidades euclidianas y las velocidades de los motores [35]:

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \begin{pmatrix} -\sin\varphi & \cos\varphi & 1 \\ -\sin\varphi & -\cos\varphi & 1 \\ \sin\varphi & -\cos\varphi & 1 \\ \sin\varphi & \cos\varphi & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ R\omega \end{pmatrix}$$

La matriz en esta expresión, que se denota como D, es la matriz de acoplamiento de velocidades. Si se expresa el vector de velocidades de motores $(v_1, v_2, v_3, v_4)^T$ como m y el vector de velocidades euclidianas como v, se tiene la siguiente expresión:

$$m = Dv$$

Los motores tienen integrado un dispositivo que les permite medir su velocidad en tiempo real con el equipo electrónico a bordo del robot. Para efectos de controlar al robot, deben medirse las velocidades de las ruedas $m = (v_1, v_2, v_3, v_4)^T$ y convertirlas a magnitudes euclidianas $v = (v_x, v_y, R\omega)^T$, es decir, sería deseable invertir la expresión $m = Dv$. Esto no es posible porque, en general, la matriz D no es cuadrada, y por lo tanto, no es invertible. Sin embargo, puede buscarse una matriz D^+ tal que $D^+D = I_3$, donde I_3 es una matriz identidad de 3 x 3. Si tal matriz existe, entonces, dadas las magnitudes euclidianas v, es posible encontrar las correspondientes velocidades de los motores m, que, a su vez, reproduce las magnitudes euclidianas originales. Esto es porque si:

$$m = Dv$$

Entonces

$$D^+m = (D^+D)v = I_3v = v$$

La matriz D^+ existe, y es llamada pseudo-inversa de la matriz D [35]. En el caso especial de un robot con dos ejes de simetría y el mismo ángulo φ para cada eje del motor (como es nuestro caso), se puede escribir una simple expresión de D^+. Suponiendo que el $\sin\varphi$ y el $\cos\varphi$ son diferentes de cero, entonces:

$$D^+ = \frac{1}{4}\begin{pmatrix} -\frac{1}{\sin\varphi} & -\frac{1}{\sin\varphi} & \frac{1}{\sin\varphi} & \frac{1}{\sin\varphi} \\ \frac{1}{\cos\varphi} & -\frac{1}{\cos\varphi} & -\frac{1}{\cos\varphi} & \frac{1}{\cos\varphi} \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Es sencillo comprobar que $D^+D = I_3$. También se puede comprobar que D^+ tiene todas las propiedades de la pseudo-inversa de D: tanto D^+D y DD^+ son simetricas, $D^+DD^+ = D^+$, y $DD^+D = D$

De lo anterior se deriva que, si se aplican ciertas velocidades de motores es posible obtener las velocidades euclidianas y de rotación respectivas, por que:

$$D^+(v_1, v_2, v_3, v_4)^T = D^+(D(v_x, v_y, R\omega)^T) = I_3(v_x, v_y, R\omega)^T = (v_x, v_y, R\omega)^T$$

Resumiendo, para transformar velocidades euclidianas en velocidades de motores se utiliza la expresión:

$$m = Dv$$

Mientras que para transformar velocidades de motores a velocidades euclidianas se usa:

$$v = D^+m$$

2.3.1.3 Los puentes H y las Señales PWM

El sentido de giro de los motores de corriente continua depende de la alimentación que tengan, si es positiva gira en un sentido, y si es negativa, en el otro. La función de los puentes H es hacer que giren en ambos sentidos sin necesidad de tener un voltaje de alimentación negativo [11]. El término "puente H" proviene de la representación gráfica del circuito como se muestra en la figura 2.13.

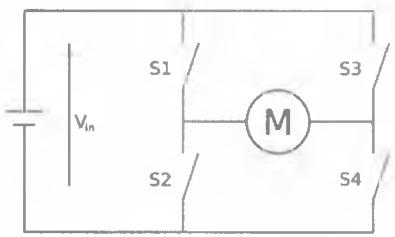


Figura 2.13 - Estructura de un Puente H

Un puente H se construye con 4 interruptores, generalmente implementados con transistores. Cuando los interruptores S1 y S4 están cerrados, y S2 y S3 abiertos, se aplica una tensión positiva en el motor, haciéndolo girar en un sentido. Abriendo los interruptores S1 y S4, y cerrando S2 y S3, el voltaje se invierte, permitiendo el giro del motor en sentido inverso. La figura 2.14 muestra los dos estados básicos del circuito descrito. Con la

nomenclatura usada, los interruptores S1 y S2 nunca podrán estar cerrados al mismo tiempo, porque esto provocaría un corto circuito. Lo mismo sucede con S3 y S4.

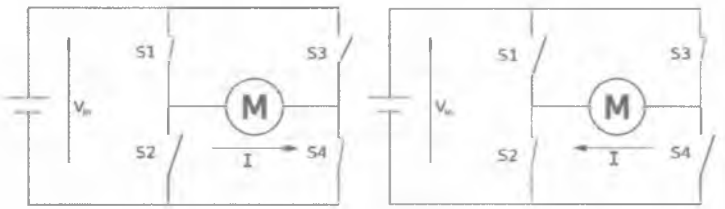


Figura 2.14 - Los dos estados básicos de un Puente H

Una vez determinado el sentido de giro del motor, es necesario calcular la velocidad con la que lo hará, la cuál puede controlarse mediante la disminución del voltaje aplicado al rotor. Esto reduce el torque y la velocidad máxima alcanzable. Sin embargo, hay dos problemas con este enfoque. El primer problema es que una señal digital del procesador debe transformarse en un valor analógico de voltaje para el motor, y el segundo es que con voltajes bajos se pierde torque y podría no ser suficiente para poner una rueda en movimiento al arranque, ya que la fricción es mayor cuando el rotor no está en movimiento que cuando ya lo está.

Para solucionar este problema se utiliza la Modulación por Ancho de Pulso (PWM por sus siglas en inglés *Pulse Width Modulation*). Ésta es una técnica en la que se modifica el ciclo de trabajo de una señal periódica generalmente cuadrada [6]. El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación al período. Matemáticamente:

$$D = \frac{\tau}{T}$$

Donde:

D	es el ciclo de trabajo
τ	es el tiempo en que la función es positiva (ancho del pulso)
T	es el período de la función

La figura 2.15 muestra una señal cuadrada con diferentes ciclos de trabajo. Como puede observarse, entre mayor es el ciclo de trabajo, mayor es el tiempo durante el cuál la función es positiva. Por ejemplo, si el ciclo de trabajo fuera del 100% observaríamos siempre una función positiva de amplitud máxima.

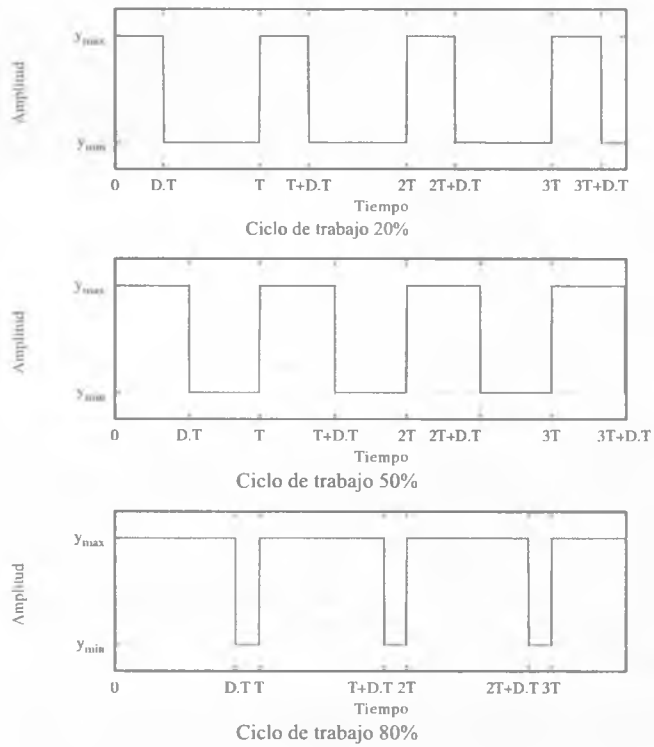


Figura 2.15 - Señal cuadrada con distintos ciclos de trabajo

Para ejemplificar el uso de la modulación PWM supongamos que se quiere reducir la velocidad de un motor a la mitad, en lugar de aplicar al motor un voltaje reducido en 50%, se aplica el voltaje máximo, pero sólo la mitad de las veces. Se selecciona una frecuencia portadora, y sólo en 50% de los ciclos la tensión máxima se aplica. Si sólo el 10% de los ciclos se utilizan para aplicar un voltaje, se dice que la señal PWM esta a 10.

Con la señal de PWM se logra que el motor se acelere a pasos discretos, es decir, cuando el voltaje máximo se aplica, la velocidad del motor aumenta a lo largo de la curva exponencial, pero cuando el voltaje se interrumpe, el motor desacelera. El efecto total es una serie de aceleraciones y desaceleraciones, que producen un efecto similar a la reducción del voltaje aplicado. Sin embargo, el máximo torque posible para cada velocidad siempre se aplica. Por lo tanto los motores pueden girar a bajas velocidades y se pueden controlar mejor.

La figura 2.16 muestra lo que ocurre cuando se aplica una señal PWM con 100%, 90%, 50% y 10% del ciclo. Las líneas sólidas muestran la velocidad promedio producida por cada señal PWM.

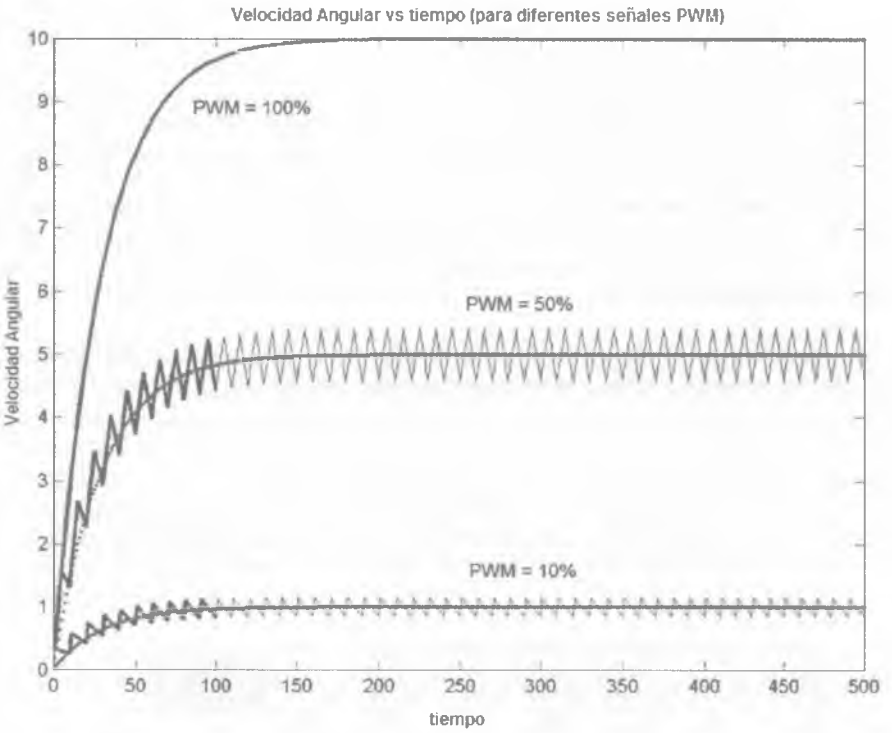


Figura 2.16 - Aceleración de un motor DC bajo diferentes señales PWM.

Aunque las señales PWM cumplen muy bien su propósito, aumentar o disminuir la velocidad del motor sin mermar el torque, resulta difícil obtener una velocidad exacta, ya que, por lo general, hay una diferencia entre la velocidad deseada (ideal) y la velocidad obtenida (real). Para disminuir esta diferencia se utilizan los controladores PID.

2.3.1.4 Controlador PID

Un controlador PID (**P**roporcional **I**ntegral **D**irivativo) es un sistema de control que, mediante un actuador, es capaz de mantener una variable o proceso en un punto deseado. Es uno de los métodos de control más frecuentes y precisos dentro de la regulación automática [2].

Para el correcto funcionamiento de un controlador PID se necesita:

1. Un sensor, que determine el estado del sistema.
2. Un controlador, que genere la señal que gobierna al actuador.
3. Un actuador, que modifique al sistema de manera controlada.

El sensor proporciona una señal que representa el *punto actual* en el que se encuentra el proceso o sistema, también llamada *process variable* (PV). El controlador lee una señal externa que representa el valor que se desea alcanzar. Esta señal recibe el nombre de *punto de referencia* o *setPoint* (SP), la cual es de la misma naturaleza y tiene el mismo rango de valores que la señal que proporciona el sensor. El controlador PID resta la señal de *punto actual* a la señal de *punto de referencia*, obteniendo así la señal de *error*, que determina en cada instante la diferencia que hay entre el valor deseado y el valor medido. La señal de error es utilizada por cada una de las 3 componentes de un controlador PID para generar 3 señales que, sumadas, componen la señal que el controlador va a utilizar para gobernar al actuador. La señal resultante de la suma de estas tres señales, se llama *variable manipulada* o *manipulated variable* (MV).

Las tres componentes de un controlador PID son: acción **P**roporcional, acción **I**ntegral y acción **D**iferencial. La figura 2.17 muestra un diagrama a bloques de un controlador PID.

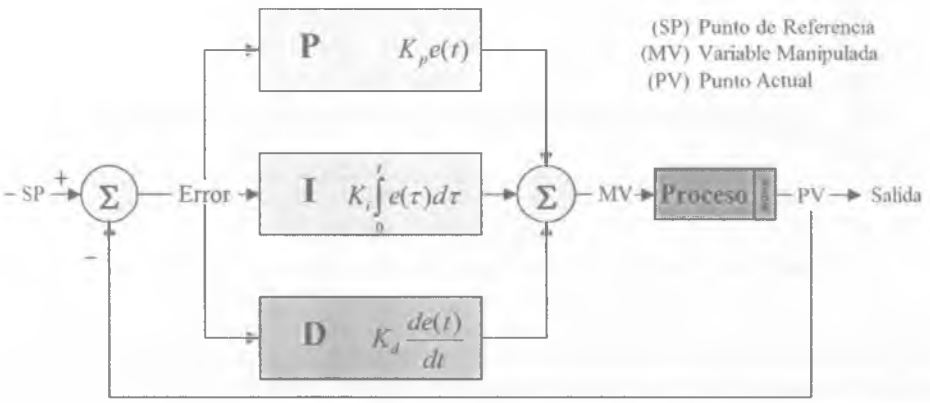


Figura 2.17 - Diagrama a bloques de un controlador PID

2.3.1.4.1 Acción de Control Proporcional

La acción proporcional se utiliza para corregir "el presente", consiste en multiplicar el error por la constante proporcional y el resultado se agrega a la señal de control. Esta componente PID toma un papel importante cuando la señal de *error* es grande, pero su acción se ve mermada con la disminución de dicha señal. Este efecto tiene como consecuencia la aparición de un *error permanente*, que hace que la parte proporcional nunca llegue a solucionar por completo el error del sistema [2]. La parte proporcional está dada por:

$$P_{\text{out}} = K_p e(t)$$

Donde:

- P_{out} : Resultado de aplicar la parte proporcional
- K_p : Constante proporcional
- e : Error = $SP - PV$
- t : Tiempo instantáneo (presente)

La constante proporcional K_p determinará el *error permanente*, siendo éste menor cuanto mayor sea el valor de la constante proporcional. Se pueden establecer valores suficientemente altos en la constante proporcional como para que el *error permanente* sea casi nulo pero, en la mayoría de los casos, estos valores solo serán óptimos en una determinada porción del rango total de control, siendo distintos los valores óptimos para cada porción del rango. Sin embargo, existe también un valor límite en la *constante proporcional* a partir del cual, en algunos casos, el sistema alcanza valores superiores a los deseados. Este fenómeno se llama *sobreoscilación* y, por razones de seguridad, no debe sobrepasar el 30%, aunque es conveniente que la parte proporcional ni siquiera produzca *sobreoscilación.* La Figura 2.18 muestra los cambios en la señal cuando se modifica la *constante proporcional*:

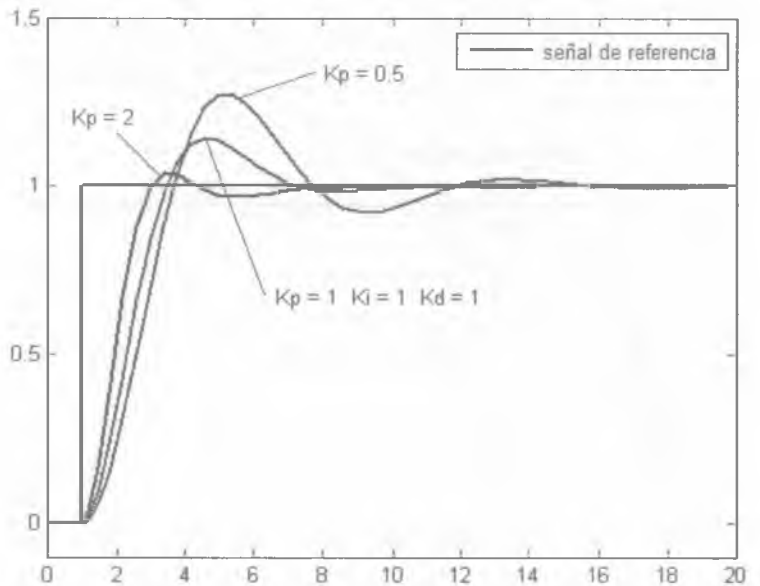


Figura 2.18 - Cambios en la señal al modificar Kp

2.3.1.4.2 Acción de Control Integral

El factor integral permite corregir "el pasado", tiene como propósito disminuir y eliminar el error en estado estacionario, provocado por el factor proporcional. El error se integra sobre un período de tiempo y se multiplica por el factor integral con el propósito de promediarlo. Luego es multiplicado por una constante de integración [2]. La parte integral esta dada por:

$$I_{out} = K_i \int_0^t e(\tau)\, d\tau$$

Donde:

- I_{out} : Resultado de la parte integral
- K_i : Constante de Integración
- e : Error = $SP - PV$
- τ : Tiempo contribuyendo a la parte integral (Pasado)

La Figura 2.19 muestra los cambios en la señal cuando se modifica la *constante de integración*:

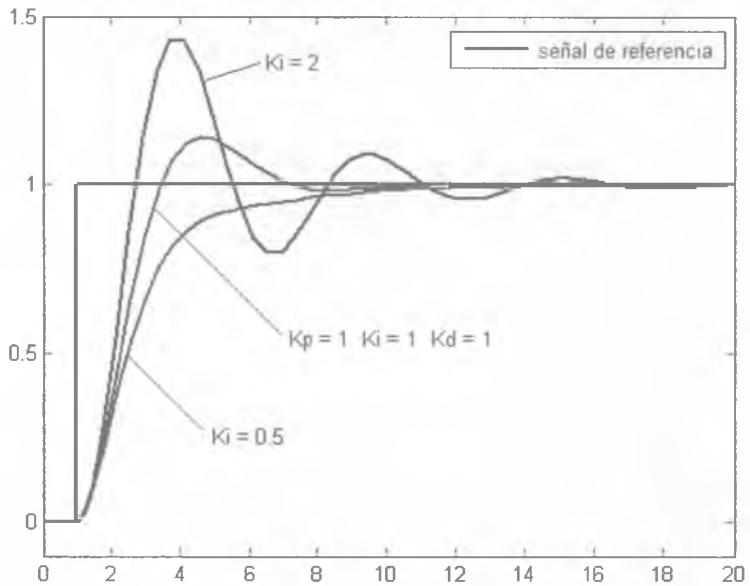


Figura 2.19 - Cambios en la señal al modificar Ki

2.3.1.4.3 Acción de Control Derivativa

La acción derivativa se manifiesta cuando hay un cambio en el valor absoluto del error; si el error es constante, solamente actúan los modos proporcional e integral [2].

La función de la acción derivativa es mantener el error al mínimo, corrigiéndolo proporcionalmente con la misma velocidad que se produce; de esta manera evita que el error se incremente. La parte Diferencial esta dada por:

$$D_{out} = K_d \frac{de}{dt}$$

Donde:

- D_{out} : Resultado de la parte Diferencial
- K_d : Constante derivativa
- e : Error = $SP - PV$
- t : Tiempo

La Figura 2.20 muestra los cambios en la señal cuando se modifica la *constante derivativa*:

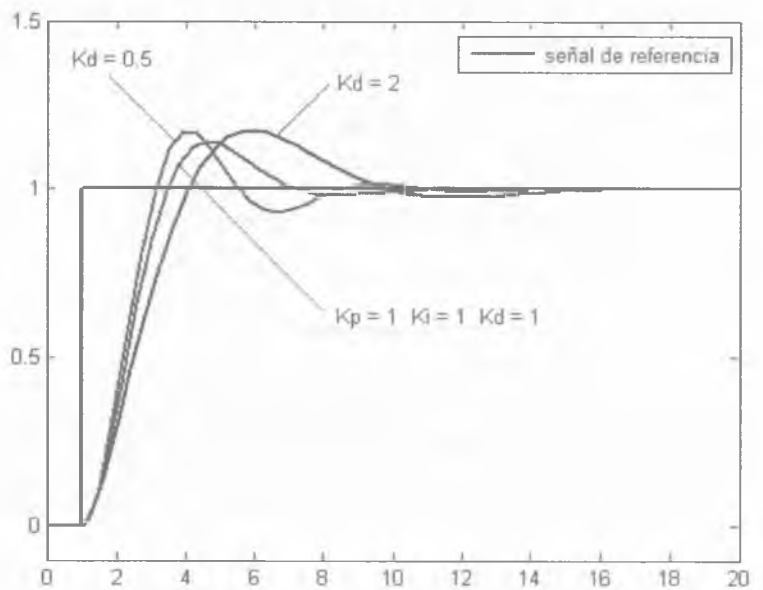


Figura 2.20 - Cambios en la señal al modificar Kd

2.3.2 Sistema de Comunicación

Como se mencionó en la sección 2.2.4, el módulo de comunicación de IA genera paquetes con las instrucciones que desea ejecuten los robots y los envía de forma inalámbrica, entonces, un sistema de comunicación residente en cada robot se encarga de recibir dicha información y entregarla al procesador central, para que éste tome sólo la información que le corresponde. Para este sistema también se utiliza un *Transceiver* por cada robot.

Debido a su gran estabilidad, el sistema de comunicación para los robots EK2007 sigue siendo el mismo que se ha usado desde la generación EK2004 [41], el cuál esta basado en un sistema de comunicación llamado RPC.

2.3.2.1 El RPC

El RPC ("*Radio Packet Controller*") es un sistema de comunicación, diseñado por la empresa *Radiometrix* [34], que requiere de una antena, una fuente de poder de 5V y algún microcontrolador con un puerto bidireccional de 1 byte. El paquete de datos que se desea enviar es recibido por un microcontrolador y alojado en el buffer de datos del RPC origen. El paquete se transmite y cuando es recibido se almacena en el buffer del RPC destino, para que el microcontrolador de éste pueda entregárselo al equipo que lo usará. La figura 2.21 ejemplifica el funcionamiento del RPC.

Figura 2.21 - Funcionamiento del RPC

El RPC utiliza 9 líneas para poder interactuar con algún sistema externo, cuatro líneas son usadas para el control, otras cuatro bidireccionales se utilizan para enviar y recibir los datos y una más de reset para restablecer la señal y reiniciar el controlador. La figura 2.22 muestra las conexiones entre el host y el RPC, mientras que la tabla 2.1 describe las funciones de dichas conexiones [34].

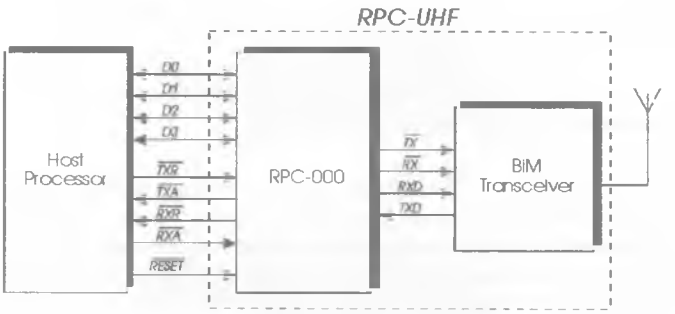


Figura 2.22 - Conexión entre el host y el RPC

Nombre del Pin	Número de PIN	Función	Descripción
TXR	6	TX Request	Solicitud de transmisión de datos desde el HOST hacia el RPC.
TXA	7	TX Accept	Transmisión de datos aceptada hacia el HOST.
RXR	8	RX Request	Solicitud de transferencia de datos del RPC hacia el HOST.
RXA	9	RX Accept	Solicitud de datos aceptada hacia el RPC
D0	2	Data 0	BUS bidireccional de 4 bits. Los datos se transfieren en nibbles, primero el LSN (Nibble menos significativo) y posteriormente el MSN (Nibble más significativo)
D1	3	Data 1	
D2	4	Data 2	
D3	5	Data 3	

Tabla 2.1 - Funciones del bus de datos del RPC

La secuencia para transmitir un byte hacia el RPC (TX download) es asíncrona y debe seguirse el siguiente procedimiento:

1. El HOST baja el nivel de la línea TXR para indicar que iniciará la transferencia.
2. El RPC baja el nivel de la línea TXA para aceptar la transferencia.
3. El HOST coloca un nibble en el BUS de datos.
4. El HOST sube el nivel de la línea TXR para notificar al RPC que el nibble está en el BUS.
5. El RPC sube la línea TXA para indicar que ha aceptado el nibble.

Para enviar un byte se envía primero el LSN y se repiten los pasos para el MSN. Y de manera análoga, para transmitir datos desde el RPC hacia el HOST se siguen los siguientes pasos:

1. El RPC baja el nivel de la línea RXR para solicitar la transferencia.
2. El HOST baja el nivel de la línea RXA para indicar que ha aceptado la transferencia.
3. El RPC coloca en el BUS de datos un nibble.

4. El RPC sube el nivel de RXR para indicar que los datos están en el bus.
5. El HOST lee los datos del bus y sube el nivel del RXA.

2.3.3 Procesamiento Central

La unidad central de procesamiento es la encargada de interpretar los comandos de IA con base en un programa residente en la memoria del mismo, para luego generar las señales que gobiernan a los actuadores del robot. Llamamos actuadores a los dispositivos que traducen señales eléctricas, provenientes del procesador central, en alguna acción concreta del robot, generalmente mecánica. A partir de la generación de robots EK2004 se ha utilizado un Procesador Digital de Señales como procesador central.

Un Procesador Digital de Señales (en adelante DSP, por sus siglas inglés *Digital Signal Processor*) es un sistema basado en un procesador o microprocesador que posee un juego de instrucciones, un hardware y un software optimizados para aplicaciones que requieren operaciones numéricas a muy alta velocidad [7]. Un DSP es único porque procesa señales en tiempo real. Esta capacidad de procesamiento hace a los DSP ideales para aplicaciones que no toleran ningún retardo. Si se tiene en cuenta que pueden trabajar con varios datos en paralelo mediante un conjunto de instrucciones específicas, nos podemos dar una idea de su enorme potencial para este tipo de aplicaciones.

Como todo sistema basado en procesador programable necesita una memoria donde almacenar los datos con los que trabajará y el programa que ejecuta, los cuales son introducidos al hardware a través de un software específico que puede o no manejar lenguajes tanto de alto como de bajo nivel.

2.3.4 Sistema de Pateo

El sistema de pateo es un mecanismo que permite a los robots impulsar la pelota dependiendo de la acción que éste quiera realizar, un golpe fuerte para disparar a gol o uno suave para mandar un pase. La dificultad de este sistema radica en encontrar un dispositivo lo suficientemente pequeño para que quepa en el robot y lo suficientemente poderoso para que la pelota salga impulsada con fuerza. Múltiples propuestas surgieron para resolver el problema utilizando mecanismos con resortes, sistemas de aire comprimido, etc. Sin embargo, a lo largo de la historia del *RoboCup* se ha generalizado el uso de un solenoide con un núcleo metálico [41].

Cuando se hace fluir corriente a través de un solenoide, un campo magnético es generado. En el interior de la bobina hay un núcleo ferromagnético similar a un pistón, llamado émbolo o *plunger*. El campo magnético, entonces, aplica una fuerza a este émbolo, ya sea atrayéndolo o repeliéndolo en nuestro caso; el impulso que toma el *plunger* es aprovechado para golpear la pelota. Cuando el campo magnético se apaga, un resorte devuelve el émbolo a su estado original. La figura 2.23, obtenida de [40], muestra el principio de funcionamiento del solenoide.

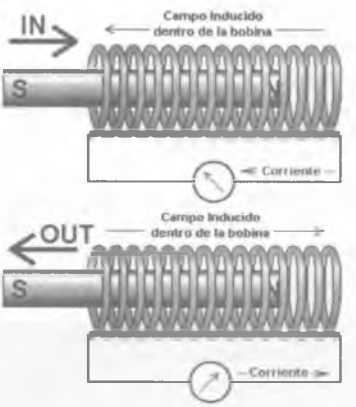


Figura 2.23 - Principio del Solenoide

Ahora bien, para generar un campo magnético capaz de expulsar al émbolo con una fuerza considerable, se requiere hacer fluir por la bobina bastante voltaje y corriente eléctrica (en el orden de las centenas de volts); sin embargo, la única fuente de energía en el robot son sus baterías, por lo tanto se requiere un circuito que permita elevar el voltaje hasta los niveles deseados y posteriormente almacenarlo para disponer de él cuando sea necesario. Para poder almacenar carga de manera temporal se puede hace uso de capacitores. Al proceso de generar alto voltaje y almacenarlo en los capacitores, se le conoce como Etapa de Carga.

A la siguiente la Etapa se le conoce como Descarga, en la cuál el voltaje de los capacitores se envía al solenoide. Es importante que el robot tenga la pelota al momento de la descarga y no se desperdicie energía al ejecutar una descarga en falso, para este motivo la etapa de descarga esta compuesta por otras dos: Sensado y Activación.

En el Sensado se detecta que el robot tenga la pelota; mientras que en la Activación, el procesador central de robot envía una señal para que se active un interruptor de descarga. De esta manera el solenoide se activará solamente si los sensores indican que el robot tiene la pelota y si la etapa de activación así lo pide. La figura 2.24 muestra los componentes básicos de un sistema de pateo con solenoide.

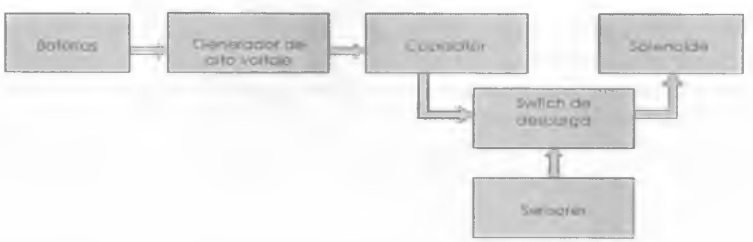


Figura 2.24 - Componentes de un sistema de pateo con solenoide

2.3.5 Sistema de Control de Pelota

Como cualquier jugador de fútbol, el robot debe hacer desplazamientos manteniendo la pelota en su poder. Para que esto sea posible se utiliza un motor que hace girar un rodillo de algún material que brinde adherencia a la pelota.

Este tipo de solución es la más popular en la liga SSL y se conoce como "dribbler". En la actualidad, la mayoría de los equipos utiliza un dispositivo de ese tipo en los robots para poder controlar la pelota. La figura 2.25 muestra el prototipo de un controlador de pelota diseñado y construido en el laboratorio del ITAM por Misael David Soto [42].

Figura 2.25 - Controlador de la pelota "Dribler"

2.3.6 Sistemas de Monitoreo

Para conseguir que el robot realice los movimientos con la adecuada precisión y velocidad, es necesario que tenga conocimiento de su propio estado. La información relacionada con su estado, fundamentalmente la velocidad con que giran sus motores, la consigue con los denominados codificadores ópticos (*encoders*).

Un codificador óptico es un tipo de sensor que convierte un desplazamiento rotacional en una señal digital sin necesidad de un convertidor analógico-digital. Los codificadores de nuestros días utilizan una fuente de luz y su correspondiente detector, el flujo de luz de la fuente hacia el sensor óptico es interrumpido por un disco rotatorio que tiene patrones con figuras transparentes y opacas. La medida del desplazamiento se realiza contando las interrupciones de dicho haz de luz. La figura 2.26 muestra los componentes de un codificador óptico [30].

Figura 2.26 - Componentes de un codificador óptico rotacional

El *encoder* entrega una serie de pulsos cuadrados, similares a una señal PWM, a través de dos canales y dependiendo de la secuencia que estos tengan se obtiene el incremento o decremento de la posición en que se encuentra el motor.

2.3.7 Suministro de Energía

Para que los sistemas del robot funcionen correctamente deben ser alimentados con el voltaje y corriente eléctrica adecuados. Sin embargo, la fuente que los proporcione debe ser tan pequeña como sea posible para caber en el robot, pero con la suficiente energía para aguantar los partidos, dichas características las encontramos en las baterías Li-Po (Litio-Polímero).

Las baterías Li-Po tienen una densidad de energía de entre 5 y 12 veces las de Ni-Cd (Niquel-Cadmio) a igualdad de peso. Además, a igualdad de capacidad, las baterías Li-Po son típicamente cuatro veces más ligeras que las de Ni-Cd. La figura 2.27 muestra una batería Li-Po.

Figura 2.27 - Batería Litio-Polímero

Sin embargo, la gran desventaja de estas baterías es que requieren un trato mucho más delicado, precisan una carga mucho más lenta y no deben descargarse tan profundamente

como es posible hacerlo con las de Ni-Cd, bajo riesgo de deteriorar su capacidad de carga irreversiblemente. Es por este motivo que debemos monitorear el nivel de voltaje de las baterías y mostrarnos, de forma clara y sencilla, cuanto voltaje queda disponible una vez que el robot esta en acción, ya sea en el transcurso de un partido o cuando se están haciendo pruebas con él; con la finalidad de saber cuando es necesario cambiarlas y de esta forma no dañarlas. Para conseguirlo se puede implementar un medidor de baterías basado en Comparadores.

Los comparadores son circuitos que, como su nombre indica, sirven para comparar dos señales (una de las cuales generalmente es una tensión de referencia) y determinar cual de ellas es mayor o menor. La tensión de salida tiene dos estados y se comporta como un convertidor analógico-digital de 1 bit. La operación de un comparador, representada en la figura 2.28, puede expresarse como:

$$\begin{cases} V_{output} = V_{low} & si \quad V_{input} < V_{ref} \\ V_{output} = V_{high} & si \quad V_{input} > V_{ref} \end{cases}$$

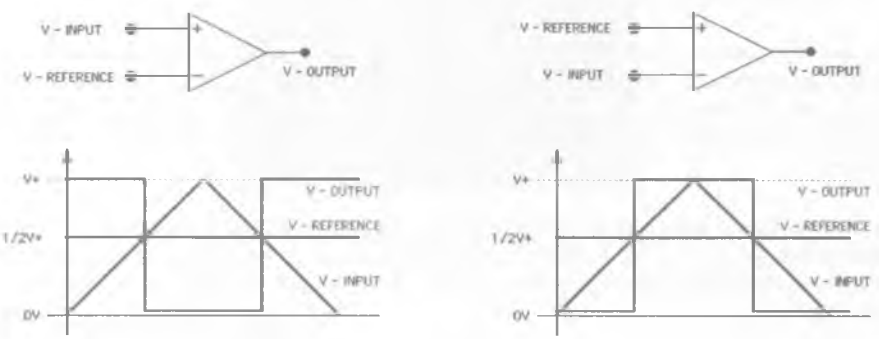


Figura 2.28 - Funcionamiento de un comparador

2.3.8 Acoplamiento

Muchos sistemas digitales, como el DSP, controlan a otros sistemas o realizan funciones de control que deben ser interconectadas a una etapa de manejo de potencia, por ejemplo en el control de velocidad de los motores. La manipulación de altas corrientes implica tener consideraciones de protección para el sistema digital. Sería deseable que la interconexión entre ambas etapas, la digital y la de potencia, se hiciera por un medio de un acoplamiento que permitiera aislar eléctricamente los dos sistemas. Esto se puede lograr con los llamados Optoacopladores, mediante los cuales se obtiene un acoplamiento óptico y un aislamiento eléctrico. La figura 2.29 muestra la conexión de un sistema digital con una etapa de potencia mediante un optoacoplador.

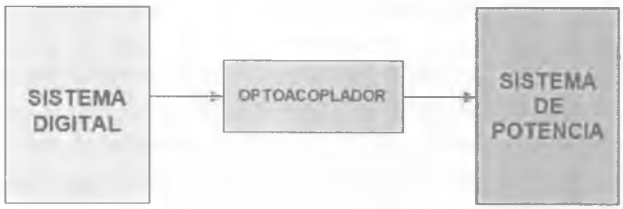


Figura 2.29 – Conexión de un Sistema Digital y un Sistema de Potencia

El Optoacoplador es un dispositivo que se compone de un *led* y un fototransistor, de manera que cuando el *led* emite luz, ilumina el fototransistor y este conduce. Estos dos elementos están acoplados de la forma más eficiente posible. Mediante el optoacoplador, el único contacto entre ambos circuitos es un haz de luz. Estos aislamientos son útiles en aplicaciones en las que los potenciales de ambos circuitos pueden diferir en varios volts, o bien en circuitos totalmente independientes, ya que el *led* puede alimentarse con el voltaje de un circuito y el fototransistor con la tensión de otro. La corriente de salida (corriente de colector del fototransistor) es proporcional a la corriente de entrada (corriente en el *led*). La relación entre estas dos corrientes se llama razón de transferencia de corriente (CTR).

2.4 Control de Trayectorias

Uno de los tantos problemas en la robótica autónoma es conseguir que los robots realicen sus comportamientos lo más apegado al modelo que los describe, es decir, que la diferencia entre el comportamiento deseado y el real sea lo menor posible.

En el caso de los robots SSL, resulta crítico que se desplacen lo más cercano a la trayectoria calculada por la IA, ya que de ello depende la evasión de obstáculos, la llegada a la pelota para hacer un pase o disparar a gol, la marcación de los robots contrarios, en fin, todo aquello relacionado con el desplazamiento del robot dentro de la cancha.

Para moverse en la cancha, el robot recibe una dirección de desplazamiento, calculada y enviada al robot por la IA, y luego debe traducir dicha información en rotación de sus motores. Para controlar la velocidad de rotación de los mismos se usa la modulación por ancho de pulso (PWM). Si la velocidad de rotación deseada no ha sido alcanzada, el controlador de los motores provee una señal PWM más alta. Lo anterior es posible gracias a la acción de un controlador PID, mismo que puede registrar la diferencia absoluta entre la velocidad deseada y la velocidad real de un motor, como se describió en la sección 2.3.1.4.

Si se consigue una buena corrección de la velocidad de los motores, se puede tener un desplazamiento muy cercano a la trayectoria calculada por IA.

La corrección con el controlador PID puede realizarse por cada motor (4 en total) o por cada grado de libertad del robot (3 grados de libertad: v_x, v_y y ω).

2.4.1 Corrección PID por motor

Desde la generación EK2004 y hasta la generación EK2006, se utilizó un controlador PID para cada motor, cuyo factor de corrección estaba dado por:

$$\Delta M_1(t) = K_p e_{M_1}(t) + K_i(\sum_{k=0}^{t} e_{M_1}(t-k)) + K_d(e_{M_1}(t) - e_{M_1}(t-1))$$

$$\vdots$$

$$\Delta M_4(t) = K_p e_{M_4}(t) + K_i(\sum_{k=0}^{t} e_{M_4}(t-k)) + K_d(e_{M_4}(t) - e_{M_4}(t-1))$$

Donde $\Delta M_i(t)$ es el factor de corrección para el motor i en el tiempo t. Y $e_{M_i}(t)$ es la diferencia entre la velocidad requerida y la velocidad actual de cada motor en el tiempo t.

2.4.2 Corrección PID por Grados de Libertad

Los controladores PID también pueden aplicarse por cada grado de libertad del robot: uno para la componente V_x, uno para la componente V_y, y otro para el ángulo de rotación ω. Las velocidades Euclidiana y angular que se requieren son transformadas en velocidades de rotación de cuatro motores, como se describió en la sección 2.3.1.2. Si las velocidades Euclidiana y angular no se logran, los controladores proporcionan correcciones que luego se transforman en correcciones de los motores. Cada controlador PID calcula un factor de corrección dado por:

$$\Delta V_x(t) = K_p e_x(t) + K_i(\sum_{k=0}^{t} e_x(t-k)) + K_d(e_x(t) - e_x(t-1))$$

$$\Delta V_y(t) = K_p e_y(t) + K_i(\sum_{k=0}^{t} e_v(t-k)) + K_d(e_v(t) - e_y(t-1))$$

$$\Delta \omega(t) = K_p e_\omega(t) + K_i(\sum_{k=0}^{t} e_\omega(t-k)) + K_d(e_\omega(t) - e_\omega(t-1))$$

Donde $e_x(t)$, $e_y(t)$ y $e_\omega(t)$ es la diferencia entre la velocidad requerida y la velocidad actual de cada componente euclidiana en el tiempo t.

2.4.3 El problema de los parámetros PID

Independientemente del tipo de corrección utilizada, por motores o por grados de libertad, en las expresiones de ambas encontramos varias constantes: K_p, K_i y K_d estas son las constantes proporcional, integral y derivativa respectivamente.

La corrección es proporcional al error (modulado por K_p). Si el error acumulado es alto, resultado de la suma de errores pasados, la corrección aumenta, modulada por la constante de integración K_i. Si el error está cambiando demasiado rápido, resultado de la diferencia de los dos últimos errores, la corrección también se ve afectada, modulada por la constante K_d. Un controlador sin los parámetros K_i y K_d, tiende a oscilar, alrededor del parámetro deseado. Un controlador con el valor de K_i demasiado grande no oscila, pero tarda en alcanzar el valor deseado. Mientras que un controlador sin parámetro K_d puede sobreactuar y la convergencia al valor deseado toma más tiempo.

Como se acaba de describir, estos parámetros son críticos y sólo pueden se calculados analíticamente cuando se dispone de un buen modelo físico del robot. En la práctica, los parámetros K_p, K_i y K_d son encontrados experimentalmente y ajustados a mano. Este procedimiento frecuentemente produce combinaciones de parámetros subóptimas, que se traducen en movimientos imprecisos del robot.

La figura 2.30 ejemplifica la trayectoria producida por una combinación de parámetros subóptima; supongamos que queremos desplazar al robot del punto A al punto B, pero el mal ajuste de parámetros produce velocidades de motores que difieren de las deseadas, esto produce un error en la trayectoria que se acumula conforme avanza el robot, llevándolo hasta el punto C. La línea punteada en la figura indica la trayectoria que debería seguir, mientras que la línea sólida indica la trayectoria real obtenida.

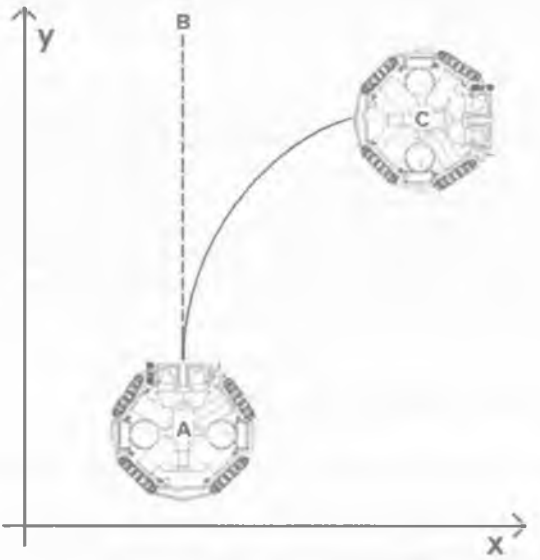


Figura 2.30 – Ejemplo de una trayectoria producida por una combinación de parámetros subóptima.

Recientemente los ingenieros se han concentrado en el desarrollo de controladores PID adaptables que ajusten automáticamente sus parámetros [1]. La mayoría de los métodos publicados se han puesto a prueba sólo con simulaciones por computadora, a los cuales se provee un modelo analítico de control. Cuando no se tiene un modelo analítico, la alternativa al ajuste manual de los parámetros, tedioso y propenso a errores, es la aplicación de los métodos de aprendizaje.

2.5 Robots autónomos y Aprendizaje por Refuerzo

En esta sección se analizará un método concreto que permite a los robots autónomos aprender mientras se encuentran inmersos en su propio entorno. Este método se denomina Aprendizaje por Refuerzo (en adelante RL por sus siglas en inglés *Reinforcement Learning*) y ha tenido un auge importante en los últimos años.

Este método de aprendizaje surge por un lado y conceptualmente, de una rama de estudios de psicología experimental, que pueden remontarse a las experiencias de Pavlov con el refuerzo condicionado, y por otro lado es heredero de los métodos de control óptimo que se originan a partir de los trabajos de Bellman [9].

2.5.1 Conceptos Básicos

Esta sección presenta de manera resumida las bases teóricas del aprendizaje por refuerzo, como son presentadas en [44].

RL es un método de aprendizaje en el cual un agente aprende un determinado comportamiento a través de la interacción con un ambiente, intentando maximizar, a largo plazo, una señal de refuerzo numérica. Tal interacción es del tipo prueba y error: diferentes acciones se prueban en el mismo estado en diferentes etapas del aprendizaje, intentando determinar la mejor. A diferencia del aprendizaje supervisado, la información disponible para el agente no es ya la acción correcta a ser tomada en un estado particular, sino sólo una medida cualitativa de la conveniencia o inconveniencia de la ejecución de una acción en dicho estado, es decir, al sistema no se le dice qué acción debe tomar, sino que él debe de "descubrir" qué acciones dan el máximo beneficio.

2.5.1.1 Modelo del aprendizaje por refuerzo

En un problema típico de aprendizaje por refuerzo se encuentran los siguientes elementos:

Agente: es el sujeto del aprendizaje. Percibe estados ($s \in S$) y ejecuta acciones ($a \in A$). La acción cambia el estado y el agente recibe una señal de refuerzo o recompensa ($r \in R$). El comportamiento del agente debe de ser tal que escoja acciones que tiendan a incrementar a largo plazo la suma de las recompensas totales.

Ambiente: es el elemento con el cual interactúa el agente. Comprende todo lo que no sea este último. En general el ambiente es no-determinístico, tomar la misma acción en el mismo estado puede dar resultados diferentes. Sin embargo, se asume que el ambiente es estacionario, esto es, las probabilidades de cambio de estado no cambian o cambian muy lentamente. En algunos problemas puede identificarse la presencia de un modelo del ambiente, es decir, una descripción (completa o incompleta) del comportamiento del ambiente.

Política ($\pi(s)$): define el comportamiento del agente en un momento dado, qué acciones toma en qué estados. Consiste en un mapeo, determinista o estocástico, entre estados y acciones:

$$\pi: \quad S \rightarrow A$$
$$s \rightarrow \pi(s) = a$$

Definición de Política

En el caso de una política estocástica, ésta queda determinada por un conjunto de probabilidades $\pi(s,a)$ de tomar la acción a en el estado s para cada acción a disponible en todo estado s.

Recompensa (r): señal escalar que define cuáles son los eventos buenos y los malos para el agente. Si es positiva equivale a un "premio", si es negativa, a un "castigo". La recompensa consiste en la única información con que cuenta el agente para lograr su aprendizaje.

El agente y el ambiente interactúan en cada paso de una secuencia de tiempo discreto ($t = 0,1,2,\ldots$) de la siguiente manera:

- En cada paso de tiempo t, el agente recibe una representación del estado en el que se encuentra, $s_t \in S$, donde S es el conjunto de posibles estados.
- Según su política actual, $\pi_t(s)$, el agente selecciona una acción $a_t \in A(s_t)$, donde $A(s_t)$ es el conjunto de acciones disponibles en el estado s_t.
- Al siguiente paso de tiempo, el ambiente responde al agente presentando una recompensa $r_{t+1} \in R$ y un nuevo estado s_{t+1}.

Esta interacción se muestra en la figura 2.32. En el marco del aprendizaje por refuerzo, el objetivo del agente es encontrar una política óptima $\pi^*(s)$ que le permita maximizar la cantidad de recompensa obtenida.

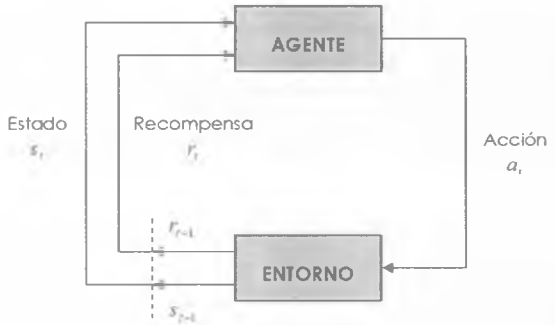


Figura 2.31 - Aprendizaje por Refuerzo.

Otro aspecto importante es el balance entre exploración y explotación. Para obtener buena ganancia uno prefiere seguir ciertas acciones, pero para saber cuáles, se tiene que hacer cierta exploración. Muchas veces depende de cuánto tiempo se espera que el agente interactúe con el medio ambiente.

2.5.1.2 Proceso de decisión markoviano (Markov Decision Process MDP)

El aprendizaje por refuerzo pretende solucionar un MDP, que define una interacción del agente y el ambiente bajo las siguientes condiciones:

1. El conjunto de estados, S, y de acciones disponibles para cada estado, $A(s_t)$, es finito.
2. La dinámica del ambiente queda determinada por la definición de:

a. Un conjunto de probabilidades de transición de estado, es decir, un conjunto de probabilidades de que el estado sea s' dado que el estado actual es s y la acción tomada en dicho estado es a:

$$P_{ss'}^{a} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$$

Probabilidad de Transición de estado

b. Conjunto de valores esperados de la recompensa recibida al tomar la acción a en el estado s y para el estado s':

$$R_{ss'}^{a} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}$$

Valor esperado de la recompensa al pasar de s a s' tomando a

La principal propiedad de un MDP es que la dinámica del ambiente no tiene memoria. En otras palabras, la respuesta del ambiente en el paso de tiempo $t+1$ (el nuevo estado y la recompensa) depende únicamente del estado en el cual se encontraba el agente y la acción tomada por éste en el paso de tiempo t, y no de estados y acciones pasadas. Esta propiedad, que define un MDP, se denomina propiedad de Markov.

2.5.1.3 Retorno

El objetivo de un agente enfrentado a un MDP es aprender el comportamiento (la política) que le permita obtener el máximo de recompensa a largo plazo. En este sentido, la recompensa no es una indicación completa de la conveniencia de la ejecución de una acción en un estado, ya que sólo informa acerca del beneficio a corto plazo que reporta dicha elección. La ejecución de una acción en un estado que reporte al agente una recompensa positiva alta (premio), pero que lo lleve a estados en los cuales reciba poca recompensa o recompensa negativa (castigo), será menos conveniente que la ejecución de una acción que reporte una recompensa menor, o incluso negativa, pero que lleve a estados donde, eventualmente, se recibirán recompensas altas.

Entonces, el agente busca maximizar el retorno definido como una suma ponderada de las recompensas recibidas a partir de un instante t. Si las recompensas recibidas después de un tiempo t se denotan como: r_{t+1}, r_{t+2}, r_{t+3}, ... lo que queremos es maximizar lo que esperamos recibir de recompensa (R_t) que en el caso más simple es:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_T$$

Si se tiene un punto terminal se llaman tareas *episódicas*, si no se tiene se llaman tareas *continuas*. En este último caso, la fórmula de arriba presenta problemas, ya que no podemos hacer el cálculo cuando T no tiene límite.

Podemos usar una forma alternativa en donde se van haciendo cada vez más pequeñas las contribuciones de las recompensas más lejanas:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Retorno

Donde γ se conoce como la *razón de descuento* y está entre: $0 \leq \gamma < 1$. Si $\gamma = 0$ se trata sólo de maximizar tomando en cuenta las recompensas inmediatas. En general, podemos pensar en los siguientes modelos:

1. Horizonte finito: el agente trata de optimizar su recompensa esperada en los siguientes h pasos, sin preocuparse de lo que ocurra despues:

$$E(\sum_{t=0}^{h} r_t)$$

Donde r_t significa la recompensa recibida t pasos en el futuro. Este modelo se puede usar de dos formas:

(i) *política no estacionaria*: donde en el primer paso se toman los h siguientes pasos, en el siguiente los $h-1$, etc., hasta terminar. El problema principal es que no siempre se conoce cuántos pasos considerar.
(ii) *receding-horizon control*: siempre se toman los siguientes h pasos.

2. Horizonte infinito: las recompensas que recibe un agente son reducidas geométricamente de acuerdo a un factor de descuento γ $(0 \leq \gamma \leq 1)$:

$$E(\sum_{t=0}^{\infty} \gamma^t r_t)$$

3. Recompensa promedio: optimizar a largo plazo la recompensa promedio:

$$\lim_{h\to\infty} E(\frac{1}{h}\sum_{t=0}^{h} r_t)$$

2.5.1.4 Funciones de valor

La política π es un mapeo de cada estado $s \in S$ y acción $a \in A(s)$ a la probabilidad $\pi(s,a)$ de tomar la acción a estando en estado s. El valor de un estado s bajo la política π, denotado como $V^\pi(s)$, es el refuerzo esperado estando en estado s y siguiendo la política π. Este valor esperado llamado *función de valor de estado* se puede expresar como:

$$V^\pi(s) = E_\pi\{R_t \mid s_t = s\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\right\}$$

Función de valor de estado

Análogamente, el valor de tomar una acción a en un estado s bajo una política π se define como el valor esperado del retorno obtenido empezando en el estado s, tomando la acción a y siguiendo en adelante la política π:

$$Q^\pi(s,a) = E_\pi\{R_t \mid s_t = s, a_t = a\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\}$$

Función de valor de acción

Informalmente, la función de valor de estado $V^{\pi}(s)$ indica "qué tan bueno" es el estado s en términos de la cantidad de recompensa que el agente puede esperar obtener a partir de él. La función de valor de acción $Q^{\pi}(s,a)$ indica "qué tan bueno" es tomar la acción a en el estado s, en términos de la misma cantidad.

Partiendo de la definición de $V^{\pi}(s)$ puede demostrarse [44] que, para cualquier política π y cualquier estado s:

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'} \left[R^{a}_{ss'} + \gamma \cdot V^{\pi}(s') \right]$$

Ecuación de Bellman para $V^{\pi}(s)$.

donde s' denota un estado siguiente y $\pi(s,a)$ la probabilidad de tomar la acción a en el estado s de acuerdo con la política π. La ecuación que establece la relación entre el valor de un estado y el valor de sus estados sucesores, se conoce como ecuación de Bellman para $V^{\pi}(s)$. La ecuación de Bellman para $Q^{\pi}(s,a)$ es:

$$Q^{\pi}(s,a) = \sum_{s'} P^{a}_{ss'} \left[R^{a}_{ss'} + \gamma \cdot Q^{\pi}(s', \pi(s)) \right]$$

Ecuación de Bellman para $Q^{\pi}(s,a)$

2.5.1.5 Funciones de valor óptimas

El objetivo esencial de las funciones de valor es servir como indicadores de calidad y permitir la comparación de diferentes políticas. Así, desde la función de valor, una política π será mejor o igual a otra política π' si y sólo si $V^{\pi}(s) \geq V^{\pi'}(s) \quad \forall s \in S$; en otras palabras, si el retorno esperado siguiendo la política π es mayor o igual al retorno esperado siguiendo la política π' para cada estado. La política óptima π^* se define como aquella para la cual $V^{\pi^*}(s) = V^{*}(s) \geq V^{\pi}(s) \quad \forall \pi \quad \forall s \in S$. La función de valor de estado óptima de π^* es, entonces:

$$V^{*}(s) = \max_{\pi} V^{\pi}(s)$$

y su función de valor de acción:

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a)$$

La ecuación de Bellman para la función de valor de la política óptima, V^{*}, a diferencia de la ecuación de Bellman para funciones de valor de otras políticas, no necesita ser escrita en términos de sus probabilidades $\pi^{*}(s,a)$:

$$V^{*}(s) = \max_{a} \sum_{s'} P^{a}_{ss'} \left[R^{a}_{ss'} + \gamma \cdot V^{*}(s') \right]$$

Ecuación de optimalidad de Bellman para $V(s)$

Esta ecuación se conoce como la ecuación de optimalidad de Bellman. La contraparte para Q^* es:

$$Q^*(s,a) = \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma \cdot Q^*(s', \pi(s)) \right]$$

Ecuación de optimalidad de Bellman para $Q^\pi(s,a)$

2.5.1.6 Métodos de Solución del Aprendizaje por Refuerzo

Existen, principalmente, tres clases de métodos para resolver el problema del aprendizaje por refuerzo: Programación Dinámica, Métodos Monte Carlo y Métodos de Diferencias Temporales. [44]

Cada clase de método tiene sus fortalezas y debilidades. Los métodos de Programación Dinámica matemáticamente están bien desarrollados, pero requieren un modelo completo y preciso del ambiente. Los métodos Monte Carlo no requieren de un modelo y son conceptualmente simples, pero no son adecuados para calcularlos gradualmente paso a paso. Por último, los métodos de diferencias temporales no requieren ningún modelo y son completamente incrementales, pero son más complejos de analizar. Los métodos también difieren en otros aspectos por ejemplo, su eficiencia y velocidad de convergencia.

Otra forma de clasificar los métodos de solución del aprendizaje por refuerzo es de acuerdo con el tipo de algoritmo que usan para resolver un MDP. Siendo así tenemos: algoritmos para encontrar una Política Óptima, en los que destacan la Programación Dinámica y los Algoritmos Genéticos; algoritmos para estimar la Función de Valor, también llamados métodos de Diferencias Temporales, por ejemplo Q-Learning y Sarsa; algoritmos Actor-Crítico y, finalmente, algoritmos que calculan el Gradiente de la Política. La figura 2.32 muestra esta última clasificación.

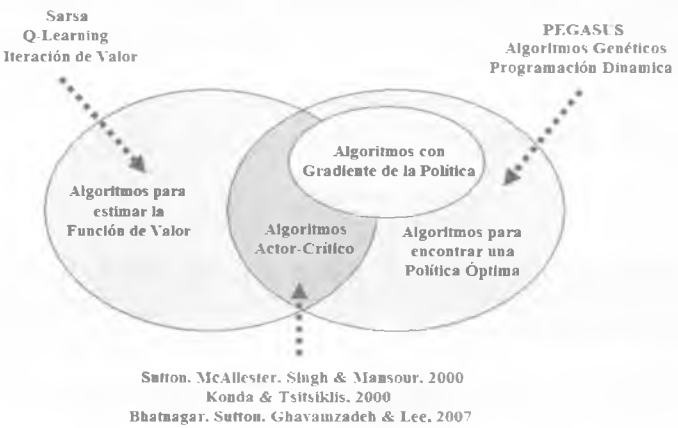


Figura 2.32 – Algoritmos para resolver MDP´s

2.5.1.7 Aprendizaje por Refuerzo con Gradiente de la Política (PGRL)

Buena parte de las técnicas de aprendizaje más conocidas y utilizadas, como *Q-Learning* y *Sarsa*, se basan en la aproximación y estimación de una función de valor y en la generación de una política a partir de dicha función. Este enfoque es conocido como Aprendizaje por Refuerzo con Función de Valor (VFRL por sus siglas en inglés). Sin embargo, a pesar de muchas aplicaciones exitosas, dichas técnicas tienen pocas garantías teóricas de eficacia.

En primer lugar, se orientan hacia la búsqueda de políticas deterministas seleccionando diferentes acciones con probabilidades específicas, mientras que la política óptima es a menudo estocástica [44]. En segundo lugar, en los métodos VFRL, un cambio infinitesimal en los parámetros de la función de valor puede impulsar el valor de una acción más que de otro, lo que provoca un cambio discontinuo en la política, los estados visitados, y el rendimiento general. Tales cambios discontinuos han sido identificados como un obstáculo clave para establecer garantías de convergencia para los algoritmos que siguen dicho enfoque [19][37]. Por ejemplo, métodos como Q-*Learning*, *Sarsa*, y métodos de programación dinámica se han mostrado incapaces de converger a una política óptima, incluso para un MDP sencillo [3][10][20].

Estos problemas han motivado la búsqueda de alternativas con mejores y más claras propiedades de convergencia. Un de los enfoques alternativos al VFRL es considerar una clase de políticas aleatorias parametrizadas, calcular el gradiente de una función de rendimiento respecto a los parámetros, y mejorar la política ajustando los parámetros en la dirección del gradiente.

Este enfoque es conocido como Aprendizaje por Refuerzo con Gradiente de la Política o *Policy Gradient Reinforcement Learning* (PGRL) [45], y recientemente ha recibido mucha atención como medio para resolver problemas cuyos estados son espacios continuos.

PGRL en lugar de aproximar una función de valor y utilizarla para calcular una política determinista, aproxima directamente una política estocástica a través de una función independiente con sus propios parámetros. Denotemos θ como el vector de parámetros de la política y ρ el rendimiento correspondiente a la política (por ejemplo, la recompensa promedio por cada paso). Entonces, en el enfoque de política de gradiente, los parámetros de la política se actualizan proporcionalmente al gradiente:

$$\Delta\theta = \alpha \frac{\partial \rho}{\partial \theta}$$

Donde α es la tasa de aprendizaje. Si lo anterior puede lograrse, entonces θ puede, generalmente, converger a una política óptima a nivel local en la medida del desempeño de ρ. A diferencia del enfoque de la función de valor, aquí los pequeños cambios en θ sólo pueden causar pequeños cambios en la política y en la distribución de los estado visitados.

En general, los algoritmos basados en PGRL consisten en tres pasos: parametrizar la política y determinar los parámetros iniciales, evaluar el gradiente, y actualizar los parámetros con base en el gradiente [15].

1. Parametrizar. Elegir π_θ y θ_0
2. Calcular el Gradiente. $\nabla\eta(\theta)$
3. Mejorar los parámetros ajustándolos en la dirección del gradiente. $\theta_{m+1} = \theta_m + \alpha_m \nabla\eta(\theta_m)$

Los pasos de la estimación del gradiente y la actualización de la política se repiten hasta que θ haya convergido.

Las ventajas de los métodos que hacen uso de la política de gradiente son numerosas. Entre las más importantes destacan:

1) Los algoritmos de PGRL tienen garantías teóricas de convergencia a políticas óptimas locales.
2) Es posible incorporar un conocimiento previo del dominio a través de una adecuada parametrización de la política.
3) La representación de la política puede elegirse con base en lo más significativo para la tarea en cuestión.
4) Las funciones de valor pueden ser muy complejas y difíciles de aproximar para algunos problemas, mientras que las políticas pueden tener una forma más simple.
5) Estos métodos pueden aplicarse sin necesidad de la comprensión profunda del problema o de la mecánica del robot.

Sin embargo, PGRL también presenta desventajas potenciales:

1) Converge sólo a un óptimo local
2) Varianza alta en la estimación del gradiente
3) Convergencia lenta.

2.5.1.7.1 Teorema del Gradiente de la Política

Para este enfoque hay que considerar el marco estándar del aprendizaje por refuerzo, en el que un agente interactúa con un MDP. El estado, acción y recompensa en cada tiempo $t \in \{0,1,2,...\}$ se denota como $s_t \in S$, $a_t \in A$ y $r_t \in R$ respectivamente. La dinámica del ambiente esta caracterizada por las probabilidades de transición de los estados $P^a_{ss'} = Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$, y la recompensas esperadas $R^a_s = \mathrm{E}\{r_{t+1} \mid s_t = s, a_t = a\}$, $\forall s, s' \in S$, $a \in A$. El comportamiento del agente en un momento dado es caracterizado por una política, $\pi(s,a,\theta) = Pr\{a_t = a \mid s_t = s, \theta\}$ $\forall s \in S$, $a \in A$ donde $\theta \in \Re^l$, para $l << |S|$, es

un vector de parámetros. Se asume que π es diferenciable con respecto a sus parámetros, es decir, $\frac{\partial \pi(s,a)}{\partial \theta}$ existe.

El objetivo del agente puede formularse de dos maneras útiles. Una de ellas es la Recompensa Promedio, en la que las políticas se clasifican en función de la recompensa esperada a largo plazo, $\rho(\pi)$

$$\rho(\pi) = \lim_{n\to\infty} \frac{1}{n} E(r_1 + r_2 + \ldots + r_n \mid \pi) = \sum_s d^\pi(s) \sum_a \pi(s,a) R_s^a$$

Donde $d^\pi(s) = \lim_{t\to\infty} Pr\{s_t = s \mid s_0, \pi\}$ es la distribución estacionaria de estados bajo la política π, la cual se asume que existe y es independiente de s_0 para todas las políticas. En el enfoque de la recompensa promedio, dada una política, el valor de un estado-acción se define como:

$$Q^\pi(s,a) = \sum_{t=1}^{\infty} E\{r_t - \rho(\pi) \mid s_0 = s, a_0 = a, \pi\} \quad \forall s \in S, a \in A$$

La segunda formulación, llamada de Estado Inicial, es aquella en el que se designa un estado inicial s_0, y hay que preocuparse sólo de la recompensa obtenida por éste a largo plazo:

$$\rho(\pi) = E\left\{\sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid s_0, \pi\right\} \; y \; Q^\pi(s,a) = E\left\{\sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k} \mid s_t = s, a_t = a, \pi\right\}$$

Donde $\gamma \in [0,1]$ es la razón de descuento ($\gamma = 1$, solo es permitido si en tareas episódicas). En esta formulación, se define $d^\pi(s)$ como un descuento ponderado de los estados visitados empezando en s_0 y después π : $d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t Pr\{s_t = s \mid s_0, \pi\}$

Teorema 1 (Política de Gradiente) [45]. Para cualquier MDP, en la formulación de Recompensa Promedio o en la de Estado Inicial

$$\frac{\partial \rho}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a)$$

Este resultado se refiere al gradiente de la métrica de desempeño con respecto al parámetro de la política.

- CAPÍTULO 3 -

REQUERIMIENTOS DE LOS ROBOTS

En este capítulo se establecen los requerimientos de la quinta generación de robots móviles autónomos *Small Size* tomando como base la generación EK2006 y las consideraciones teóricas del capítulo 2.

3.1 Los requerimientos de la generación EK2007

Los resultados obtenidos por el equipo *Eagle Knights* en la liga *Small Size* durante el *RoboCup* 2006 [47], celebrado en Bremen Alemania, condujeron a la necesidad de contar con un nuevo equipo robots que reunieran las características necesarias para ser más competitivos en las próximas competencias. Prácticamente lo que se necesitaba era un diseño electrónico más estable.

El punto de partida para el diseño de un equipo de robots SSL es la funcionalidad básica mencionada en la sección 2.2.5; sin embargo, como puede observarse en la figura 3.1, para establecer los requerimientos de los robots EK2007 se partió de las de debilidades de la generación anterior. Esto se hizo con la finalidad de aprovechar los avances logrados año tras año. Por lo tanto, las próximas secciones estarán dedicadas a describir las características que los nuevos robots deberían cubrir, así como los puntos débiles de la generación anterior que motivaron los cambios.

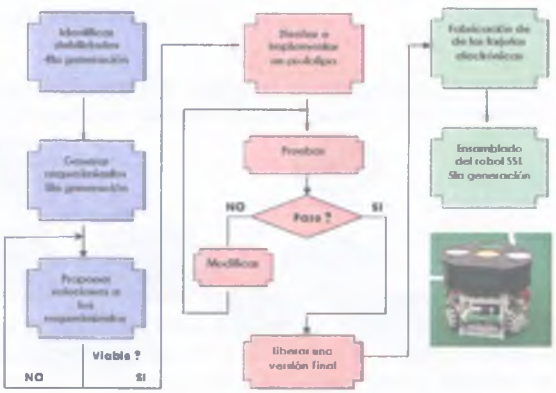


Figura 3.1 - Pasos para el diseño e implementación de la generación EK2007

3.1.1 Locomoción

Se había mencionado que el desplazamiento omnidireccional con cuatro ruedas es utilizado en los robots EK desde 2005; sin embargo, la disposición de sus motores era asimétrica, lo que complicaba su control. Para simplificar el modelo omnidireccional, tal y como se describió en la sección 2.3.1.1, se decidió que los nuevos robots tendrían dos ejes de simetría.

3.1.2 Arquitectura Electrónica Modular

Por cuestiones de espacio, las generaciones anteriores agrupaban todos los componentes electrónicos en una o dos tarjetas de circuito impreso (PCB por sus siglas en inglés); sin embargo, si se deseaba hacer modificaciones en algún sistema era necesario rediseñar toda la tarjeta de nuevo. Además, la aglomeración de varios circuitos en un espacio tan reducido produjo errores en la fabricación de las tarjetas, por ejemplo, confusión entre las referencias de los circuitos. Por lo tanto, se vio conveniente separar los componentes electrónicos del robot con base en la función que desempeñan y/o la energía que demandan, e integrarlos en una tarjeta de propósito específico; esto gracias a la nueva estructura mecánica del robot diseñada por Misael David Soto [42]. Clasificados de tal forma, se pueden obtener cuatro módulos:

- Suministro de Energía
- Módulo Analógico
- Módulo Digital
- DSP

La Figura 3.2 muestra de manera gráfica los módulos de la electrónica de un Robot SSL generación EK2007. Cada módulo integra diversos sistemas con una función específica.

Figura 3.2 - Módulos de la Electrónica de un Robot SSL generación EK2007

A continuación se describen de forma más detallada los requerimientos de los módulos de la electrónica mostrados en la figura 3.2 con los respectivos sistemas que los integran.

3.1.3 Suministro de energía

El módulo para el suministro de energía debe agrupar todas las conexiones necesarias para abastecer de voltaje y corriente eléctrica al robot, además debe incluir un par de medidores que indiquen el voltaje de las baterías. Para que los circuitos del robot funcionen adecuadamente, el tipo de suministro debe ser congruente con la cantidad de energía que estos demandan y el tipo de señales eléctricas que utilizan, siendo así, es conveniente tener dos tipos: Suministro Analógico y Suministro Digital.

3.1.3.1 Suministro de Energía Analógico

El propósito de este sistema es proveer la energía necesaria para que funcionen los circuitos cuyos componentes demandan bastante voltaje y corriente, principalmente los controladores de los motores y el de sistema de pateo. La máxima tensión e intensidad requerida por dichos circuitos es de 16 volts y 2.8 amperes respectivamente.

3.1.3.2 Suministro de Energía Digital

De forma análoga al anterior, el propósito de este sistema es proporcionar energía a los circuitos para la comunicación, identificación y sensado del robot, además del suministro para el DSP. Los componentes de los circuitos digitales necesitan 5 volts para funcionar correctamente y una corriente constante de 0.5 A en el caso del DSP.

3.1.3.3 Suministro para el DSP

El Procesador Digital de Señales requiere un voltaje de 5V y una corriente mínima y constante de 500mA para su correcto funcionamiento. Dicha energía puede obtenerse del Suministro Digital.

3.1.3.4 Medidores de Baterías

La gran desventaja de las baterías Li-Po es que requieren un trato mucho más delicado, precisan una carga mucho más lenta y no deben descargarse tan profundamente como es posible hacerlo con las de Ni-Cd. Por este motivo los nuevos robots deben contar con un medidor de baterías que muestre su nivel de voltaje con la finalidad de cambiarlas cuando sea necesario.

3.1.4 Módulo Digital

El módulo Digital debe agrupar todos aquellos circuitos lógicos para la comunicación, identificación y sensado del robot. Todos estos circuitos se caracterizan por que sus señales están codificadas en dos únicos estados 1 y 0, refiriéndose a que en un circuito eléctrico hay tensión de voltaje (1) o hay ausencia del mismo (0).

3.1.4.1 Identificador del Robot

Este sistema sirve para identificar al robot con un número del 0 al 9, aunque en la práctica sólo se utilizan los números del 1 al 5. El dígito asignado al robot se tiene que almacenar en una variable del DSP y desplegarse en un display de 7 segmentos.

3.1.4.2 Transceiver

Un sistema de comunicación en cada robot debe encargase de recibir la información de IA y entregarla al procesador central, para que éste tome sólo la información que le corresponde. Este sistema, basado en el RPC descrito en la sección 2.3.2.1, ha funcionado bien el todas las generaciones donde se ha implementado, por lo tanto en los robots EK2007 no presenta modificaciones.

3.1.4.2.1 La Trama de Comunicación

En las generaciones anteriores, el Sistema de Inteligencia Artificial calculaba el vector de movimiento para cada robot y posteriormente transformaba las velocidades euclidianas de cada uno en velocidades individuales para cada motor, estas últimas eran enviadas en una trama de comunicación a todos los robots. Sin embargo, para la quinta generación se decidió que IA sólo enviaría el vector de movimiento y el DSP de cada robot haría la conversión a velocidades de motores, de esta forma se aprovecharía más el DSP y se libaría al sistema de IA de dicho procesamiento.

3.1.5 Módulo Analógico

Debe agrupar todos los circuitos de potencia, es decir, circuitos que demandan gran cantidad de voltaje y corriente, y con los cuales pueden funcionar todos los actuadores del robot. Llamamos actuadores a los dispositivos que traducen señales eléctricas, provenientes del DSP, en alguna acción del robot. La electrónica analógica considera y trabaja con valores continuos; podemos acotar que trata con señales que cambian en el tiempo de forma continua, por ejemplo las señales PWM.

3.1.5.1 Control de motores

En la generación EK2006, los controladores de los motores se calentaban después de funcionar cierto tiempo debido a algunos errores en la implementación, lo que se traducía en movimientos imprecisos de los robots, sobre todo cuando los motores tenían que girar a bajas velocidades. Además después de las competencias de 2006 muchos motores quedaron inservibles. Esto provocó que para la nueva generación se cambiaran los motores por un modelo más resistente y compacto.

Debido a que el modelo de motores fue reemplazado, fueron necesarios nuevos controladores con una nueva configuración para tener mejor control.

3.1.5.2 Optoacopladores

El DSP es un dispositivo muy sensible a señales eléctricas que no están dentro de su rango de operación, además tiene que interactuar con circuitos de la parte analógica que manejan bastante potencia. Por lo tanto, para evitar daños al DSP, es necesario utilizar una etapa de acoplamiento entre él y los circuitos analógicos. Para tal propósito deben utilizarse optoacopladores, cuyo funcionamiento fue descrito en la sección 2.3.8.

3.1.5.3 Codificadores de motores (Encoders)

Las señales producidas por los *encoders,* utilizadas para calcular la velocidad real de los motores, llegaban al DSP con un voltaje justo en el límite de su rango de operación (3.3V) y con alguna pequeña variación excedían dicho umbral provocándole daños.

Para no seguir dañando DSPs, todas las señales que llegan a éste tendrían que ser ajustadas para que, en caso de pequeñas variaciones, no excedieran los 3.3V con los que trabaja el procesador.

3.1.5.4 Sistema de Control de pelota

Electrónicamente, el sistema de control de pelota o *Dribbler* simplemente requiere un motor para hacer girar un rodillo que atraiga la pelota hacia el robot. Aunque el sentido y velocidad de giro de dicho motor siempre es el mismo, se puede utilizar el mismo controlador de los motores para el desplazamiento.

3.1.5.5 Sistema de Pateo

Desafortunadamente, el sistema de pateo de la generación EK2006 presentó diversos problemas: inestabilidad, golpeo débil y componentes de difícil adquisición. Provocados,

en gran medida, por la forma en que funcionaba su generador de alto voltaje para la etapa de Carga descrita en la sección 2.3.4.

Para elevar el voltaje usaban un transformador; sin embargo, los transformadores funcionan con voltaje alterno y las baterías proporcionan voltaje directo, por lo tanto era necesario agregar una etapa de oscilación y con ella alimentar el transformador. Además, los transformadores mantienen la potencia constante a la entrada y a la salida, es decir, si un transformador eleva el voltaje en un factor de 10, entonces la corriente se reduce en un factor de 10. Debido a esta propiedad era necesario agregar una etapa de potencia que incrementara la corriente.

Todo lo anterior resultaba complicado e inestable, sobre todo por que los transformadores se comportaban de forma distinta en cada robot debido a que se fabricaron artesanalmente para reducir su tamaño y que de esta forma cupieran en el robot.

Ante esta situación, se volvió necesario diseñar un sistema que prescindiera del transformador y que necesitara menos procesamiento para golpear la pelota con mayor fuerza. También se vio conveniente separar este sistema del resto de los circuitos y tener una tarjeta exclusiva para su propósito, de esta manera, en caso de tener problemas con él, simplemente habría que buscar otra solución pero sin tener que rediseñar todos los demás circuitos, como se había venido haciendo.

3.1.6 DSP

A grandes rasgos, el DSP tiene que interpretar las instrucciones de IA y las señales de ciertos circuitos para realizar lo siguiente:

- Obtener el número identificador del robot y mostrarlo a través de un display de 7 segmentos.
- Con base en el identificador, obtener del Sistema de Comunicación los paquetes con los comandos de IA que le corresponden.
- Activar o desactivar los sistemas de pateo y control de pelota.
- Actualizar las variables correspondientes con el nuevo vector de movimiento
- Mientras no reciba nuevas instrucciones, controlar los motores con base en el último vector de movimiento recibido, lo que implica hacer la transformación de velocidades euclidianas a velocidades de motores.
- Registrar en todo momento la velocidad y sentido de giro de los motores.
- Transformar las velocidades de motores en velocidades euclidianas para hacer la corrección con los controladores PID.

El DSP que utilizan los robots desde 2004 tiene una gran capacidad de procesamiento, por lo tanto no tendría inconveniente en hacer la transformación de velocidades euclidianas a velocidades de motores, liberando al sistema de IA de dicha tarea. De esta forma, la transformación inversa también puede ser utilizada cambiando el tipo de corrección de los

controladores PID, es decir, pasar de la Corrección por Motores a la Corrección por Grados de Libertad.

3.1.7 Control de Trayectorias

Para corregir el error en la trayectoria, producido por una combinación subóptima de parámetros PID que se explicó en la sección 2.4.3, se hace uso de los sistemas de Visión e IA, como se describe a continuación:

Supongamos que el sistema de IA envía un vector de movimiento al robot para llevarlo del punto A al punto B; en cada momento el Sistema de Visión proporciona la posición real del robot a IA, en cuanto esta última detecta una desviación del objetivo, redefine el vector de movimiento para compensar el error y se lo envía al robot, este ciclo se repite hasta llegar al punto destino. La figura 3.3 muestra esta forma de corregir el error en la trayectoria.

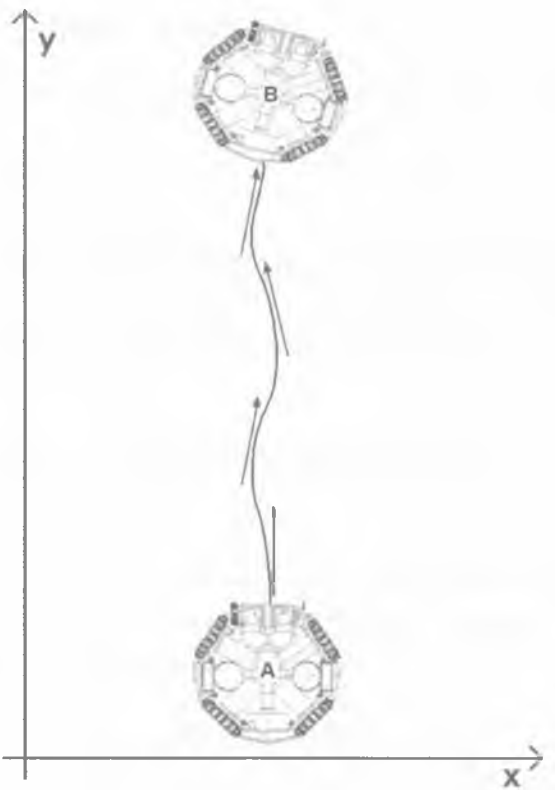


Figura 3.3 – Ejemplo de Control de una Trayectoria mediante los SV e IA

Sin embargo, como puede observarse en la figura anterior, este tipo de corrección provoca oscilaciones durante la trayectoria, además, la orientación con que llega el robot al destino ya no es la deseada.

Por este motivo se vio conveniente implementar el algoritmo de PGRL descrito en [18] y [23] para encontrar los parámetros óptimos de los controladores PID que permitan al robot conducirse con el menor error, de esta forma se estaría atacando al problema desde el origen.

- CAPÍTULO 4 -

DISEÑO DE LOS ROBOTS

En este capítulo se aborda el diseño de los robots EK2007-EK2008, con los que el Laboratorio de Robótica del ITAM participó en la competencia *RoboCup* 2007 y *RoboCup* 2008. Además, en las últimas secciones, se muestra como los parámetros óptimos para varios controladores PID pueden ser aprendidos de forma adaptiva tras conducir al robot en su entorno mientras se evalúa su comportamiento.

4.1 Diseño de los Robots EK2007

Tomando en cuenta los requerimientos del capítulo anterior, se procedió con el diseño de los robots.

4.1.1 Locomoción

Por simplicidad, los robots EK2007 tienen dos ejes de simetría con un ángulo $\varphi = 38°$, figura 4.1, por lo tanto las ecuaciones de transformación quedan como sigue:

Para transformar velocidades euclidianas en velocidades de motores:

$$m = Dv$$

Para transformar velocidades de motores a velocidades euclidianas:

$$v = D^{+} m$$

Donde $m = (v_1, v_2, v_3, v_4)^T$ es el vector que agrupa la velocidad individual de los cuatro motores, y $v = (v_x, v_y, \omega)^T$ es el vector con las velocidades euclidianas y de rotación del robot. Y las matrices D y D^{+} quedan como sigue:

$$D = \begin{pmatrix} -\sin\varphi & \cos\varphi & 1 \\ -\sin\varphi & -\cos\varphi & 1 \\ \sin\varphi & -\cos\varphi & 1 \\ \sin\varphi & \cos\varphi & 1 \end{pmatrix} = \begin{pmatrix} -0.6157 & 0.7880 & 1 \\ -0.6157 & -0.7880 & 1 \\ 0.6157 & -0.7880 & 1 \\ 0.6157 & 0.7880 & 1 \end{pmatrix}$$

$$D^{+} = \frac{1}{4}\begin{pmatrix} -\frac{1}{\sin\varphi} & -\frac{1}{\sin\varphi} & \frac{1}{\sin\varphi} & \frac{1}{\sin\varphi} \\ \frac{1}{\cos\varphi} & -\frac{1}{\cos\varphi} & -\frac{1}{\cos\varphi} & \frac{1}{\cos\varphi} \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -0.4061 & -0.4061 & 0.4061 & 0.4061 \\ 0.3173 & -0.3173 & -0.3173 & 0.3173 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{pmatrix}$$

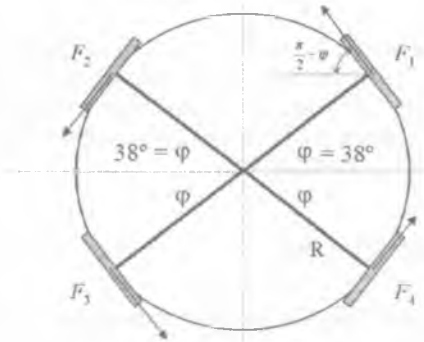


Figura 4.1 - Disposición de los Motores en los Robots EK2007

4.1.2 Arquitectura Electrónica Modular

Para tener una arquitectura más flexible en caso de futuros cambios, los sistemas de los robots EK2007 fueron diseñados con base en la modularización planteada en la sección 3.1.2, tal y como se muestra en las próximas secciones.

4.1.3 Suministro de Energía

La fuente de alimentación elegida para los robots EK2007 fueron las baterías Li-Po, la cantidad de baterías así como su capacidad varió dependiendo del tipo de suministro, digital o analógico, como se describe a continuación.

4.1.3.1 Suministro de Energía Analógico

Para conseguir los 16V y 2.8A (máximos) que requieren los circuitos analógicos se utilizó un arreglo de 6 baterías Li-Po de 7.4V y 730mAh. Dicho arreglo consta de 2 grupos de baterías conectadas en serie, cada uno de los cuales se conforma por 3 baterías conectadas en paralelo (BA1A, BA1B, BA1C y BA2A, BA2B, BA2C), como se muestra en la figura 4.2.

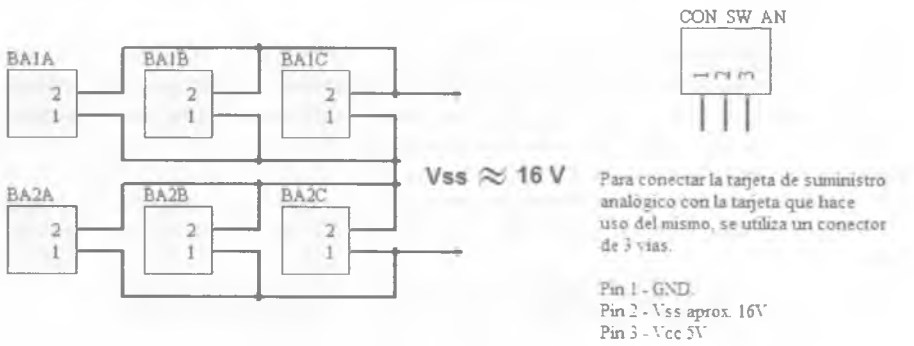


Figura 4.2 - Diagrama lógico para suministro de energía analógico

Cabe mencionar que, cuando están completamente cargadas, las baterías alcanzan una diferencia de potencial de 8V aproximadamente, y no los 7.4V indicados. Entonces, un grupo de 3 conectadas en paralelo proporciona aproximadamente 8V y 2190mAh, pero al conectar dos grupos en serie obtenemos 16V y 2190mAh, mismos que son utilizados por los controladores de los motores y el sistema de pateo mediante un conector de tres vías (CON_SW_AN). Algunas señales de control en estos circuitos deben ser de 5V, por lo tanto hay que reducir los 16V mediante un regulador.

La razón de utilizar un arreglo de baterías, y no una sola que proporcione el total deseado, fue la restricción del espacio en el diseño mecánico. Pero antes de seguir es necesario establecer la terminología adecuada para identificar los niveles de voltaje utilizados en el diseño de los circuitos, tal como lo muestra la tabla 4.1.

Nombre	Símbolo utilizado en los diagramas	Valor máximo	Descripción
V_{SS}	Vss	16V	Voltaje total del arreglo de baterías
V_{CC_A}	+5a	5V	Voltaje de salida del regulador para alimentar a los circuitos integrados
GND_A		---	Tierra general del circuito analógico

Tabla 4.1 - Voltajes de los circuitos analógicos

4.1.3.2 Suministro de Energía Digital

Para conseguir los niveles de voltaje y corriente deseados se utilizó un arreglo de 3 baterías de 7.4 volts y 910 mAh conectadas en paralelo, por lo tanto el voltaje total del conjunto es de 8V aproximadamente cuando las baterías están completamente cargadas. A pesar que una sola batería cubre los requerimientos de energía, se usan tres para que los circuitos tengan energía por más tiempo. Además, en los requerimientos se estableció que el voltaje con el que trabajan los circuitos digitales es de 5V, por lo tanto, también es necesario reducir los 8V mediante un regulador. En la figura 4.3 se muestra la conexión de las baterías.

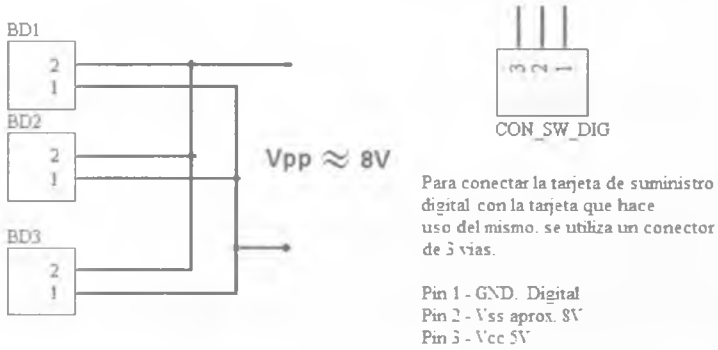


Figura 4.3 - Diagrama lógico para suministro de energía digital

La tabla 4.2 muestra los niveles de voltaje utilizados en el diseño de los circuitos digitales.

Nombre	Símbolo utilizado en los diagramas	Valor máximo	Descripción
V_{PP}	----	8V	Voltaje total del arreglo de baterías
V_{CC_D}	+5d	5V	Voltaje de salida del regulador para alimentar a los circuitos
GND_D		---	Tierra general del circuito digital.

Tabla 4.2 - Voltajes de los circuitos analógicos

4.1.3.3 Medidor de Baterías del Suministro Analógico

Para monitorear el voltaje de las baterías se diseñó un medidor basado en el funcionamiento del comparador LM339DR2. El medidor analógico cuenta con tres *leds* indicadores. Un *led* verde permanece encendido cuando el voltaje es superior a 16 volts; cuando el voltaje es inferior a los 15 volts pero mayor a 14 volts el *led* verde se apaga y se enciende un *led* amarillo; finalmente si el voltaje es inferior a 14 volts el *led* amarillo se apaga y enciende uno rojo. La figura 4.4 describe el funcionamiento del medidor de baterías analógico.

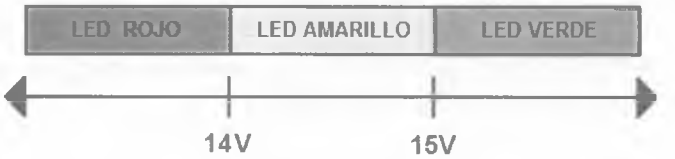


Figura 4.4 - Comportamiento del Medidor de Baterías analógico

Según lo descrito en la sección 2.3.7, necesitamos dos tensiones para hacer la comparación: en primer lugar el de entrada (V_{input}) que proviene directamente de las pilas, y cuyo voltaje máximo es de 16V (V_{SS}); y en segundo lugar, el de referencia (V_{ref}) que se decidió obtener del voltaje regulado de la parte analógica (V_{CC_A}), ya que permanece fijo en 5V la mayor parte del tiempo, siempre y cuando las baterías, en conjunto, no se descarguen por debajo de dicha tensión. Sin embargo, el comparador utilizado solo funciona con voltajes en el rango de 0V a 5V, por lo tanto las tensiones a comparar son reducidas mediante un divisor de voltaje, es decir, una configuración de circuito eléctrico que reparte la tensión de una fuente entre una o más resistencias conectadas en serie. La figura 4.5 muestra dicha configuración con la fórmula que describe su comportamiento.

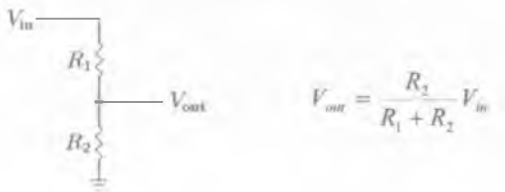


Figura 4.5 - Divisor de voltaje

Donde V_{in} es la tensión a repartir y V_{out} es sólo la fracción de voltaje que deseamos.

Para tener un margen de comparación, se decidió fijar el voltaje de referencia en $V_{ref} = 2.5V$ mismo que obtuvimos de la tensión analógica regulada $V_{CC\_A} = 5V$ haciendo uso de la formula antes descrita:

$$V_{ref} = \frac{R_2}{R_1 + R_2} V_{CC\_A} \rightarrow 2.5V = \frac{R_2}{R_1 + R_2} 5V$$

Para que la igualdad anterior se cumpla y que $V_{ref} = 2.5V$, debe ocurrir que $R_1 = R_2$. A continuación se debe obtener la señal de entrada V_{input} que será comparada contra la referencia V_{ref} y que permitirá generar la señal para encender el *led* verde siempre que la primera sea mayor que la segunda, para esto otro divisor debe ajustarse de tal forma que una tensión igual a la cota superior del medidor permita obtener exactamente el valor de la referencia (2.5V), de esta forma, un valor de V_{SS} superior a la cota ($V_{SS} > 15V$) dará por resultado una señal mayor que V_{ref}. Dicho esto, las condiciones del divisor que permite obtener la señal para el encendido del *led* verde son:

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{SS} \rightarrow 2.5V = \frac{R_2}{R_1 + R_2} 15V$$

De donde se obtiene que, para cumplir la igualdad $R_1 = 5R_2$. El voltaje V_{out} resultado del divisor anterior, se convierte en la segunda entrada del comparador.

Para encender el *led* rojo se sigue un procedimiento similar al del verde, sólo se debe cambiar el valor de la cota superior por el de la inferior e invertir las entradas del comparador, ya que a la salida del mismo queremos respuesta cuando la tensión de entrada sea menor que la referencia $V_{SS} < 14V$.

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{SS} \rightarrow 2.5V = \frac{R_2}{R_1 + R_2} 14V$$

Para esto se requiere que $R_1 = 4.6R_1$. De esta forma, mientras $V_{SS} > 14V$ no tendremos salida en el comparador pero en cuanto $V_{SS} < 14V$ se obtiene una señal que enciende el *led* rojo.

Ya que el *led* amarillo enciende cuando el voltaje de las baterías está en el rango intermedio (14V a 15V), simplemente aplicamos una operación NOR entre las señales que encienden los *leds* verde y el rojo, es decir, enciende cuando no esta en rojo ni esta en verde.

En lugar de utilizar resistencias con un valor fijo para los divisores, se optó por usar potenciómetros, ya que, al ser resistencias variables, permiten modificar los divisores de voltaje y con ello el límite para el encendido de los *leds*. La figura 4.6 muestra el diagrama lógico del medidor de baterías analógico.

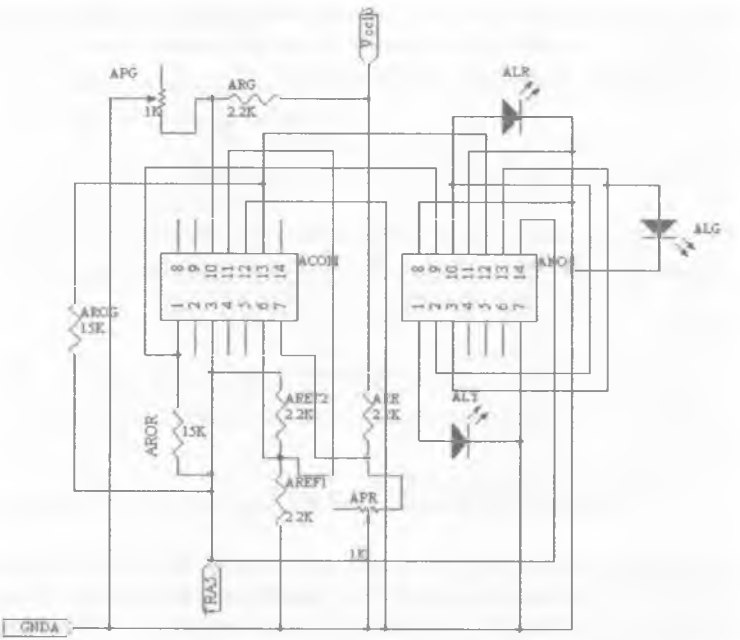


Figura 4.6 - Diagrama lógico Medidor de Baterías Analógico

4.1.3.4 Medidor de Baterías del Suministro Digital

Este medidor monitorea el nivel de voltaje de las baterías que alimentan la parte digital del robot. También cuenta con tres *leds* indicadores: un *led* verde permanece encendido cuando el voltaje es superior a 7.5 volts; cuando el voltaje es inferior a los 7.5 volts pero mayor a 7 volts el *led* verde se apaga y se enciende el *led* amarillo; finalmente si el voltaje es inferior a 7 volts el *led* amarillo se apaga y enciende uno rojo. Este comportamiento puede observarse en la figura 4.7.

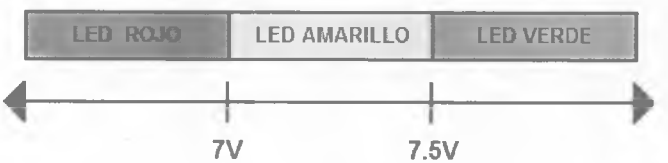


Figura 4.7 - Comportamiento del Medidor de Baterías Digital

Los principios en los que se basa son los mismos que el analógico, la principal diferencia radica en el valor de las cotas para encender los diodos emisores de luz. El divisor de voltaje para la tensión de referencia quedo como sigue:

$$V_{ref} = \frac{R_2}{R_1 + R_2} V_{CC\_A} \rightarrow 2.5V = \frac{R_2}{R_1 + R_2} 5V$$

Por lo tanto si queremos que $V_{ref} = 5V$ entonces $R_1 = R_2$. Mientras que los divisores para generar las señales que encienden el *led* verde y el rojo quedaron de la siguiente manera:

Para el *led* verde:

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{SS} \rightarrow 2.5V = \frac{R_2}{R_1 + R_2} 7.5V \quad \therefore \quad R_1 = 2R_2$$

Para el *led* rojo:

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{SS} \rightarrow 2.5V = \frac{R_2}{R_1 + R_2} 7V \quad \therefore \quad R_1 = 1.8R_2$$

Para encender el *led* amarillo, cuando el voltaje esta entre 7V y 7.5V, se procede le la misma forma que en el otro medidor, es decir, simplemente aplicando una operación NOR entre las señales que activan los *leds* verde y el rojo. La figura 4.8 muestra el diagrama lógico del medidor de baterías digital.

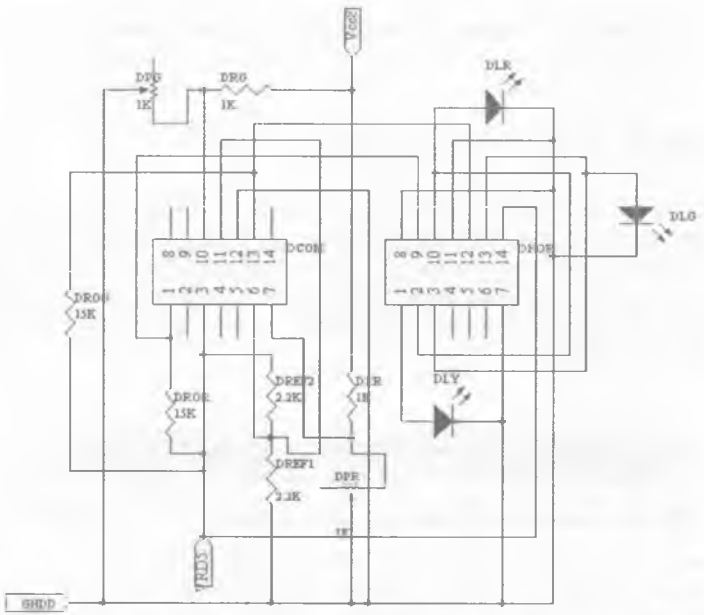


Figura 4.8 - Diagrama lógico Medidor de Baterías Digital

4.1.3.5 Suministro para el DSP

La tensión y corriente eléctrica para el DSP (5V y 500mA) se obtienen del suministro digital de 8V una vez que son regulados a 5V.

El principal componente de este sistema es sólo un conector que entrega al DSP el voltaje de las baterías digitales regulado a 5V.

Figura 4.9 - Componentes para el suministro de energía del DSP

4.1.3.6 Diagrama Eléctrico para el Suministro de Energía

Una vez establecidos los voltajes para alimentar al robot, tanto en la parte digital como en la analógica, y diseñados los medidores de baterías, se procedió a integrar todos los sistemas en un solo diagrama, incorporando un interruptor maestro que enciende a apaga el robot como se muestra en la figura 4.10. A través de los conectores CON_SW_AN y CON_SW_DIG, la energía de las baterías pasa a los circuitos que harán uso de la misma.

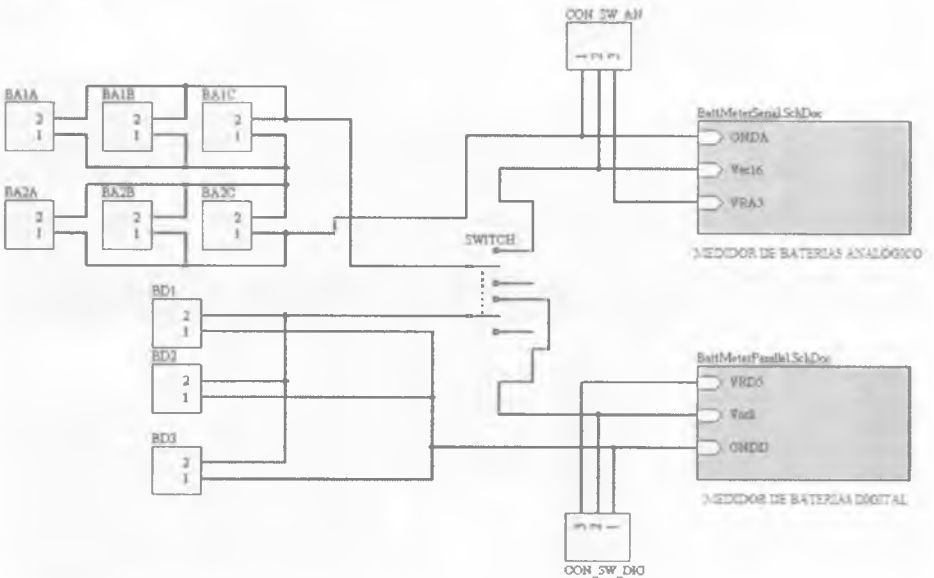


Figura 4.10 - Diagrama Lógico para el Suministro de Energía

4.1.4 Módulo Digital

En el módulo Digital las señales de los circuitos están codificadas en dos únicos estados 1 y 0, "1" refiriéndose a que en un circuito eléctrico hay tensión de voltaje (V_{CC_D}) o "0" cuando hay ausencia del mismo (GND_D).

4.1.4.1 Identificador del Robot

Éste sistema se conforma por un interruptor giratorio (SWID) mediante el cual se le asigna un identificador al robot. SWID proporciona una señal lógica de 4 bits (ID0, ID1, ID2 e ID3) que representan el complemento del código BCD, el cual es leído y codificado por el DSP para después entregarlo a un controlador (7SEGDRV) a través de las terminales SEGA, SEGB, SEGC y SEGD. Dicho controlador se encarga de traducir el código BCD en 7 señales (SA, SB, … SG) que nos permitirán ver el número asignado al robot en un display de 7 segmentos (DSPLY). La figura 4.11 nos muestra como funciona el interruptor.

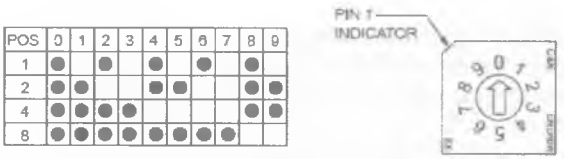


Figura 4.11 - Funcionamiento del interruptor giratorio

Para que el DSP pueda codificar la señal del interruptor debemos indicarle la correspondencia entre el código BCD Complemento y el número respectivo, así como la correspondencia entre número y el código BCD Natural que se envía al controlador del display como se muestra en la tabla 4.3.

Identificador	Código leído del interruptor (BCD Complemento)	Código entregado por el DSP al controlador del Display (BCD Natural)
0	1111	0000
1	1110	0001
2	1101	0010
3	1100	0011
4	1011	0100
5	1010	0101
6	1001	0110
7	1000	0111
8	0111	1000
9	0110	1001

Tabla 4.3 - Códigos para la identificación del robot

La figura 4.12 muestra los componentes de este sistema y la forma en que se conectan. Cabe mencionar que los elementos CONDSP1 y CONDSP2 son los conectores que permiten la comunicación del DSP con los demás circuitos a través de sus puertos de entrada y/o salida.

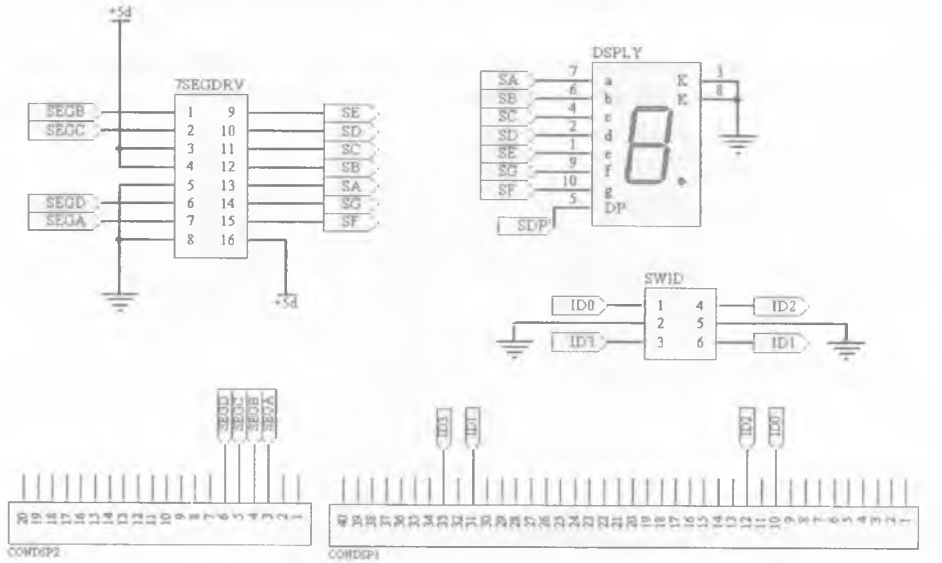


Figura 4.12 - Diagrama Lógico del Identificador del Robot

4.1.4.2 Transceiver

Al igual que en la generación anterior se utilizó un dispositivo de *Radiometrix* (RPC-914/869-64) que consta de un radio transmisor/receptor de UHF con frecuencias de 914MHz o 869MHz y un controlador de paquetes de datos de 64Kbit/s, la figura 4.13 muestra el dispositivo utilizado, que funciona con base en el RPC descrito en la sección 2.3.2.1.

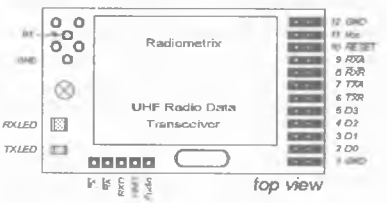


Figura 4.13 – Transceiver de Radiometrix

Desde los robots EK2005, se diseño un sistema de comunicación bidireccional, de modo que los robots también pudieran enviar información hacia IA. Esto es posible gracias a un adaptador de voltaje bidireccional. El funcionamiento de dicho adaptador es sencillo: las señales que recibe el *transceiver* tienen un voltaje de 5V, al pasar por el adaptador las señales se ajustan a 3.3V para ser entregadas al DSP; en el sentido inverso, las señales del DSP de 3.3V al pasar por el adaptador salen hacia el *transceiver* con 5V.

El diseño de este sistema únicamente implica hacer las conexiones entre el *Transceiver*, el adaptador de voltaje y el DSP, con un como se muestra en la figura 4.14

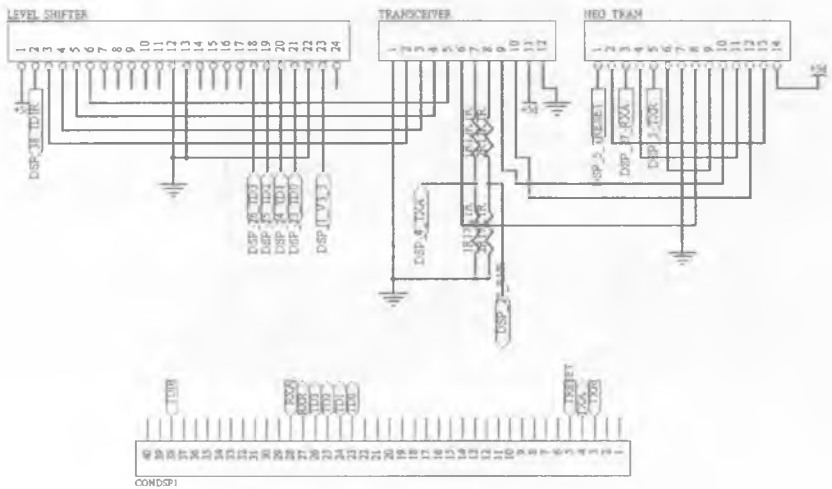


Figura 4.14 - Diagrama lógico del *Transceiver*

4.1.4.2.1 La Trama de Comunicación

En la sección 2.3.2.1 se describió como se envía la información por bytes a través de *Transceiver*, ahora se describe el tipo de datos que deben recibir los robots.

Se debe considerar que IA decide la posición y orientación futura de cada uno de los robots, el movimiento que cada robot debe hacer es distinto y por lo tanto, el módulo de comunicación debe encargarse de que cada robot reciba sólo la información que le corresponde. También es importante conocer la cantidad de información que cada robot recibe y para ello basta considerar que para que el robot llegue a su posición futura debe mover sus motores a una determinada velocidad, y si es necesario deberá activar los dispositivos de control y pateo de la pelota. Entonces la cantidad de información que IA debe enviar a cada robot se puede agrupar en dos bloques: control y vector de movimiento.

Control: En esta sección se ubica la información relativa al estado de los dispositivos de control y pateo de la pelota, así como el sentido de los vectores de movimiento. El estado de cada dispositivo puede ser encendido o apagado y la dirección de un vector puede ser positiva o negativa y por lo tanto, el estado de cada dispositivo y la dirección de cada vector se puede representar con un bit por dispositivo, con excepción del pateo ya que usa tres bits para manejar 7 intensidades de pateo.

Vector de movimiento: A diferencia de la generación anterior, en la que se enviaban las velocidades de los 4 motores (v_1, v_2, v_3, v_4), ahora en este bloque se especifica la magnitud de las componentes del vector de movimiento en que deberá desplazarse el robot (v_x, v_y, ω). Si se asume que una resolución de 256 velocidades diferentes es suficiente, entonces se necesitan 8 bits para las magnitudes de cada vector.

Por lo anterior, el paquete de información que cada robot recibe tiene la estructura que se muestra en la tabla 4.4.

Bloque	Dispositivo	Cantidad de información
Control	Control de pelota	1 bit
	Pateo de pelota	3 bits
	Dirección Componente Vx	1 bit
	Dirección Componente Vy	1 bit
	Dirección Componente ω	1 bit
Vector de Movimiento	Magnitud Componente Vx	8 bits
	Magnitud Componente Vy	8 bits
	Magnitud Componente ω	8 bits

Tabla 4.4 - Estructura del paquete de información para un robot.

De la tabla anterior se puede asumir que si se reserva un byte para el bloque de control, entonces el tamaño del paquete para un robot es de 4 bytes. Cada paquete debe ser enviado a cada uno de los robots con su información correspondiente, de esta manera se forma una trama. La trama debe contener un bloque de control para especificar el tamaño de la trama y otro bloque con la información de los robots. La tabla 4.5 y la figura 4.15 muestran la estructura de la trama del módulo de comunicación.

Control	Robot1	Robot2	Robot3	Robot4	Robot5
1 byte	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes

Tabla 4.5 - Estructura de la trama del módulo de comunicación.

Por lo tanto el tamaño de la trama es de:

$$T = 5(4 \text{ bytes}) + 1 \text{ byte} = 21 \text{ bytes}$$

De esta manera IA, una vez que toma la decisión de la posición y la orientación futura y calcula velocidades, construye la trama y la envía por medio del módulo de comunicación.

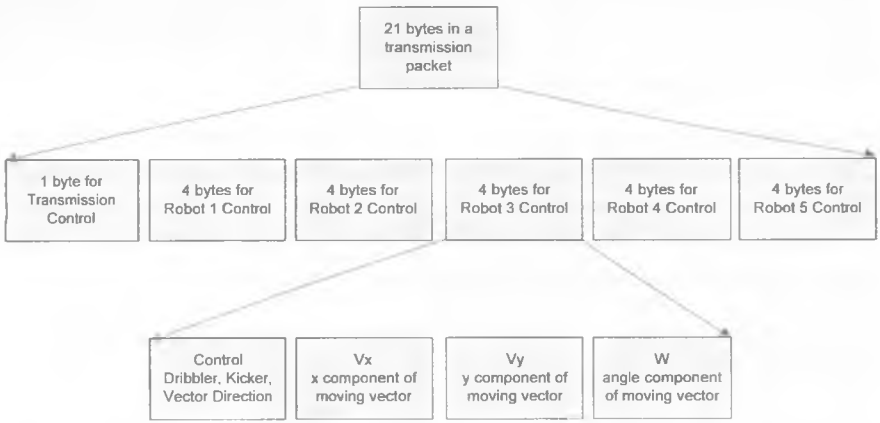


Figura 4.15 - Estructura de la Trama de Comunicación

4.1.5 Módulo Analógico

Este módulo se compone por todos los circuitos de potencia que demandan gran cantidad de voltaje y corriente, y con los cuales pueden funcionar los actuadores del robot: motores y solenoide.

4.1.5.1 Control de motores

En la sección 2.3.1.3 se describió una forma de controlar a los motores de corriente directa a través de los puentes H, para controlar el sentido de giro, y las señales de PWM, para controlar la velocidad de giro. Las señales PWM son calculadas y generadas por el DSP, mientras que los puentes H pueden construirse a partir de componentes discretos, pero también están disponibles como circuitos integrados; siendo ésta última opción más conveniente por cuestiones de espacio, se decidió utilizar el Circuito Integrado L6207D. Cada puente H del encapsulado es controlado por tres señales lógicas:

- Una para habilitar el puente H (EN) y
- Dos para indicar el sentido de movimiento (DIR1 y DIR2, respectivamente).

Para que el controlador opere las señales DIR1 y DIR2 siempre deber ser complementarias (una es la inversa de la otra). Las señales de control son generadas por el DSP dependiendo de las instrucciones que reciba de IA y comunicadas al controlador. La figura 4.16 muestra las conexiones entre el DSP y el L6207D con las señales EN, DIR1 Y DIR2.

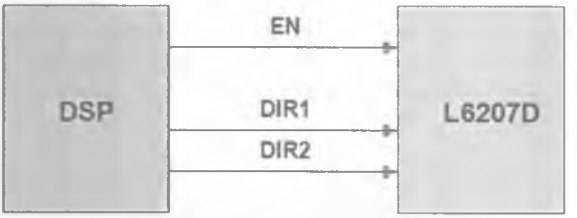


Figura 4.16 - Conexión entre el DSP y el L6207D

El funcionamiento del controlador se describe en la tabla 4.6 [39]:

Entradas		Función
EN = V_{CC_A}	DIR1 = V_{CC_A} DIR2 = GND_A	Adelante
	DIR1 = GND_A DIR2 = V_{CC_A}	Atrás
EN = GND_A	DIR1 = X DIR2 = X	Detenido
X = Cualquier valor V_{CC_A} o GND_A		

Tabla 4.6 – Funcionamiento del L6207D

Los circuitos L6207D permiten controlar a dos motores simultáneamente; sin embargo, para evitar llevarlos al límite se utilizó un encapsulado por cada motor.

Para controlar un solo motor con un L6207D se configuran los dos puentes en paralelo según lo indicado en la documentación del circuito. La conexión en paralelo supone que los dos puentes recibirán las mismas señales para que operen con las mismas funciones, las cuatro salidas se asignan a un mismo motor. De este modo, los dos puentes proporcionan el doble de corriente a un motor.

La figura 4.17 muestra la configuración del L6207D para ese propósito, en ella se pueden observar tres señales de entrada al controlador (IN1_DSP, IN2_DSP y EN_GEN) y dos señales de salida (OUT1_MOT y OUT2_MOT). Las tres primeras son las señales de control DIR1, DIR2 y EN respectivamente, mientras que las otras dos son las salidas del controlador que van directo al motor.

Se había mencionado que las señales DIR1 y DIR2 deber ser complementarias, para no desperdiciar puertos de entrada/salida del DSP ocupando uno por señal, se implementó un circuito inversor, como se muestra en la figura 4.18, de esta forma basta una señal de dirección para generar su complemento.

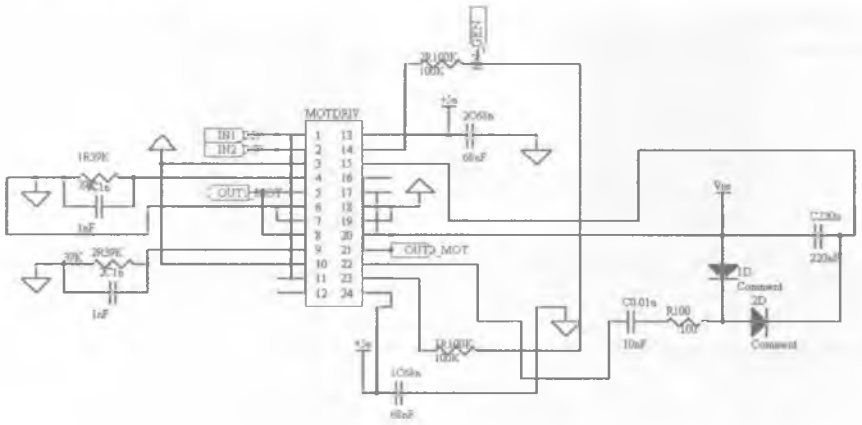


Figura 4.17 - Diagrama lógico para el controlador de los motores

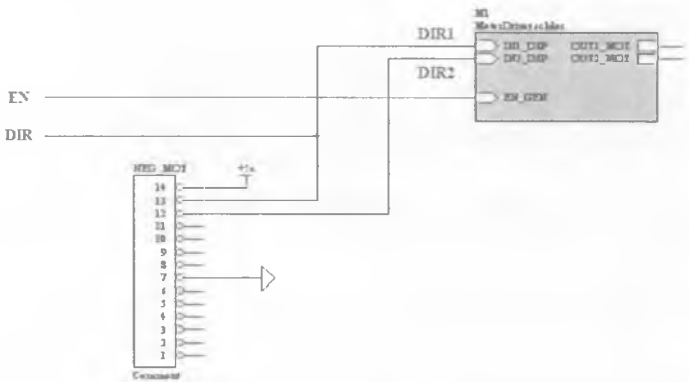


Figura 4.18 - Generación de Señales de Dirección Complementarias

4.1.5.2 Optoacopladores

Las señales que más nos interesa acoplar son las de control de motores, es decir, las señales entre el DSP y el L6207D (EN, DIR1 y DIR2). La figura 4.19 muestra el uso de una etapa de acoplamiento para proteger al DSP.

Figura 4.19 - Acoplamiento entre DSP y L6207D

Este tipo de acopladores los podemos encontrar en un encapsulado, para nuestro propósito se seleccionó el 740L6010. El 740L6010 está formado por un diodo emisor infrarrojo y un fototransistor de silicón empacados en un chip de 6 patas. La figura 4.20 muestra la conexión de los encapsulados con las señales que acoplaran (EN y DIR).

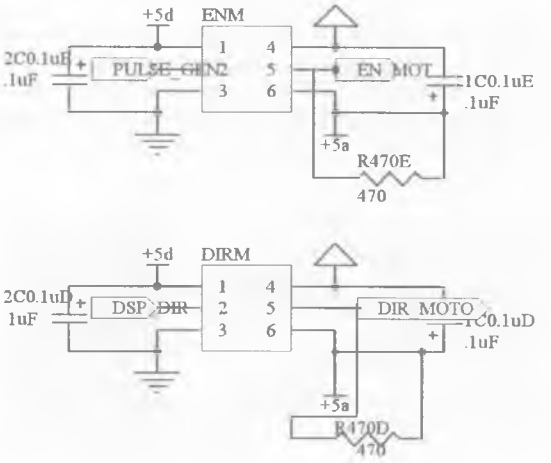


Figura 4.20 - Diagrama lógico para los optoacopladores (Controladores de Motores)

Este mismo diseño se aplica para otras señales como las del sistema de pateo y el sistema de control de pelota, que también utilizan el voltaje de la parte analógica, figura 4.21.

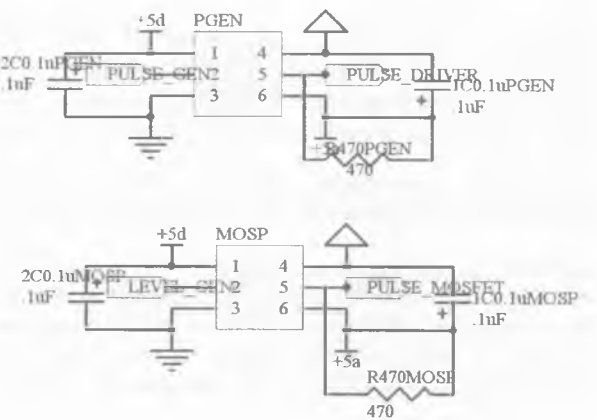


Figura 4.21 - Diagrama lógico para los optoacopladores (Sistema de Pateo)

4.1.5.3 Codificadores de motores (Encoders)

Los motores utilizados en la generación EK2007 cuentan con un codificador de dos canales (CHANNEL_A y CHANNEL_B) cuyo funcionamiento es similar al descrito en la sección 2.3.6, y requieren un voltaje de 5V DC para su operación.

Las señales de los canales del codificador deben estar conectadas al DSP para calcular la velocidad real del motor. La salida de cada canal del codificador es una señal cuadrada de 5Vpp. El DSP trabaja con señales de 3.3V por lo que fue necesario utilizar un divisor de voltaje para reducir la amplitud de la señal de los canales a 3.3V. La figura 4.22 muestra la conexión entre los *encoders* y el DSP.

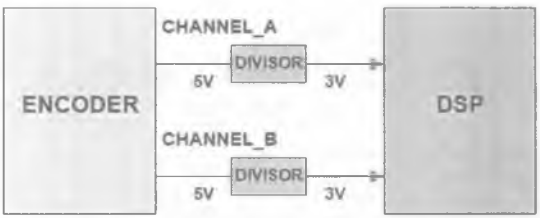


Figura 4.22 - Conexión entre *encoder* y DSP

Para poder disminuir la diferencia de potencial desde el *encoder* hacia el DSP se utilizaron dos resistencias por canal una de 17KΩ y otra de 12KΩ. Utilizando la expresión para el divisor de voltaje tenemos:

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in} = \frac{17K\Omega}{(17+12)K\Omega} 5V = 2.93V$$

Se decidió utilizar un voltaje un poco menor a los 3.3V (2.93V) que utiliza el DSP para dar un margen de tolerancia, en caso de que tengamos variaciones bruscas en señal de 5V.

Los motores se unen a los circuitos mediante un conector de 6 vías (CONM). Cada pin del conector tiene una función específica: los pines 1 y 2 (IN2_CON, IN1_CON) reciben las salidas del controlador del motor (OUT1_MOT y OUT2_MOT de la figura 4.17), los pines 3 y 4 (IN_TO_+5d e IN_TO_GND) son para alimentar al *encoder* integrado en el motor, y los pines 5 y 6 (CHANNEL_A y CHANNEL_B) entregan las señales del codificador, figura 4.23.

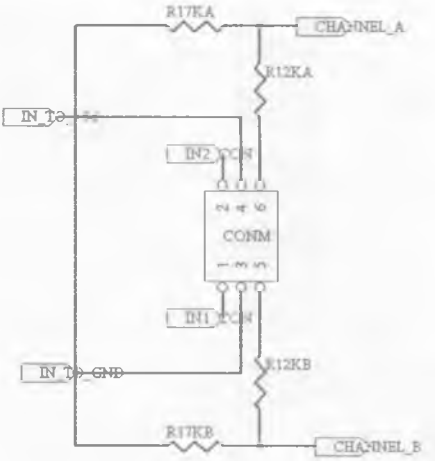


Figura 4.23 - Conector de Motores

4.1.5.4 Sistema de Control de pelota

Se había mencionado que este sistema utiliza un motor que hace girar un rodillo de algún material que brinde adherencia a la pelota. Dicho motor es de características similares a los utilizados para las ruedas, por lo tanto usamos el mismo tipo de controlador descrito en la sección 4.1.5.1, así como un acoplador para recibir la señal del DSP que lo activa.

4.1.5.5 Diagrama eléctrico para los módulos Digital y Analógico

Todos los diseños anteriores, tanto digitales como analógicos, fueron agrupados en un solo esquema, figura 4.24, en el caso de los circuitos para el control de motores el diseño se replicó cinco veces.

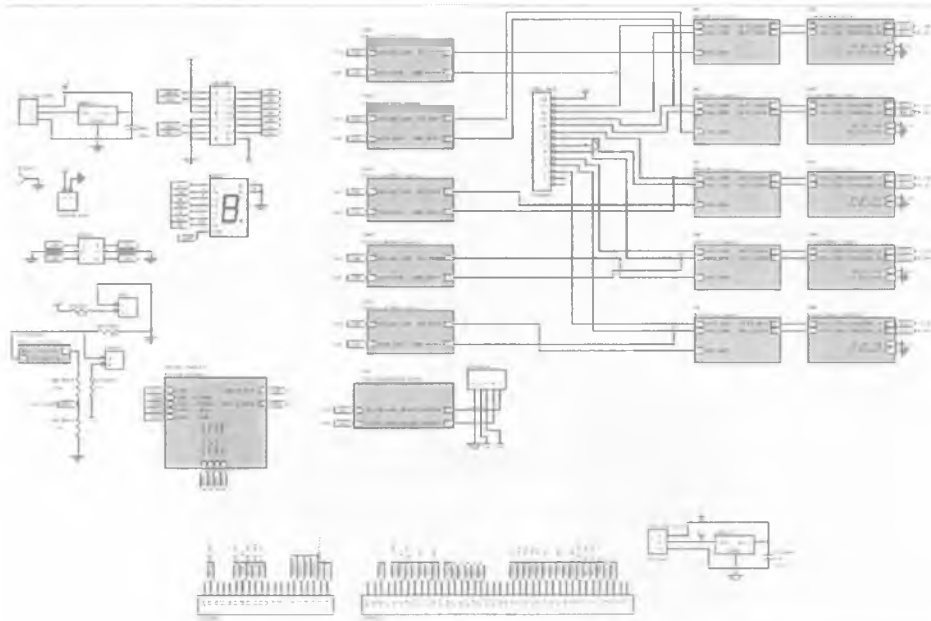


Figura 4.24 - Diagrama Lógico para los Módulos Digital y Analógico

4.1.5.6 Sistema de Pateo

Por lo complicado e inestable de los transformadores, la generación EK2007 dejó de utilizarlos. La alternativa a dichos dispositivos fue un convertidor DC-DC [32] (PicoElectronics IRF100S).

El convertidor DC-DC es un dispositivo que cuando esta encendido y recibe a la entrada un voltaje de 5-15V en corriente directa (DC), lo eleva hasta 100V también en corriente directa (DC), de ahí su nombre. La figura 4.25 muestra el convertidor que se utilizó como generador de alto voltaje en el sistema de pateo de la generación EK2007.

El dispositivo adquirido cuenta con un pin de control para activar o desactivar la conversión. Si el DC-DC no es desactivado continúa elevando el voltaje, con el consecuente consumo de baterías, por lo tanto fue necesario utilizar un interruptor capaz de cambiar su estado al recibir una señal digital (DSP_CHRG). Para esta tarea se optó por un relevador o *relay*.

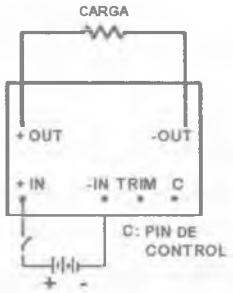


Figura 4.25 - Convertidor DC-DC

Un *relay* es un dispositivo electromagnético que controla el estado de un interruptor mediante una señal eléctrica, en nuestro caso de 5V. Existen relevadores con interruptores normalmente abiertos y normalmente cerrados. Para el caso de los primeros, normalmente abiertos, si no reciben la señal de control el switch permanece abierto, impidiendo la circulación de la corriente, pero cuando reciben la señal eléctrica cambian su estado a cerrado, con lo cual la corriente empieza a fluir; en cambio, los normalmente cerrados, siempre permiten el flujo de la corriente mientras no reciban la señal de control. En nuestro caso no queremos que el convertidor DC-DC funcione todo el tiempo, por lo tanto se utilizó un relevador normalmente abierto.

Con el convertidor funcionando cada vez que se desee, el siguiente paso es almacenar la carga que produce en un par de capacitores, los cuales deben ser lo suficientemente grandes para trabajar con el voltaje de 100V. Los condensadores elegidos poseen una capacitancia de 2200μF y pueden recibir hasta 200V, además se conectan en paralelo para obtener una capacitancia neta de 4400μF. Cuando los condensadores están completamente cargados el convertidor debe dejar de funcionar, el tiempo que le toma a los capacitores cargarse hasta 100V es aproximadamente 20 segundos. Transcurrido dicho tiempo el convertidor debe dejar de funcionar y activarse hasta que se le indique, generalmente después de la etapa de descarga que se describe a continuación.

En la etapa de descarga el voltaje de los capacitores se envía al solenoide, es importante que el robot tenga la pelota al momento de la descarga y no se desperdicie energía al ejecutar una descarga en falso, para eso la etapa de activación esta compuesta por otras dos etapas: Sensado y activación. De esta manera el solenoide se activará solamente si los sensores indican que el robot tiene la pelota y si la etapa de activación así lo pide.

En el sensado se detecta que el robot tenga la pelota. En la activación el DSP envía la señal para que el robot active el interruptor de descarga (a través de la señal DSP_KICK).

Para implementar el sensor de pelota se utilizó un diodo emisor de luz infrarroja y un fototransistor. El led y el fototransistor están colocados debajo del dispositivo de control de

pelota. El voltaje de alimentación se obtiene de V_{CC_D}. Se utiliza una resistencia de 330Ω entre V_{CC_D} y el ánodo del led para evitar que se queme. El cátodo se conecta a GND_D.

El colector del fototransistor también se conecta a V_{CC_D} y hay un resistencia de 220KΩ entre estos dos puntos. El emisor se conecta directamente a GND_D. La señal de salida se toma en el colector después de la resistencia, de esta manera se tiene V_{CC_D} cuando el fototransitor está saturado (recibe la señal del *led* infrarrojo), y GND_D cuando el fototransistor está en corte (cuando la pelota obstruye la línea de vista entre el *led* infrarrojo y el fototransistor). La activación del sistema de pateo se realiza por software, de modo que la señal del sensor llega al DSP para que el programa del robot tome la decisión de golpear. Sólo por seguridad del DSP, la señal que indica la presencia de la pelota pasa a través de un optoacoplador y posteriormente por un divisor de voltaje para reducir la señal a 3.3V. La figura 4.26 muestra el diagrama lógico del sensor de pelota, LED y FTRAN corresponden al *led* infrarrojo y al fototransistor respectivamente.

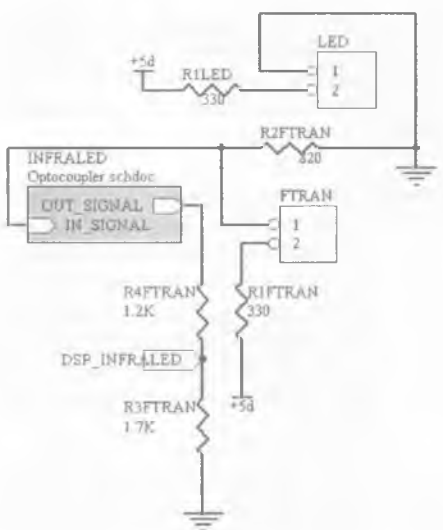


Figura 4.26 - Sensor de Pelota

Para tener diversas intensidades de pateo, se utilizó un control electrónico de intensidad de la corriente de descarga, por medio de un MOSFET. La modulación de la intensidad se realizó con una señal PWM en el canal de activación del DSP hacia el MOSFET. La señal de activación permite utilizar hasta siete distintas intensidades de pateo. La figura 4.27 muestra los bloques del circuito de activación de disparo de pelota.

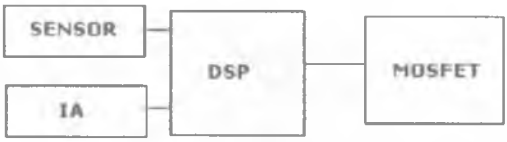


Figura 4.27 - Bloques del circuito de activación de disparo de pelota

4.1.5.7 Diagrama Eléctrico para el Sistema de Pateo (*Kicker*)

Durante las etapas de carga y descarga el flujo de energía es muy elevado, por lo tanto fue conveniente separar el circuito del sistema de pateo de los demás. La figura 4.28 muestra el diagrama eléctrico de este sistema.

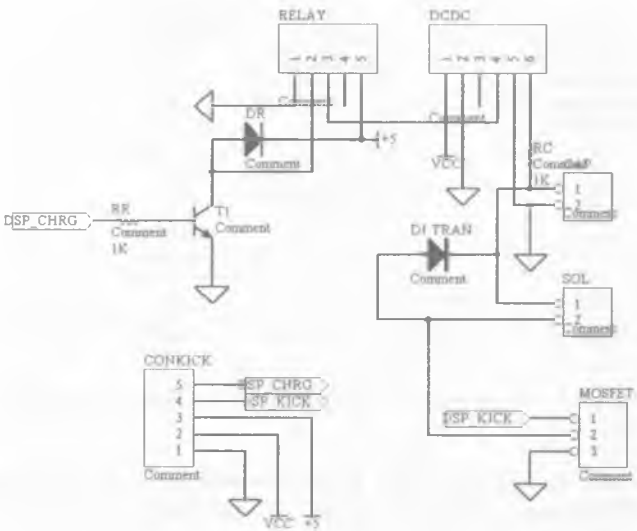


Figura 4.28 - Diagrama lógico del sistema de pateo

4.1.6 DSP

El DSP es considerado en el diseño de los robots por las implicaciones en la programación del mismo, ya que el programa residente en su memoria le permite al robot interpretar y ejecutar los comandos de IA.

El diagrama de flujo del programa principal del DSP se muestra en la figura 4.29.

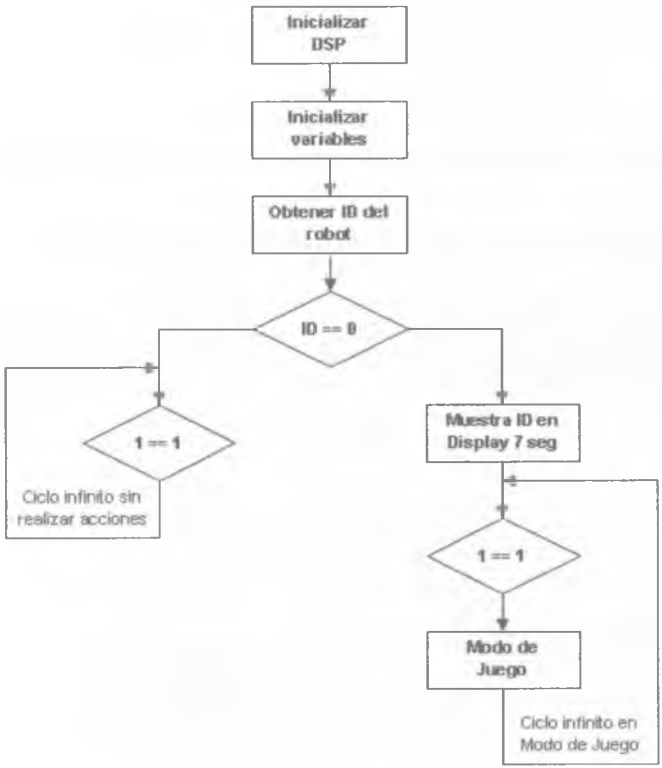


Figura 4.29 - Diagrama de Flujo del Programa Principal del DSP

Tal como lo muestra el diagrama, antes de realizar cualquier acción relacionada con el control del robot es necesario inicializar el DSP, que consiste en activar los periféricos de los que hará uso. Posteriormente se establecen los valores iniciales para las variables de control, por ejemplo las velocidades de los motores en cero. El siguiente paso, en caso de tener identificado al robo, es entrar en un ciclo infinito denominado Modo de Juego. En este ciclo el robot esta listo para recibir y ejecutar las instrucciones de IA. El diagrama de flujo de la figura 4.30 muestra más a detalle en que consiste dicho ciclo.

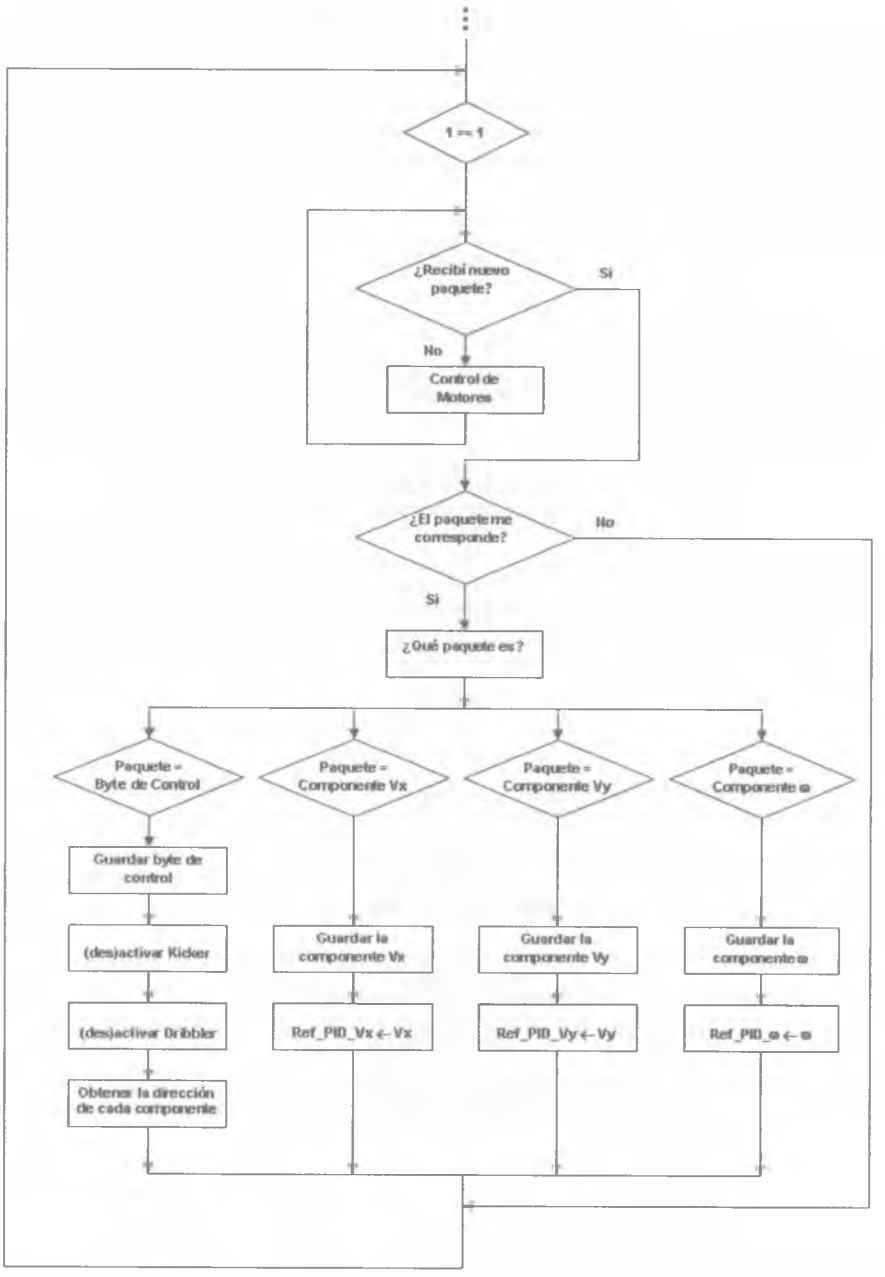


Figura 4.30 - Diagrama de flujo del Modo de Juego

En el Modo de Juego el DSP tiene que obtener, de su sistema de comunicación, sólo los paquetes que le corresponden, para interpretarlos y traducirlos en señales para los circuitos. Mientra no recibe una nueva instrucción el DSP debe controlar a los motores con la última trayectoria recibida. Durante el Control de Motores se generan las señales PWM que activarán los motores con su respectivo sentido de giro (EN, DIR1 y DIR2), además, si es necesario, actúan los controladores PID, como se muestra en la figura 4.31.

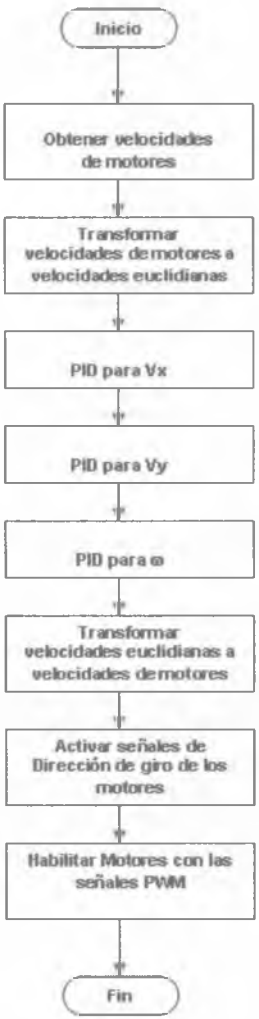


Figura 4.31 - Diagrama de Flujo para el Control de Motores

Como se describió en la sección 2.3.1.4, los controladores PID reciben dos señales para hacer la corrección: una señal que representa el *punto actual* (PV) en el que se encuentra el sistema, en nuestro caso la velocidad actual de cada motor; y una señal que representa el valor que se desea alcanzar (SP), la velocidad de cada motor resultado de transformar la velocidades euclidianas a velocidades de motores. El controlador PID resta la señal de *punto actual* a la señal de *punto de referencia*, obteniendo así la señal de *error*, que determina en cada instante la diferencia que hay entre el valor deseado y el valor medido. La señal de error es utilizada por cada una de las 3 componentes de un controlador PID para generar 3 señales que, sumadas, componen la señal que el controlador va a utilizar para gobernar a los motores.

La señal PV es calculada mediante las señales que proporcionan los *encoders,* las cuales son procesadas prácticamente en paralelo con la ejecución del programa principal mediante las llamadas interrupciones (señales recibidas por el DSP indicando que debe "interrumpir" el curso de ejecución actual y pasar a ejecutar código específico para tratar esta situación).

4.1.7 La generación EK2008

La generación EK2008 no necesita un capítulo exclusivo para su descripción, ya que el cambio más significativo entre esta generación y su antecesora fue el material con que se fabricó la estructura mecánica de los robots y un ajuste en el consumo de energía.

4.1.7.1 Nueva Estructura Mecánica

En algunos partidos del *RoboCup* 2007, la potencia de disparo de los contrincantes produjo daños en la estructura mecánica de nuestros robots, principalmente en los soportes del sensor de la pelota, lo que impedía que algunos de nuestros robots golpearan la pelota o bien realizaban descargas en falso. Este problema nos condujo a la necesidad de cambiar las partes plásticas del robot por partes fabricadas en algún material mucho más resistente.

El diseño mecánico empleado en los robots EK2007 siguió siendo el mismo para los del 2008 con la diferencia que en lugar de plástico se utilizó aluminio para su fabricación, dicho material, además de darle mayor rigidez, ayuda a disipar el calor generado por los motores. Sin embargo, se hicieron algunos cambios para ajustar la altura del robot, los cuales repercutieron en el espacio para alojar el solenoide, ante este problema no hubo más opción que cambiar el solenoide por uno más pequeño.

4.1.7.2 Consumo de Energía

En lo relacionado con la electrónica, después del *RoboCup* 2007 la mayoría de los robots continuaban funcionando correctamente, por lo tanto, para el *RoboCup* 2008 se decidió

utilizar la misma arquitectura pero con algunos ligeros cambios, relacionados principalmente con el consumo de energía. Este cambio fue motivado por dos razones:

En primer lugar, las baterías de los robots 2007 sólo duraban el primer tiempo del partido, principalmente las baterías del suministro analógico, después del cual había que cambiarlas por un juego nuevo. Para no perder tiempo en el cambio de baterías, se vio conveniente cambiarlas por algunas de mayor duración. El segundo motivo fue el más decisivo, ya que en las nuevas reglas se incrementó el tamaño de la cancha, figura 4.32, y por lo tanto, al tener que recorrer una mayor distancia, las baterías tendrían un desgaste mayor.

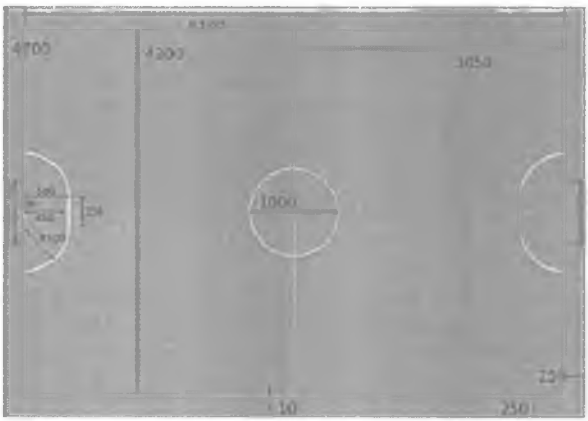


Figura 4.32 - Dimensiones de la cancha de juego para el *RoboCup* 2008

Siendo así, las seis baterías de la parte analógica, agrupadas como se mencionó en la sección 3.2.3.1, fueron sustituidas por cuatro que proporcionan 2100mAh, esto quiere decir que si la baterías consumen una corriente constante de 2.1A durarán 1 hora.

4.2 Control de Trayectorias

En esta parte del documento, se trata de mostrar como eliminar el ajuste manual de los parámetros PID implementando la técnica de Aprendizaje por Refuerzo con Gradiente de la Política propuesta en [18] y [23]. A grandes rasgos, se explica como los parámetros óptimos para varios controladores PID pueden ser aprendidos de forma adaptiva tras conducir al robot en su entorno mientras se evalúa su comportamiento.

4.2.1 Aprendiendo los Parámetros PID

Tal y como se mencionó en la sección 2.5.1.7, los algoritmos basados en PGRL consisten en tres pasos [15]:

1. Parametrizar la Política. (Elegir π_θ y θ_0)
2. Calcular el Gradiente. $\nabla\eta(\theta)$
3. Mejorar los parámetros ajustándolos en la dirección del gradiente. $\theta_{m+1} = \theta_m + \alpha_m \nabla\eta(\theta_m)$

Estos mismos pasos son seguidos para aprender los parámetros PID. La idea principal asume que, aunque están correlacionados, los parámetros PID pueden variar independientemente. Siendo así, es posible modificar aleatoriamente un conjunto de parámetros, calcular el error parcial derivado de cada uno de ellos y corregir los valores, de tal forma que mejoren el rendimiento y precisión de la función de valor. Buenos resultados refuerzan buenas combinaciones, un mal desempeño penaliza la combinación. Esta técnica encuentra los parámetros que permiten al robot conducirse con el menor error.

4.2.1.1 Parametrizar la Política

El método considera cada posible conjunto de parámetros PID como la definición de la política que puede ser ejecutada por el robot:

$$\pi = \{p_1, \ldots, p_N\}$$

El número de parámetros N es independiente del número de ruedas del robot, ya que se usa un controlador PID por cada grado de libertad y no por cada motor. Para los robots EK2007-EK2008, el conjunto de parámetros π consiste en 9 elementos que corresponden a las constantes proporcional, integral y derivativa de cada controlador PID. Considerada de tal forma, la política parametrizada quedó como sigue:

$$\pi = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$$

Donde:

Parámetro	Descripción
$p_1 = K_p$ $p_2 = K_i$ $p_3 = K_d$	Parámetros para el controlador de la componente V_x
$p_4 = K_p$ $p_5 = K_i$ $p_6 = K_d$	Parámetros para el controlador de la componente V_y
$p_7 = K_p$ $p_8 = K_i$ $p_9 = K_d$	Parámetros para el controlador de la componente ω

4.2.1.2 Evaluar las Políticas y Calcular el Gradiente

El siguiente paso es estimar la derivada parcial de la función objetivo de π con respecto a cada parámetro, éste cálculo servirá para determinar en que dirección es más favorable ajustar los parámetros. Para conseguirlo, primero generamos t políticas aleatorias θ^j cerca de π:

$$\theta^1 = \{p_1 + \Delta_1^1, ..., p_9 + \Delta_9^1\}$$
$$\theta^2 = \{p_1 + \Delta_1^2, ..., p_9 + \Delta_9^2\}$$
$$\vdots$$
$$\theta^t = \{p_1 + \Delta_1^t, ..., p_9 + \Delta_9^t\}$$

En las cuales, el valor Δ_i^j es elegido con probabilidad uniforme del conjunto $\{-\varepsilon_i, 0, +\varepsilon_i\}$ y donde ε_i es una constante pequeña en relación con p_i.

En seguida se avalúa el comportamiento del robot para cada política θ^j. Para esto, los parámetros de los 3 controladores PID del robot se actualizan con los del conjunto θ^j. La evaluación de un conjunto de parámetros θ^j consiste en una sencilla prueba:

Con los nuevos parámetros, el robot tiene que acelerar desde el reposo en una dirección determinada y con cierto ángulo respecto a su orientación. Debe avanzar por un intervalo de tiempo constante, sin dar vueltas ni desviarse mucho del punto de partida. Luego, el robot debe ser detenido abruptamente tan rápido como sea posible. Durante esta fase de la prueba, el robot no debe recibir ninguna información de retroalimentación ya sea del Sistema de Visión o del de Inteligencia Artificial.

La evaluación de cada una de las políticas genera una puntuación que es una medida del desempeño del robot descrito por ese conjunto de parámetros. La puntuación está determinada por la función de calidad $Q(\theta^j)$, la cual es una función ponderada por dos criterios: desviación del robot de su destino y rotación acumulada sobre el eje del robot.
La forma en que se calcula cada criterio se describe a continuación:

La desviación del robot de su destino puede ejemplificarse como sigue: supongamos que el robot se encuentra en el punto A y se le ordena desplazarse en línea recta hasta el punto B; sin embargo, la combinación de parámetros produce una mala corrección que se traduce en una trayectoria que lo desvía de su objetivo, llevándolo hasta el punto C. El ángulo formado por los tres puntos, tomando como vértice el punto A, nos da una estimación de la desviación del robot. Generalmente, este error es producido por una corrección inadecuada en los controladores de las componentes V_x (desplazamientos horizontales) y V_y (desplazamientos verticales).

La figura 4.33 muestra como es considerada la desviación; la línea punteada (del punto A al punto B) indica la trayectoria ideal que debería describir el robot, mientras que la línea sólida (del punto A al C) ejemplifica una trayectoria distinta a la ideal, producida a una mala combinación de parámetros.

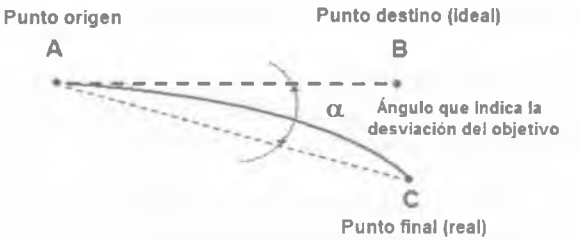


Figura 4.33 - Determinación de la Desviación del Robot de su destino

Ahora bien, supóngase que el robot parte del punto A con destino al punto B, además se le indica que debe llagar con la misma orientación con la que partió; sin embargo, el robot, además del error que lo llevó al punto C, también llegó con una orientación distinta a la que se le indicó. Ésta diferencia entre la orientación final deseada y la orientación final real, es lo que definimos como rotación acumulada, generalmente producida por una mala corrección de la componente ω. La figura 4.34 ejemplifica dicho error.

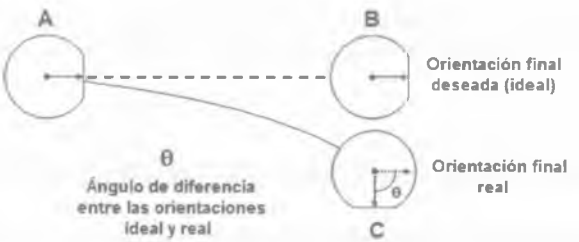


Figura 4.34 - Determinación de la Rotación Acumulada

Una vez determinada la desviación del robot y la rotación acumulada, se le asigna una puntuación a cada una, misma que esta en función del error medido y que toma valores en el intervalo [-1, 0], siendo -1 la peor calificación, correspondiente al máximo error (180° de diferencia en ambos casos), y 0 la mejor, que corresponde a un comportamiento sin errores (llegó al punto correcto con la orientación indicada). Por ejemplo, supongamos el caso extremo en el que al robot se le ordenó desplazarse del punto A al punto B, pero llegó al punto C, que se encuentra completamente del lado contrario al B, entonces la puntuación para la desviación de su destino sería -1; sin embargo, la orientación con la que llegó fue la misma que se deseaba, entonces la puntuación de la rotación acumulada sería de 0.

Aunque ambos criterios son importantes en la conducción del robot, para efectos prácticos, tiene mayor peso llegar al punto correcto sin desviarse, ya que la rotación del robot es mucho más fácil corregir. Por este motivo se mencionó que $Q(\theta^j)$ es una función ponderada por ambos criterios, quedando como sigue:

$$Q(\theta^j) = (\textit{Puntuación de la desviación}) * 0.8 + (\textit{Puntuación de la rotación acumulada}) * 0.2$$

Después de evaluar el comportamiento de cada θ^j, calculamos la derivada parcial de la función objetivo de π con respecto a cada parámetro. Lo hacemos agrupando cada θ^j en uno de tres conjuntos para cada dimensión:

$$\theta^j \in \begin{cases} S_i^{+\varepsilon} & si\ el\ i\text{-}ésimo\ parámetro\ de\ \theta^j es \quad p_i + \varepsilon_i \\ S_i^{+0} & si\ el\ i\text{-}ésimo\ parámetro\ de\ \theta^j es \quad p_i + 0 \\ S_i^{-\varepsilon} & si\ el\ i\text{-}ésimo\ parámetro\ de\ \theta^j es \quad p_i - \varepsilon_i \end{cases}$$

A continuación, se calcula una puntuación promedio $A_i^{-\varepsilon}$ A_i^{+0} y $A_i^{+\varepsilon}$ para $S_i^{-\varepsilon}$, S_i^{+0}, y $S_i^{+\varepsilon}$ respectivamente.

$$A_i^{+\varepsilon} = \frac{\sum_{R^j \in S_i^{+\varepsilon}} Q(\theta^j)}{\left|S_i^{+\varepsilon}\right|} \quad A_i^{-\varepsilon} = \frac{\sum_{R^j \in S_i^{-\varepsilon}} Q(\theta^j)}{\left|S_i^{-\varepsilon}\right|} \quad A_i^{+0} = \frac{\sum_{R^j \in S_i^{+0}} Q(\theta^j)}{\left|S_i^{+0}\right|}$$

Este cálculo nos proporciona un gradiente para cada parámetro. En otras palabras, estos tres promedios nos dan una estimación de los beneficios de alterar el i-ésimo parámetro por $-\varepsilon_i$, 0 ó ε_i.

4.2.1.3 Actualizar la Política en la Dirección indicada por el Gradiente

Si el gradiente no presenta ambigüedades, calculamos un nuevo valor para el parámetro de acuerdo con la ecuación:

$$p_i = \begin{cases} p_i + \eta_i & si \quad A_i^{+\varepsilon} > A_i^{+0} > A_i^{-\varepsilon} \quad ó \quad A_i^{+\varepsilon} > A_i^{-\varepsilon} > A_i^{+0} \\ p_i - \eta_i & si \quad A_i^{-\varepsilon} > A_i^{+0} > A_i^{+\varepsilon} \quad ó \quad A_i^{-\varepsilon} > A_i^{+\varepsilon} > A_i^{+0} \\ p_i & otro\ caso \end{cases}$$

Donde η_i es la constante de aprendizaje para cada parámetro. El viejo conjunto de parámetros π es remplazado por uno nuevo con los nuevos parámetros ajustados en la dirección más favorable, y el proceso sigue iterando mientras no se consigue un progreso.

El Pseudocódigo para el algoritmo *Policy Gradient* N-Dimensional [23], seguido en el aprendizaje de los parámetros PID, se muestra en la figura 4.35. Durante cada iteración del ciclo principal generamos t políticas cerca de π para estimar el gradiente alrededor de la misma, y luego moverla con una tasa η en la dirección más favorable.

$\pi \leftarrow$ *Política Inicial*
while !done **do**
    Generar t políticas aleatorias cerca de π: $\{\theta^1, \theta^2, \ldots, \theta^t\}$
    Evaluar las t políticas $\{\theta^1, \theta^2, \ldots, \theta^t\}$
    for $i = 1$ to N **do**
        $A_i^{+\varepsilon} \leftarrow$ *puntuación promedio para todo* θ^j *que tiene una perturbación positiva en la dimensión i*
        $A_i^{+0} \leftarrow$ *puntuación promedio para todo* θ^j *que tiene una perturbación igual a cero en la dimensión i*
        $A_i^{-\varepsilon} \leftarrow$ *puntuación promedio para todo* θ^j *que tiene una perturbación negativa in la dimensión i*
        if $A_i^{+\varepsilon} > A_i^{+0} > A_i^{-\varepsilon}$ or $A_i^{+\varepsilon} > A_i^{-\varepsilon} > A_i^{+0}$ **then**
        $p_i = p_i + \eta_i$
        else
            if $A_i^{-\varepsilon} > A_i^{+0} > A_i^{+\varepsilon}$ or $A_i^{-\varepsilon} > A_i^{+\varepsilon} > A_i^{+0}$ **then**
                $p_i = p_i - \eta_i$
            else
                $p_i = p_i$
            end if
        end if
    end for
    $\pi \leftarrow \pi_{ajustado}$
end while

Figura 4.35 - Pseudocódigo para el algoritmo de *Policy Gradient* N-dimensional.

- CAPÍTULO 5 -

IMPLEMENTACIÓN DE LOS ROBOTS

En este último capítulo se describe el proceso que se llevó a cabo para construir al equipo de robots EK2007. Además se explica como fue implementado el Algoritmo de PGRL en el Sistema de IA para obtener los parámetros óptimos de los controladores PID.

5.1 Implementación de los Robots EK2007

El punto de partida para el proceso de la construcción de los robots lo constituye la selección de los dispositivos que cumplen con las características establecidas en el diseño. El proceso de selección de dispositivos involucra una revisión de la documentación de sus fabricantes para conocer sus características y seleccionar al que cumpla con los requerimientos. Posteriormente se construyen circuitos de prueba de cada sistema con los elementos adquiridos para verificar su adecuado funcionamiento y liberar una versión final. Si el diseño es estable se fabrican las Tarjetas de Circuito Impreso.

5.1.1 Locomoción

Los motores elegidos para los robots EK2007-EK2008 fueron de la marca Faulhaber modelo 2224P0212, ya que son motores DC con bastante torque, muy compactos y resistentes. Las ecuaciones para transformar las velocidades euclidianas a velocidades de los motores y viceversa fueron implementadas en el programa del DSP siguiendo los diagramas de flujo de la sección 4.1.6.

5.1.2 Arquitectura Electrónica Modular

Para reflejar la modularidad descrita en los requerimientos, se trató de implementar cada módulo en una Tarjeta de Circuito Impreso, de tal forma que si fuera necesario cambiar algún sistema, no se tuviera que rediseñar toda la electrónica; con lo cual el diseño quedó plasmado en cuatro tarjetas como se muestra en la Figura 5.1:

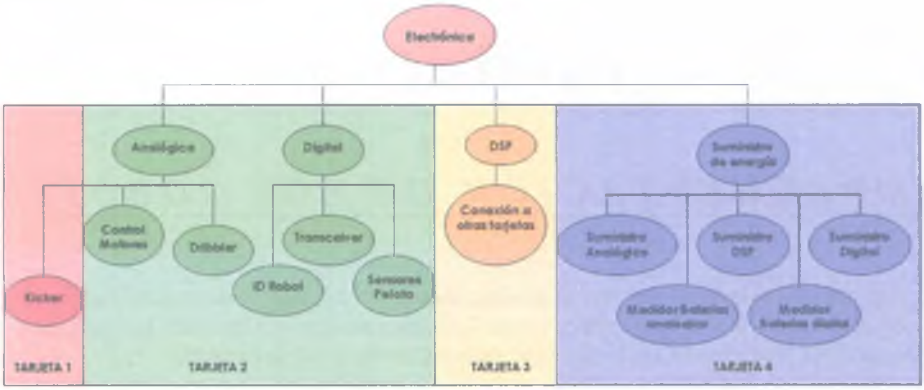


Figura 5.1 - Agrupación de los sistemas en las Tarjetas de Circuito Impreso

Cabe mencionar que el DSP (Tarjeta 3 en la figura 5.1) es un dispositivo electrónico que no fue diseñado ni construido en el Laboratorio de Robótica, éste es fabricado y comercializado por la empresa *Spectrum Digital Incorporated* [43], y su funcionamiento se basa en el procesador TMS320LF2812 de *Texas Instrument*.

Los diagramas eléctricos mostrados en las figuras 4.10, 4.24 y 4.28 fueron utilizados para fabricar las Tarjetas de Circuito Impreso (PCB por sus siglas en inglés) a través de un software de diseño electrónico llamado Protel 2004. Protel es un programa CAD de la empresa Altium para el diseño de circuitos electrónicos en su fase esquemática y el diseño del circuito impreso. Dicho programa permite generar automáticamente las pistas de la tarjeta a partir de un diagrama lógico. En este software también se especifican las dimensiones y forma de la tarjeta.

Los archivos generados en Protel, fueron enviados a una empresa estadounidense dedicada a la fabricación de tarjetas electrónicas, llamada PCBexpress [31]. Además de fabricar la tarjeta impresa, ofrecen el servicio para soldar los componentes que se les indique, este servicio resultó muy conveniente sobre todo por la dificultad que implica soldar los componentes de superficie.

La descripción de los componentes electrónicos seleccionados para la fabricación de las tarjetas se incluye en el Apéndice A. Cada componente se puede identificar en los diagramas del capítulo 4 mediante su etiqueta.

5.1.3 Suministro de Energía (Tarjeta de Circuito Impreso)

Por seguridad, todas las conexiones que abastecen de energía al robot se integraron en una tarjeta independiente de los módulos críticos para su control, en especial el DSP, ya que nos permite mantenerlos aislados en caso de corto circuito. La figura 5.2 muestra el diseño final de la tarjeta para el suministro de energía.

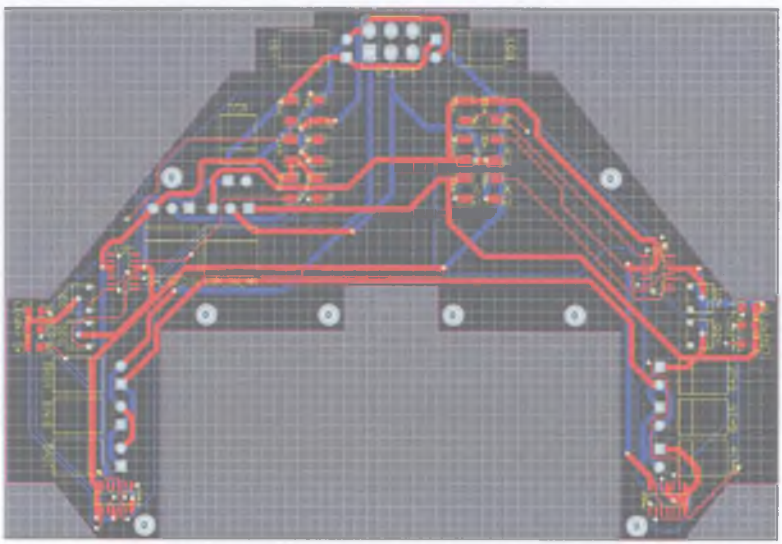


Figura 5.2 - PCB Suministro de Energía

5.1.4 Módulos Digital-Analógico (Tarjeta de Circuito Impreso)

A pesar que se intentó tener una tarjeta exclusiva para el Módulo Digital y una para el Módulo Analógico, las dimensiones del robot lo impidieron, ya que si se añadía una tarjeta extra la altura del robot rebasaría los 15cm permitidos, por tal motivo no quedó más opción que integrar los dos módulos en una sola tarjeta; sin embargo, se cuidó que los voltajes y referencias del los circuitos quedaran bien definidas para evitar confusiones. El resultado del la unión de los módulos fue la tarjeta de la figura 5.3.

La tarjeta Digital-Analógica integra los diversos módulos, por la parte digital: Identificador del Robot, *Transceiver* y Sensor de Pelota. Mientras que por la parte analógica agrupa: Control de Motores, incluyendo el del Sistema de Control de Pelota (*Dribbler*), y Optoacopladores.

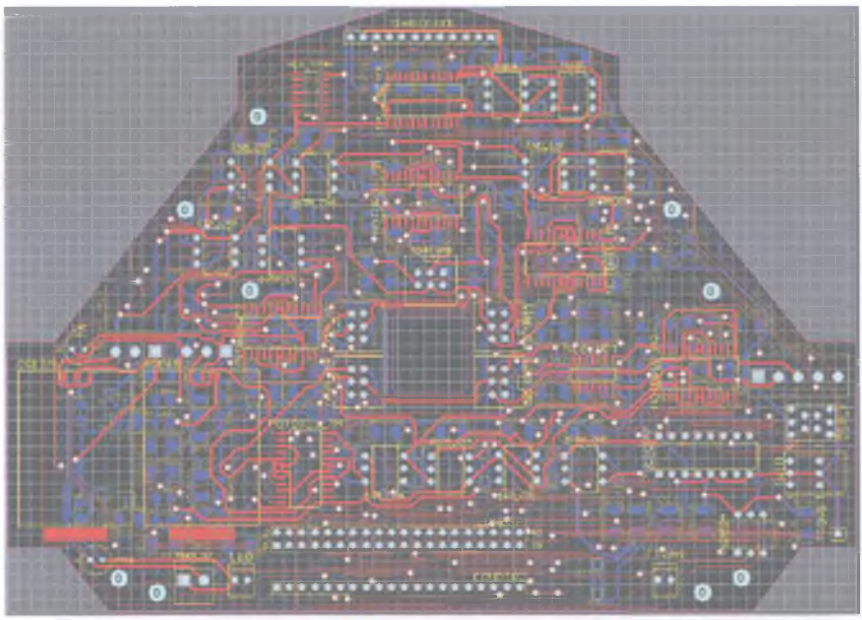


Figura 5.3 - PCB Digital-Analógica

5.1.5 Sistema de Pateo (Tarjeta de Circuito Impreso)

Históricamente el sistema de pateo ha presentado diversos problemas, por lo tanto, se decidió fabricar una tarjeta exclusiva para el mismo, ya que da mucha flexibilidad en caso de tener que cambiarlo por fallas o inefectividad, además permite mantener alejados a los circuitos de carga y descarga, que en algún momento podrían afectar a otros circuitos por el alto voltaje que circula en los primeros. A diferencia de las otras tarjetas, la de este sistema fue completada en el Laboratorio de Robótica, ya que los componentes a soldar eran muy pocos, figura 5.4.

La tarjeta sólo contempla los circuitos para las etapas de carga y descarga, ya que los circuitos del sensor de la pelota se incorporaron en la tarjeta Digital-Analógica.

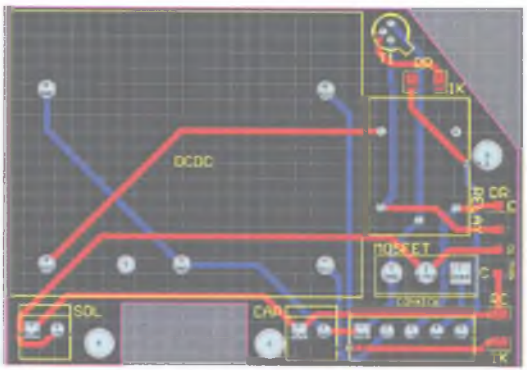


Figura 5.4 - PCB Sistema de Pateo

5.1.6 DSP

Como se mencionó en la sección 5.1.2, la tarjeta del DSP es fabricada por la empresa *Spectrum Digital*, el modelo de la misma es eZdsp TMF2812, figura 5.5.

Figura 5.5 - eZdsp TMF2812

El eZdsp TMF2812 es una tarjeta de evaluación que permite examinar el procesador digital de señales TMS320F2812 de *Texas Instrument*. Además, es una excelente plataforma para desarrollar y ejecutar el software para el procesador. Las características del eZdsp TMF2812 son las siguientes.

- DSP TMS320F2812 de *Texas Instrument*
- Velocidad de operación de 150 MIPS
- 18K words (1 word = 16 bits) de memoria RAM
- 128K words de memoria Flash
- 64K words memoria SRAM

- Reloj de 30 MHz.
- 2 conectores de expansión (I / O)
- Operación con 5 volts y 500 mA
- Puertos de Entrada/salida

El entorno de desarrollo esta formado por:

- *Code Composer Studio* (*Texas Instruments*)
- Compilador C
- Ensamblador
- Debugger sobre código fuente C
- Sistema operativo en tiempo real: DSPBIOS

Con las herramientas anteriores se codificaron, en lenguaje de programación C, los diagramas de flujo de la sección 4.1.6. El programa resultante fue introducido al hardware a través de *Code Composer Studio*. De esta forma, cada vez que se enciende el robot el programa se empieza a ejecutar.

5.1.7 Ensamblado del Robot

Con las tarjetas terminadas, el paso siguiente fue verificar su correcto funcionamiento para su posterior ensamblado en la estructura mecánica. El resultado final: un robot EK2007.

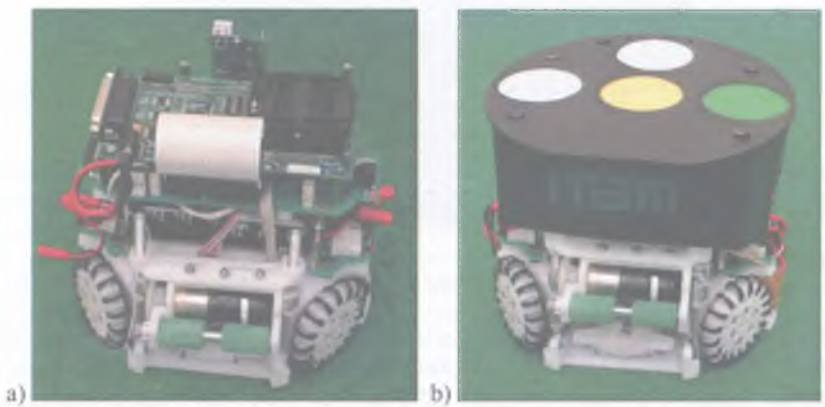


Figura 5.6 - Robot EK2007 Completamente ensamblado. a) Sin cubierta b) Con cubierta

Durante el proceso de ensamblado se deben tener en cuenta ciertas precauciones, sobre todo con la polaridad de las baterías y de los capacitores del sistema de pateo, ya que una mala conexión podría provocar un corto circuito o incluso la ignición de los capacitores. Por otro

lado, los motores deben ser conectados en el orden especificado en la tarjeta, de lo contrario, no habrá coherencia con el modelo omnidireccional programado en el DSP y el robot no se moverá correctamente.

5.1.8 Errores en el diseño electrónico

Desafortunadamente, al pasar del diseño a la implementación de los robots, se cometieron algunos errores, mismos que se describen a continuación:

5.1.8.1 Error en los Medidores de Baterías

Al trasladar el diseño de los medidores al software para la fabricación de las tarjetas, se omitió una resistencia entre los *Leds* indicadores y su respectiva tierra (GND_D en el caso del medidor digital y GND_A para el analógico). Esta omisión provoca un corto circuito mientras alguno de los *leds* está encendido, ya que cuando se genera la señal para activarlo (5V V_{CC_D}), ésta lo polariza en forma directa permitiendo el paso de la corriente sin encontrar resistencia hacia la tierra. La figura 5.7 a) muestra el error cometido, mientras que b) ejemplifica como debió haberse hecho la conexión.

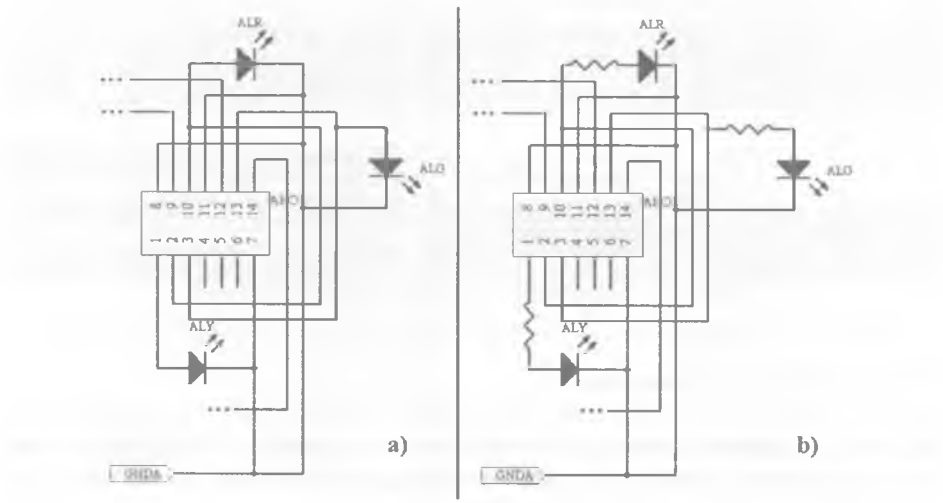


Figura 5.7 - Error en les medidores de baterías. a) Diagrama erróneo b) Diagrama correcto

Aunque la omisión no genera un corto circuito que ponga en riesgo los demás circuitos del robot, por que el *led* opone un poco de resistencia, provoca un desgaste excesivo de las baterías.

5.1.8.2 Error en la conexión del Sistema de Pateo

Originalmente, se pensó suministrar energía a la tarjeta del *Kicker* mediante un conector de 5 vías ubicado en la tarjeta Digital-Analógica; sin embargo, la proximidad de dicho conector con el controlador de un motor produjo daños en el último, ya que durante la etapa de carga del sistema de pateo, la gran cantidad de corriente que circulaba hacia el conector inducía un campo magnético que afectaba al circuito integrado provocando su calentamiento. Por este motivo se decidió que a través de dicho conector sólo recibiera las señales lógicas para indicarle al sistema la carga o descarga, mientras que la alimentación se realizó directamente de la tarjeta para el suministro de energía.

5.1.8.3 Error en los cambios para la generación EK2008

Quizá el error más perjudicial para los robots se produjo con los cambios para la generación 2008, cuando se cambió la fuente de alimentación analógica. En teoría, los motores utilizados soportaban la corriente que proporcionaban las nuevas baterías; sin embargo, los motores no resistieron y se dañaron durante la competencia.

Otro cambio que perjudico el desempeño de los robots fue la disminución del tamaño del solenoide, debido a los ajustes en la nueva mecánica. Al ser más pequeño, la potencia de disparo también se redujo considerablemente. La tabla 5.11 muestra una comparación de las intensidades de pateo del los robots EK2007 y EK2008, en ambos casos el sistema de pateo es el mismo descrito en la sección 4.1.5.7, con la diferencia del solenoide utilizado para la descarga.

Versión	Tiempo de Carga	Intensidad de Pateo
EK2007	20 seg	5 m/s
EK2008	20 seg	3 m/s

Tabla 5.1 - Comparación de las intensidades de pateo

5.1.9 Control de Trayectorias

Para estimar el gradiente que nos indique la dirección de ajuste de los parámetros, en cada iteración se deben generar t políticas aleatorias θ^j cerca de π y evaluar su comportamiento tal como se indicó en el apartado 4.2.1.2.

La generación de las políticas se hizo mediante una hoja de cálculo, mientra que para la evaluación de las mismas se utilizaron los sistemas de IA y Visión como se describe a continuación.

Evaluar una política implica tres pasos:

1. Actualizar los parámetros de los controladores con los del conjunto θ^{j}.
2. Con los nuevos parámetros el robot tiene que acelerar desde el reposo en una dirección determinada y con cierto ángulo respecto a su orientación y después debe ser detenido abruptamente.
3. En cuanto el robot deja de moverse se debe medir la desviación de la trayectoria y la rotación acumulada para darle una puntuación a la política.

La implementación del primero requirió modificar la trama de comunicación entre IA y el robot, para que éste último recibiera el nuevo conjunto de parámetros de forma inalámbrica y así poder agilizar las pruebas, ya que anteriormente, los parámetros eran incluidos en la programación del DSP como valores constantes que permanecían sin alteración en tanto no se deseara ajustarlos, en caso contrario había que modificar el código y volver a cargar el programa en el DSP, lo cual podía volver muy lentas las pruebas.

La figura 5.8 muestra la trama de comunicación utilizada durante las pruebas, como puede observarse, se añadieron seis bytes para enviar los parámetros de los controladores, dos bytes por cada uno de ellos.

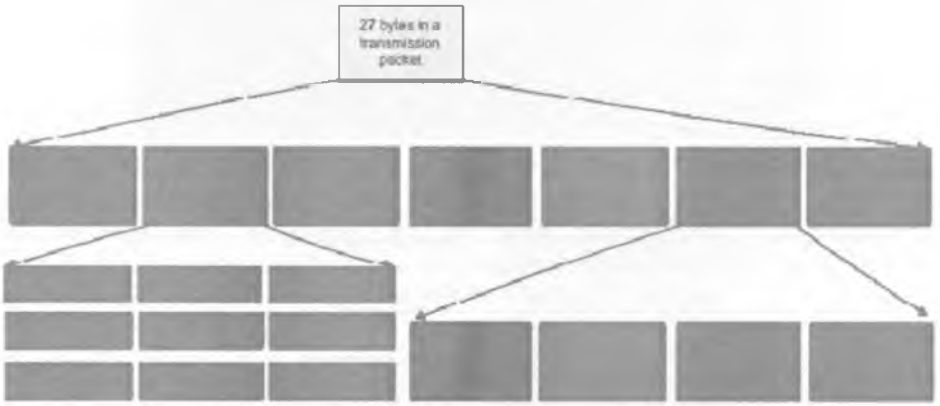


Figura 5.8 - Trama de Comunicación utilizada durante el entrenamiento de los robots.

Para el segundo paso se incluyó una interfaz en el sistema de IA que permite ingresar los datos para la prueba: parámetros del conjunto θ^{j} a evaluar, las coordenadas de las que parte el robot, la dirección en la que debe desplazarse y la orientación final deseada, figura 5.9.

Una vez proporcionadas todos los datos requeridos para la prueba, se presiona el botón "PIDs", que actualiza de forma inalámbrica los parámetros del conjunto θ^{j}. Posteriormente se presiona el botón "Iniciar", al hacerlo, el robot recibe el vector en el cual debe desplazarse y la orientación con la que debe llegar, enseguida empieza a moverse con base

en los datos indicados por un intervalo de 2 segundos, después del cual es detenido abruptamente.

Durante esta fase de la prueba la corrección de la trayectoria mediante los Sistemas de IA y Visión, descrita en la sección 3.1.7, no debe estar activada.

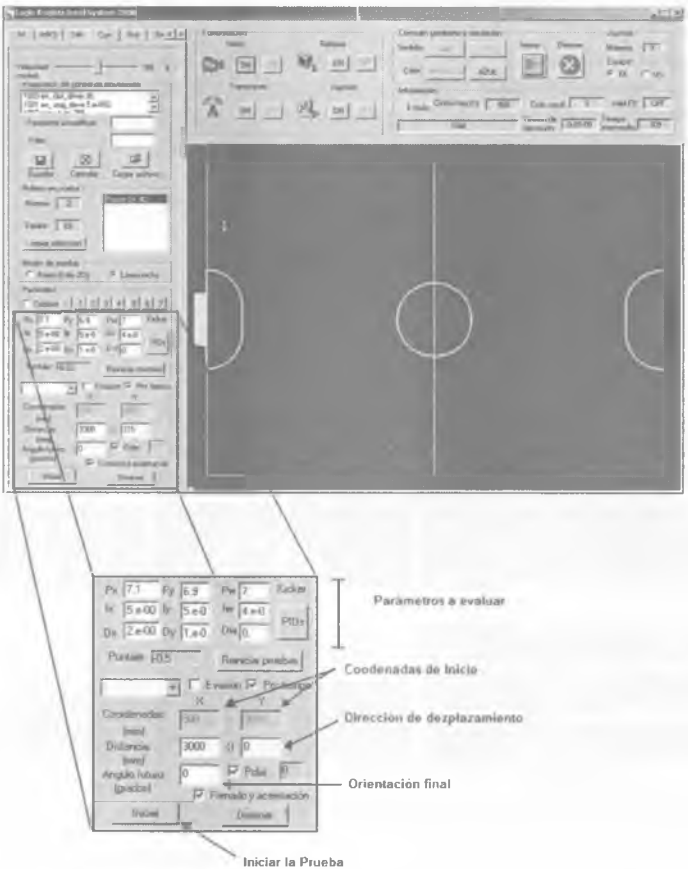


Figura 5.9 - Interfaz para realizar la evaluación de las políticas

Para obtener la desviación del robot de su destino y la rotación acumulada, necesarias para darle un puntaje a la política evaluada, se hace uso del Sistema de Visión, el cual tiene implementados diversos métodos que permiten medir la posición y orientación real del robot a través del procesamiento de las imágenes que captura, como la mostrada en la figura 5.10.

La posición y orientación del robot son comunicadas al Sistema de IA, que se encarga de obtener la puntuación para cada criterio de la función $Q(\theta^j)$.

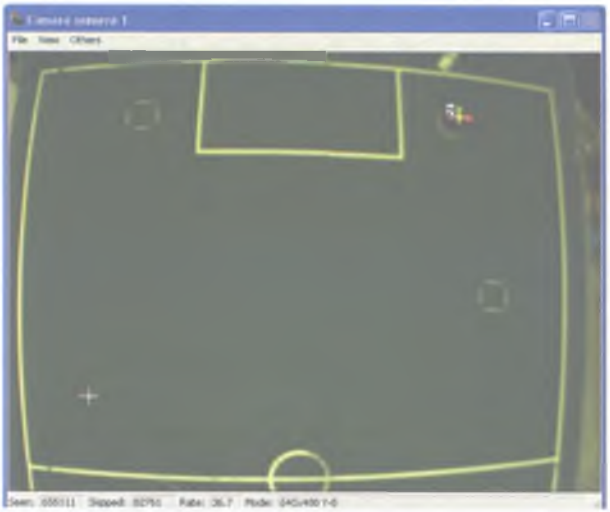


Figura 5.10 - Imagen capturada por el Sistema de Visión

Cuando finaliza la prueba de una política θ^j, el sistema de IA entrega la puntuación de los dos criterios evaluados, la cual se registra en una hoja de cálculo. Al final de cada iteración, se calcula el valor de la función $Q(\theta^j)$ para todas las políticas y se calcula el gradiente tal como se indicó en el apartado 4.2.1.2, para después actualizar los parámetros en la dirección más favorable. Las iteraciones continúan hasta que la nueva política ajustada es igual a la política ajustada en iteración inmediata anterior.

- CAPÍTULO 6 -

PRUEBAS Y RESULTADOS

En este capítulo se mencionan los resultados que obtuvo el equipo *Eagle Knights* en las competencias oficiales donde participó con los robots EK2007 y EK2008. También se describen las pruebas realizadas y los resultados obtenidos durante el entrenamiento del robot para "aprender" los parámetros óptimos de los controladores PID.

6.1 RoboCup 2007

La prueba definitiva para los robots EK2007 llegó con el campeonato mundial *RoboCup* 2007, celebrado en Altanta, Georgia, E.U.A. En esa ocasión calificaron un total de 19 equipos, de los cuales participaron 12, distribuidos en dos grupos como lo muestra la tabla 6.1.

Equipo	País	Grupo
CMDragons	USA	A
RoboDragons	Japón	A
Botnia	Finlandia	A
Wright Eagle	China	A
B-Smart	Alemania	A
Strive	China	A
Plasma-Z	Tailanda	B
ZJUNlict	China	B
Eagle Knights	**México**	**B**
KIKS	Japón	B
RFC Cambridge	USA	B
FURGBol	Brasil	B

Tabla 6.1 - Equipos participantes en el *RoboCup* 2007

6.1.1 Los partidos del equipo Eagle Knights

La tabla 6.2 muestra los resultados de los seis partidos disputados en el *RoboCup* 2007, de los cuales los primeros cinco corresponden a una etapa eliminatoria y el sexto corresponde al partido de cuartos de final.

Partido	Contrincante	Observaciones	Marcador Final	
1	*RFC Cambridge*	Por primera vez en la historia de nuestro equipo se ganó un partido de competencia mundial por la máxima diferencia de goles permitida 10-0, sin que esta fuera resultado de la no participación del contrincante (las reglas estipulan que si un equipo no se presenta al juego, se asigna una victoria de 10-0 en favor del equipo contrario).	*RFC Cambridge*	0
			Eagle Knights	10
2	*FURGBol*	Variaciones de luz en el entorno afectaron el Sistema de Visión, impidiendo que en momentos no encontrara la ubicación de la pelota o de los robots.	*FURGBol*	0
			Eagle Knights	4
3	*KIKS*	Hubo problemas de interferencia con el sistema de comunicación, ya que las frecuencias con las que trabaja también son usadas por otros equipos, incluso de otras ligas. Esto provocó que los robots no respondieran adecuadamente a los comandos de IA.	*KIKS*	5
			Eagle Knights	0
4	*ZJUNlict*	Si bien, el sistema del contrincante en conjunto no era tan superior al nuestro, una falla en los motores de un robot nos dejó con sólo 4 robots en la cancha. La ausencia provocó que empezaran a caer goles en nuestra contra.	*ZJUNlict*	10
			Eagle Knights	0
5	*Plasma-Z*	A pesar que nuestros robots respondieron adecuadamente, el nivel de juego del equipo tailandés era superior al nuestro, sobre todo si los comparamos con base en la intensidad de su sistema de pateo, que lograba impulsar la pelota a una sorprendente velocidad de 14 m/s.	*Plasma-Z*	10
			Eagle Knights	0
6	*CMDragons*	El reto para los *Eagle Knights* en los cuartos de final consistió en resistir por lo menos hasta el final del primer tiempo sin que los campeones anotaran los 10 goles necesarios para terminar el juego, ya que la velocidad de desplazamiento de sus robots hacían muy difícil la marca de los nuestros.	*CMDragons*	10
			Eagle Knights	0

Tabla 6.2 - Resultado de los Partidos del Equipo *Eagle Knights* en el *Robocup* 2007

Finalizada la competencia, los *Eagle Knights* quedamos posicionados en séptimo lugar a nivel mundial en la categoría SSL [48].

Equipo	País	Lugar
CMDragons	USA	1
Plasma-Z	Tailandia	2
RoboDragons	Japón	3
ZJUNlict	China	4
KIKS	Japón	5
Wright Eagle	China	6
Eagle Knights	**México**	**7**
B-Smart	Alemania	8
Botnia	Finlandia	9
FURGBol	Brasil	10
RFC Cambridge	USA	11
Strive	China	12

Tabla 6.3 - Clasificación Final *RoboCup* 2007

6.2 RoboCup 2008

Debido a la estabilidad observada en los robots del 2007, para la competencia del 2008, celebrada en Zuzhou, China, se decidió participar con los mismos robots, pero con las "mejoras" descritas en la sección 4.1.7 y con un nuevo Sistema de Inteligencia Artificial basado en un Sistema Experto [54]. En esta edición participaron 17 equipos divididos en 4 grupos como se describen en la tabla 6.4.

Equipo	País	Grupo
CMDragons	USA	A
KIKS	Japón	A
FANTASIA	China	A
Skuba	Tailandia	A
RoboDragons	Japan	B
Wright Eagle	China	B
Parsian	Iran	B
Khainui	Tailandia	B
Plasma-Z	Tailandia	C
Eagle Knights	**México**	**C**
MRL	Iran	C
AUA Ares	China	C
ZjuNlict	China	D
B-Smart	Alemania	D
Botnia-Dragon Knights	Finlandia/China	D
RoboJackets	USA	D
Strive	China	D

Tabla 6.4 - Equipos Participantes en el *RoboCup* 2008

6.2.1 Los partidos del equipo Eagle Knights

Para el *RoboCup* 2008 se esperaba que el equipo obtuviera mejores resultados que en el torneo anterior; lamentablemente, desde el primer partido empezaron a surgir problemas derivados de las modificaciones descritas en la sección 4.1.7. En primer lugar, con la reducción de tamaño del soleniode también disminuyó la intensidad de pateo, que si bien era suficiente para hacer pases, no lo era para disparar a gol o despejar la pelota cuando estaba cerca de la portería. Por si fuera poco, la mala decisión de aumentar la capacidad de las baterías provocó que algunos motores se dañaran durante el partido y que los robots que los tenían puestos no se movieran o lo hicieran incorrectamente. La tabla 6.5 muestra el marcador de los tres partidos disputados.

Partido	Contrincante	Observaciones	Marcador Final	
1	*MLR*	Fallas en los motores de dos robots provocaron que se jugara el partido con el equipo incompleto y, por lo tanto, no fue posible marcar a todos los robots contrincantes, esta deficiencia se tradujo como goles en contra casi al final del partido.	*MLR*	4
			Eagle Knights	0
2	*AUA Ares*	Para el segundo partido se detectó el problema con las baterías y se decidió retirar dos de las cuatro colocadas, de esta manera los circuitos estarían alimentados igual que en los robot del 2007. Sin embargo, el daño ya estaba hecho en algunos motores. Al igual que en el primer partido, los goles cayeron cuando nuestro equipo de robots estaba incompleto.	*AUA Ares*	4
			Eagle Knights	0
3	*Plasma Z*	Jugar contra dicho equipo representó un gran reto, sobre todo por las condiciones en las que llegaban los robots. Más que ganar el encuentro, nuestro objetivo fue resistir el mayor tiempo posible con los robots dañados. Al final sólo pudimos soportar el primer tiempo, cuando los contrarios anotaron la máxima diferencia.	*Plasma Z*	10
			Eagle Knights	0

Tabla 6.5 - Resultado de los Partidos del Equipo *Eagle Knights* en el *Robocup* 2008

La derrota en los tres encuentros nos impidió avanzar a la siguiente ronda, quedando ubicados al final del torneo en el décimo tercer lugar [49].

Equipo	País	Lugar
Plasma-Z	Tailandia	1
CMDragons	USA	2
Skuba	Tailandia	3
ZjuNlict	China	4
RoboDragons	Japan	5
AUA Ares	China	6
FANTASIA	China	7
B-Smart	Alemania	8
Wright Eagle	China	9
Botnia/Dragon Knights	Finlandia/China	10
MRL	Iran	11
Khainui	Tailandia	12
Eagle Knights	**México**	**13**
Strive	China	14
RoboJackets	USA	15
KIKS	Japón	16
Parsian	Iran	17

Tabla 6.6 - Clasificación Final del *RoboCup* 2008

6.3 Comparando los robots EK2007-EK2008 con los de otros Equipos

Podría considerarse que el factor que permite evaluar el desempeño de los robots EK2007-EK2008 es el número de partidos jugados y los resultados obtenidos en el *RoboCup*; sin embargo al utilizar esos datos como parámetro de desempeño se debe considerar que el éxito o fracaso en un encuentro depende de la eficacia de cada uno de los sistemas que conforman la arquitectura de un equipo SSL: Sistema de Visión, Sistema de Inteligencia Artificial, Sistema de Comunicación y los Robots. Por lo tanto, es necesario realizar un análisis más objetivo que permita evaluar el desempeño de los robots con base en datos provenientes de los mismos y compararlos con los de otros equipos, para determinar las fortalezas y debilidades que guíen al desarrollo de nuevos y mejores robots.

En los siguientes apartados se comparan algunas características de los robots EK contra las de otros equipos de la liga SSL, tabla 6.7. Se decidió que las características a comparar fueran aquellas que afectan directamente el desempeño del robot durante un partido, siendo así, se consideraron los siguientes puntos:

1. Velocidad de Desplazamiento y tipo de Motores
2. Intensidad del Sistema de Pateo
3. Tipo de Procesador Central
4. Duración de las Baterías

Equipo	País	Referencia
Plasma-Z	Tailandia	[22]
CMDragons	USA	[12]
Skuba	Tailandia	[38]
ZjuNlict	China	[55]
RoboDragons	Japan	[27]
AUA Ares	China	[56]
FANTASIA	China	[33]
B-Smart	Alemania	[25]
Wright Eagle	China	[13]
Botnia/Dragon Knights	Finlandia/China	[26]
MRL	Iran	[4]
Khainui	Tailandia	[21]
Eagle Knights	**México**	[53]
Strive	China	[36]
RoboJackets	USA	[5]
KIKS	Japón	[57]
Parsian	Iran	[17]

Tabla 6.7 - Equipos a comparar

6.3.1 Velocidad de Desplazamiento y Tipo de Motores

La velocidad de desplazamiento de los robots SSL esta determinada en gran medida por el tipo de motores utilizados y por la forma en que estos últimos transmiten su fuerza hacia la rueda.

En la competencia se ha generalizado el uso de dos tipos de motores de DC: Motores DC con escobillas (*Brushed*) y Motores DC sin escobillas (*Brushless*). Los primeros, *brushed*, son altamente eficientes y tienen grandes características para hacerlos funcionar como servo-motores. Pese a estas grandes ventajas, cuentan con un conmutador y unas escobillas las cuales están sujetas al desgaste y por esta razón necesitan mantenimiento o bien tienen un tiempo de vida limitado. La característica principal de los segundos, *brushless*, es que realizan la misma función de un motor DC normal pero reemplazando el conmutador y las escobillas por *switches* de estado sólido que funcionan con una lógica para la conmutación de los embobinados. La gran ventaja de los motores *brushless*, frente a los demás motores de DC, es que no requieren de un mantenimiento periódico.

En cuanto a la forma en que el motor transmite su fuerza a la rueda puede ser de dos tipos: transmitiéndola directamente del eje del motor hacia la rueda o a través de engranes entre el eje y la rueda. A igualdad de motores, con la primera es posible alcanzar mayor velocidad; sin embargo se pierde precisión, en cambio con la segunda se gana precisión pero se sacrifica velocidad. La tabla 6.8 muestra la máxima velocidad alcanzada por los robots de distintos equipos, la cual está relacionada con el tipo de motor utilizado y la forma en que transmite su fuerza a la rueda.

Equipo	Máxima Velocidad Desplazamiento (m/s)	Tipo de Motores DC	Transmisión de la fuerza del motor a la rueda
Plasma-Z	2.5	Brushless	A través de engranes
CMDragons	3	Brushless	A través de engranes
Skuba	2.5	Brushless	A través de engranes
ZjuNlict	3	Brushless	A través de engranes
RoboDragons	3	Brushless	A través de engranes
AUA Ares	3	Brushless	A través de engranes
FANTASIA	2.5	Brushless	A través de engranes
B-Smart	2	Brushed	A través de engranes
Wright Eagle	2.5	Brushed	A través de engranes
Botnia	2	Brushless	A través de engranes
MRL	2	Brushed	A través de engranes
Khainui	2	Brushed	Directa
Eagle Knights	**2.5**	**Brushed**	**Directa**
Strive	3	Brushed	A través de engranes
RoboJackets	2	Brushless	A través de engranes
KIKS	2.5	Brushed	Directa
Parsian	2	Brushed	Directa

Tabla 6.8 - Velocidad de desplazamiento de los robots de diversos equipos SSL

En la tabla anterior se puede ver que la combinación más socorrida para la locomoción de los robots es: motores *brushless* cuya fuerza es transmitida a la rueda a través de un sistema de engranes. Resulta interesante notar que los primeros 7 lugares de la competencia utilizan dicha combinación, mientras que en las últimas posiciones de la tabla se encuentran los equipos que transmiten la fuerza de los motores directamente a la rueda, incluyendo a los EK. En la figura 6.1 se puede observar que los motores *brushless* están siendo más utilizados que los *brushed*, es muy probable que sean preferidos por el poco mantenimiento que requieren y su mayor tiempo de vida.

Figura 6.1 - Combinaciones utilizadas para la locomoción de los robots SSL en el *RoboCup* 2008

6.3.2 Intensidad del Sistema de Pateo

En la liga SSL, cuando se habla de intensidad de pateo se hace referencia a la velocidad que alcanza la pelota al ser golpeada por el *kicker*. El sistema de pateo de todos los equipos en SSL esta basado en el principio del solenoide; como se describió en la sección 2.3.4, dichos sistemas requieren un generador de alto voltaje y capacitores para almacenarlo. La velocidad de la pelota es directamente proporcional a la fuerza que genera el solenoide y a la cantidad de carga almacenada en los capacitares, la cual esta en función del voltaje máximo que proporcione su generador. De acuerdo con la tabla 6.8, para el torneo del 2008, prácticamente todos los equipos disponían de un generador de alto voltaje que les permitía incrementarlo hasta 200V en promedio, mientras que el de los robots EK solamente llegaba a 110 V. Estos datos evidencian que el problema del sistema de pateo EK radica principalmente en su generador de alto voltaje.

Equipo	Velocidad Promedio del Sistema de Pateo (m/s)	Máximo Voltaje de su generador
Plasma-Z	10	250V
CMDragons	10	200V
Skuba	9	200V
ZjuNlict	9	200V
RoboDragons	9	200V
AUA Ares	10	200V
FANTASIA	9.8	150V
B-Smart	8	200V
Wright Eagle	9	200V

Botnia	8	200V
MRL	6	200V
Khainui	5	200V
Eagle Knights	**4**	**110V**
Strive	7	200V
RoboJackets	6	200V
KIKS	9.5	200V
Parsian	5	180V

Tabla 6.9 - Velocidad del sistema de pateo de diversos equipos SSL

6.3.3 Tipo de Procesador Central

En la arquitectura de un robot SSL, el procesador central juega un papel muy importante para su desempeño, por ser el encargado de interpretar y ejecutar las instrucciones de IA. Por lo tanto la velocidad y precisión con que ejecute los comandos, principalmente los relacionados con el control motriz, pueden hacer la diferencia durante un encuentro. Sin embargo, para el procesamiento que se requiere actualmente en la liga, la diferencia entre utilizar un FPGA o un DSP es mínima. Pero si se considera la posibilidad de que en un futuro la liga decida cambiar la visión global por visión local, lo que implicaría realizar procesamiento de imágenes en el robot, entonces la mejor opción sería un DSP, ya que esta diseñado para dicho tipo de aplicaciones. En la figura 6.2 se pueden observar que en la competencia se está generalizando el uso del FPGA.

Equipo	**Tipo de Procesador Central**
Plasma-Z	FPGA
CMDragons	ARM7 y FPGA
Skuba	FPGA
ZjuNlict	FPGA
RoboDragons	Procesador Hitachi SH2
AUA Ares	DSP
FANTASIA	FPGA
B-Smart	Microcontrolador ATMEGA128
Wright Eagle	DSP
Botnia	FPGA-ARM7
MRL	Microcontrolador AVR
Khainui	ARM7
Eagle Knights	**DSP**
Strive	FPGA
RoboJackets	FPGA-ARM7
KIKS	Procesador Hitachi SH2
Parsian	Microcontrolador AVR

Tabla 6.10 - Tipo de Procesador Central que utilizan diversos equipos SSL

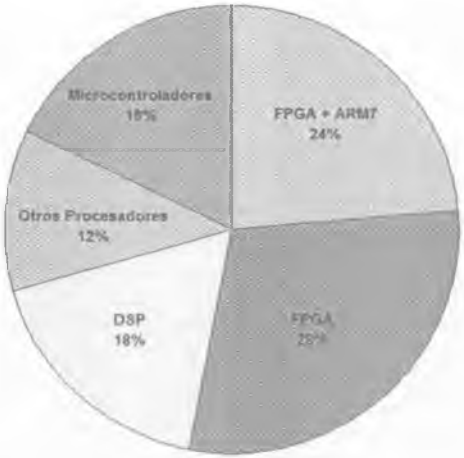


Figura 6.2 - Tipos de procesador central utilizados en el *RoboCup* 2008

6.3.4 Duración de las Baterías

Prácticamente los robots de todos los equipos duran un partido completo sin recargar o cambiar sus baterías; en cambio las baterías de los robots EK2007-EK2008 solamente resistían hasta el medio tiempo, después del cual había que cambiarlas. Se detectó que el motivo del desgaste tan rápido fue el error en el diseño electrónico descrito en la sección 5.1.8.1. Sin embargo, es posible evitar el consumo excesivo dejando de alimentar a los medidores de baterías.

Equipo	Duración Promedio de las Baterías
Eagle Knights	20 min
Resto de la Liga	45 min

Tabla 6.11 - Duración promedio de las baterías en el *RoboCup* 2008

6.4 Entrenamiento del robot para aprender los parámetros PID

A continuación se describen las consideraciones para las pruebas realizadas durante el entrenamiento del robot, así como los resultados obtenidos al final del mismo.

Valores de la Política Inicial π: Los valores del conjunto inicial de parámetros pueden partir de cero; sin embargo, una de las ventajas del método PGRL es la posibilidad incorporar un conocimiento previo del dominio a través de una adecuada parametrización de la política. Por lo tanto, si la política inicial parte de los valores obtenidos manualmente,

es posible converger más rápido al óptimo local. Siendo así, la política inicial quedó de la siguiente forma:

$$\pi = \{7.0,0.05,0.05,7.0,0.05,0.05,7.0,0.05,0.05\}$$

Generar t políticas aleatorias θ^l cerca de π: El total de políticas aleatorias a generar en cada iteración debe ser lo suficientemente grande como para poder observar una tendencia en el gradiente, pero no tanto que vuelva tediosas las mediciones. Para los experimentos se decidió generar 10 políticas aleatorias por iteración. La tabla 6.12 muestra las políticas generadas en la primera.

Políticas	Controlador para V_x			Controlador para V_y			Controlador para ω		
	Kp	Ki	Kd	Kp	Ki	Kd	Kp	Ki	Kd
θ^1	6.9	0.04	0.04	7.1	0.04	0.05	7	0.06	0.06
θ^2	7	0.05	0.05	7	0.05	0.04	7	0.06	0.04
θ^3	6.9	0.05	0.04	6.9	0.06	0.06	6.9	0.06	0.06
θ^4	7.1	0.04	0.06	7	0.04	0.05	7	0.06	0.05
θ^5	6.9	0.04	0.04	7	0.05	0.04	7.1	0.06	0.04
θ^6	6.9	0.04	0.04	7	0.05	0.05	7	0.04	0.05
θ^7	7	0.06	0.04	6.9	0.04	0.04	7	0.04	0.04
θ^8	7.1	0.05	0.04	7	0.04	0.04	7.1	0.06	0.06
θ^9	7	0.06	0.05	7	0.05	0.06	7.1	0.06	0.04
θ^{10}	7	0.06	0.06	7.1	0.04	0.05	7.1	0.05	0.04

Tabla 6.12 - Políticas Aleatorias Generadas en la Primera Iteración del Entrenamiento

Cada conjunto θ^l es el resultado de alterar el parámetro p_i con un valor Δ_i^l que es elegido con probabilidad uniforme del conjunto $\{-\varepsilon_i,0,+\varepsilon_i\}$ y donde ε_i es una constante pequeña en relación con el tipo de parámetro a ser alterado. La tabla 6.13 indica el valor las constantes ε_i utilizadas en los experimentos:

Valor de ε_i	Parámetros en los que se utilizó
0.1	p_1, p_4 y p_7, correspondientes a la constante K_p
0.01	p_2, p_5 y p_8, correspondientes a la constante K_i
0.01	p_3, p_6 y p_9, correspondientes a la constante K_d

Tabla 6.13 - Valores de las constantes para alterar las políticas

Evaluación de las Políticas: Con cada conjunto de parámetros θ^l el robot aceleraba desde el reposo en una dirección de 45° sin girar sobre su eje durante un lapso de 2 segundos, después del cual el robot era detenido abruptamente. Entonces, el Sistema de Visión proporcionaba al Sistema de Inteligencia la posición y orientación final del robot, para que

el segundo calculara la desviación de su destino y la rotación acumulada y con ello obtener la puntuación de la función $Q(\theta')$. La puntuación de la función $Q(\theta')$ es una medida del desempeño del robot descrito por ese conjunto de parámetros.

Política	Desviación	Rotación acumulada	$Q(\theta^j)$
θ^1	-9.66E-02	-6.50E-02	-0.04514
θ^2	-9.27E-02	-7.08E-02	-0.04416
θ^3	-0.124	-0.115	-0.0611
θ^4	-4.92E-02	-2.09E-02	-0.02177
θ^5	-9.86E-02	-9.01E-02	-0.04845
θ^6	-7.34E-02	-4.77E-02	-0.03413
θ^7	-0.123	-8.76E-02	-0.05796
θ^8	-9.15E-02	-9.92E-02	-0.04652
θ^9	-8.44E-02	-8.68E-02	-0.04244
θ^{10}	-7.82E-02	-6.35E-02	-0.03763

Tabla 6.14 - Puntuaciones de las Políticas Evaluadas en la Primera Iteración

En las figuras 6.3 y 6.4 es más fácil de apreciar como el robot tiene un comportamiento distinto con cada política. Se puede observar que la política con mejor desempeño fue θ^4, esto hace más probable que la dirección de ajuste en la próxima iteración sea muy similar a la de dicho conjunto.

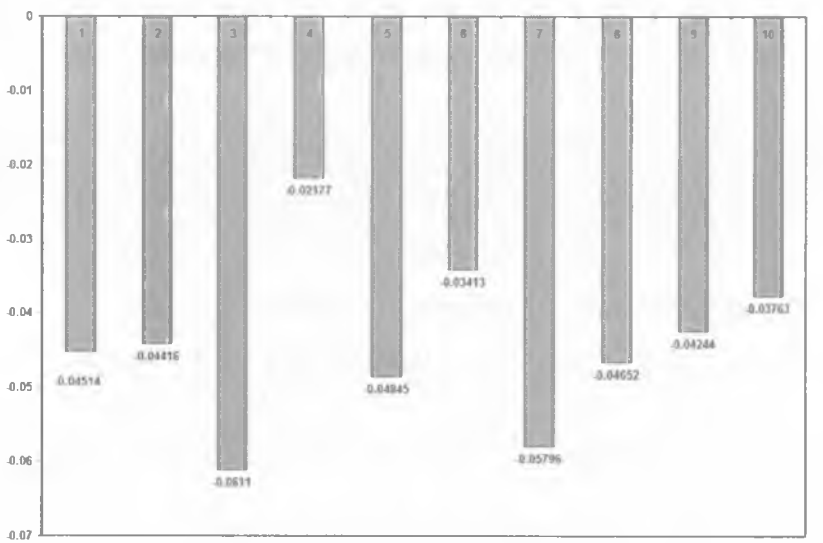


Figura 6.3 - Puntuación de las Políticas evaluadas en la Primera Iteración. Entrenamiento a 45°

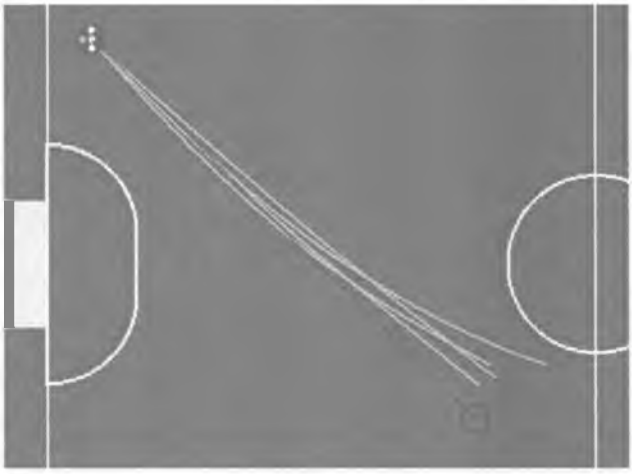


Figura 6.4 - Trayectorias Producidas por algunas Políticas de la Primera Iteración. Entrenamiento a 45°

Cálculo del Gradiente: El siguiente paso fue agrupar cada θ^j en uno de los tres conjuntos $S_i^{-\varepsilon}$, S_i^{+0}, ó $S_i^{+\varepsilon}$, con base en el tipo de perturbación aleatoria $\{-\varepsilon_i, 0, +\varepsilon_i\}$ para cada dimensión. Por ejemplo, si el primer parámetro de la política θ^j fue alterado por $+\varepsilon_i$, entonces θ^j se agrupará en $S_i^{+\varepsilon}$; si fue alterado por $-\varepsilon_i$, se agrupará en $S_i^{-\varepsilon}$; y si no se sufrió cambio en S_i^{+0}. Las distintas perturbaciones para el primer parámetro se muestran en la tabla 6.15.

Política	θ^1	θ^2	θ^3	θ^4	θ^5	θ^6	θ^7	θ^8	θ^9	θ^{10}
Perturbación en p_1	$-\varepsilon_i$	0	$-\varepsilon_i$	$+\varepsilon_i$	$-\varepsilon_i$	$-\varepsilon_i$	0	$+\varepsilon_i$	0	0

Tabla 6.15 - Tipo de Perturbaciones para el Primer Parámetro

Por lo tanto, los conjuntos para el primer parámetro quedaron como sigue:

$$S_1^{-\varepsilon} = \{\theta^1, \theta^3, \theta^5, \theta^6\} \qquad S_1^{+0} = \{\theta^2, \theta^7, \theta^9, \theta^{10}\} \qquad S_1^{+\varepsilon} = \{\theta^4, \theta^8\}$$

Con base en los conjuntos obtenidos, se calculó la puntuación promedio para cada uno:

$$A_1^{-\varepsilon} = \frac{\sum_{\theta^j \in S_1^{-\varepsilon}} Q(\theta^j)}{|S_1^{-\varepsilon}|} = \frac{Q(\theta^1) + Q(\theta^3) + Q(\theta^5) + Q(\theta^6)}{4} = -0.047205$$

$$A_1^{+0} = \frac{\sum_{\theta^j \in S_1^{+0}} Q(\theta^j)}{|S_1^{+0}|} = \frac{Q(\theta^2) + Q(\theta^7) + Q(\theta^9) + Q(\theta^{10})}{4} = -0.0455475$$

$$A_1^{+\varepsilon} = \frac{\sum_{\theta^j \in S_1^{+\varepsilon}} Q(\theta^j)}{|S_1^{+\varepsilon}|} = \frac{Q(\theta^4) + Q(\theta^8)}{2} = -0.034145$$

Estos tres promedios dieron una estimación de los beneficios de alterar el primer parámetro por $-\varepsilon_i$, 0 ó $+\varepsilon_i$. Como puede observarse, el mejor promedio lo obtuvo la perturbación positiva $A_i^{+\varepsilon}$, por lo tanto, el parámetro se incrementó con una tasa de aprendizaje η_i. En caso que la mejor puntuación hubiera sido la negativa $A_i^{-\varepsilon}$, el parámetro se tendría que haber ajustado en la dirección contraria, es decir, disminuyéndolo en η_i. Y si la mejor hubiera sido A_i^{+0}, el parámetro quedaría sin cambios. La tasa de aprendizaje η_i varió dependiendo del tipo de parámetro según lo indicado en la tabla 6.16.

Valor de η_i	Parámetros en los que se utilizó
0.1	p_1, p_4 y p_7, correspondientes a la constante K_p
0.01	p_2, p_5 y p_8, correspondientes a la constante K_i
0.01	p_3, p_6 y p_9, correspondientes a la constante K_d

Tabla 6.16 - Taza de aprendizaje por el tipo de parámetro. Entrenamiento a 45°

Los cálculos descritos se realizaron para los nueve parámetros, la tabla 6.17 muestra las puntuaciones para cada uno de ellos durante la primera iteración; con base en ellas se determinó la dirección del gradiente, última fila de la tabla, para remplazar el conjunto de parámetros inicial π por otro ajustado en la dirección más favorable, tabla 6.18.

	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
$A_i^{+\varepsilon}$	-0.0341	-0.0460	-0.0297	-0.0413	-0.0611	-0.0517	-0.0437	-0.0442	-0.0509
A_i^{+0}	-0.0455	-0.0505	-0.0433	-0.0395	-0.0422	-0.0346	-0.0406	-0.0376	-0.0279
$A_i^{-\varepsilon}$	-0.0472	-0.0373	-0.0488	-0.0595	-0.0418	-0.0492	-0.0611	-0.0460	-0.0461
Dirección del Gradiente	+	-	+	0	-	0	0	0	0

Tabla 6.17 - Puntuaciones Promedio para cada Parámetro en la Primera Iteración

	Controlador para V_x			Controlador para V_y			Controlador para ω		
	Kp	Ki	Kd	Kp	Ki	Kd	Kp	Ki	Kd
Parámetro	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9
π **inicial**	7.0	0.05	0.05	7.0	0.05	0.05	7.0	0.05	0.05
π **ajustada**	7.1	0.04	0.06	7.0	0.04	0.05	7.0	0.05	0.05

Tabla 6.18 - Ajuste de la Política Inicial al finalizar la Primera Iteración

Con el nuevo conjunto de parámetros ajustados se volvió a repetir el proceso hasta converger a un óptimo local, es decir, cuando la nueva política ajustada fue igual a la política anterior. En la sección siguiente se analizan los resultados del entrenamiento.

6.4.1 Resultados del entrenamiento a 45°

Como se mencionó, las iteraciones del proceso continuaron hasta converger a un óptimo local, esto sucedió en la sexta iteración. Era de esperarse una convergencia tan rápida por haber tomado como política inicial los valores de los parámetros obtenidos manualmente.

La tabla 6.19 y las figuras 6.5, 6.6 y 6.7 muestran la evolución de los parámetros para cada controlador PID. El conjunto de parámetros de la sexta iteración es considerado como el óptimo local.

Iteración	Controlador para V_x			Controlador para V_y			Controlador para ω		
	Kp	Ki	Kd	Kp	Ki	Kd	Kp	Ki	Kd
1	7.0	0.05	0.05	7.0	0.05	0.05	7.0	0.05	0.05
2	7.1	0.04	0.06	7.0	0.04	0.05	7.0	0.05	0.05
3	7.1	0.03	0.07	7.1	0.05	0.04	7.1	0.05	0.05
4	7.1	0.02	0.08	7.0	0.06	0.03	7.2	0.06	0.04
5	7.1	0.01	0.07	6.9	0.05	0.03	7.1	0.06	0.03
6	**7.1**	**0.01**	**0.07**	**6.9**	**0.05**	**0.03**	**7.1**	**0.06**	**0.03**

Tabla 6.19 - Evolución de los parámetros para cada Controlador PID.
Entrenamiento a 45°

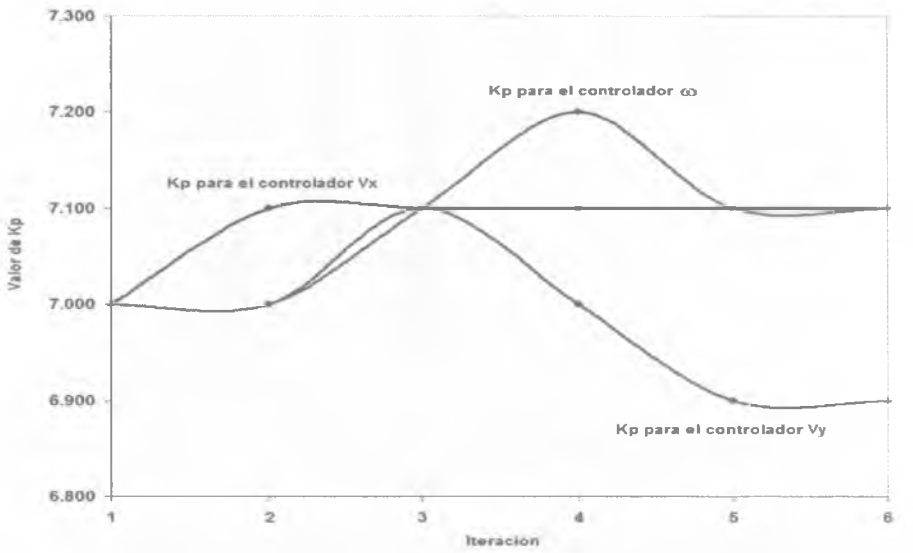


Figura 6.5 - Evolución del Parámetro Kp para cada Controlador.
Entrenamiento a 45°

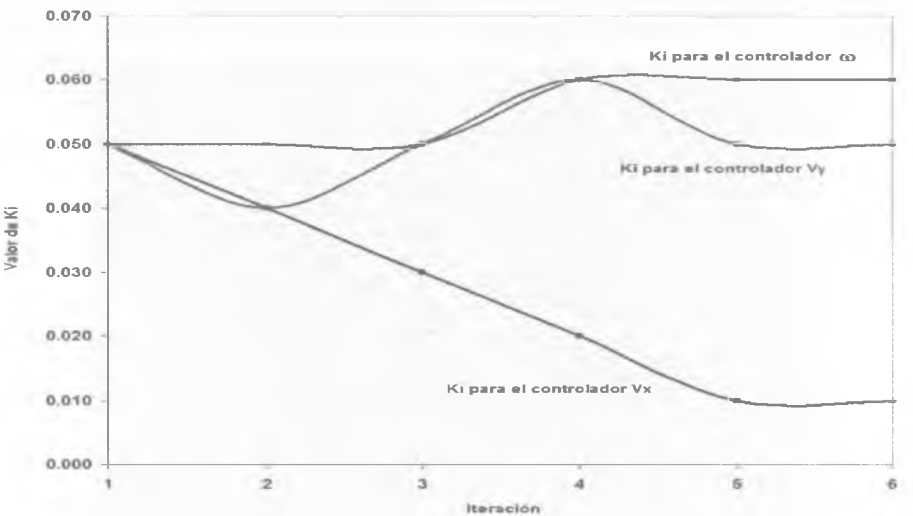


Figura 6.6 - Evolución del Parámetro Ki para cada Controlador.
Entrenamiento a 45°

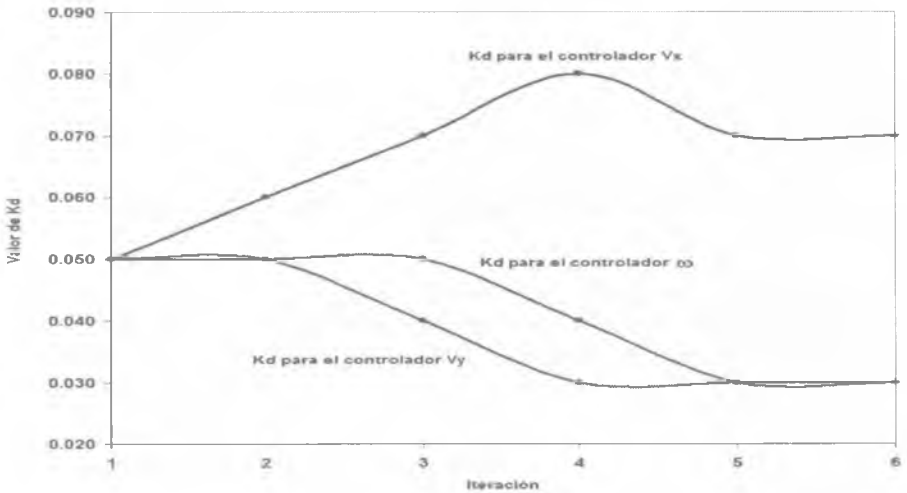


Figura 6.7 - Evolución del Parámetro Kd para cada Controlador.
Entrenamiento a 45°

La política ajustada al final de cada iteración fue evaluada de la misma forma que las políticas aleatorias para comprobar si su desempeño mejoraba conforme avanzaba el entrenamiento. La tabla 6.20 indica la puntuación obtenida por cada uno de ellas, mientras que en la figura 6.8 se puede observar la tendencia de su desempeño. Hay que recordar que las puntuaciones de $Q(\theta')$ se encuentran en el intervalo [-1,0], -1 la peor y 0 la mejor.

Desviación	Rotación Acumulada	$Q(\theta')$
-0.075	-0.0664	-0.03664
-0.056	-0.0167	-0.02407
-0.069	-0.0584	-0.03344
-0.0265	-0.00713	-0.011313
-0.00651	-0.0212	-0.004724
-0.0152	-0.0246	-0.00854

Tabla 6.20 - Puntuación de las Políticas Ajustadas durante cada Iteración

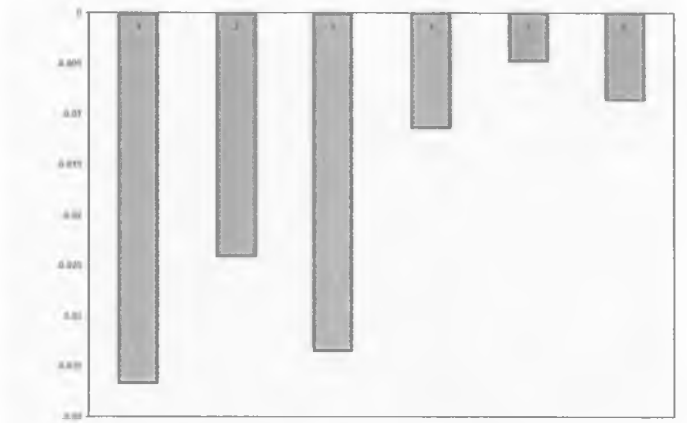


Figura 6.8 – Evolución del Desempeño de las Políticas Ajustadas durante cada Iteración. Entrenamiento a 45°

En la gráfica anterior y en la figura 6.9 se puede observar que hubo una diferencia notable entre el desempeño del robot con los parámetros ajustados en la primera iteración y el desempeño con los parámetros "aprendidos" al final del entrenamiento, iteraciones cinco y seis. Sin embargo, y a pesar que la trayectoria producida por los parámetros ajustados con el método de PGRL es mejor que la trayectoria producida por los parámetros ajustados a mano, figura 6.10, podemos ver que la trayectoria sigue teniendo un error de desviación de entre 5% y 10%. Este error es producido por un derrape del robot al inicio del movimiento.

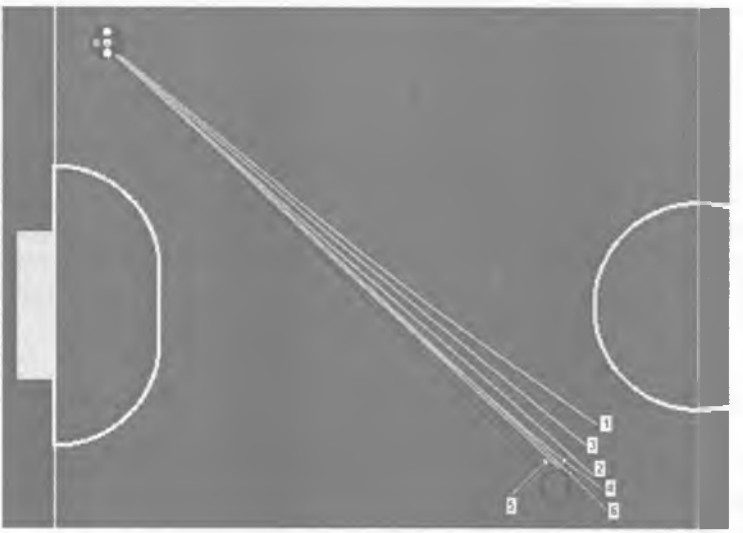


Figura 6.9 - Trayectorias Producidas por las Políticas Ajustadas en cada Iteración. Entrenamiento a 45°

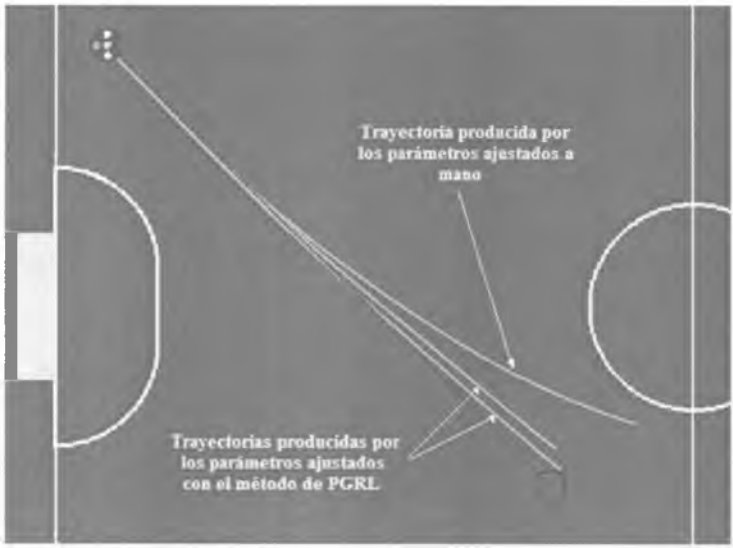


Figura 6.10 - Trayectorias producidas por los parámetros ajustados a mano y los ajustados con el método PGRL. Entrenamiento a 45°

Es de notar que, si bien el conjunto de parámetros de las dos ultimas iteraciones son los mismos, lo puntuación obtenida en su respectiva prueba es diferente. El motivo de esta variación, como se explicó en la sección 2.5.1.1, es la naturaleza no-determinística del ambiente, es decir, tomar la misma acción en el mismo estado puede dar resultados diferentes. Sin embargo, podemos notar que a pesar de ello el desempeño del robot sigue siendo mejor que el obtenido con el primer conjunto.

Ya que el entrenamiento se llevó a cabo siempre con el mismo tipo de prueba, moviendo al robot en una dirección de 45° sin rotar sobre su eje, había que comprobar si su desempeño era similar para otras direcciones de desplazamiento. En este punto se pudo notar que los nuevos parámetros sólo funcionaban para ángulos cercanos a los simétricos de 45° (135°, 225° y 315°), ángulos con el mismo valor absoluto pero con signo contrario. Para desplazamientos totalmente verticales u horizontales la trayectoria era bastante mala, figuras 6.11 y 6.12.

Figura 6.11 - Trayectoria obtenida al mover al robot en un ángulo de 0° con los parámetros ajustados en el entrenamiento a 45°

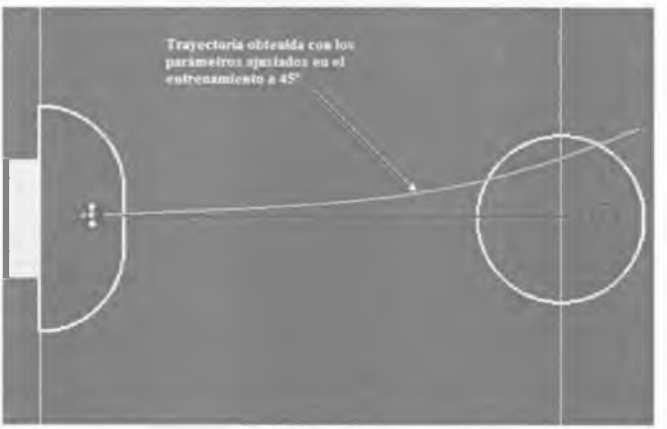


Figura 6.12 - Trayectoria obtenida al mover al robot en un ángulo de 90° con los parámetros ajustados en el entrenamiento a 45°

El error en los movimientos completamente verticales u horizontales, puede explicarse por la distinta contribución de los motores con respecto al ángulo de desplazamiento del robot debido a la forma en que están distribuidos. Es decir, cada motor se utiliza en un porcentaje

distinto de los demás dependiendo del ángulo en que se desplace el robot. Por ejemplo, si se moviera en un ángulo de ± 38°, la tracción sería proporcionada sólo por los motores 2 y 4, en cambio si se desplazara a ± 135° la tracción estaría dada por los motores 1 y 3. Por este motivo se vio conveniente entrenar al robot para los ángulos que más se le dificultan por la disposición de los motores: 0° y 90°.

6.4.2 Resultados del entrenamiento a 0°

El proceso de entrenamiento fue prácticamente igual que el de 45°, con la diferencia de la dirección en la cual fue enviado el robot para evaluar su desempeño. En la primera iteración se pudo observar que las trayectorias producidas por las políticas aleatorias fueron bastante malas, ya que la mayoría se desvió demasiado de su objetivo como lo refleja la figura 6.13.

Figura 6.13 - Trayectorias producidas por algunas Políticas de la Primera Iteración. Entrenamiento a 0°

Si comparamos el desempeño de las políticas durante la primera iteración para el entrenamiento a 45° y para el entrenamiento a 0°, podemos darnos cuenta de lo difícil que le resulta al robot desplazarse en la segunda dirección, figura 6.14. Todas las políticas analizadas con desplazamiento a 0° tuvieron una puntuación inferior, incluso, al de la peor política evaluada a 45°. Tan malas puntuaciones son consecuencia de haberse desviado considerablemente del objetivo.

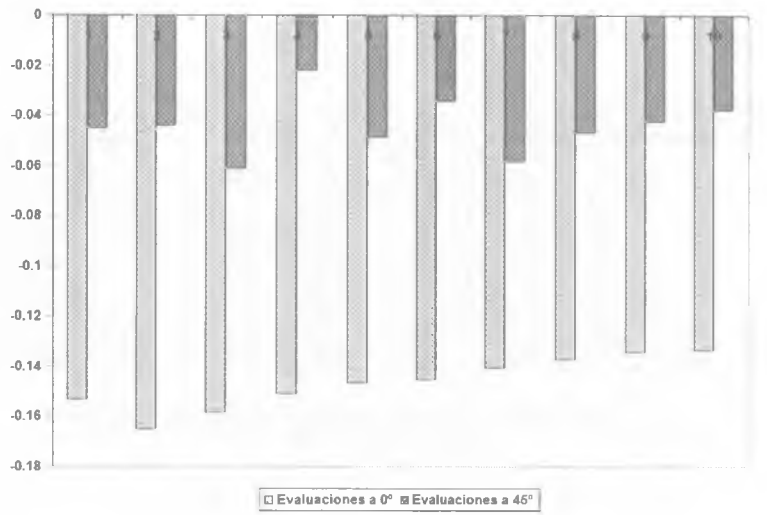


Figura 6.14 - Puntuación de las políticas en la Primera Iteración.
Entrenamiento a 45° y 0°

Antes de hacer el primer ajuste de parámetros en la dirección indicada por el gradiente, se previó que, si el entrenamiento continuaba igual que el anterior, la convergencia tardaría mucho más tiempo, debido a que el óptimo podría estar más alejado de los valores iniciales. Por tal motivo, se decidió acelerar el proceso modificando la tasa de aprendizaje en las primeras iteraciones, tabla 6.21. En cuanto hubo un progreso significativo se regresó a las tasas iniciales para hacer darle mayor precisión al ajuste.

Valor de η_i	**Parámetros en los que se utilizó**
0.3	p_1, p_4 y p_7, correspondientes a la constante K_p
0.03	p_2, p_5 y p_8, correspondientes a la constante K_i
0.03	p_3, p_6 y p_9, correspondientes a la constante K_d

Tabla 6.21 - Tasa de Aprendizaje por el tipo de Parámetro. Entrenamiento a 0°

Después de seis iteraciones los parámetros evolucionaron según lo indicado en la tabla 6.22. Como puede observarse en la tabla y en las figuras 6.15, 6.16 y 6.17 los ajustes más significativos fueron en los parámetros del controlador V_x.

Iteración	Controlador para V_x			Controlador para V_y			Controlador para ω		
	Kp	Ki	Kd	Kp	Ki	Kd	Kp	Ki	Kd
1	7.0	0.05	0.05	7.0	0.05	0.05	7.0	0.05	0.05
2	7.0	0.08	0.08	7.3	0.02	0.05	7.3	0.05	0.02
3	7.2	0.06	0.10	7.1	0.04	0.03	7.3	0.03	0.02
4	7.2	0.07	0.11	7.2	0.04	0.02	7.3	0.04	0.01
5	7.3	0.07	0.12	7.2	0.04	0.01	7.3	0.04	0.01
6	**7.4**	**0.07**	**0.12**	**7.2**	**0.04**	**0.01**	**7.3**	**0.04**	**0.01**

Tabla 6.22 - Evolución de los parámetros para cada Controlador PID.
Entrenamiento a 0°

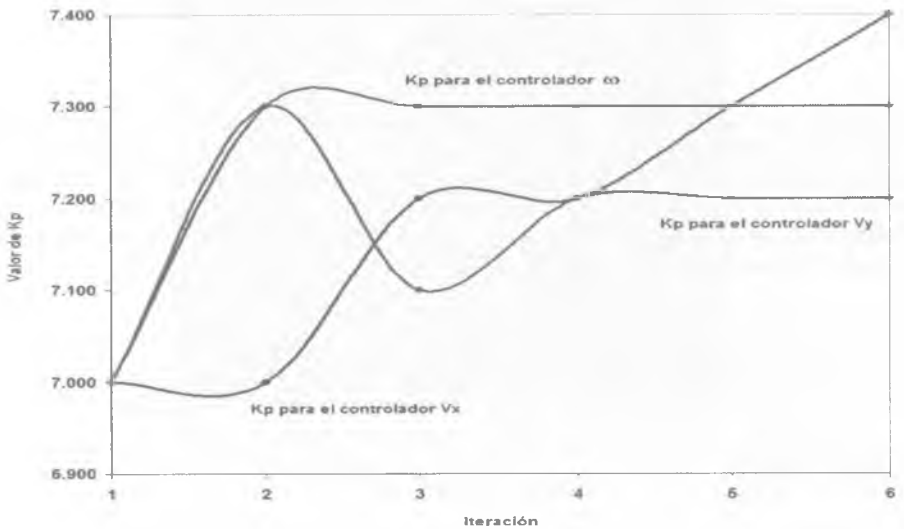


Figura 6.15 - Evolución del Parámetro Kp para cada Controlador.
Entrenamiento a 0°

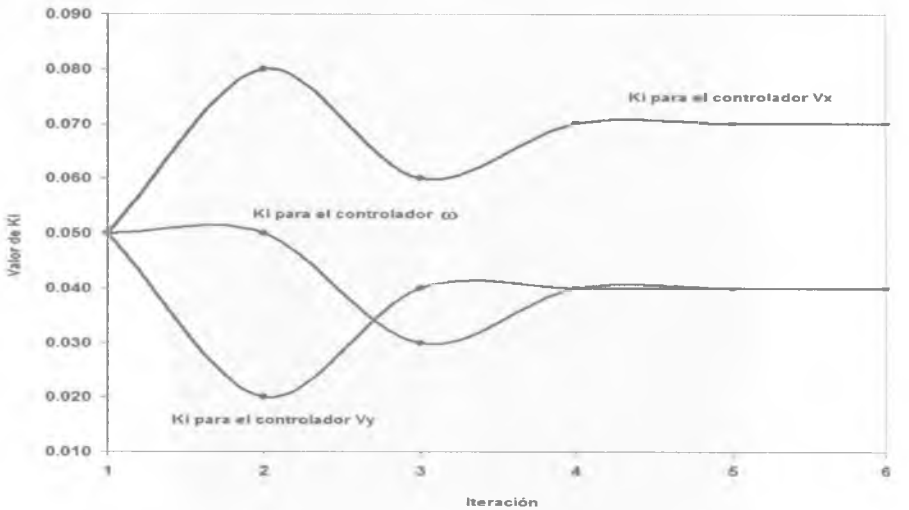


Figura 6.16 - Evolución del Parámetro Ki para cada Controlador.
Entrenamiento a 0°

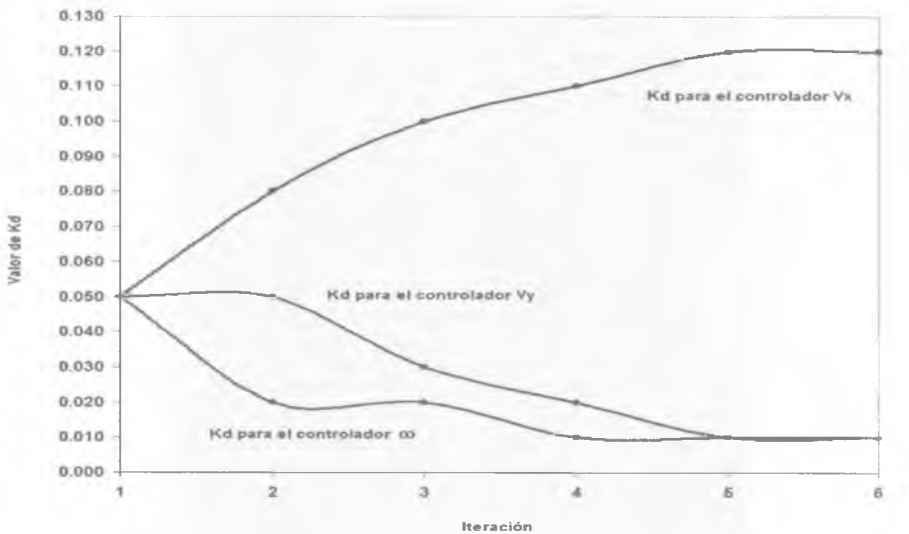


Figura 6.17 - Evolución del Parámetro Kd para cada Controlador.
Entrenamiento a 0°

A pesar que todos los parámetros convergieron en la sexta iteración, con excepción de Kp para el controlador V_x, y que el desempeño de cada política ajustada mejoraba conforme avanzaba el entrenamiento, figura 6.18 y 6.19, las trayectorias producidas por los parámetros aprendidos no resultaron completamente satisfactorias, figura 6.20. Es muy probable que esto se deba a que el óptimo local al que se llegó difiere bastante del global.

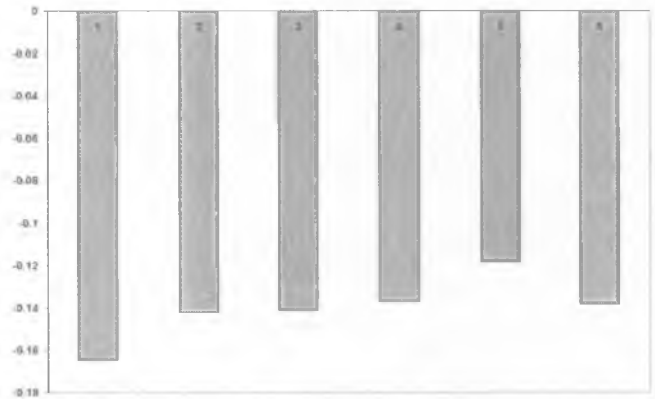


Figura 6.18 - Evolución del Desempeño de las Políticas Ajustadas durante cada Iteración. Entrenamiento a 0°

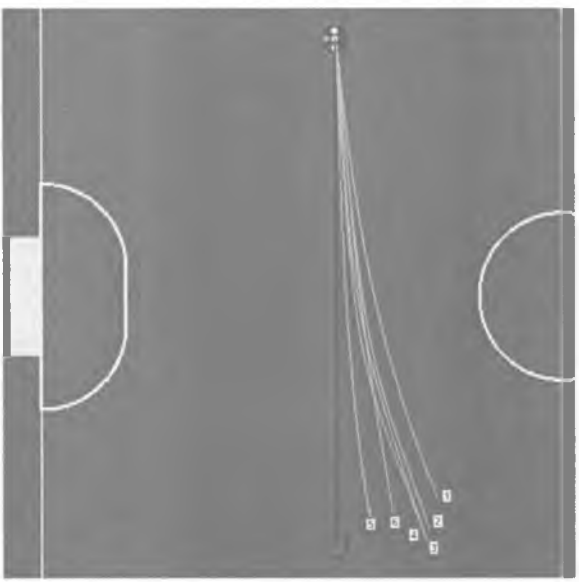


Figura 6.19 - Trayectorias Producidas por las Políticas Ajustadas en cada Iteración. Entrenamiento a 0°

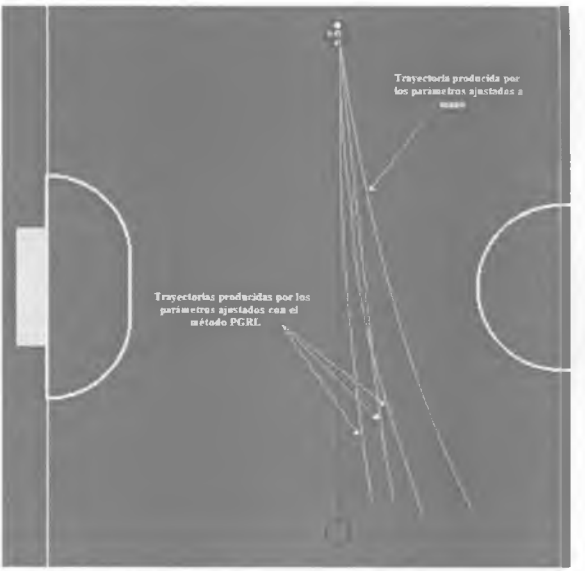


Figura 6.20 - Trayectorias producidas por los parámetros ajustados con el método PGRL. Entrenamiento a 0°

6.4.3 Resultados del entrenamiento a 90°

Contrario a lo que se esperaba, el desplazamiento a 90° tuvo menos complicaciones que el de 0° e incluso menos que el de 45°, esto puede apreciarse con el desempeño de las políticas aleatorias evaluadas en la primera iteración, figura 6.21 y las trayectorias producidas por las mismas, figura 6.22.

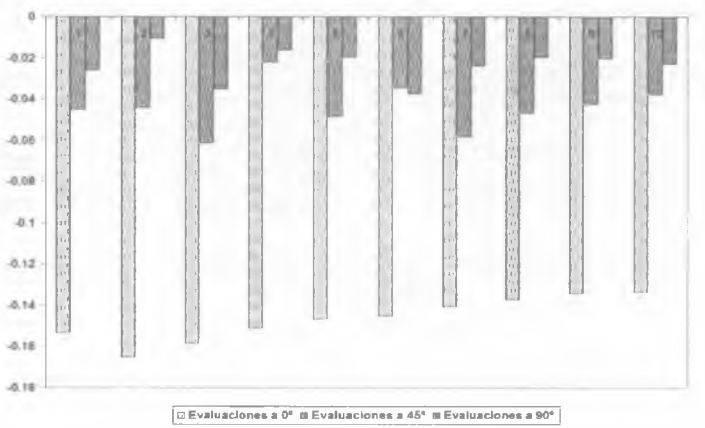


Figura 6.21 - Puntuación de las políticas en la Primera Iteración. Entrenamiento a 0°, 45° y 90°

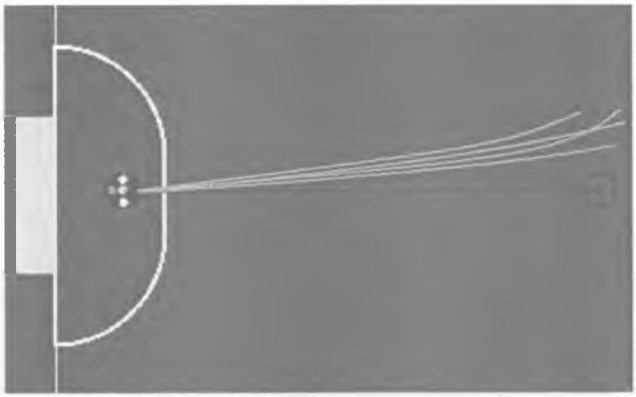


Figura 6.22 - Trayectorias producidas por algunas Políticas de la Primera Iteración. Entrenamiento a 90°

Las tasas de aprendizaje fueron las mismas del entrenamiento a 45°, ya que el ajuste sólo necesitaba mayor precisión. A partir de la cuarta Iteración empezaron a verse resultados satisfactorios y en la quinta los parámetros convergieron. Sin embargo, para analizar que ocurría si se realizaba otra iteración, se llevó a cabo una sexta. Como lo indica la tabla 6.23, en la última iteración sólo fueron afectados dos parámetros, Kd del controlador V_y y Ki del controlador ω, el resto permaneció constante. Este resultado puede damos una idea de la efectividad del algoritmo.

Iteración	**Controlador para V_x**			**Controlador para V_y**			**Controlador para ω**		
	Kp	Ki	Kd	Kp	Ki	Kd	Kp	Ki	Kd
1	7.0	0.05	0.05	7.0	0.05	0.05	7.0	0.05	0.05
2	7.1	0.05	0.05	7.0	0.04	0.04	7.1	0.06	0.04
3	7.2	0.05	0.06	7.1	0.05	0.03	7.1	0.07	0.03
4	7.1	0.05	0.05	7.1	0.06	0.02	7.1	0.08	0.02
5	7.1	0.05	0.05	7.1	0.06	0.02	7.1	0.08	0.02
6	**7.1**	**0.05**	**0.05**	**7.1**	**0.06**	**0.01**	**7.1**	**0.09**	**0.02**

Tabla 6.23 - Evolución de los parámetros para cada Controlador PID. Entrenamiento a 0°

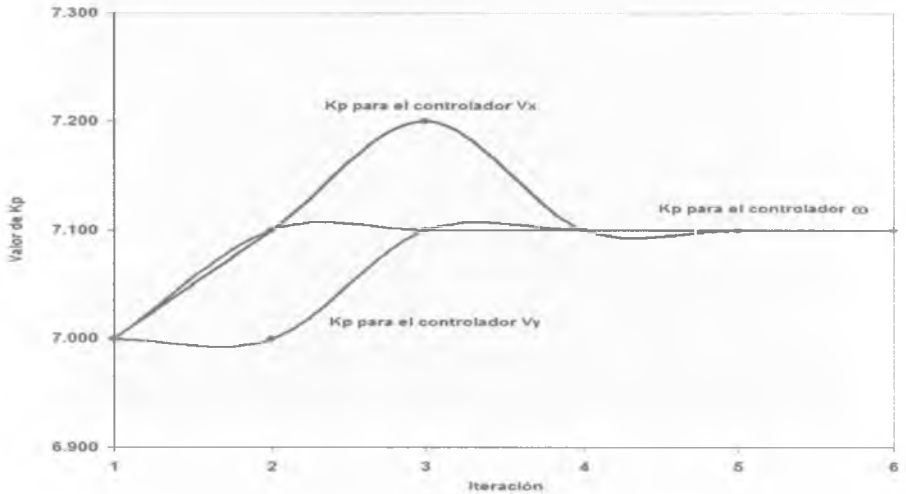


Figura 6.23 - Evolución del Parámetro Kp para cada Controlador.
Entrenamiento a 90°

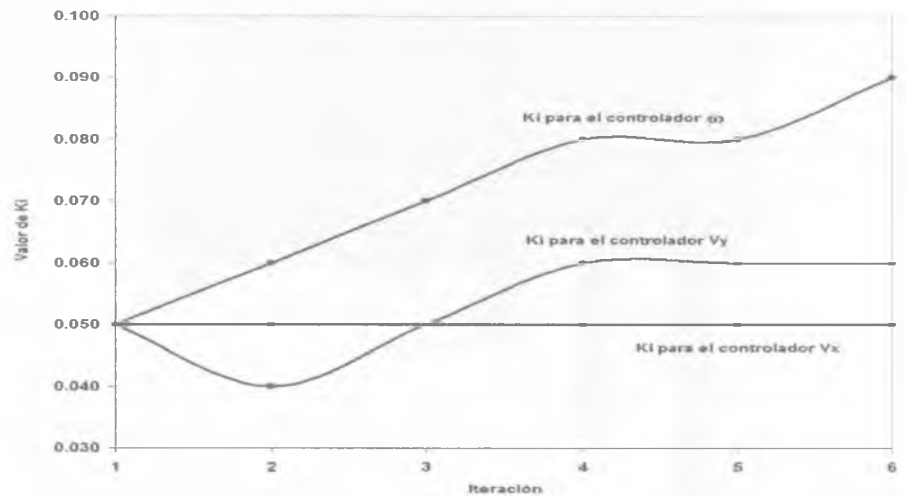


Figura 6.24 - Evolución del Parámetro Ki para cada Controlador.
Entrenamiento a 90°

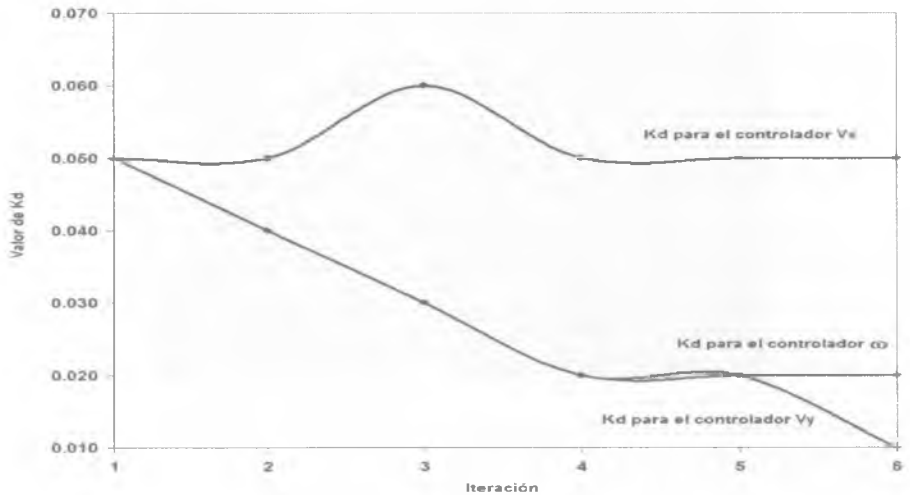


Figura 6.25 - Evolución del Parámetro Kd para cada Controlador.
Entrenamiento a 90°

En las tres figura anteriores podemos darnos cuenta que el ajuste fue hecho principalmente en los parámetros de los controladores V_v y ω. Los cambios V_v eran esperados por ser el vector en el cual se movió al robot, mientras que los ajustes en ω son para mantener al robot estable mientras avanza, ya que si gira sobre su eje la trayectoria variaría en la misma proporción.

En la figura 6.26 y 6.27 se muestra como fue mejorando el desempeño de las políticas ajustadas en cada iteración, de los tres entrenamientos éste fue el que mejores resultados obtuvo.

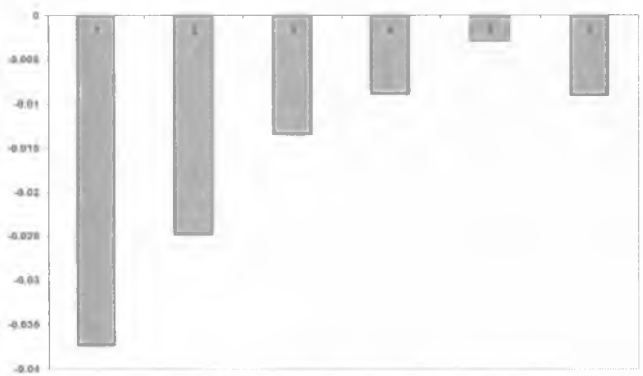


Figura 6.26 - Desempeño de las Políticas Ajustadas durante cada Iteración.
Entrenamiento a 90°

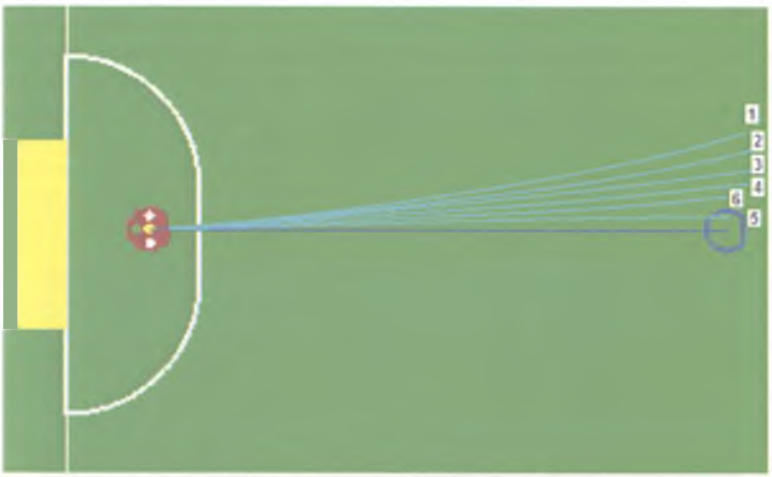


Figura 6.27 - Trayectorias Producidas por las Políticas Ajustadas en cada Iteración. Entrenamiento a 90°

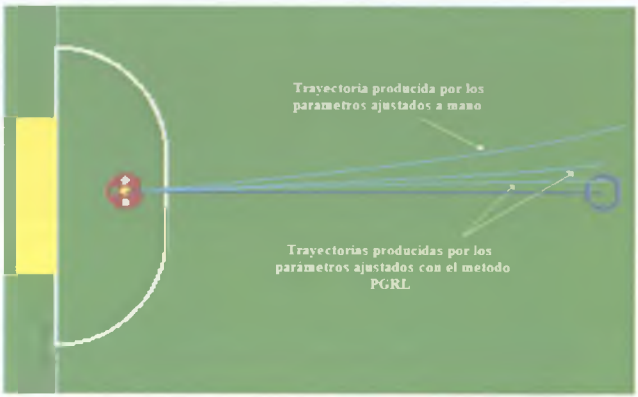


Figura 6.28 - Trayectorias producidas por los parámetros ajustados con el método PGRL. Entrenamiento a 90°

6.4.4 Resultado de los tres entrenamientos

Después de los entrenamientos se obtuvieron tres conjuntos distintos de parámetros, tabla 6.24, cada uno de los cuales funciona relativamente bien para desplazamientos en ángulos similares a los del entrenamiento, o bien, en ángulos con magnitud similar pero con signo contrario. Por ejemplo, los parámetros obtenidos para 90° también funcionan para desplazamientos a 270° o cercanos; los ajustados para 0° sirven para moverse a 180°; y los

de 45° para 135°, 225° y 315°. Este comportamiento nos condujo a establecer el criterio de la figura 6.29 para utilizar los parámetros obtenidos dependiendo del ángulo de desplazamiento requerido.

Entrenamiento	Controlador								
	V_x			V_y			ω		
	Kp	Ki	Kd	Kp	Ki	Kd	Kp	Ki	Kd
45°	7.1	0.01	0.07	6.9	0.05	0.03	7.1	0.06	0.03
0°	7.4	0.07	0.12	7.2	0.04	0.01	7.3	0.04	0.01
90°	7.1	0.05	0.05	7.1	0.06	0.01	7.1	0.09	0.02

Tabla 6.24 - Parámetros PID obtenidos en cada Entrenamiento

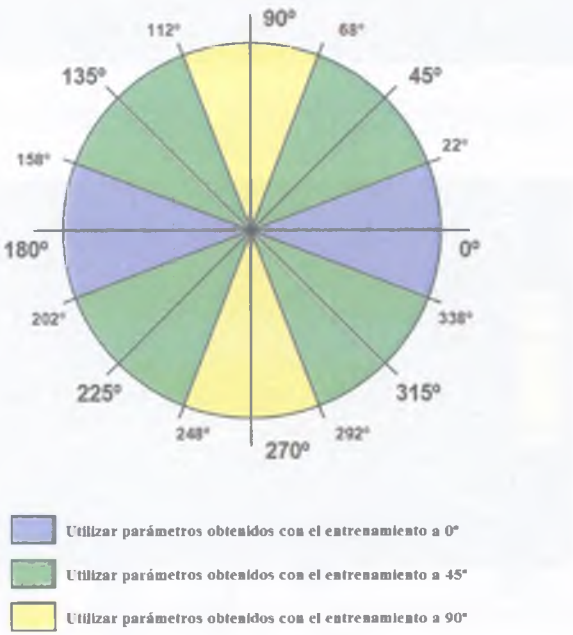


Figura 6.29 - Criterio para utilizar los parámetros obtenidos en cada entrenamiento

Con el criterio anterior no es necesario que el sistema de IA envíe los parámetros requeridos dependiendo del ángulo, como se hizo en el entrenamiento, simplemente bastó con verificar, en el programa del DSP, en que sector se encuentra el ángulo para actualizar los parámetros con base en el mismo. De este modo, es posible regresar a la trama de comunicación descrita en la sección 4.1.4.2.1.

Después de integrar en el robot los parámetros obtenidos con el respectivo criterio de uso de los mismos, se realizaron pruebas enviando al robot en diferentes direcciones, y se comprobó que pudieron seguir la trayectoria deseada de una forma más rápida y precisa que con los parámetros ajustados a mano. Sin embargo, las pruebas mostraron que persistió una pequeña desviación del objetivo de aproximadamente 10% en el punto final, con excepción de trayectorias en ángulos cercanos a los 0° y 180°, cuya desviación fue mayor, 15%, como se muestra en la figura 6.30.

Figura 6.30 - Trayectorias obtenidas con los nuevos parámetros

En la figura anterior se puede observar que, con los nuevos parámetros PID, las trayectorias mejoraron considerablemente; sin embargo, en todas ellas persiste un error que es producido por un derrape en las ruedas, el cual es independiente de la dirección de desplazamiento.

El derrape se produce cuando las ruedas pierden adherencia sobre la superficie y el robot se desplaza lentamente ajeno al control indicado. Durante los experimentos se pudo observar que dicho efecto ocurre al iniciar el movimiento, en cuanto deja de producirse, el robot continua su camino pero con cierto error. Con el fin de eliminar o reducir el efecto que el robot siga su trayectoria con el menor error posible, por lo tanto, la siguiente sección explicará una forma de conducir al robot sin derrape y, en caso de que ocurra, como detectarlo y corregirlo tal como se explicada en [35].

6.4.5 Como conducir al robot sin derrape

Supóngase que se desea desplazar un robot SSL hacia adelante tratando de evitar cualquier derrape en las ruedas. Ahora bien, suponiendo que se ha determinado experimentalmente que cuando el voltaje para los motores es un máximo de X volts las ruedas no giran (es posible determinar el valor de X colocando al robot en el piso y aumentando de la tensión en el motor hasta que las ruedas comienzan a girar).

Lo que se tendría que hacer es comenzar con los motores en $V_0 = X$ y dejar que el robot ruede hacia adelante. Después de algunos milisegundos, la corriente inducida en el rotor E disminuye el voltaje eficaz en el rotor del solenoide para $V_0 - E$, y el par motor correspondiente. Entonces, se puede aumentar el valor de la tensión a $V_1 = V_0 + E$, y ahora el par motor corresponde al voltaje eficaz $V_0 + E - E = V_0$. Repetir periódicamente este ajuste, permite girar los motores con el máximo torque posible sin que las ruedas se derrapen. En el límite, cuando el ajuste se hace con mucha frecuencia, el torque se mantiene constante, la aceleración de la rueda también, y la velocidad angular del motor aumenta linealmente. Sin embargo, este resultado depende de una perfecta distribución de la masa del robot, a fin de que todas las ruedas ejerzan la misma presión sobre el suelo. Esta hipótesis no se puede garantizar cuando el robot es desplazado rápidamente, porque a veces la aceleración puede levantar del piso la parte frontal o lateral del robot, o al menos disminuir la presión que ejercen las ruedas sobre el suelo. Por lo tanto el derrape de las ruedas es algo que no puede ser evitado al cien por ciento y tienen que ser detectado y corregido en el momento que ocurra, como se muestra en la siguiente sección.

6.4.5.1 Detección y Corrección del Derrape

Considerando el diseño simétrico del robot, sea m el vector de las velocidades individuales de los motores $(v_1, v_2, v_3, v_4)^T$, D la matriz de acoplamiento de velocidades, y v el vector de las velocidades en el plano euclidiano $(v_x, v_y, R\omega)^T$.
El hecho de que las velocidades euclidianas del robot y las velocidades de los motores estén conectadas por las expresiones:

$$m = Dv \quad v = D^+ m$$

abre la posibilidad de probar inconsistencias en la velocidad de los motores y, por lo tanto, detectar el derrape de las ruedas.

El robot obtiene de IA el vector v y lo transforma en las velocidades de motores necesarias m. Enseguida, a través de los *encoders*, se calcula la velocidad real en que giran los motores m'.

Si $v' = D^+ m'$ y $m' = Dv'$, entonces debe ser cierto que $m' = DD^+ m$. Si no es así, entonces una o más ruedas se están derrapando sobre el suelo. Un simple cálculo muestra que para un robot simétrico con ángulo φ:

$$DD^+ = \begin{pmatrix} 3 & 1 & -1 & 1 \\ 1 & 3 & 1 & -1 \\ -1 & 1 & 3 & 1 \\ 1 & -1 & 1 & 3 \end{pmatrix}$$

Al multiplicar por la matriz anterior es muy fácil comprobar si las ruedas se derrapan, ya que se debe obtener $(I - DD^+)m' = 0$. La matriz $I - DD^+$ esta dada por:

$$I - DD^+ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} - \frac{1}{4}\begin{pmatrix} 3 & 1 & -1 & 1 \\ 1 & 3 & 1 & -1 \\ -1 & 1 & 3 & 1 \\ 1 & -1 & 1 & 3 \end{pmatrix}$$

$$= \frac{1}{4}\begin{pmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{pmatrix}$$

La prueba de consistencia, que muestra dicha matriz, se puede reducir a la siguiente prueba:

$$v_1 - v_2 + v_3 - v_4 = 0$$

o de forma equivalente:

$$v_1 + v_3 = v_2 + v_4$$

Es decir, la suma de las velocidades del motor 1 con su opuesto, motor 3, es igual a la suma de las velocidades del motor 2 con su opuesto, motor 4. Las velocidades utilizadas anteriormente, que son $(1,1,1,1)$ para rotación, $(1,-1,-1,1)$ para avanzar al frente, y $(-1,-1,1,1)$ para moverse a los lados, cumplen con la prueba de consistencia, incluso cualquier combinación lineal de ellas también la cumple.

Es importante señalar que esta detección de derrape no depende del ángulo φ. Se trata de una prueba para cualquier robot omnidireccional con cuatro ruedas simétricamente distribuidas con un ángulo φ.

Si se detecta que la rotación de las ruedas no es coherente, es decir, que una o más ruedas están derrapando, es posible corregirlo, para ello se debe calcular la proyección de

$m = (v_1, v_2, v_3, v_4)^T$ en la dirección del vector unitario $\frac{1}{2}(1,-1,1,-1)^T$ y restarlo de m. Entonces el vector m corregido es:

$$m' = m - (m \cdot \frac{1}{2}(1,-1,1,-1)^T)\frac{1}{2}(1,-1,1,-1)^T$$

La corrección puede simplificarse a:

$$m = m - \frac{v_1 - v_2 + v_3 - v_4}{4}(1,-1,1,-1)^T$$

Implementar la detección y corrección del derrape permitiría mejorar aún más los resultados obtenidos en el control de trayectorias.

- CAPÍTULO 7 -

CONCLUSIONES Y LÍNEAS FUTURAS

En este último capítulo se presentan las conclusiones del trabajo desarrollado en esta tesis. Primero se exponen las conclusiones particulares de la generación EK2007-EK2008 y enseguida se presentan las del Control de Trayectorias. Posteriormente se analizan las líneas futuras del proyecto para finalizar con unas conclusiones generales.

7.1 Conclusiones de la generación EK2007-EK2008

Los robots EK2007-EK2008, más que un gran avance comparado con las primeras generaciones, representan la estabilización de varios años de trabajo de los anteriores integrantes del Laboratorio.

Se mejoraron aspectos importantes como el sistema de pateo, cuyo nuevo diseño prescindió de los componentes inestables; se le dio mayor modularidad a la arquitectura para preveer futuros cambios en algún sistema; los nuevos motores requirieron controladores más eficientes; al cambiar el tipo de corrección PID por motores a corrección por grados de libertad se tuvo un mejor aprovechamiento del DSP y se libró al sistema de IA de hacer la transformación de velocidades euclidianas a velocidades de motores. Además, los DSP's dejaron de ser dañados gracias al ajuste de las señales que recibían.

En general, los robots tuvieron un buen desempeño, prueba de ello fue el séptimo lugar mundial en el *RoboCup* 2007, en el que su movilidad y control estuvieron a la altura de la competencia. Obviamente, esa posición no fue sólo el resultado de los nuevos robots, fue el resultado conjunto de todos los sistemas del equipo.

Sin embargo, aún hay algunos aspectos por mejorar, sobre todo en lo que históricamente ha sido nuestro mayor problema: el sistema de pateo. Si bien el sistema descrito en este trabajo fue estable y permitió anotar goles en el *RoboCup* 2007, su potencia aún dejó mucho que desear, ya que de poco sirve tener una buena estrategia y movilidad en la cancha si el robot no golpea con la suficiente fuerza cuando es necesario. Incluso, equipos que tenían un desempeño inferior al nuestro, en varios sentidos, lograron ganarnos en el *RoboCup* 2008 simplemente por que golpeaban muy fuerte la pelota.

Otro aspecto que se debe considerar para el diseño de nuevos robots es la forma en que los motores transmiten su fuerza hacia las ruedas, según lo descrito en la sección 6.3.1, el 76 % de los equipos comparados utiliza un sistema de engranes para darle tracción a las ruedas,

lo que les da mayor precisión en el control del robot. Adicionalmente, de ese 76% un 70 % utiliza motores *brushless*, cuya principal ventaja es el poco mantenimiento que requieren y su mayor tiempo de vida.

En lo que respecta al tipo de procesador central, el DSP ha dado muy buenos resultados al cubrir las necesidades de procesamiento de los robots actuales; sin embargo, el potencial de que dispone continua sin ser explotado. Pero, si se considera la posibilidad de que en un futuro la liga decida cambiar la visión global por visión local, por ser esta última la que más se apega a la realidad, implicaría realizar procesamiento de imágenes en el robot, en cuyo caso el DSP resultaría la mejor opción, al ser un procesador optimizado para dicho tipo de aplicaciones.

Finalmente, es preciso mencionar que a pesar de las derrotas en el torneo de 2008, producto de haber hecho cambios sin probarlos exhaustivamente, el haber mantenido la misma arquitectura electrónica por dos años consecutivos permitió al equipo explorar nuevas soluciones para los problemas planteados en la liga. Por ejemplo, el nuevo Sistema de Inteligencia Artificial basado en un Sistema Experto [54] que desarrolló Ernesto Torres para el *RoboCup* 2008, o la implementación del método de Aprendizaje por Refuerzo descrito en este trabajo para obtener los parámetros óptimos de los controladores PID. Esto pone al descubierto que, para seguir aportando innovaciones a la liga, es necesario que el equipo supere cuanto antes los problemas de hardware que le impiden obtener mejores resultados.

7.2 Conclusiones del Control de Trayectorias

El control de trayectorias resulta crítico para el desempeño individual y colectivo de los robots SSL, ya que de ello depende todo lo relacionado con el desplazamiento dentro de la cancha: seguir y llegar correctamente a la pelota, bloquear los tiros del contrincante, evadir obstáculos, posicionarse para recibir un pase, etc. Por lo tanto, se decidió aplicar el enfoque del Aprendizaje por Refuerzo con Gradiente de la Política con el fin de optimizar el comportamiento de conducción de los robots.

La técnica que se utilizó encuentra los parámetros óptimos que permiten al robot conducirse con un menor error; una conducción más precisa se traduce en un mejor movimiento general, en un posicionamiento más robusto, y una mejor predicción de la posición futura del robot por parte el Sistema de IA.

La ventaja del método de Aprendizaje por Refuerzo con Gradiente de la Política (PGRL) es que no requiere un modelo analítico del hardware del robot y puede ser especialmente utilizado cuando se hacen modificaciones al mismo: cambios en el peso, cambios en el tamaño u orientación de las ruedas, cambio de motores, etc. Por ejemplo, en el Laboratorio de Robótica del ITAM se está trabajando en el diseño de nuevos robots con motores *brushless*, en este caso es claro que los parámetros PID deberán ajustarse para las nuevas características mecánicas y electrónicas.

Sin embargo, al igual que otras técnicas de carácter más general, el enfoque sólo converge hacia un óptimo local. En contraste con algunos algoritmos basados en la Función de Valor (VFRL), tales como *Q-Learning*, que probablemente convergen a un óptimo global, pero en dicho tipo de métodos un cambio infinitesimal en los parámetros de la función puede impulsar el valor de una acción más que de otro, lo que provoca un cambio discontinuo en la política, los estados visitados y el rendimiento general. Tales cambios discontinuos han sido identificados como un obstáculo clave para establecer garantías de convergencia. En cambio con el método PGRL, los pequeños cambios en los parámetros sólo pueden causar pequeños cambios en la política y en la distribución de los estados visitados.

Según lo indicado en la sección 6.4.4, las trayectorias obtenidas con los nuevos parámetros aún tienen un pequeño error producido por el derrape de las ruedas, para reducirlo es conveniente implementar lo indicado en la sección 6.4.5, con lo cual se conseguiría una trayectoria mucho mejor; sin embargo, en el entorno de un partido, de poco sirve al robot llegar correctamente a la pelota si no la golpea con la suficiente fuerza. Por lo tanto, para ganar más partidos, los resultados del presente trabajo se deben combinar con un mejor sistema de pateo.

7.3 Líneas Futuras

Los Sistemas de Visión e Inteligencia Artificial del equipo *Eagle Knights* han alcanzado un nivel de madurez que está a la par de los mejores equipos de la liga, a tal punto que el factor decisivo para figurar entre los primeros lugares del mundo esta determinado por el hardware de los robots.

Ante dicha situación, es necesario que los nuevos integrantes del Laboratorio enfoquen sus esfuerzos en el diseño de robots mucho más estables y duraderos con, al menos, las siguientes características: motores *brushless*, un diseño mecánico que permita transmitir de forma más eficiente la fuerza de los motores a las ruedas y, para el sistema de pateo, un generador de alto voltaje que proporcione un mínimo de 200V. Si se desarrollan robots más competitivos, es factible conseguir mejores resultados en los próximos torneos, incluso, utilizando los sistemas de Visión e Inteligencia Artificial actuales.

Sin embargo, para conquistar el objetivo de *RoboCup*, los progresos conseguidos en cada liga deben adaptarse a la de humanoides, ya que la meta de 2050 esta basada en ellos. En este sentido, la liga SPL (*Standard Plaftorm League*) ya dio el siguiente paso al sustituir los robots AIBO de Sony (cuadrúpedos) por los robots NAO, humanoides de la empresa *Aldebaran Robotics*.

Uno de los grandes problemas a los que se enfrenta dicha liga es la caminata de los robots, ya que, al ser bípedos, requieren equilibrarse para no caer. El software de control de los robots NAO provee una caminata predeterminada pero, como pudo comprobarse en el *RoboCup* 2008, no es la óptima para el tipo de superficie donde se juegan los partidos, los robots difícilmente avanzaban más de dos metros sin perder el equilibrio y caer. Sin embargo, también es posible ajustar ciertos parámetros para establecer una caminata propia.

La característica anterior puede ser aprovechada como una línea de futuro trabajo: el algoritmo de Aprendizaje por Refuerzo con Política de Gradiente podría ser adaptado para aprender los parámetros óptimos que permitan a los robots NAO conseguir una caminata más rápida y eficiente, tal como lo hizo en su momento el equipo de la Universidad de Texas en Austin para los robots AIBO [23].

7.4 Conclusiones Generales

Desde 2003, el Laboratorio de Robótica del ITAM ha enfocado sus esfuerzos para representar dignamente a México en las competencias internacionales, año tras año los nuevos integrantes consiguen mejorar los sistemas desarrollados por sus antecesores, en el caso de la liga *Small Size*, los avances se producen no sólo en el hardware de los robots, sino también en los sistemas de Visión e Inteligencia Artificial. Para continuar con ese progreso, es necesario que el conocimiento generado dentro del Laboratorio no se pierda, en ese sentido, documentos como el presente llevan implícito un objetivo extra: sentar memoria del trabajo realizado para que los nuevos integrantes del laboratorio tengan una guía que los encamine al desarrollo de nuevas soluciones que les permitan mejorar los resultados en las competencias. Sin embargo, para darle mayor sentido a los progresos alcanzados, es preciso que la motivación para mejorar a los robots no sea sólo figurar en los primeros lugares de los torneos, sino buscarles nuevas aplicaciones, fuera del contexto *RoboCup*, que sean relevantes para la sociedad en que vivimos.

REFERENCIAS BIBLIOGRÁFICAS

[1] Aeström Karl J. y Hägglund Tore, "*Automatic tuning and adaptation for PID controllers—A survey"*, en Dugard L, M'Saad M. y Landau I., "*Adaptive Systems in Control and Signal Processing"*, Pergamon Press, Oxford, 1992, pp. 371–376.

[2] Aeström Karl J. y Hägglund Tore, *"PID Controllers: Theory, Design, and Tuning"*, Segunda Edición, Research Triangle Park, NC, Instrument Society of America, 1995.

[3] Baird L. C. "*Residual algorithms: Reinforcement learning with function approximation"*. En *"Proc. of the Twelfth Int. Conf on Machine Learning"*, Morgan Kaufmann, 1995, pp. 30-37.

[4] Bakhshandeh O, Azidehak A., Gorji M. y Ahmad S., *"MRL Small Size 2008 Team Description"*, Islamic Azad Universit of Qazvin, Electrical Engineering and Computer Science Department, Mechatronics Research Lab, Qazvin, Iran, 2008. [http://small-size.informatik.uni-bremen.de/tdp/2008/MRL_SSL_TDP_2008.pdf]

[5] Bardagjy A., Shtylman R., Posey S., Kulpe J., Marks P., Johnson B., McConville J., Donnan S., Nemat Y., y Nieman J., *"RoboJackets 2008 Team Description Paper"*, Georgia Institute of Technology - Atlanta, Georgia, USA, 2008. [http://small-size.informatik.uni-bremen.de/tdp/2008/RoboJackets_SSL_TDP_2008.pdf]

[6] Barr Michael. "*Pulse Width Modulation*" *Embedded Systems Programming*, Septiembre 2001, pp. 103-104.

[7] Bateman Andy y Paterson-Stephens Iain, "*The DSP Handbook*", Prentice Hall, 2001.

[8] Baxter J. y Bartlett P. L. "*Infinite-horizon policy-gradient estimation*". En "*Journal of Artificial Intelligence Research"*, vol. 15, 2001. pp. 319–350

[9] Bellman R. E., "*Dynamic Programming"*. Princeton University Press, NJ.

[10] Bertsekas D. P., y Tsitsiklis, J. N., "*Neuro-Dynamic Programming"*. En Cao, X. R., Chen, H. F. 1997. "*Perturbation realization, potentials, and sensitivity analysis of Markov Processes"*, *IEEE Trans. on Automatic Control 42*(10):1382-1393.

[11] Brown Jim, "*Brief H-Bridge Theory Operation*", Abril 1998. [http://www.dprg.org/tutorials/1998-04a]

[12] Bruce, James; Zickler, Stefan; Licitra, Mike y Veloso, Manuela *"CMDragons 2006 Team Description Paper"* Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 2006.
[http://small-size.informatik.uni-bremen.de/tdp/2006/CMDragons_SSL_TDP_2006.pdf]

[13] Department of Precision Machinery and Instrumentation University of Science & Technology of China, *"Wright Eagle Team of USTC"*, China, 2008.

[14] Digi-Key Corporation. "*Digi-Key Corporation – USA Home Page*"
[http://www.digikey.com]

[15] Field Timothy, "*Policy-Gradient Learning for Motor Control*", Tesis Victoria University of Wellington 2005. pp 17.

[16] Francois Aragón, Juan Pablo, *Arquitectura de Visión Local para Robots Móviles del ITAM.* Tesis (Ingeniero en Computación) ITAM, 2006.

[17] Ghiasvand A., Mehdi K., Rohani M., Ghaednia H., Sadatmand S., y Monajjemi V., *"Parsian Team Description for Robocup 2008",* Electrical Engineering Department, Amirkabir University of Technology (Tehran Polytechnic) Tehran, Iran, 2008.
[http://small-size.informatik.uni-bremen.de/tdp/2008/Parsian_SSL_TDP_2008.pdf]

[18] Gloye A., Göktekin C., Egorova A., Tenchio O. y Rojas R., *"Learning to Drive and Simulate Autonomous Mobile Robots"*. En: "*RoboCup-2004: Robot Soccer World Cup VIII"*, Springer, 2004.

[19] Gordon G. J., *"Stable function approximation in dynamic programming"*. En "*Proceedings of the Twelfth Int. Conf. on Machine Learning"*, Morgan Kaufman, 1995, pp. 261-268.

[20] Gordon G. J., "*Chattering in SARSA"*, En CMU Learning Lab Technical Report. Jaakkola T., Singh, S. P. y Jordan, M. I. "*Reinforcement learning algorithms for partially observable Markov decision problems"*, *NIPS 7*, Morgan Kaufman. 1996, pp. 345-352.

[21] Jindarak K., Anongthong N., Srithong A., Saelee K., Boonprakob V., Surakamhaeng T., y Karnjanadecha M., *"Khainui Team Description for Robocup 2008",* Department of Computer Engineering, Faculty of Engineering Prince of Songkla University, Songkhla, Thailand.
[http://small-size.informatik.uni-bremen.de/tdp/2008/Khainui_SSL_TDP_2008.pdf]

[22] Kriengwattanakul A., Wattanasarn S., Suntharasantic S., Chanpichaigosol S., Kokaewwichian C., Wattanavekin T., Ovatlarnporn P., Phayong S., Tongoa N., Changwichukarn T., Leesatapornwongsa T., Wannasuphoprasit W., Wongsaisuwan M., *"PLASMA-Z 2008 Team Description Paper".* Chulalongkorn University, Thailand, 2008.

[23] Kohl Nate y Stone Peter, *"Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion"*. En "*Proceedings of the IEEE International Conference on Robotics and Automation"*, New Orleans, LA, May 2004. pp. 2619-2624.

[24] Laboratorio de Robótica ITAM, "Eagle Knights". [http://robotica.itam.mx/beta/index.html].

[25] Laue Tim, Burchardt A., Cierpka K., Fritsch S., Göde N., Huhn K., Kirilov T., Lassen B., Lyatif E., Miezal M., Modzelewski M., Nehmiz U., Schwarting M., Seekircher A., Stein R., *"B-Smart (Bremen Small Multi-Agent Robot Team) Team Description for RoboCup 2008"*, Fachbereich 3 – Mathematik, Informatik Universität Bremen, Bremen, Germany, 2008. [http://small-size.informatik.uni-bremen.de/tdp/2008/BSmart_SSL_TDP_2008.pdf]

[26] Liu Yang, Chomentowski Rafal y Glocker Tobias, *"Botnia Dragon Knights - Team Description Paper for RoboCup 2008 Small Size League"*, Department of Information Technology and Department of Computer Science Vaasa University of Applied Sciences, Finland, 2008. [http://small-size.informatik.uni-bremen.de/tdp/2008/BotniaDragonKnights_SSL_TDP_2008.pdf]

[27] Maeno J., Achiwa H., Moribayashi T., Tamaki J., Hiramoto N., Nakanishi R., Narita R., Otake K., Tanaka M., Ueno S., Murakami K. y Tadashi, *"RoboDragons 2008 Team Description"*, Naruse Aichi Prefectural University, Japan, 2008. [http://small-size.informatik.uni-bremen.de/tdp/2008/RoboDragons_SSL_TDP_2008.pdf]

[28] Martínez Gómez, Luis, *Sistema de Visión Para el Equipo de Robots Autónomos del ITAM*. Tesis (Ingeniero en Computación) ITAM, 2004.

[29] Moneo Soler, Francisco, *Sistema de Planeación de Alto Nivel para RoboCup*. Tesis (Ingeniero en Telemática e Ingeniero en Computación) ITAM, 2005.

[30] Ollero Baturone, Aníbal, *Robótica Manipuladores y robots móviles*, Marcombo, primera edición, 2001. pp. 171

[31] PCBexpress "*PCBexpress home prototype printed circuit board manufacturer – RoHS Compliant Circuit Boards*". [http://www.pcbexpress.com]

[32] PICO Electronics, Inc. "*Pico Electronics: Surface Mount Series IR DC to DC Coverters, Single and Dual*". [http://www.picoelectronics.com/dcdclow/pe88_89.htm]

[33] Pu Li, Zhirui Wang, Fuben He, Yijie Huangfu, Jinghui Jia, Liang Zhao, Taiyun He y Hang Li, *"FANTASIA 2008 Team Description Paper"*, School of Innovation and Experiment Dalian University of Technology China, 2008. [http://small-size.informatik.uni-bremen.de/tdp/2008/FANTASIA_SSL_TDP_2008.pdf]

[34] Radiometrix Ltd, "*UHF Radio Packet Controller*", Inglaterra, Noviembre 2004.

[35] Rojas, Raúl. "*Omnidirectional Control*". Freie Universität Berlín, Alemania.

[36] Shanghai University, *"Strive(Small Size) Team Description 2008"*, China, 2008. [http://small-size.informatik.uni-bremen.de/tdp/2008/Strive_SSL_TDP_2008.pdf]

[37] Singh, S. P., Jaakkola, T., Jordan, M. I. "*Learning without state-estimation in partially observable Markovian decision problems". Proc. ICML-94*, pp. 284-292.

[38] Srisabye J., Hoonsuwan P., Bowarnkitiwong S., Onman C., Wasuntapichaikul P., Signhakarn A., Wechsuwanmanee P., Dumnernkittikul R., Sukvichai K., y Tipsuwan Y., *"Skuba 2008 Team Description*" Faculty of Engineering, Kasetsart University, Bangkok. 2008 [http://small-size.informatik.uni-bremen.de/tdp/2008/Skuba_SSL_TDP_2008.pdf]

[39] STMicroelectronics, *"L6207 - DMOS Dual Full Bridge Driver with PWM Current Controller"*, 2002.

[40] Society of Robots, *"Actuators – Solenoids"* [http://www.societyofrobots.com/actuators_solenoids.shtml]

[41] Sotelo Iniesta, Edgar, *Diseño e Implementación de los Robots F180 del ITAM.* Tesis (Ingeniero en Telemática) ITAM, 2006.

[42] Soto Ruíz, Misael David, *Diseño e implementación del sistema de control de pelota* (Ingeniero Industrial) ITAM, 2007.

[43] Spectrum Digital Inc. "*Spectrum Digital Inc.*" [http://www.spectrumdigital.com]

[44] Sutton Richard y Barto Andrew, "*Reinforcement Learning: An Introduction*". Cambridge, Massachusetts- USA. Bradford Book, MIT Press. 1998. 320p.

[45] Sutton R., McAllester D., Singh S., y Mansour Y., "*Policy Gradient Methods for Reinforcement Learning with Function Approximation*". En: "*Advances in Neural Information Processing Systems"*, Solla S. A., Leen T. K., y Muller K. R., vol. 12. The MIT Press, Cambridge, USA 2000. pp. 1057–1063.

[46] The RoboCup Federation, "*RoboCup Official Site*". [http://www.robocup.org]

[47] The RoboCup Federation, Small-Size League, *"Small Size Robot League – robocup2006:results"*. [http://small-size.informatik.uni-bremen.de/robocup2006:results]

[48] The RoboCup Federation, Small-Size League, *"Small Size Robot League – robocup2007:results"*.
[http://small-size.informatik.uni-bremen.de/robocup2007:results]

[49] The RoboCup Federation, Small-Size League, *"Small Size Robot League – robocup2008:schedule_and_results"*.
[http://small-size.informatik.uni-bremen.de/robocup2008:schedule_and_results]

[50] The RoboCup Federation, Small-Size League, *"Small Size Robot League – rules:main"*.
[http://small-size.informatik.uni-bremen.de/rules:main#rules_2007]

[51] The RoboCup Federation, Small-Size League, *"Small Size Robot League - start"*.
[http://small-size.informatik.uni-bremen.de]

[52] Torres E., Martínez L., Muñoz M., Rodríguez J., Soto M., Silva J., Murrieta Y., Faba A., Chavarría C, y Weitzenfeld A., *"Eagle Knights-RoboBulls 2007: Small Size League"*, Laboratorio de Robótica, ITAM ,México, 2007.

[53] Torres E., Rodríguez J., Aguilar P., Ponce O., Moses A., y Weitzenfeld A., *"Eagle Knights-RoboBulls 2008: Small Size League"*, Laboratorio de Robótica, ITAM, México y RoboCup Laboratory, College of Engineering, University of South Florida, Tampa FL, 2008.
[http://small-size.informatik.uni-bremen.de/tdp/2008/EagleKnights_SSL_TDP_2008.pdf]

[54] Torres Vidal, Ernesto, *Sistema de Inteligencia Artificial para el Control de Robots Autónomos "Small Size"*, Tesis (Ingeniero en Telemática e Ingeniero en Computación) ITAM, 2009.

[55] Wang W., Jiang W., Yi Li, Qiu X., Kan Li, Zheng D. y Xiong R., *"ZJUNLICT Team Description for ROBOCUP 2008"*, State Key Laboratory of Industrial Control Technology Zhejiang, China, 2008.
[http://small-size.informatik.uni-bremen.de/tdp/2008/ZJUNlict_SSL_TDP_2008.pdf]

[56] Wei Cui, Liang Liu, Buo Guo Jun Tao, Xiang Gao y Wenlan Cai, *"AUA_Ares 2008 Team Description Paper"*, Department of Control Engineering Aviation University of Airforce, China, 2008.
[http://small-size.informatik.uni-bremen.de/tdp/2008/AUAAres_SSL_TDP_2008.pdf]

[57] Yamauchi K., Yoshimoto T., Horii S., Chiku T., Watanabe M., Itoh K. y Sugiura T., *"The description of team KIKS"*, Toyota National College of Technology Department of Electrical and Electronic Engineering, Japan, 2008.
[http://small-size.informatik.uni-bremen.de/tdp/2008/KIKS_SSL_TDP_2008.pdf]

- APÉNDICE A -

COMPONENTES ELECTRÓNICOS

En este apéndice se especifican, mediante tablas, las características de los componentes electrónicos seleccionados para la implementación cada sistema, con sus respectivos números de parte que los identifican en la empresa donde fueron adquiridos [14].

A.1 Suministro de Energía

A.1.1 Suministro de Energía Analógico

Etiqueta	DigiKey#	No. Parte	Descripción
BA1A	455-1648-ND	B2PS-VH(LF)(SN)	CONN HEADER VH SIDE 2POS 3.96MM
BA1B	455-1648-ND	B2PS-VH(LF)(SN)	CONN HEADER VH SIDE 2POS 3.96MM
BA1C	455-1648-ND	B2PS-VH(LF)(SN)	CONN HEADER VH SIDE 2POS 3.96MM
BA2A	455-1648-ND	B2PS-VH(LF)(SN)	CONN HEADER VH SIDE 2POS 3.96MM
BA2B	455-1648-ND	B2PS-VH(LF)(SN)	CONN HEADER VH SIDE 2POS 3.96MM
BA2C	455-1648-ND	B2PS-VH(LF)(SN)	CONN HEADER VH SIDE 2POS 3.96MM
CON_SW_AN	455-1649-ND	B3PS-VH(LF)(SN)	CONN HEADER VH SIDE 3POS 3.96MM

Tabla A.1 - Componentes para el suministro de energía analógico

A.1.2 Suministro de Energía Digital

Etiqueta	DigiKey#	No. Parte	Descripción
BD1	455-1648-ND	B2PS-VH(LF)(SN)	CONN HEADER VH SIDE 2POS 3.96MM
BD2	455-1648-ND	B2PS-VH(LF)(SN)	CONN HEADER VH SIDE 2POS 3.96MM
BD3	455-1648-ND	B2PS-VH(LF)(SN)	CONN HEADER VH SIDE 2POS 3.96MM
CON_SW_DIG	455-1649-ND	B3PS-VH(LF)(SN)	CONN HEADER VH SIDE 3POS 3.96MM
SWITCH	CKN5003-ND	1201M2S3CQE2	SWITCH SLIDE DPDT 6A PC MOUNT

Tabla A.2 - Componentes para el suministro de energía digital

A.1.3 Medidor de Baterías Suministro Analógico

Etiqueta	DigiKey#	No. Parte	Descripción
ALR	P11532CT-ND	LNJ211R82RA	LED RED FACE UP 1206
ALY	P11536CT-ND	LNJ411K84RA	LED AMBER FACE UP 1206
ALG	P11537CT-ND	LNJ311G83RA	LED GREEN FACE UP 1206
APG	306JC102B-ND	306JC102B	POT 1.0K OHM 6MM CERMET VERT
APR	306JC102B-ND	306JC102B	POT 1.0K OHM 6MM CERMET VERT
ARG	P2.2KASCT-ND	ERJ-P14J222U	RES ANTI-SURGE 2.2K OHM 5% 1210
AROG	P15KASCT-ND	ERJ-P14J153U	RES ANTI-SURGE 15K OHM 5% 1210
AROR	P15KASCT-ND	ERJ-P14J153U	RES ANTI-SURGE 15K OHM 5% 1210
AREF1	P2.2KASCT-ND	ERJ-P14J222U	RES ANTI-SURGE 2.2K OHM 5% 1210
AREF2	P2.2KASCT-ND	ERJ-P14J222U	RES ANTI-SURGE 2.2K OHM 5% 1210
ARR	P2.2KASCT-ND	ERJ-P14J222U	RES ANTI-SURGE 2.2K OHM 5% 1210
ANOR	296-14874-1-ND	SN74LS02DR	IC QUAD 2-INPUT NOR 14-SOIC
ACOM	296-1013-1-ND	LM339DR2	IC COMP QUAD SGL SUPPLY 14SOIC

Tabla A.3 - Componentes para el Medidor de Baterías Analógico

A.1.4 Medidor de Baterías Suministro Digital

Etiqueta	DigiKey#	No. Parte	Descripción
DLR	P11532CT-ND	LNJ211R82RA	LED RED FACE UP 1206
DLY	P11536CT-ND	LNJ411K84RA	LED AMBER FACE UP 1206
DLG	P11537CT-ND	LNJ311G83RA	LED GREEN FACE UP 1206
DPG	306JC102B-ND	306JC102B	POT 1.0K OHM 6MM CERMET VERT
DPR	306JC102B-ND	306JC102B	POT 1.0K OHM 6MM CERMET VERT
DRG	P1.0KASCT-ND	ERJ-P14J102U	RES ANTI-SURGE 1K OHM 5% 1210
DROG	P15KASCT-ND	ERJ-P14J153U	RES ANTI-SURGE 15K OHM 5% 1210
DROR	P15KASCT-ND	ERJ-P14J153U	RES ANTI-SURGE 15K OHM 5% 1210
DREF1	P2.2KASCT-ND	ERJ-P14J222U	RES ANTI-SURGE 2.2K OHM 5% 1210
DREF2	P2.2KASCT-ND	ERJ-P14J222U	RES ANTI-SURGE 2.2K OHM 5% 1210
DRR	P1.0KASCT-ND	ERJ-P14J102U	RES ANTI-SURGE 1K OHM 5% 1210
DNOR	296-14874-1-ND	SN74LS02DR	IC QUAD 2-INPUT NOR 14-SOIC
DCOM	296-1013-1-ND	LM339DR2	IC COMP QUAD SGL SUPPLY 14SOIC

Tabla A.4 - Componentes para el Medidor de Baterías Digital

A.1.5 Suministro de energía para del DSP

Etiqueta	DigiKey#	No. Parte	Descripción
POWER_DSP	455-1639-ND	B2P-VH(LF)(SN)	CONN HEADER VH TOP 2POS 3.96MM

Tabla A.5 - Componentes para el suministro de energía del DSP

A.2 Modulo Digital

A.2.1 Identificador del Robot

Etiqueta	DigiKey#	No. Parte	Descripción
7SEGDRV	296-3528-1-ND	CD4511BNSR	IC 7-SEG LED DECOD/DRVR 16-SOP
DSPLY	160-1514-5-ND	LTS-2301AB	LED 7-SEG .28" 1DIGIT BLUE CC
SWID	CKN6177-ND	CRD16RM0AB	SWITCH ROTARY DIP HEX FLUSH RA

Tabla A.6 - Componentes para el Identificador del Robot

A.2.2 Transceiver

Etiqueta	DigiKey#	No. Parte	Descripción
LEVEL_SHIFTER	296-8514-1-ND	SN74LVC4245ADBR	IC OCT BUS XCVR/SHIFTER 24-SSOP
TRANSCEIVER	----------	RPC-914/869-64	RADIOMETRIX TRANSCEIVER
NEG_TRAN	296-14081-1-ND	CD40106BM96	IC HEX SCHMITT TRIGGER 14-SOIC
1R12KTR	P12KWCT-ND	ERJ-12ZYJ123U	RES 12K OHM 1/2W 5% 2010 SMD
1R17KTR	P16.9KACCT-ND	ERJ-12SF1692U	RES 16.9K OHM 1/2W 1% 2010 SMD
2R12KTR	P12KWCT-ND	ERJ-12ZYJ123U	RES 12K OHM 1/2W 5% 2010 SMD
2R17KTR	P16.9KACCT-ND	ERJ-12SF1692U	RES 16.9K OHM 1/2W 1% 2010 SMD

Tabla A.7 - Componentes para el Sistema de Comunicación

A.3 Módulo Analógico

A.3.1 Control de motores

Etiqueta	DigiKey#	No. Parte	Descripción
1R39K_MN	P39KWCT-ND	ERJ-12ZYJ393U	RES 39K OHM 1/2W 5% 2010 SMD
2R39K_MN	P39KWCT-ND	ERJ-12ZYJ393U	RES 39K OHM 1/2W 5% 2010 SMD
1R100K_MN	P100KWCT-ND	ERJ-12ZYJ104U	RES 100K OHM 1/2W 5% 2010 SMD
2R100K_MN	P100KWCT-ND	ERJ-12ZYJ104U	RES 100K OHM 1/2W 5% 2010 SMD
1D_MN	1N4148WXTPMSCT-ND	1N4148WX-TP	DIODE SWITCH 75V 300MA SOD323
2D_MN	1N4148WXTPMSCT-ND	1N4148WX-TP	DIODE SWITCH 75V 300MA SOD324
1C1n_MN	311-1163-1-ND	CC1206JRNP09BN102	CAP CERAMIC 1000PF 50V NP0 1206
2C1n_MN	311-1163-1-ND	CC1206JRNP09BN103	CAP CERAMIC 1000PF 50V NP0 1207
2C68n_MN	399-1247-1-ND	C1206C683J5RACTU	CAP 68000PF 50V CERAMIC X7R 1206
1C68n_MN	399-1247-1-ND	C1206C683J5RACTU	CAP 68000PF 50V CERAMIC X7R 1206
C0.01u_MN	445-1803-1-ND	C1220X7R1H103K	CAP CER .01UF 50V 10% X7R 0508
R100_MN	P100WCT-ND	ERJ-12ZYJ101U	RES 100 OHM 1/2W 5% 2010 SMD
C220n_MN	399-1251-1-ND	C1206C224K5RACTU	CAP .22UF 50V CERAMIC X7R 1206
MOTDRIV_MN	497-4571-5-ND	E-L6207D	IC DUAL FULL BRIDGE DVR 24 SOIC

Tabla A.8 - Componentes para los controladores de los Motores.

En la tabla anterior N corresponde al número del motor (1, 2, 3, 4, 5)

A.3.2 Optoacopladores

Etiqueta	DigiKey#	No. Parte	Descripción
ENM_CMN	74OL6010-ND	74OL6010	OPTOCOUPLER CMOS BUFFER 6-DIP
DIRM_CMN	74OL6010-ND	74OL6010	OPTOCOUPLER CMOS BUFFER 6-DIP
2C0.1uE_CMN	PCC2188CT-ND	ECY-29RE104KV	CAP .1UF 25V CERAMIC 0508 X7R
1C0.1uE_CMN	PCC2188CT-ND	ECY-29RE104KV	CAP .1UF 25V CERAMIC 0508 X7R
2C0.1uD_CMN	PCC2188CT-ND	ECY-29RE104KV	CAP .1UF 25V CERAMIC 0508 X7R
1C0.1uD_CMN	PCC2188CT-ND	ECY-29RE104KV	CAP .1UF 25V CERAMIC 0508 X7R
R470E_CMN	P470WCT-ND	ERJ-12ZYJ471U	RES 470 OHM 1/2W 5% 2010 SMD
R470D_CMN	P470WCT-ND	ERJ-12ZYJ471U	RES 470 OHM 1/2W 5% 2010 SMD
PGEN	74OL6010-ND	74OL6010	OPTOCOUPLER CMOS BUFFER 6-DIP
MOSP	74OL6010-ND	74OL6010	OPTOCOUPLER CMOS BUFFER 6-DIP
2C0.1uPGEN	PCC2188CT-ND	ECY-29RE104KV	CAP .1UF 25V CERAMIC 0508 X7R
1C0.1uPGEN	PCC2188CT-ND	ECY-29RE104KV	CAP .1UF 25V CERAMIC 0508 X7R
2C0.1uMOSP	PCC2188CT-ND	ECY-29RE104KV	CAP .1UF 25V CERAMIC 0508 X7R
1C0.1uMOSP	PCC2188CT-ND	ECY-29RE104KV	CAP .1UF 25V CERAMIC 0508 X7R
R470PGEN	P470WCT-ND	ERJ-12ZYJ471U	RES 470 OHM 1/2W 5% 2010 SMD
R470MOSP	P470WCT-ND	ERJ-12ZYJ471U	RES 470 OHM 1/2W 5% 2010 SMD

Tabla A.9 - Componentes para los optoacopladores

En la tabla anterior N corresponde al número del motor (1, 2, 3, 4, 5)

A.3.3 Codificadores de motores (Encoders)

Etiqueta	DigiKey#	No. Parte	Descripción
R17KA_SMN	P16.9KACCT-ND	ERJ-12SF1692U	RES 16.9K OHM 1/2W 1% 2010 SMD
R17KB_SMN	P16.9KACCT-ND	ERJ-12SF1692U	RES 16.9K OHM 1/2W 1% 2010 SMD
R12KA_SMN	P12KWCT-ND	ERJ-12ZYJ123U	RES 12K OHM 1/2W 5% 2010 SMD
R12KB_SMN	P12KWCT-ND	ERJ-12ZYJ123U	RES 12K OHM 1/2W 5% 2010 SMD
CONM_SMN	WM8233-ND	90130-1106	CONN HEADER 6POS .100" STR TIN

Tabla A.10 - Componentes para los conectar los motores

En la tabla anterior N corresponde al número del motor (1, 2, 3, 4, 5)

A.3.4 Sistema de Control de pelota

Los componentes para el motor de este sistema son los mismos de las secciones A.3.1, A.3.2 y A.3.3.

A.3.5 Sistema de Pateo

Etiqueta	DigiKey#	No. Parte	Descripción
DCDC	IRF100S	IRF100S	SERIES IR / 10 WATTS DC-DC CONVERTERS
CAP	455-1639-ND	B2P-VH(LF)(SN)	CONN HEADER VH TOP 2POS 3.96MM
SOL	455-1639-ND	B2P-VH(LF)(SN)	CONN HEADER VH TOP 2POS 3.96MM
MOSFET	IXFH50N20-ND	IXFH50N20	MOSFET N-CH 200V 50A TO-247AD
CONKICK	455-1642-ND	B5P-VH(LF)(SN)	CONN HEADER VH TOP 5POS 3.96MM
RR	P1.0KWCT-ND	ERJ-12ZYJ102U	RES 1.0K OHM 1/2W 5% 2010 SMD
RC	P1.0KWCT-ND	ERJ-12ZYJ102U	RES 1.0K OHM 1/2W 5% 2010 SMD
DR	497-2500-1-ND	STTH102A	DIODE ULTRAFAST 200V 1A SMA
D1 TRAN	497-2500-1-ND	STTH102A	DIODE ULTRAFAST 200V 1A SMA
T1	497-2598-ND	2N2222A	TRANSISTOR NPN 75V 0.6A TO-18

Tabla A.11 - Componentes del Sistema de Pateo

Etiqueta	DigiKey#	No. Parte	Descripción
R1LED	P330WCT-ND	ERJ-12ZYJ331U	RES 330 OHM 1/2W 5% 2010 SMD
R2FTRAN	P820WCT-ND	ERJ-12ZYJ821U	RES 820 OHM 1/2W 5% 2010 SMD
R1FTRAN	P330WCT-ND	ERJ-12ZYJ331U	RES 330 OHM 1/2W 5% 2010 SMD
R3FTRAN	P1.69KACCT-ND	ERJ-12SF1691U	RES 1.69K OHM 1/2W 1% 2010 SMD
R4FTRAN	P820WCT-ND	ERJ-12ZYJ821U	RES 820 OHM 1/2W 5% 2010 SMD
LED	WM4200-ND	22-23-2021	CONN HEADER 2POS .100 VERT TIN
FTRAN	WM4200-ND	22-23-2021	CONN HEADER 2POS .100 VERT TIN
ENINFRA	74OL6010-ND	74OL6010	OPTOCOUPLER CMOS BUFFER 6-DIP
1C0.1uE	PCC2188CT-ND	ECY-29RE104KV	CAP .1UF 25V CERAMIC 0508 X7R
2C0.1uE	PCC2188CT-ND	ECY-29RE104KV	CAP .1UF 25V CERAMIC 0508 X7R
R470E	P470WCT-ND	ERJ-12ZYJ471U	RES 470 OHM 1/2W 5% 2010 SMD

Tabla A.12 - Componentes del Sensor de Pelota

A.3.6 Conexión del DSP con la Tarjeta Digital-Analógica

Etiqueta	DigiKey#	No. Parte	Descripción
CONDSP1	--------	--------	CONECTOR DE 40 VÍAS
CONDSP2	--------	--------	CONECTOR DE 20 VÍAS

Tabla A.13 - Componentes para la conexión del DSP